

# Unity Game Engine

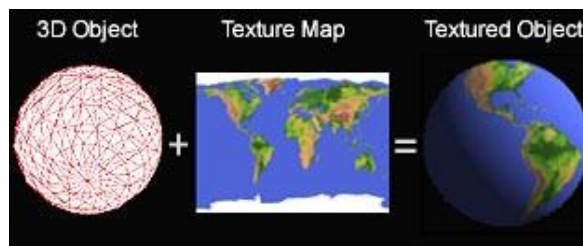
Introduction to Unity – Textures

<https://docs.unity3d.com/Manual/class-TextureImporter.html>

3D Computer Game Programming

## 2D Texture Image

- **Textures** bring your **Meshes**, **Particles**, and **interfaces** to life!
- A **texture map** or **texture image** is a two-dimensional image file.

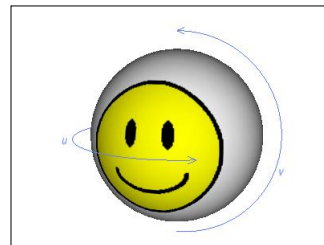
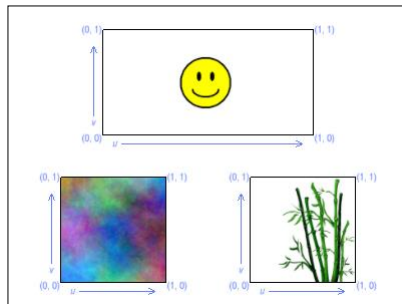


3D Computer Game Programming



## Texture Coordinates

- **Texture coordinates**--a special  $(u, v)$  coordinate pair that is associated with each vertex of your model
  - $(u, v)$  coordinate range such that the  $u$  coordinate ranges from 0 to 1 from right to left, and the  $v$  coordinate ranges from 0 to 1 from bottom to top.

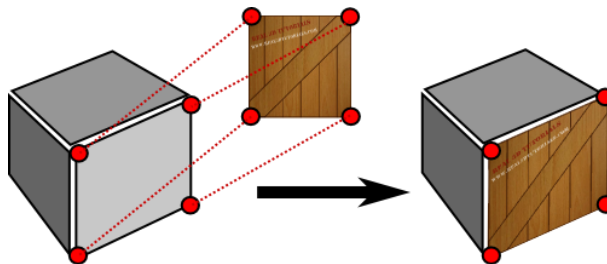


3D Computer Game Programming



## UV Mapping (Texture Mapping)

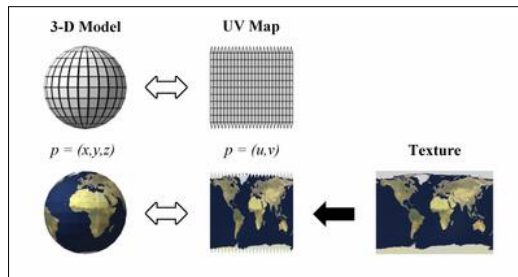
- When mapping a 2D texture onto a 3D model, some sort of wrapping is done. This is called **UV mapping** and is done in your 3D modelling app.



3D Computer Game Programming



## UV Mapping (Texture Mapping)



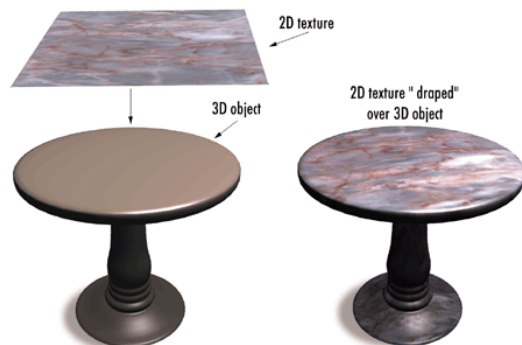
$$u = 0.5 + \frac{\arctan 2(d_z, d_x)}{2\pi}$$

$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}$$

3D Computer Game Programming



## UV Mapping (Texture Mapping)



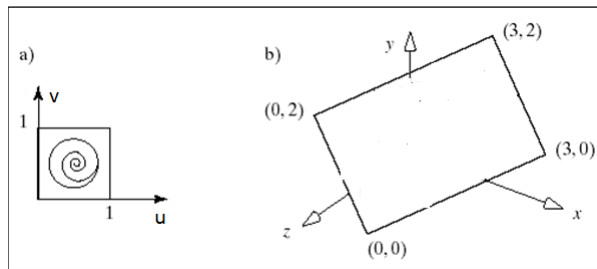
$$(u, v) \leftrightarrow (x, y, z)$$

3D Computer Game Programming



## Texture Wrap Modes

- It is also legal to use texture coordinates  $(u,v)$  that go outside this range; you can have negative values, for instance, or numbers higher than 1.



4 corner vertices of the 3D plane have the corresponding  $(u,v)$ s as follows:

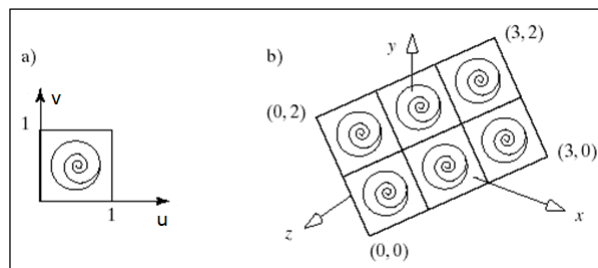
$(0,0)$ ,  $(0,2)$ ,  
 $(3,0)$ ,  $(3,2)$

3D Computer Game Programming



## Repeat - Tiling a Texture

- **Tile** the texture = making it repeat.
- Use  $u$  and  $v$  values that are larger than 1.0.
  - For example,  $u$  value of 2.67 causes the renderer to use  $u = 0.67$ .
  - The integer part is the current number of repeats of the pattern.



3D Computer Game Programming

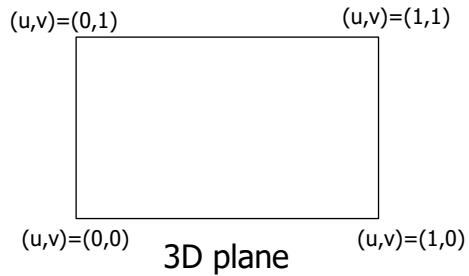


## Repeat - Tiling a Texture

- $(u,v)$  values associated with  $(x,y,z)$  can be **scaled** and **translated**.



texture



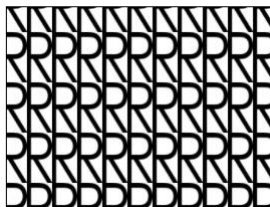
Scale  $(u,v)$  values by  $sx=2$  and  $sy=2$ .  
Translate  $(u,v)$  values by  $tx=-1$ ,  $ty=0$ .

3D Computer Game Programming



## Texture Wrap Modes

R



**Repeat**

R



**Clamp**

3D Computer Game Programming



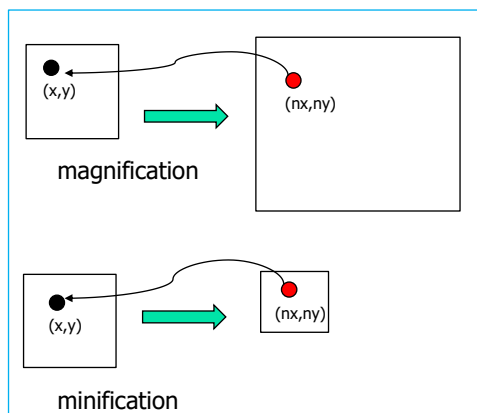
## Texture Filter Types

- The pixels of a texture image rarely matches one-to-one with actual screen pixels. Usually, it is the case that either it is stretched over (**texture magnification**--the texture image is stretched bigger), or the opposite (**texture minification**--the texture image is squished smaller).
- You can control how the texture looks when it is magnified or minified by setting its **filter type/mode**.

3D Computer Game Programming



## Magnification and Minification



Given the input texture  $T$ , we need to find magnified or minified texture image  $N$ .

In other words, for every pixel at  $(nx, ny)$  of  $N$ , we need to calculate which pixels from  $T$  can be used.

Given  $(nx, ny)$ , get  $(x, y)$  by multiplying a magnification/minification factor.

3D Computer Game Programming



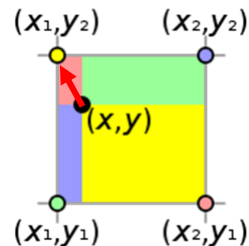
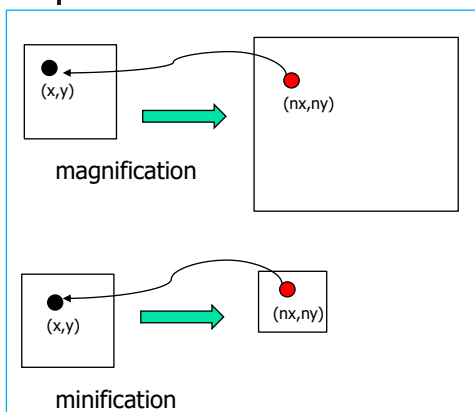
## Texture Filter Types

- Filter types are for both magnification and minification
  - Sample the nearest pixel
  - Sample the four nearest pixels, and linearly interpolate them.

3D Computer Game Programming



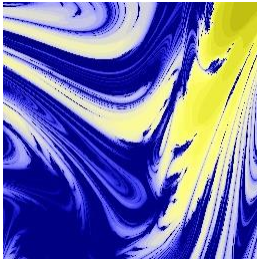
## Nearest Neighbor



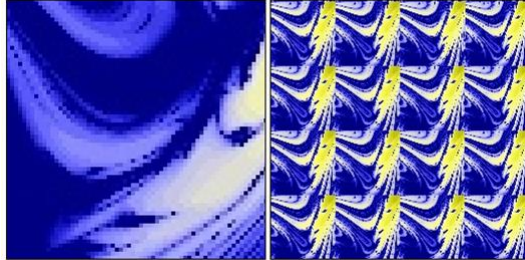
Given  $(nx, ny)$ , get  $(x,y)$  by multiplying a magnification/minification factor. Then, choose the nearest neighbor.

3D Computer Game Programming

## Texture Filter - Nearest Point



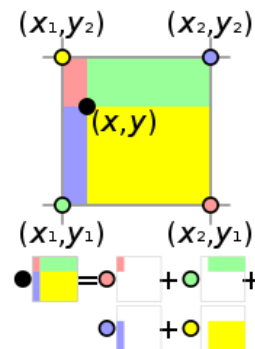
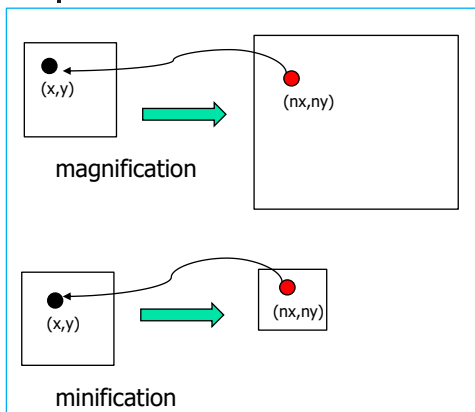
Texture



Magnification (left) and Minification (right)

3D Computer Game Programming

## Bilinear Interpolation



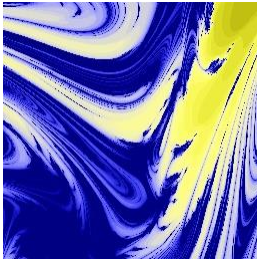
Given  $(nx, ny)$ , get  $(x, y)$  by multiplying a magnification/minification factor. Then, perform bilinear interpolation as shown in the diagram.

3D Computer Game Programming

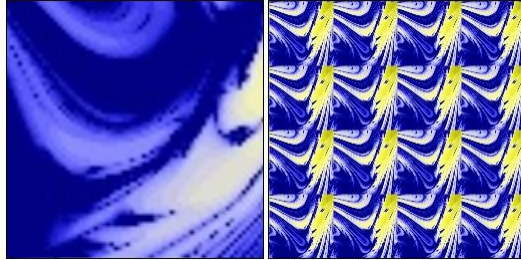




## Texture Filter – Bilinear Interpolation



Texture

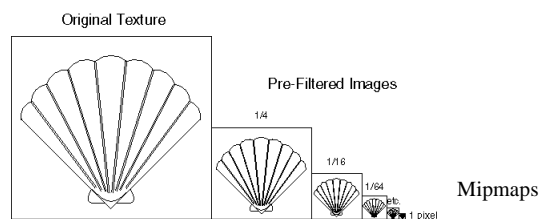


Magnification (left) and Minification (right)

3D Computer Game Programming

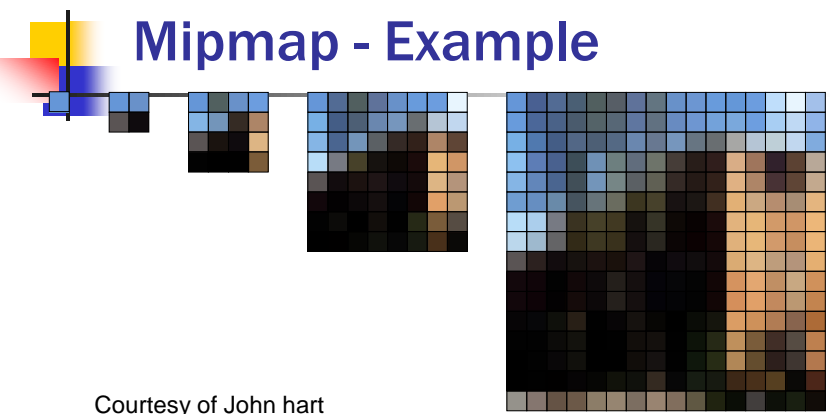
## Mipmaps: Multiple Levels of Detail

- As textured object moves farther from the camera, the texture map must decrease in size. If the object is moved farther away from the viewer until it appears on the screen as a single pixel, then the filtered textures may appear to change abruptly at certain transition points.
- To avoid such artifacts, you can specify a series of prefiltered texture maps of decreasing resolutions, called *mipmaps*. (*Mip* stands for the Latin *multum in parvo*)



3D Computer Game Programming

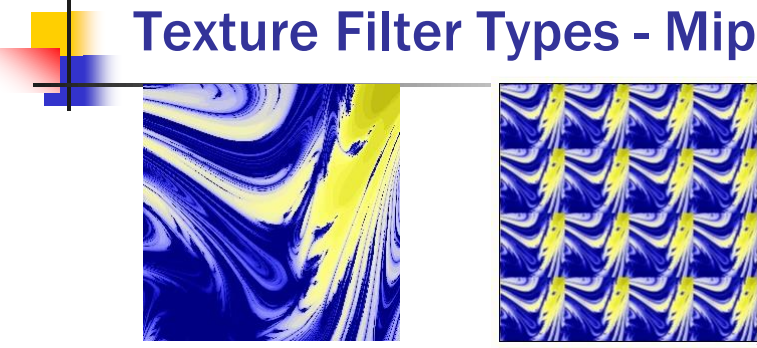
## Mipmap - Example



Courtesy of John hart

3D Computer Game Programming

## Texture Filter Types - Mipmaps



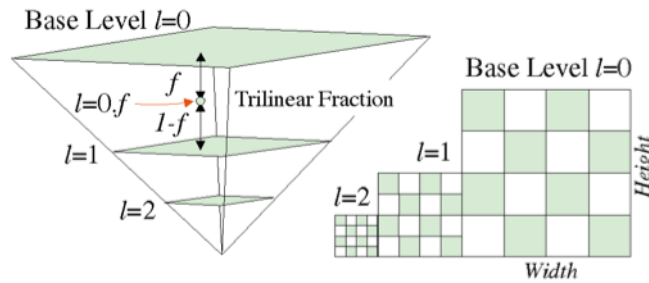
- Mipmaps
  - (1) It requires 33% more texture memory (per mipmapped texture), but it renders quickly.
  - (2) It helps the texture look much smoother than filtering alone when it is minified.
  - (3) Mipmapping doesn't have anything at all to do with magnification.

3D Computer Game Programming



## Mipmaps – Trilinear Filtering

- Trilinear Filtering

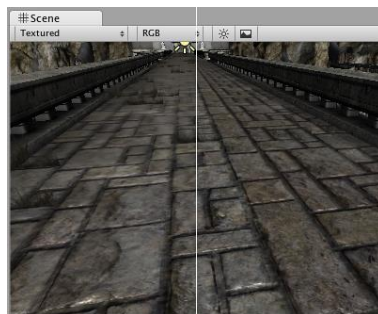


3D Computer Game Programming



## Texture Filter Types - Anisotropic filtering

- Anisotropic filtering increases texture quality when viewed from a grazing angle, at some expense of rendering cost (the cost is entirely on the graphics card). Increasing anisotropy level is usually a good idea for ground and floor textures.

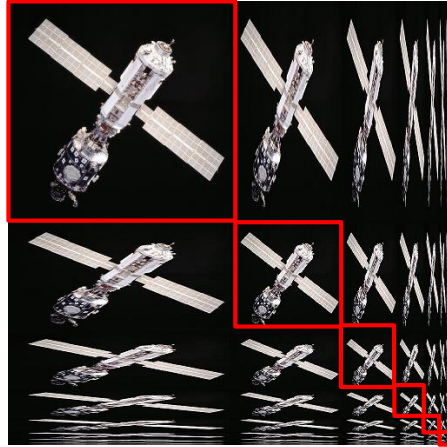


Good for surfaces that are at oblique viewing angles with respect to the camera

3D Computer Game Programming

## Texture Filter Types - Anisotropic filtering

- In addition to downsampling to  $128 \times 128$ , images are also sampled to  $256 \times 128$  and  $32 \times 128$  etc.
- These *anisotropically downsampled* images can be probed when the texture-mapped image frequency is different for each texture axis. Therefore, one axis need not blur due to the screen frequency of another axis



An example of *anisotropic* mipmap image storage

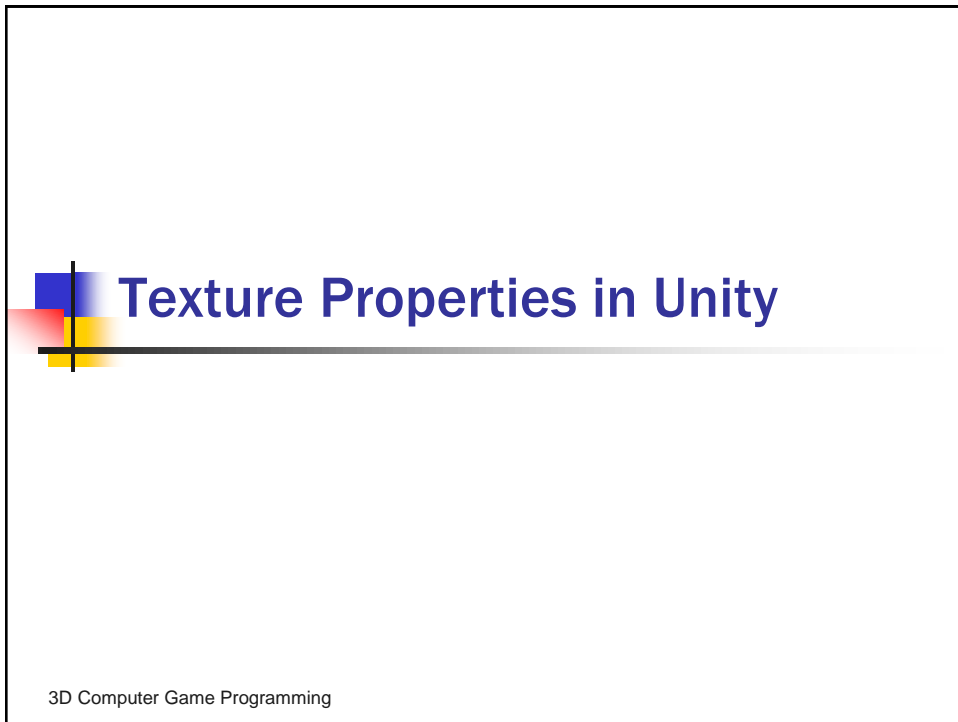
3D Computer Game Programming


## Trilinear vs Anisotropic



Image from Wikipedia. An illustration of texture filtering methods showing a trilinear mipmapped texture on the left and the same texture enhanced with anisotropic texture filtering on the right.

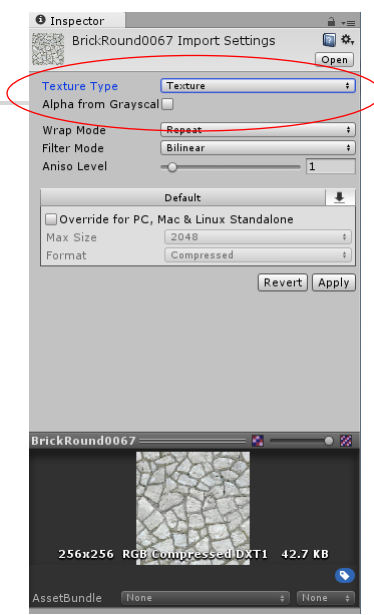
3D Computer Game Programming





## Texture Inspector

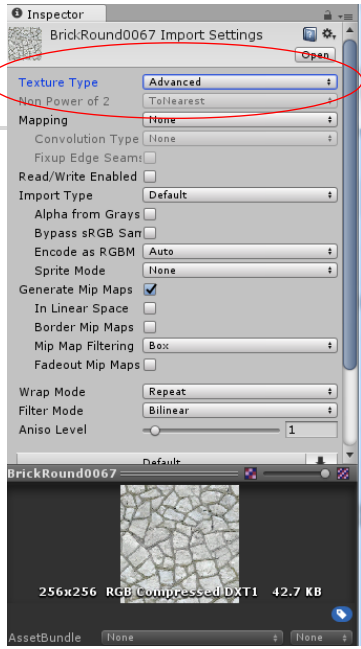
- **Texture Type:** Select this to set basic parameters depending on the purpose of your texture.
  - **Texture:** This is the most common setting used for all the textures in general.
  - **Normal Map:** Select this to turn the color channels into a format suitable for real-time normal mapping.
  - **Editor GUI :** Use this if your texture is going to be used on any HUD/GUI Controls.
  - **Sprite (2D and UI):** This must be selected if your texture will be used in a 2D game as a **Sprite**.



3D Computer Game Programming

# Texture Inspector

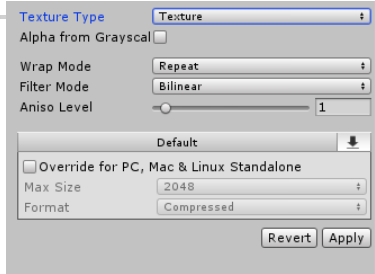
- Texture Type:** Select this to set basic parameters depending on the purpose of your texture.
  - Cubemap:** Cubemap, often used to create reflections.
  - Cookie :** This sets up your texture with the basic parameters used for the Cookies (masks) of your lights.
  - Advanced:** Select this when you want to have specific parameters on your texture and you want to have total control over your texture.



3D Computer Game Programming

# Texture Type – “Texture”

- Texture**
  - Alpha From Grayscale:** If enabled, an alpha transparency channel will be generated by the image's existing values of light and dark.
  - Wrap Mode**
    - Repeat :** The Texture repeats (tiles) itself
    - Clamp:** The Texture's edges get stretched
  - Filter Mode**
    - Point**
    - Bilinear**
    - Trilinear**
  - Aniso Level :** Increases texture quality when viewing the texture at a steep angle. Good for floor and ground textures.



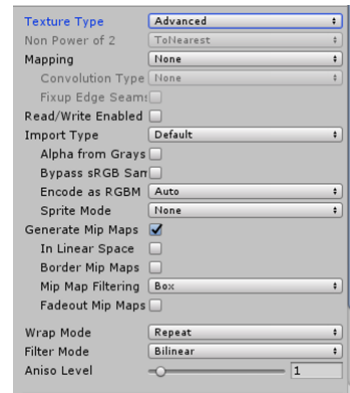
3D Computer Game Programming



## Texture Type – “Advanced”

### ■ Advanced

- **Non Power of 2:** If texture has non-power-of-two size, this will define a scaling behavior *at import time*.
  - **None:** Texture size will be kept as-is.
  - **To nearest:** Texture will be scaled to the nearest power-of-two size at import time.
  - **To larger:** Texture will be scaled to the next larger power-of-two size at import time.
  - **To smaller:** Texture will be scaled to the next smaller power-of-two size at import time.



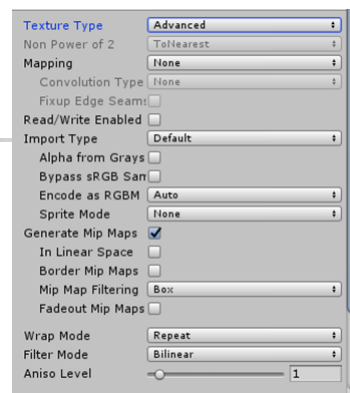
3D Computer Game Programming



## Texture Type

### ■ Advanced

- **Mapping:** Should a Cubemap be generated from this texture?
- **Read/Write Enabled :** Select this to enable access to the texture data from scripts using GetPixels, SetPixels and other Texture2D functions.
- **Import Type:** The way the image data is interpreted.
  - **Default (standard texture), Normal Map (texture as a normal map), Light Map (texture as a light map) .**
  - **Alpha from grayscale:** (Default mode only) Generates the alpha channel from the luminance information in the image.
  - **Bypass sRGB sampling :** (Default mode only) Use the exact colour values from the image rather than compensating for gamma (useful when the texture is for GUI or used as a way to encode non-image data)



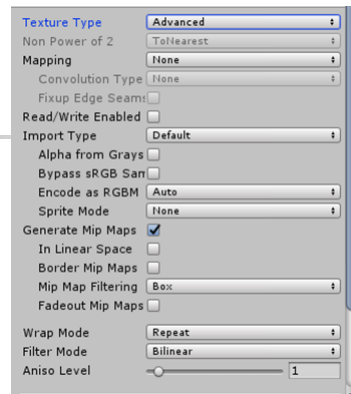
3D Computer Game Programming



## Texture Type

### ■ Advanced

- **Generate Mip Maps:** Select this to enable mip-map generation.
  - **In Linear Space:** Generate mipmaps in linear colour space.
  - **Border Mip Maps:** Select this to avoid colors seeping out to the edge of the lower Mip levels.
  - **Mip Map Filtering:** **Box** (the mip levels become smoother and smoother as they go down in size.) or **Kaiser** (sharper mipmaps).
  - **Fade Out Mipmaps:** Enable this to make the mipmaps fade to gray as the mip levels progress.
- **Wrap Mode, Filter Mode, Aniso Level :** Same as “Texture” Texture Type.



3D Computer Game Programming



## Supported Formats in Unity

- Unity can read the following file formats: PSD, TIFF, JPG, TGA, PNG, GIF, BMP, IFF, PICT.
- Unity can import multi-layer PSD and TIFF files just fine.
  - They are flattened automatically on import but the layers are maintained in the assets themselves. This is important as it allows you to just have one copy of your textures that you can use from Photoshop, through your 3D modelling app and into Unity.

3D Computer Game Programming





## Texture Sizes

- Texture sizes should be powers of two on the sides. For example, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 etc. pixels.
- The textures do not have to be square, i.e. width can be different from height.
- Note that each platform may impose maximum texture sizes.
- Non power of two (NPOT) texture sizes generally take slightly more memory and might be slower to read by the GPU.
  - So for performance it's best to use power of two sizes whenever you can.

3D Computer Game Programming



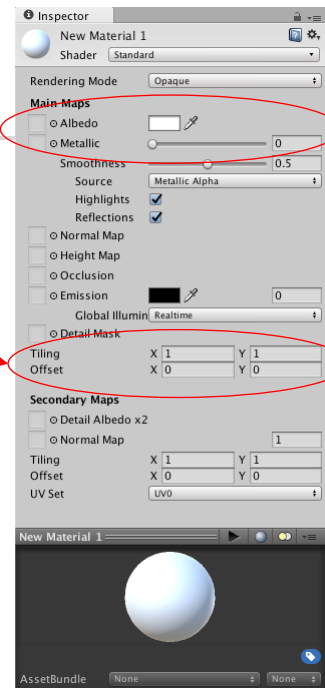
## Texture Sizes (2)

- If the platform or GPU does not support NPOT texture sizes, then Unity will scale and pad the texture up to next power of two size
  - This will use even more memory and make loading slower (in practice, this can happen on some older mobile devices).
  - In general you'd want to use non power of two sizes only for GUI purposes.
- Non power of two texture assets can be scaled up at import time using the **Non Power of 2** option in the advanced texture type in the import settings.

3D Computer Game Programming

# Material with Texture

- Albedo
- Altering/transforming (u,v)s of the mesh
  - Scale
  - Translation



3D Computer Game Programming