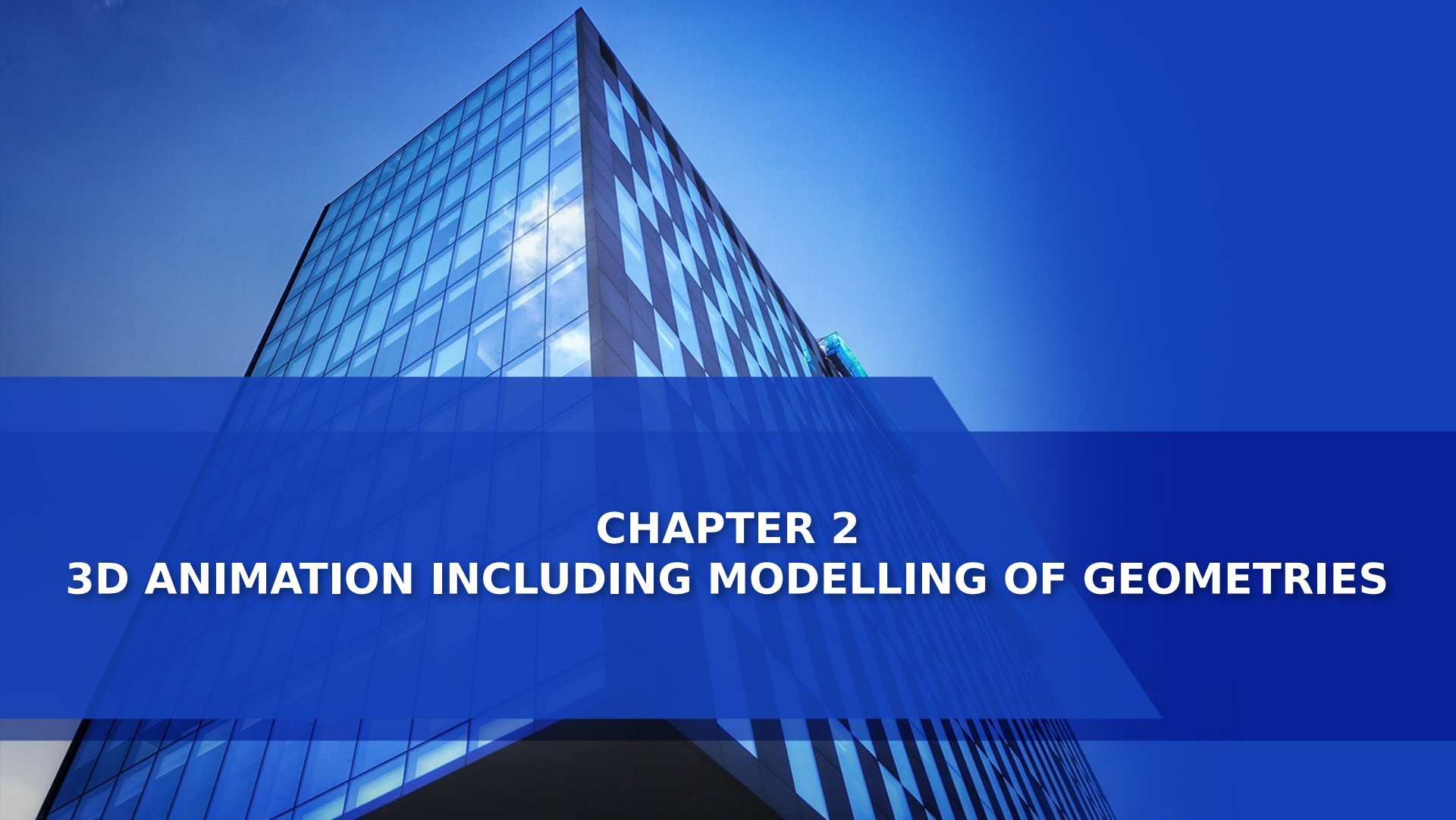
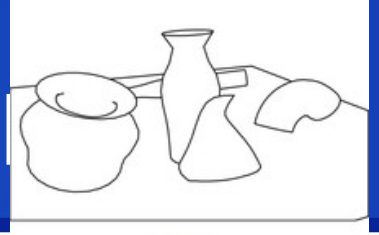


CHAPTER 2
3D ANIMATION INCLUDING MODELLING OF GEOMETRIES

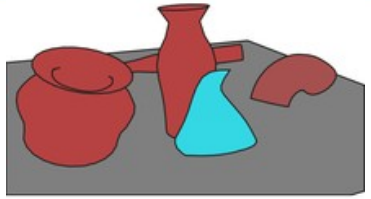
Contents

1. **Rendering (computer graphics)**
   - How this 3d rendering works
   - Camera projection using blender
   - Programming Environment in Blender
   - 3D Shapes Geometry – Faces, Edges, and Vertices
   - Rendering Features
   - The rendering equation
   - 3D modeling
2. **Wire-frame model**
3. **Scanline rendering**
4. **Radiosity (computer graphics)**
5. **Subsurface scattering**
6. **Shadow mapping**
7. **High-dynamic-range rendering**

Rendering (computer grap



- Rendering or image synthesis is the automatic process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program.
- The resulting image is referred to as the render.
- Multiple models can be defined in a scene file containing objects in a strictly defined language or data structure.
- The scene file contains geometry, viewpoint, texture, lighting, and shading information describing the virtual scene.
- A variety of rendering techniques applied to a single 3D scene









How this 3d rendering works

Camera Projection:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f * pX}{zX * sX} & s & 0 & 0 \\ 0 & \frac{f * pY}{zY * sY} & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(rotX) & -\sin(rotX) & 0 \\ 0 & \sin(rotX) & \cos(rotX) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(rotY) & \sin(rotY) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(rotY) & \cos(rotY) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(rotZ) & -\sin(rotZ) & 0 & 0 \\ \sin(rotZ) & \cos(rotZ) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -locX \\ 0 & 1 & 0 & -locY \\ 0 & 0 & 1 & -locZ \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

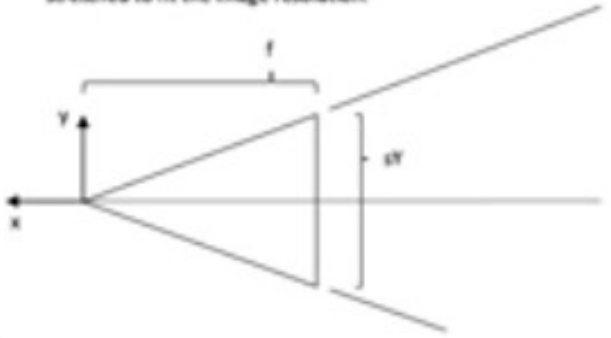
X, Y, Z ...	3d world coordinates (in world units)
locX, locY, locZ ...	camera translation (in world units)
rotX, rocY, rocZ ...	camera rotation (in radian -> 2π = 360°)
pX, pY ...	image resolution (in Pixel)
sX, sY ...	sensor size (in world units)
f ...	focal length
s ...	skew
x, y, z ...	2d image coordinates

Perspective:

The variables x and y represent the coordinates of a projection onto the image plane. However, at this stage the points are projected orthographically. In order to introduce perspective, the 2d image coordinates **x** and **y** need to be divided by z. Z itself remains as depth, channel.

Camera Model:

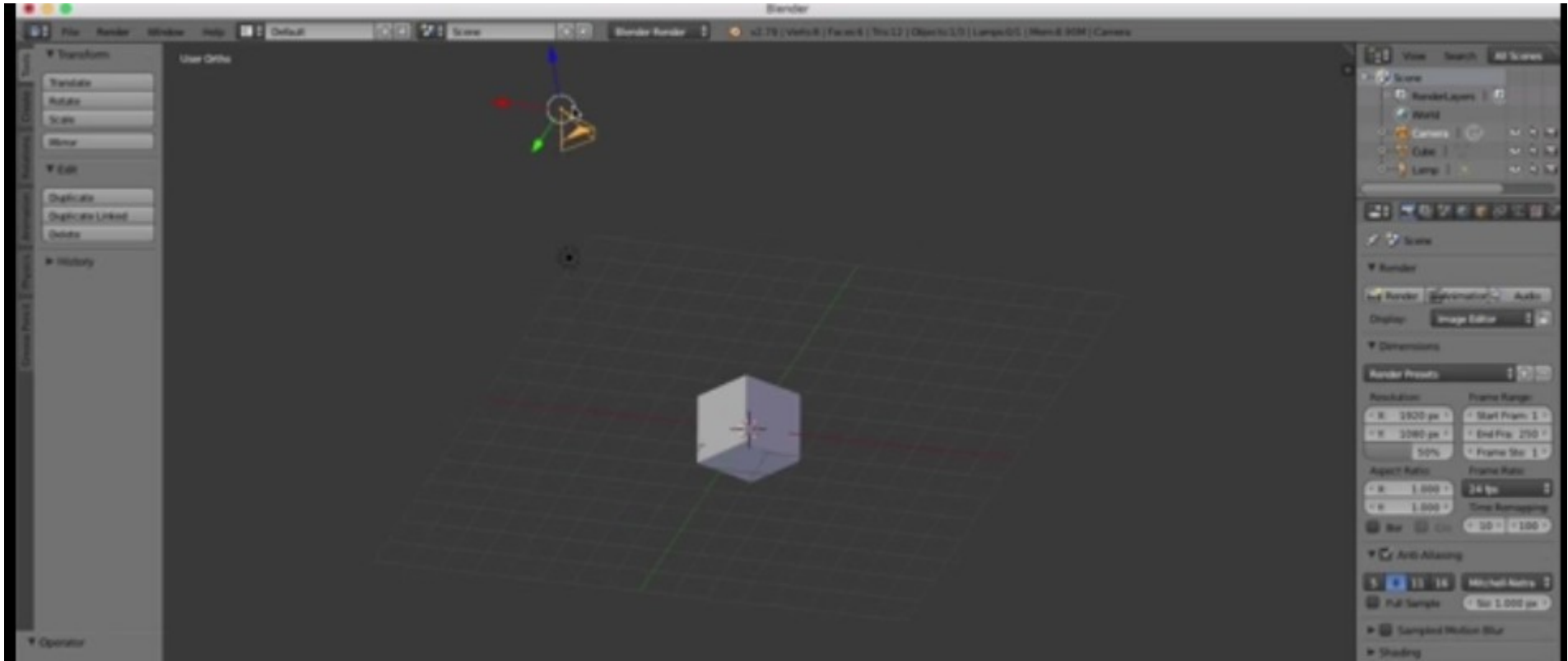
The ratio between focal length and the sensor size determine the field of view. The result will be stretched to fit the image resolution.



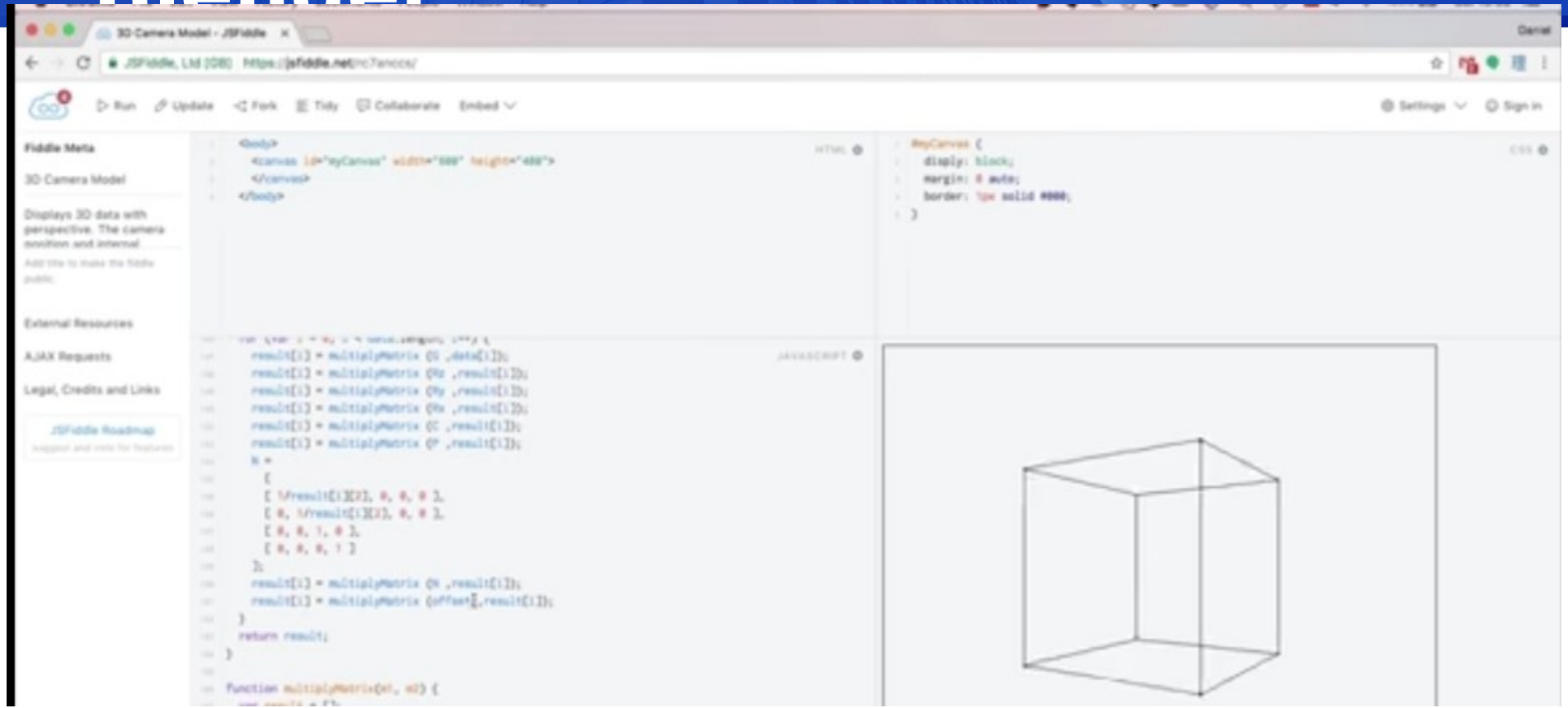


- The input coordinates are multiplied to get the result of output coordinates.
- In a 3d space, cube can be represented in an image form.
- A camera is positioned at the middle of the object origin representing x,y and z axis.
- Sort of move all of information happens away from the centre into camera space where the translation begins.
- In case of translation and scaling calculation becomes simple, but determining rotation is quite hard. Not that hard to make it simple rotX, rotY and rotZ should be substituted with the corresponding radians that is the degrees. For instance if the radians is π for the rotX/rotY/rotZ then 3.14 is assigned.
- Camera uses a certain resolution coordinates with respect to the camera model in order to make camera projection and make perspectives.

Camera projection using blender



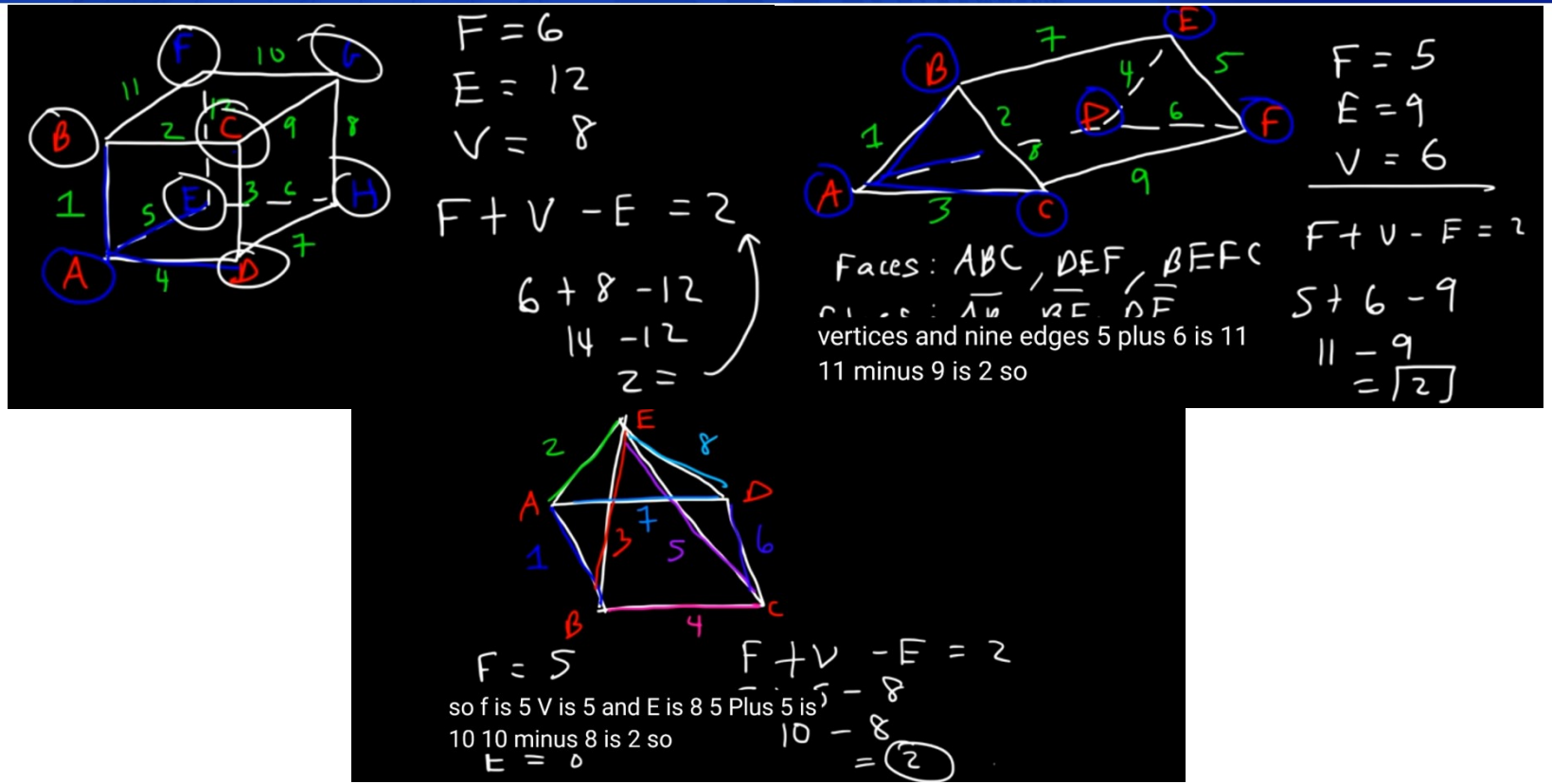
Programming Environment in Blender



3D Shapes Geometry – Faces, Edges, and Vertices

- Sphere
- Cylinder
- Cone
- Rectangular prism
- Triangular prism
- Square pyramid
- Cube
- Faces, Edges and Vertices for cube, triangular prism, square based pyramid – Harmony with Euler's formula

Faces, Edges and Vertices for cube, triangular prism, square based pyramid



Rendering Features

- Shading – how the color and brightness of a surface varies with lighting
- Texture-mapping – a method of applying detail to surfaces
- Bump-mapping – a method of simulating small-scale bumpiness on surfaces Techniques
- Therefore, a few loose families of more-efficient light transport modelling **techniques** have emerged: rasterization, including scanline rendering, geometrically projects objects in the scene to an image plane, without advanced optical effects;
- Ray casting considers the scene as observed from a specific point of view, calculating the observed image based only on geometry and very basic optical laws of reflection intensity, and perhaps using Monte Carlo techniques to reduce artifacts;
- Ray tracing is similar to ray casting, but employs more advanced optical simulation, and usually uses Monte Carlo techniques to obtain more realistic results at a speed that is often orders of magnitude faster.

The rendering equation

This is the key academic/theoretical concept in rendering. It serves as the most abstract formal expression of the non-perceptual aspect of rendering. All more complete algorithms can be seen as solutions to particular formulations of this equation.

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

Meaning: at a particular position and direction, the outgoing light (L_o) is the sum of the emitted light (L_e) and the reflected light. The reflected light being the sum of the incoming light (L_i) from all directions, multiplied by the surface reflection and incoming angle. By connecting outward light to inward light, via an interaction point, this equation stands for the whole 'light transport' — all the movement of light — in a scene.

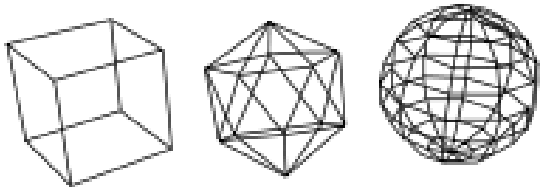
Wire-frame model

- A wire-frame model, also wireframe model, is a visual representation of a three-dimensional (3D) physical object used in 3D computer graphics.
- It is created by specifying each edge of the physical object where two mathematically continuous smooth surfaces meet, or by connecting an object's constituent vertices using (straight) lines or curves.
- An object is specified by two tables: (1) Vertex Table, and, (2) Edge Table.
- The vertex table consists of three-dimensional coordinate values for each vertex with reference to the origin.
- Edge table specifies the start and end vertices for each edge.

Vertex	X	Y	Z
1	1	1	1
2	1	-1	1
3	-1	-1	1
4	-1	1	1
5	1	1	-1
6	1	-1	-1
7	-1	-1	-1
8	-1	1	-1

Edge	Start Vertex	End Vertex
1	1	2
2	2	3
3	3	4
4	4	1
5	5	6
6	6	7
7	7	8
8	8	5
9	1	5
10	2	6
11	3	7
12	4	8

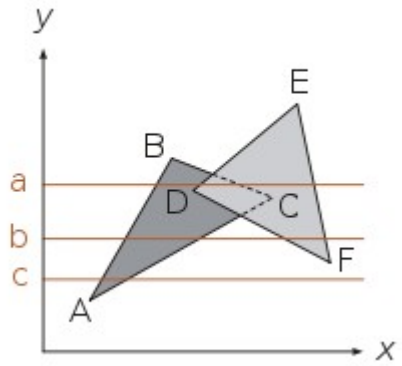
- Sample rendering of a wire-frame cube, icosahedron, and approximate sphere
- A naive interpretation could create a wire-frame representation by simply drawing straight lines between the screen coordinates of the appropriate vertices using the edge list.



Scanline rendering

- **Scanline rendering** (also **scan line rendering** and **scan-line rendering**) is an algorithm for visible surface determination, in 3D computer graphics, that works on a **row-by-row basis** rather than a polygon-by-polygon or pixel-by-pixel basis.
- All of the polygons to be rendered are first sorted by the top y coordinate at which they first appear, then each row or scan line of the image is computed using the intersection of a scanline with the polygons on the front of the sorted list, while the sorted list is updated to discard no-longer-visible polygons as the active scan line is advanced down the picture.
- The main advantage of this method is that sorting vertices along the normal of the scanning plane reduces the number of comparisons between edges.

- Another advantage is that it is not necessary to translate the coordinates of all vertices from the main memory into the working memory—only vertices defining edges that intersect the current scan line need to be in active memory, and each vertex is read in only once.
- The main memory is often very slow compared to the link between the central processing unit and cache memory, and thus avoiding re-accessing vertices in main memory can provide a substantial speedup.
- This kind of algorithm can be easily integrated with many other graphics techniques, such as the Phong reflection model or the Z-buffer algorithm.



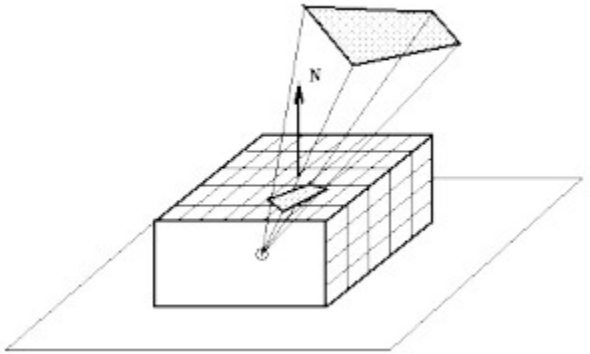
Radiosity (computer graphics)

- Radiosity is an application of the finite element method to solving the rendering equation for scenes with surfaces that reflect light diffusely.
- Radiosity is a global illumination algorithm in the sense that the illumination arriving on a surface comes not just directly from the light sources, but also from other surfaces reflecting light.
- Radiosity is viewpoint independent, which increases the calculations involved, but makes them useful for all viewpoints.
- Radiosity algorithm overview: The surfaces of the scene to be rendered are each divided up into one or more smaller surfaces (patches).

- A view factor (also known as form factor) is computed for each pair of patches; it is a coefficient describing how well the patches can see each other.
- Patches that are far away from each other, or oriented at oblique angles relative to one another, will have smaller view factors.
- If other patches are in the way, the view factor will be reduced or zero, depending on whether the occlusion is partial or total.

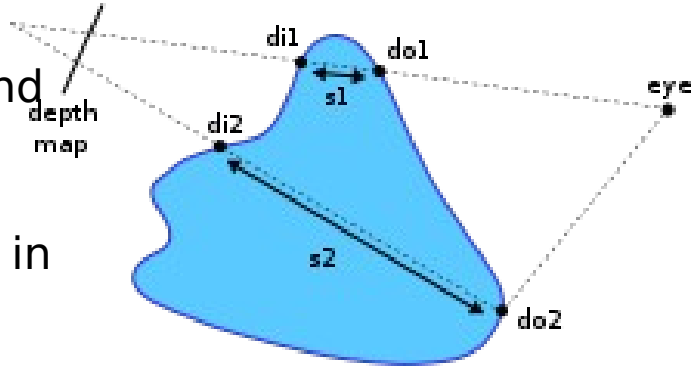
Rendering of a patch onto the surface of the hemicube

- You can think of this as ”Pixelating a Patch”
- As patches are rendered onto the surface of the hemicube, they will occupy hemicube pixels.
- The farther away a surface is, the fewer pixels it will occupy.
- The greater the angle of relative orientation, the fewer pixels it will occupy.
- Finally, the greater the angle, the fewer pixels it will occupy.
- Using a z-buffer, we can let some patches partially (or fully) occlude other patches, causing them to occupy even fewer pixels (or none) which gives us visibility information.



Subsurface scattering

- Subsurface scattering (SSS), also known as subsurface light transport (SSLT), is a mechanism of light transport in which light that penetrates the surface of a translucent object is scattered by interacting with the material and exits the surface at a different point.
- Depth Map based SSS: One method of estimating this distance is to use depth maps, in a manner similar to shadow mapping.
- The scene is rendered from the light's point of view into a depth map, so that the distance to the nearest surface is stored.



Wrap lighting

- One simple trick that approximates scattering is *wrap lighting*.
- Normally, diffuse (Lambert) lighting contributes zero light when the surface normal is perpendicular to the light direction.
- Wrap lighting modifies the diffuse function so that the lighting wraps around the object beyond the point where it would normally become dark.
- This reduces the contrast of the diffuse lighting, which decreases the amount of ambient and fill lighting that is required.
- Wrap lighting is a crude approximation to the Oren-Nayar lighting model, which attempts to more accurately simulate rough matte surfaces (Nayar and Oren 1995).
- The graph in Figure illustrate how to change the diffuse lighting function to include the wrap effect.
- The value wrap is a floating-point number between 0 and 1 that controls how far the lighting will wrap around the object.

- To compute this efficiently in a fragment program, the function can be encoded in a texture, which is indexed by the dot product between the light vector and the normal.
- This texture can also be created to include a slight color shift toward red when the lighting approaches zero.
- This is a cheap way to simulate scattering for skin shaders.

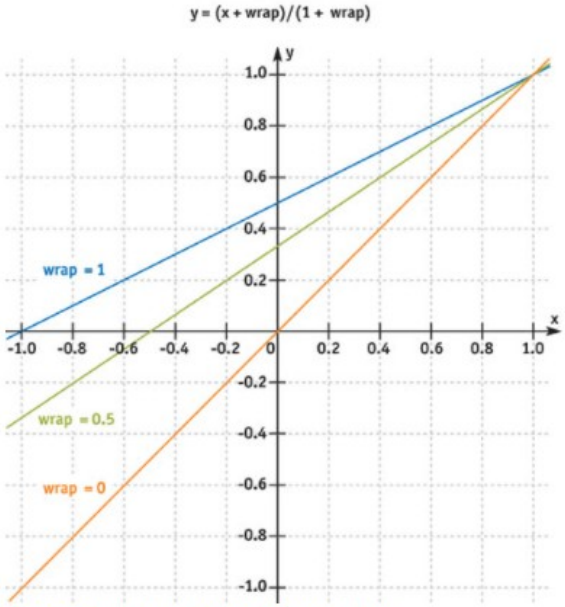
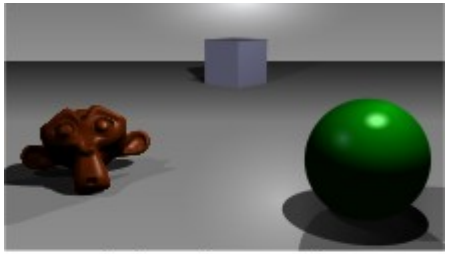


Figure 16-1 Graph of the Wrap Lighting Function

```
float diffuse = max(0, dot(L, N)); float wrap_diffuse = max(0, (dot(L, N) + wrap) / (1 + wrap));
```

Shadow mapping

- Shadow mapping or shadowing projection is a process by which shadows are added to 3D computer graphics.
- Shadows are created by testing whether a pixel is visible from the light source, by comparing the pixel to a z-buffer or depth image of the light source's view, stored in the form of a texture.
- A depth buffer, also known as a z-buffer, is a type of data buffer used in computer graphics used to represent depth information of objects in 3D space from a particular perspective.



A simple three-dimensional scene

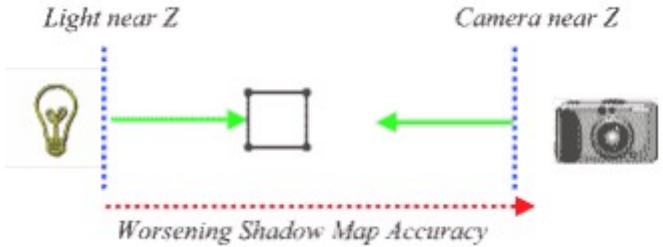


Z-buffer representation

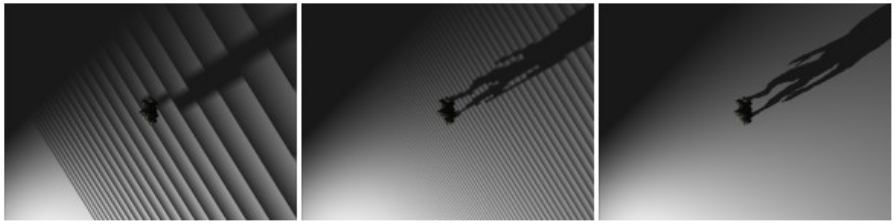
- Shadow maps effectively simulate the shadows cast upon a surface when there is an object between a light source and the surface - these objects are known as occluders.
- Although not a new technique, it is only relatively recently that graphics hardware has had the processing and memory capacity to utilise shadow maps effectively.
- Shadow mapping is a two pass process, meaning that the geometry that makes up the world is rendered twice.
- In the first pass, the world is rendered from the light’s point of view, as if it were a camera, with the depth buffer of this pass saved.

- In the second pass, the world geometry is drawn from the camera’s point of view, using a shader that has the first pass depth texture, and a transformation matrix that will transform vertices into the light’s view space as variables - just as if the scene were being rendered from the light’s point of view.
- By comparing the resulting transformed z value, and the depth value in the depth texture, it can be worked out whether a vertex is ’behind’ or ’in front’ of the depth values - if it is behind, then it is behind whatever object wrote to the depth buffer, and so has been occluded from the light.

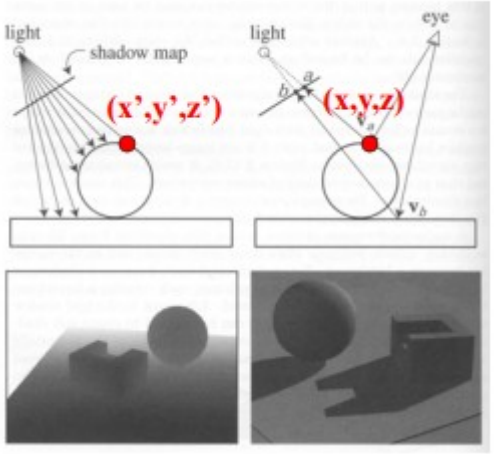
- This results in aliasing, noticeable blockiness around the edges of the shadows.
- What’s worse is that something that’s in the distance from the light’s point of view, and therefore receiving few depth values, may be close from the camera’s point of view - this results in a case of duelling frusta, where the closer to being parallel the light and camera frusta are, the fewer depth values there are where it counts, up close:



- The easiest solution to this is to increase the resolution of the shadow map, so that even far away surfaces have more shadow map resolution covering them in top figure.
- This results in ’sharper’ shadows that more accurately represent the object that is casting the shadows, at the expense of taking up more space.
- The bottom figure depicts the 3D shadow map. It has the light focus on left and camera focus on right with a shadow map.



A scene shadowed using shadowmaps of a resolution of 32, 128, and 512 texels square, respectively



High-dynamic-range rendering

- High-dynamic-range rendering (HDRR or HDR rendering), also known as high-dynamic-range lighting, is the rendering of computer graphics scenes by using lighting calculations done in high dynamic range (HDR).
- This allows preservation of details that may be lost due to limiting contrast ratios.
- Video games and computer-generated movies and special effects benefit from this as it creates more realistic scenes than with the more simplistic lighting models used.
- Graphics processor company Nvidia summarizes the motivation for HDR in three points: bright things can be really bright, dark things can be really dark, and details can be seen in both.

Development of HDRR through DirectX

- Complex shader effects began their days with the release of Shader Model 1.0 with DirectX 8. Shader Model 1.0 illuminated 3D worlds with what is called standard lighting. Standard lighting, however, had two problems:

1. Lighting precision was confined to 8 bit integers, which limited the contrast ratio to 256:1. Using the HVS color model, the value (V), or brightness of a color has a range of 0 – 255. This means the brightest white (a value of 255) is only 255 levels brighter than the darkest shade above pure black (i.e.: value of 0).
2. Lighting calculations were integer based, which didn't offer as much accuracy because the real world is not confined to whole numbers.

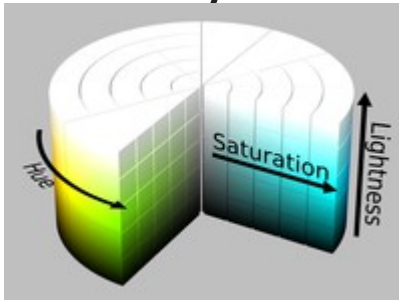
Development of HDRR through OpenGL

- It is possible to develop HDRR through GLSL shader starting from OpenGL 1.4 onwards. OpenGL Shading Language (GLSL) is a high-level shading language with a syntax based on the C programming language.

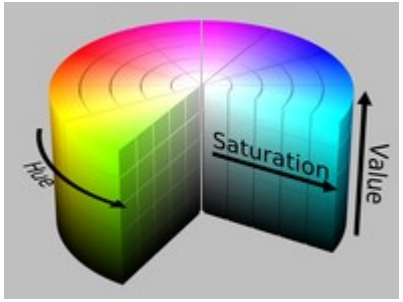
- HSL (for hue, saturation, lightness) and HSV (for hue, saturation, value; also known as HSB, for hue, saturation, brightness) are alternative representations of the RGB color model, designed to more closely align with the way human vision perceives color-making attributes.
- In these models, colors of each hue are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top.
- The HSL representation models the way different paints mix together to create colour in the real world, with the lightness dimension resembling the varying amounts of black or white paint in the mixture (e.g. to create "light red", a red pigment can be mixed with white paint; this white paint corresponds to a high "lightness" value in the HSL representation).

- Fully saturated colors are placed around a circle at a lightness value of ½, with a lightness value of 0 or 1 corresponding to fully black or white, respectively.
- Meanwhile, the HSV representation models how colors appear under light.
- The difference between HSL and HSV is that a color with maximum lightness in HSL is pure white, but a color with maximum value/brightness in HSV is analogous to shining a white light on a colored object (e.g. shining a bright white light on a red object causes the object to still appear red, just brighter and more intense, while shining a dim light on a red object causes the object to appear darker and less bright).

- HSL cylinder



- HSV cylinder



- The advantages of HDR compared to LDR from the application point of view.
- The quality of the LDR image have been reduced on purpose to illustrate potential differences between the HDR and LDR visual contents as seen on an HDR display.
- The given numbers serve as an example and are not meant to be a precise reference.



Standard (Low) Dynamic Range



High Dynamic Range

up to 500 cd/m²	peak brightness	2 000-10 000 cd/m²
50 dB	camera dynamic range	120 dB
1:1 000	display contrast	1:1 000 000
from 8 to 16 bit	quantization	floating point or variable
display-referred	image representation	scene-referred
display-limited	fidelity	as good as the eye can see

- A complete pipeline for HDR image and video processing from acquisition, through compression and quality evaluation, to display is presented.
- At the first stage digital images are acquired either with cameras or computer rendering methods.
- At the second stage, digital content is efficiently compressed and encoded either for storage or transmission purposes.
- Finally, digital video or images are displayed on display devices.
- Tone mapping is required to accommodate HDR content to LDR devices, and conversely LDR content upgrading (the so-called inverse tone mapping) is necessary for displaying on HDR devices.

- Apart from considering technical capabilities of display devices, the viewing conditions such as ambient lighting and amount of light reflected by the display play an important role for proper determination of tone mapping parameters.
- Quality metrics are employed to verify algorithms at all stages of the pipeline.

