cisc3660
game programming
fall 2012
lecture # III.1

**topics:**

- transformations
- camera
- visual effects: lighting, shading, materials
- texture mapping

**references:**

- `http://www.evl.uic.edu/spiff/class/cs426/`, by Prof Jason Leigh, University of Illinois at Chicago (`http://www.evl.uic.edu/spiff/`) and Prof Robert Kooima, Louisiana State University (`http://csc.lsu.edu/~kooima/`)
- Blender Game Engine Overview, User Manual version 2.6
  `http://wiki.blender.org/index.php/Doc:2.6/Manual/Lighting`
  `http://wiki.blender.org/index.php/Doc:2.6/Manual/Materials`
  `http://wiki.blender.org/index.php/Doc:2.6/Manual/Textures`

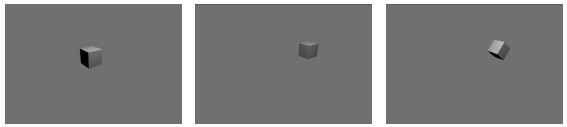


## transformations

- there are 3 types of transformations:
  - translation (called "grab" in blender) — move object along $x$, $y$ or $z$ axes
  - scale — change size of object along $x$, $y$ or $z$ axes
  - rotation — rotate object about $x$, $y$ or $z$ axes
- origin of object for transformation matters
- order of transformations matters, if the transformations are performed within the local coordinate system of the object



- example: translate, then rotate



start | translate $y+=1$ | then rotate $x+=10$

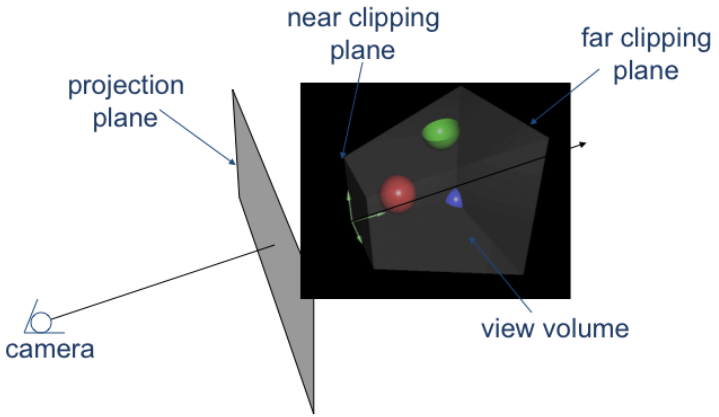
- example: rotate, then translate

start | rotate $x+=10$ | then translate $y+=1$



## camera

- not only is position of camera important, but also the planes that define what it sees
- the *near clipping plane* defines the closest objects the camera can see
- the *far clipping plane* defines the furthest objects the camera can see
- the *view volume* is the 3D volume bounded by the near and far clipping planes that define the volume that the camera can see
- the *projection plane* defines how the 3D view volume is projected onto a 2D region for displaying on a computer screen
- illustration on next page:




defining what the camera can see

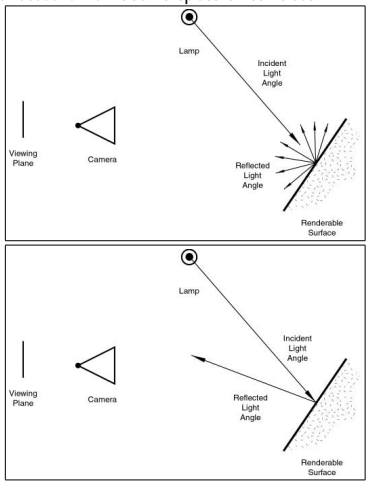


## lighting

- there are different types of light:
  - *ambient*—evenly illuminates all objects in scene
  - *directional*—all light rays are parallel (i.e., in the same direction)
  - *point*—all light rays emanate from one point
  - *spot*—all light rays emanate from one point and radiate out in a cone shape
- reflection:
  - *diffuse*
    - the greater the angle between the vector normal (perpendicular) to the surface and a vector to the light source, the less light is reflected
    - most light is reflected at $0^\circ$
    - no light is reflected at $90^\circ$
  - *specular*
    - maximum specular reflectance occurs when viewpoint is along path of perfectly reflected ray



- how light is reflected on diffuse vs specular surfaces:

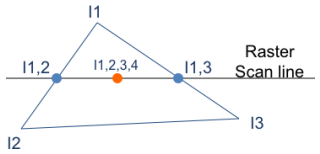




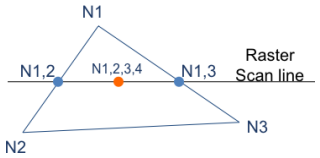

- lighting effects in blender:
  - color of ambient light in the world
  - ambient occlusion
  - effect of ambient light on object's material
  - indirect lighting (color that another object radiates onto object of concern)
  - lamps
- lighting settings:
  - *type* of light
  - color
  - position and direction
  - other settings include energy level and falloff (attenuation)
    - attenuation can be linear or quadratic or mixed

## shading

- shading defines how light behaves when it hits the surface of an object
- various methods are used to interpolate value of shadow
  - Gouraud shading: interpolate intensity of light along edges of polygon

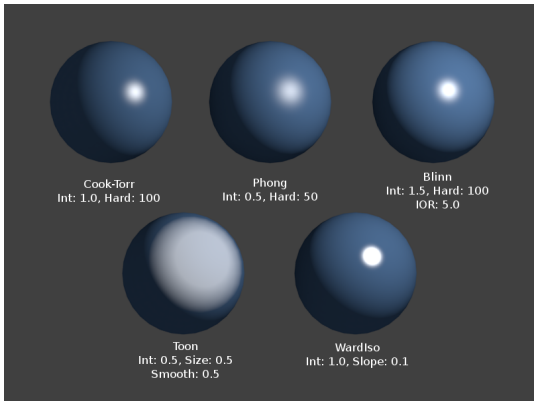



  - Phong shading: interpolate normals



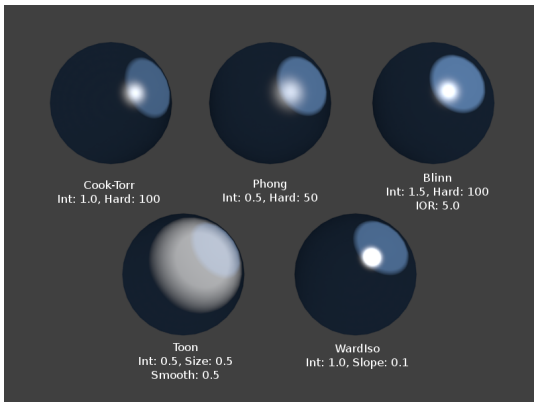



- Lambert diffuse shaders (in blender):



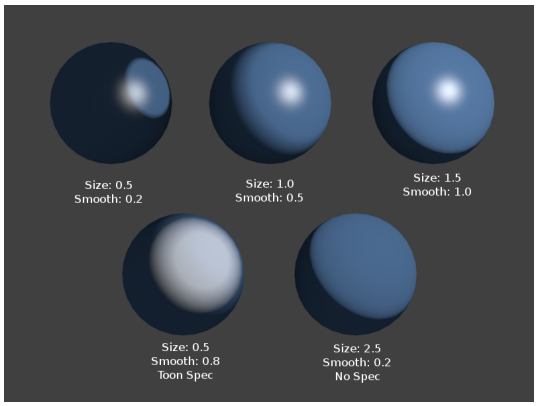



- Toon diffuse shader (in blender):






- various parameter settings for Toon shader, for example (in blender):



## materials properties in blender

- diffuse shaders
  - defines color of material when light hits it
  - shadows controlled by *falloff* settings
  - color/shading are independent of viewpoint
- specular shaders
  - defines bright highlights of glossy materials
  - specular reflection defined by *Snell's Law*, which basically says that light will be reflected with regard to the surface normal, based on the incident angle of the light source
  - color/shading are dependent on viewpoint
  - note that specular reflection is *not* mirroring (which is achieved in blender using raytracing)



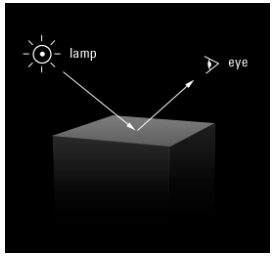
- ambient light effect
  - defines the amount of ambient light that hits an object
  - can be modulated using *environment lighting* (e.g., lamp property settings) and *ambient occlusion* (i.e., object(s) blocking the light, placed between the light source and the object of concern)
- color ramps
  - defines range of color that is blended in when object is in shadow
  - controls *color gradient*
  - takes precedence over textures in blender
- raytraced reflection
  - defines "mirror effect"
  - light ray emanates from camera and bounces off nearest object—depending on transparency settings: opaque objects cause light to bounce off with the same color settings as the original light; transparent objects cause light to go through the object with a modulated color, depending on the color of the transparent object and the amount of transparency
  - note that, in blender, *raytracing* needs to be turned on in the scene properties



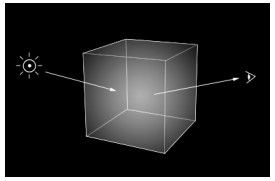
- raytraced transparency
  - defines refraction of light rays when they travel through transparent objects
  - as above, transparent objects cause light to go through the object with a modulated color, depending on the color of the transparent object and the amount of transparency
  - *index of refraction* for object's material defines how light is reflected
- subsurface scattering
  - some materials (e.g., human skin) have "layers", so reflection properties are essentially a combination of the properties of each layer, modulated according to which layer is on the "top" (closest to light source)
- strands
  - defines how hair is rendered in blender
  - based on multiple polygons, which can be rendered in different ways
  - refer to blender manual for details



- volume materials
  - defines rendering method for when light passes through an object
  - two types: *solid rendering* (e.g., solid object) and *volume rendering* (e.g., cloud, mist)




solid rendering



volume rendering

## texture mapping

- use images (e.g., jpg, gif) of textures and *wrap* these on the faces of objects in blender
- the *UV* editor in blender handles this
  - a 3D object is defined in $x$-$y$-$z$ space
  - a 2D face is defined in $u$-$v$ space (where $u$ refers to up/down; and $v$ refers to left/right)
- a good source for textures is:
  `http://www.mayang.com/textures`
- how to map a texture to a surface in blender:
  - select object whose face you want to map with a texture
  - go into *object Edit Mode*
  - click on "U" and then "Unwrap" and the "Face select" button to be able to select one face
  - right-click the face to edit
  - select *UV/Image editor* panel (on properties panel)
  - select "Image" - "Open image" and select an image file (e.g., jpg) to map to selected object face



  - back in the 3D view window, set the viewport shading mode to "Textured"
  - now you should see the texture-mapped face
- modify texture:
  - you can use the object manipulation controls to change the scale and shape of the texture map
  - for example, make the texture map small in order for it to repeat on the object surface
  - or make it big to zoom it on the object surface
  - or modify vertices of texture map to distort it on the object surface