Vitrual Tour

Architectural Model

Keshav Prasad Singh
107CS023, B.Tech, 2011

Rohit Man Amatya
107CS056, B.Tech, 2011

In Supervision of
Prof. Pankaj Kumar Sa

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela–769 008, Orissa, India

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. `www.nitrkl.ac.in`

Dr. Pankaj K Sa
Assistant Professor

May 09, 2011

Certificate

This is to certify that the work in the project entitled *Virtual Tour* by *Keshav Prasad Singh* and *Rohit Man Amatya* is a record of an original work carried out by them under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering.* Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Pankaj K Sa

Acknowledgements

My gratitude and indebtedness to the entire people who help me to shape up this project. I am grateful towards my supervisor **Prof. Pankaj Kumar Sa** for his genius thinking, without his flexibleness and inception this project would never have been made possible. I am thankful my friends **Abdulla**, **Dipesh**, **Chandan**, **Gary Mo**, **Toketo** and **Kamlesh** for there awesome ideas and support.

I finally thank to the **Andrew Price** *(blenderguru.com)* for his awesome free blender tutorial and video which inspired me to start using blender and **Ira Krakow** for his Tips and Tricks youtube videos at the time when I got into trouble.

Rohit Man Amatya
rhohitman@gmail.com

Abstract

Interactive 3D Visualization of Architectural models might be the best way to get some idea about an Architecture Plan. Photo-realistic visualization often attracts the investors and customers for whom the architectural blueprints are obscure.

Architectural Visualization is considered to have a bright future ahead of it as more and more architects and real estate developers are using this technology.

Virtual Walk–through can give not only ideas about your building but its interiors and design too. The Architectural Virtual Environment also most widely used in Gaming and Entertainment Industry in creating a complex movie scenes or a game environment.

Contents

Chapter 1

Introduction

Virtual Tour computer generated environment, (mostly related with architectural model), created along with landscape and sometimes moving people and vehicles. Architectural models are drafted with exact proportion and scale as the real ones, and are often added with real life textures and material to mimic reality.

Entertainment Industry, which spends millions in film production high uses CG techniques. Creating a non-existing location set or to pull off the incredible visual stunt requires lots of manpower and resources. Rich visual quality, camera movement, etc. are some of the prevailing features making CG a much preferable option.

Photo-realistic visualization often attracts the investors and customers for whom the architectural blueprints are obscure, giving designer leverage in conveying his project ideas. Photo-realistic visualization and animation actually play major role in real estate sales and construction.

Architectural models is the heart of the Gaming Industry. 3D gaming Environment are created using architectural models where the game play takes palace. Gaming has much to deal with the Real-time Processing than the accurate simulation.

1.1 Background

Making the Architectural model used to be the made with boxes, cardboard using glue and paint. [1] Similar to like making the building model project for primary-school. A knife or scissors to cut doors and windows holes in boxes, landscape with clay or sand and wood for furnitures. Using natural materials to make it look realistic. Early '60 to '70 Si-Fi movies used these kinds of models to depict frictional location or the future city.

The first use of a Virtual Tour and the derivation of the name was in 1994 as a

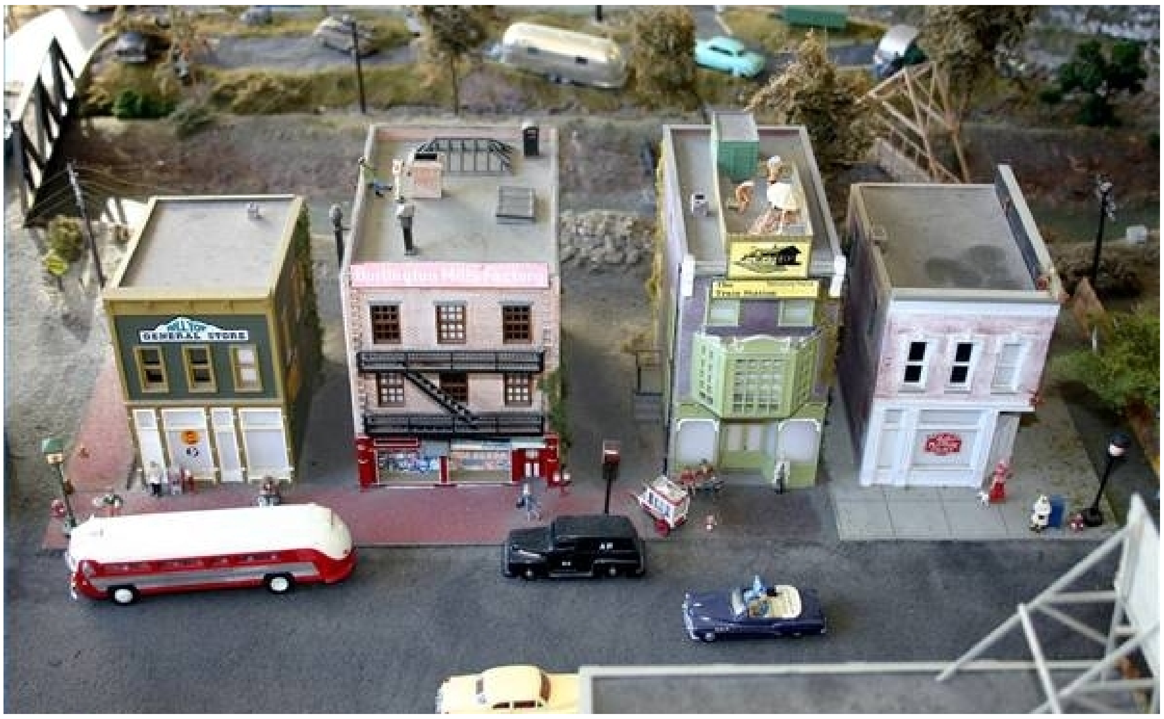


Fig. 1.1: Physical Architectural Model [2]

museum visitor interpretation, providing a 'walk-through' of a 3D reconstruction of Dudley Castle [3] in England designed by British engineer Colin Johnson. The system featured in a conference held by the British Museum in November 1994 and in the subsequent technical paper.

1.2 Overview

3D Model consists of 3D object, created using collection of points in 3D space connected by several entities such as lines, curves, triangles, and surfaces. Its displayed as a 2D image through a process called 3D rendering.

A 3D model are created manually or using algorithm and scanning which is not in the scope of our project. In manual modeling process geometric data for 3D model is similar prepared as to Architectural drawing but in 3D space. Pre-image (wireframe) are then added with textures, lights and relative positioned to other objects to create completed scene. These scene are used for movie animations.

The Interactive Models are made with the same scene in place but adding the navigation and physical phenomena. These Interactive models have much do with speed rendering as the animation would take much much time to get rendered.

1.3 Implementation

In this project we aim to built the Virtual Tour of Department of Computer Science & Engineering. Here we present how we built project using open source tools, with main focused on Blender. Our primary objectives are to create:

- Animated Visualization (movie)
- Virtual Walk-through (Realtime 3D Render)

List of software and tools were used in building this project are given in Table 1.1.

Processor	AMD Turion™ Dual-Core Mobile M500
3D Tool	Blender 2.5x
Renderer	Blender Render, YafaRay, LuxRender
Satellite Image Source	Google™ Earth
Programing & Scripting	C/C++, Python 3.1.2+
Other Tools	GIMP, Inkscape

Table 1.1: Software and Hardware used during Project

1.3.1 Project Pipeline

Most of the Work has be done using Blender. We also have tried compare with some external render engines. And finally worked on the Interactive Model.

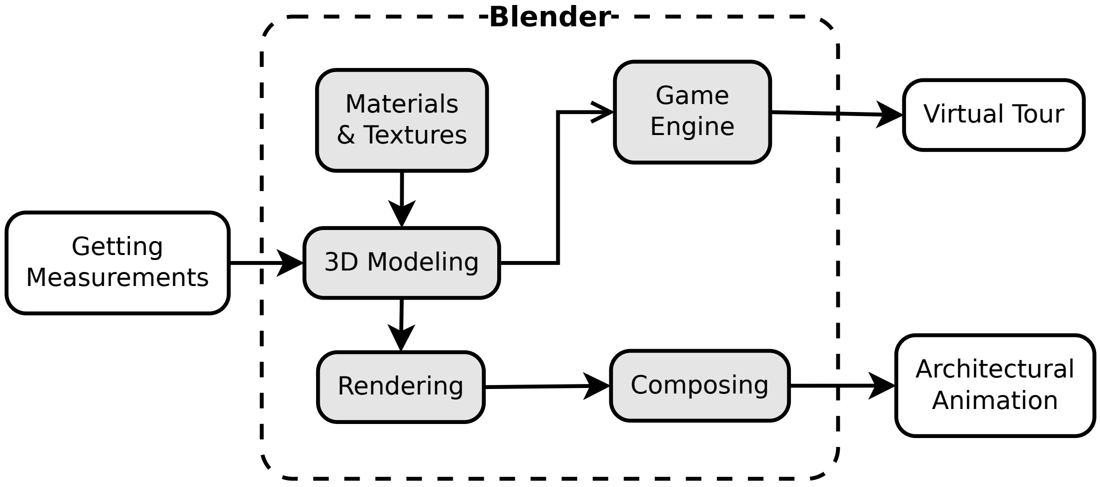


Fig. 1.2: Project Flow

Chapter 2

3D Modeling

The 3D model is completely described in 3D space and can be viewed from any angle. Building the 3D modeling is the complex process which need the help specialized tool like 3D modeling software. Although you can draw complex and interesting models, they're all constructed from a small number of primitive graphical items. In 3D art almost all models are built from triangles [1]. It may not seem so at first, because many modeling tools let you work with quadrangles, curves, bevels, mathematical surfaces etc. But in the end, it's all triangles.

2.1 3D Representation

Graphics scenes can contain various kinds of objects: rocks, water, tree, marble, steel, glass etc. So, there is no one method that we can use to describe objects that will include all characteristics of these different materials. Producing a realistic displays of scenes, requires accurate representations that accurately model object characteristics. [4]

Polygon and quadric surfaces provide precise descriptions for simple Euclidean objects such as polyhedrons and ellipsoids; spline surfaces end construction techniques are useful for designing aircaft wings, gears, and other engineering structures with curved surfaces; procedural methods, such as fractal constructions and particle systems, allow us to give accurate representations for clouds, clumps of grass, and other natural objects; physically based modeling methods using systems of interacting forces can be used to describe the nonrigid behavior of a piece of cloth; octree encodings are used to represent internal features of objects, such as those obtained from medical C images; and isosurface displays, volume renderings, and other visualization techniques are applied to 3D discrete data sets to obtain visual representations of the data. [4]

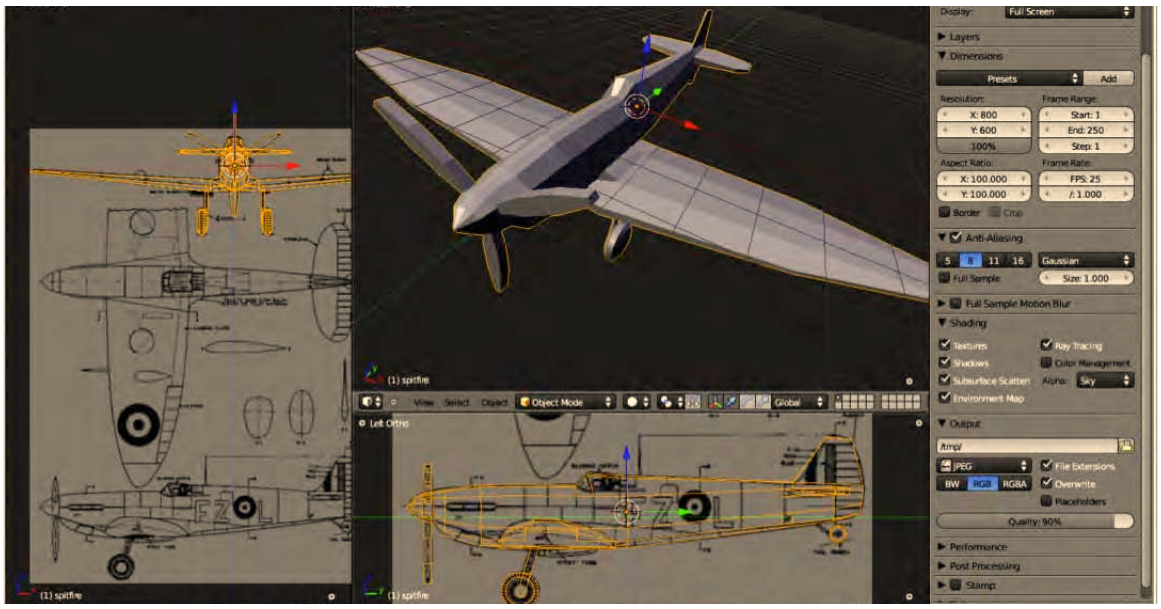
Representation schemes for solid objects are often divided into two broad categories, although not all representations fall neatly into one or the other of these two categories *Boundary* and *Space-partitioning* representations.

2.1.1 Boundary representations

Boundary representation describe a 3D object as a set of surfaces that separate the object interior from the environment. Typical examples of boundary representations are polygon facets and spline patches.

Wireframe: Its a representation is one of its kind where is the outlines of an object represented by vertices and edges only. This is an advantage for simple objects or while creating initial modeling.

Surface: Surface models represent the object as an ordered set of surfaces in 3D space. Surface models are mainly used for the generation of models, whose surfaces consist of analytical not easily describable faces having different curvatures in different directions. This is often the case for models of automobiles, ships or airplanes.



Display: Full Screen
Resolution: X: 800, Y: 600, 100%
Frame Range: Start: 1, End: 250, Step: 1
Aspect Ratio: X: 100.000, Y: 100.000
Frame Rate: FPS: 25, /: 1.000
Border, Crop
Anti-Aliasing: 5 [8] 11 16, Gaussian, Full Sample, Size: 1.000
Full Sample Motion Blur
Shading: ✓ Textures, ✓ Ray Tracing, ✓ Shadows, Color Management, ✓ Subsurface Scatter, Alpha: Sky, ✓ Environment Map
Output: /tmp/
JPEG, BW [RGB] RGBA, ✓ File Extensions, ✓ Overwrite, Placeholders
Quality: 90%

Fig. 2.1: Modeling a Plane [5]

2.1.2 Space-partitioning representations

Space-partitioning representations are used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids. A common space-partitioning description for a three-dimensional object is an octree representation. [4]

2.2 Polygon Surfaces

Its most commonly used boundary representation for a 3D graphics object is a set of surface polygons that enclose the object interior. Many graphics systems store all object descriptions as sets of surface polygons. This simplifies and speeds up the surface rendering and display of objects, since all surfaces are described with linear equations. Objects to be described with other schemes, such as spline surfaces, that are then converted to polygonal representations for processing.

A polygon representation for a polyhedron precisely defines the surface features of the object. But for other objects, surfaces are tesselated (or tiled) to produce the polygon-mesh approximation. Such representations are common in design and solid-modeling applications, since the wireframe outline can be displayed quickly to give a general indication of the surface structure. Realistic renderings are produced by interpolating shading patterns across the polygon surfaces to eliminate or reduce the presence of polygon edge boundaries. And the polygon-mesh approximation to a curved surface can be improved by dividing the surface into smaller polygon facets.

2.3 Modeling Processes

There are mainly five ways of modeling:

Polygonal modeling: Also Known as the Mess modeling, is the technique where vertices are placed 3D space, connected line to form a polygonal mesh. The vast majority of 3D models today are built as textured polygonal models.

NURBS modeling: NURBS Surfaces are defined by spline curves which are truly smooth surfaces, not approximations using small flat surfaces.

Patches modeling: Depend on curved lines to define the visible surface. Patches fall somewhere between NURBS and polygons in terms of flexibility and ease of use.

Primitives modeling: Using geometric primitives (i.e. sphere, cubes) building blocks for more complex models converted them meshes for further operations and rendering.

Sculpt modeling: Still fairly new method of modeling 3D. Sculpting allows artistic exploration as the model will have a new topology created over it.

2.4 Measurement

Re-constructing 3D Model of existing structure requires the proper measurement. Following section we will discuss about few basic technique to get approximate dimensions.

2.4.1 Area

Big round earth's surfaces can be consider as a flat plain for short distances. **Haversine formula** [6] gives such approximate distance measure using longitudes and latitudes between points. Let the two points in spherical coordinates be [lon_1, lat_1] and [lon_2, lat_2] then distance will be:

$$\boxed{d = R \times c}$$

$$\begin{aligned}
&\triangle lon = lon_2 - lon_1 \qquad \triangle lat = lat_2 - lat_1 \\
&a = \sin(\tfrac{\triangle lat}{2})^2 + cos(lat_1) \times cos(lat_2) \times sin(\tfrac{\triangle lon}{2})^2 \\
&c = 2 \times atan2(\sqrt{a}, \sqrt{(1-a)})
\end{aligned}$$

where,

R is radius of earth (mean radius *6,371 km*)

c is great circle distance in *radians.*

d is great circle distance d (same units as R).

Limitation: But, even though the circumference of the Earth is about 40,000 km, flat-Earth formulas for calculating the distance between two points start showing noticeable errors when the distance is more than about 20 km.

2.4.2 Height

Reference object: Including something in the photo of known height (*eg. a meter stick or a person*) close to the building, photo scale can be easily determined. Using this scale we can calculate any height in the photograph by multiplying $1/scale$ with measured length in the photo. The vertical distortion error can arise in this method, which can be minimize by shooting photo from adequate distance.

Basic trigonometry: With a few simple measurements, it's possible to estimate heights with some accuracy which is $height = (\tan\theta \times distance) + eyeheight$.

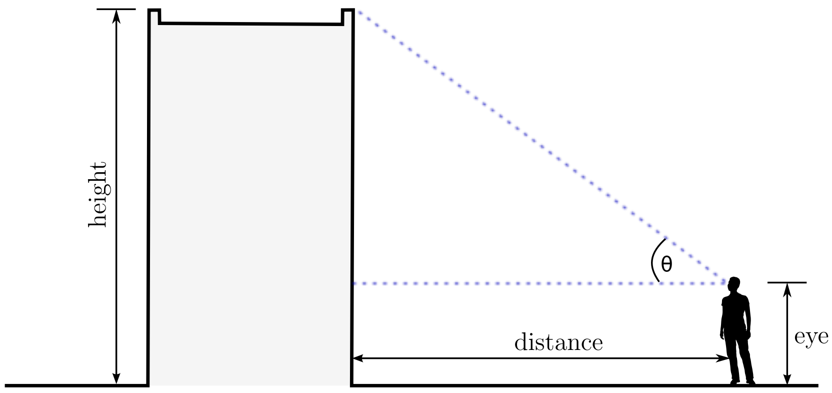


Fig. 2.2: Basic trigonometry

Repeated units: Buildings are constructed modular materials *i.e bricks, blocks* Overall height can be approximated using the no of units multiplied height of a single unit.

2.5 Implementation

The 3D models are generally made before the construction takes place. In our project we are re-constructing the 3D model of the existing building which is our Department building. We start with the basic measurement survey before hand start building model.

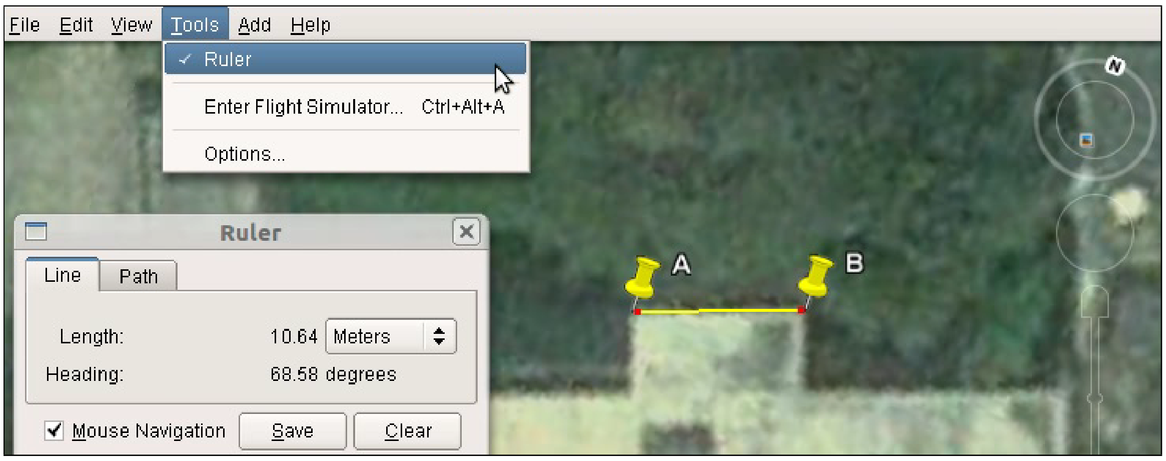
2.5.1 Comparing Measurements

After failed attempts to create the model with photo reference and approximate map overlay. We took the different step to approximate the measurement starting with comparing and calculating errors. We took choose test points on site, took down longitude and latitude of those point using Google Earth.

Example: Sample Test Case

Point A: $22°15'06.73''N, 84°54'2.26''E$ **Point B**: $22°15'06.86''N, 84°54'2.62''E$

Measurement	Value (in m)	Error
Actual	10.87	0%
Haversine	11.05	1.65%
Google Earth Ruler	10.64	2.12%


File Edit View Tools Add Help
Ruler
Enter Flight Simulator... Ctrl+Alt+A
Options...
Ruler
Line
Path
Mouse Navigation
Save
Clear
A
B
N

Length: 10.64 Meters
Heading: 68.58 degrees

Fig. 2.3: Google Earth Ruler

Conclusion

We found that a some small error, due to the fact that the Earth is very slightly ellipsoidal, using spherical model gives errors typically up to 0.3 %.

2.5.2 Making Model

Overlaying model onto the reference image acquired we start tracing the boarders of the building and scale the model to proportion to measurements taken during survey.

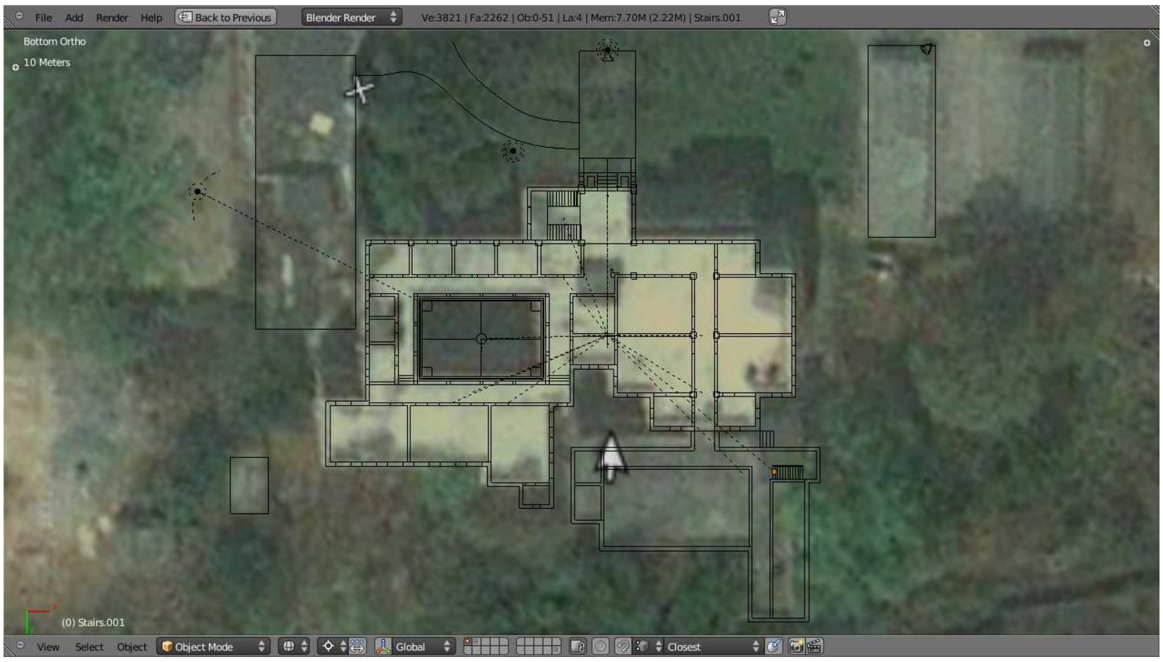


Fig. 2.4: Outline of Building using Reference Background

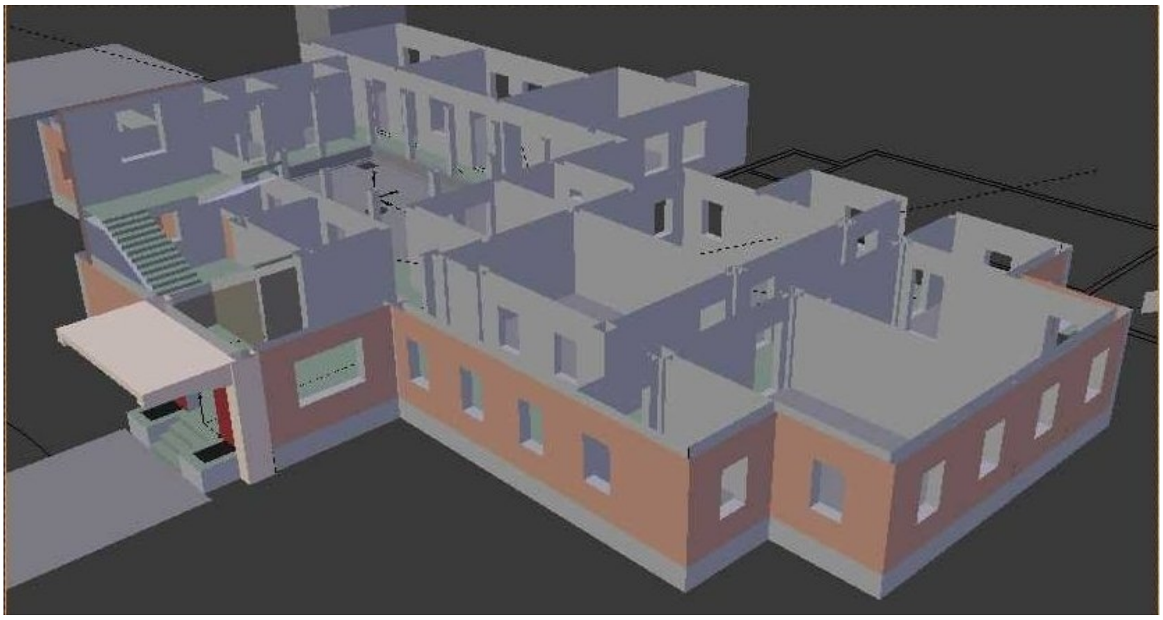


Fig. 2.5: Ground floor Model

Chapter 3

Rendering

Rendering is the process of generate its images 2D images using 3D model. Rendering engines are fed with the view point and 3D scene information like geometry, lighting & shading, textures etc which processed to give a digital image output. Rendering comprises of performing complex mathematical calculations which requires intensive processing, therefore rendering engines are outlined with graphics pipeline. [7]

In 3D graphics rendering are done slowly, as in pre-rendering to giver real-time effect to the displaying scene. Real-time rendering is used in 3D video games relying on the use of graphics cards bearing 3D hardware accelerators. While Pre-rendering, a computationally intensive process, is used in movie creation.

Due to these problems, different modeling techniques for the transportation of light have been devised. They are categorized into four different families **Rasterization**, **Ray Casting**, **Ray Tracing** and **Radiosity**.

Radiosity isn't generally used for rendering, but it helps in calculation the amount of light leaving the source of light and illuminating the surface. Excluding radiosity, the other three techniques are often used for the surface rendering. Cost efficient good results are achieved by softwares which use a combination of one or more rendering techniques.

3.1 Rasterization

Rasterization is most fastest rendering technique in comparison to other rendering techniques. Its the process of computing the mapping from scene geometry to pixels and does not prescribe a particular way to compute the color of those pixels. Shading, may be based on physical light transport, or artistic intent.

Rasterization is most popular technique for producing real-time 3D computer graphics. Real-time applications need to respond immediately to user input, and generally need to produce frame rates of at least 25 frames per second to achieve smooth animation.

3.2 Ray Casting

The geometry which has been modeled in ray casting is parsed line by line, pixel by pixel, from the point of view outward, as if the casting rays coming out from the point of view. The point where an object is intersected, value of the color value at the very point can be evaluated using several possible methods. The simplest for that, the value of color at that point of intersection becomes the pixel value. We can determine the color from a texture-map. The more challenging and sophisticated method would be modifying the value of the color by an factor for illumination, but we don't have to calculate the relationship to a simulated light source. For reducing artifacts, a certain number of rays may be averaged in slightly different directions.

The rough simulations may be additionally employed for optical properties: its a very simple calculation in which the ray from the object to the point of view is made. one more calculation is made for the angle of incidence of light rays from the very light source(s), and from these as well as the specified intensities of the light sources, then the value of pixel is calculated. One more simulation is illumination plotted from an algorithm called radiosity algorithm, or cb be combination of these two.

Primary use of Raycasting is for realtime simulations, it is same as those used in 3D cartoon animation and computer games, in which details are not important, or where the more efficient way is to manually fake the details in order to obtain a better performance in the very computational stage. This is mostly the case when a huge number of frames are needed to be animated. The result is the surfaces which have a characteristic 'flat' appearance without any use of additional tricks, as if objects in the scene were all painted with matte finish.

3.3 Ray Tracing

Ray tracing a technique which used to generate an image by tracing the path of light. This technique is very proficient for producing a visual realism of high degree reality, the reality is usually higher than that of rendering methods done by typical scanline, but the computational cost is quite high. This helps in making ray tracing best suited for applications in which the image can be rendered slowly and ahead of time, its just same as in still images and film and in the television special effects, but it real time applications like video games where speed is critical it is poorly suited for these kind of applications. Capable of simulating optical effects, such as scattering, reflection and refraction, and chromatic aberration; ray tracing can simulate the natural flow of light, reded as particles.

Often, ray tracing methods are utilized to approximate the stimulated solution or behavior to the rendering equation by applying Monte Carlo methods to it. Bidirectional Path Tracing, Path Tracing, or Metropolis light transport are the mostly used methods, also Whitted Style Ray Tracing, or hybrids methods are used which are semi realistic. While light propagate on straight lines in most implementations, still relativistic spacetime effects stimulation applications exist.

For the final quality rendered product of a ray traced work, each pixels are generally shot with multiple rays, and traced not just from the interaction of the first object, but rather, through a series of sequential bounces, utilizing the basic principles of optics as "angle of incidence equals angle of reflection" and other advanced laws dealing with surface roughness and refraction.

Once the ray either encounters a light source, or more probably once a set limiting number of bounces has been evaluated, then the surface illumination at that final point is evaluated using techniques described above, and the changes along the way through the various bounces evaluated to estimate a value observed at the point of view. This is all repeated for each sample, for each pixel.

In distribution ray tracing, at each point of intersection, multiple rays may be spawned. In path tracing, however, only a single ray or none is fired at each intersection, utilizing the statistical nature of Monte Carlo experiments.

3.4 Radiosity

Radiosity is the method to stimulate ways in which surfaces are used to illuminate using reflection from other surfaces which are directly illuminated. This some what produces a realistic form of shading and thereby capture a better ambience of an indoor scene. Shadows caressing the corners of rooms is a classic example.

The simulation optical basis is that a given area is illuminated by a scattered spectrum of light which are reflected from a given surface whose source light can be from any point.

Complexity in the stimulation technique varies. A rough radiosity estimate can be made in renderings i.e. a simple illumination of a scene with factors such the ambiance factor. Realism is exerted when advanced radiosity estimation along with high quality algorithm for tracing ray like those for the indoor scenes.

The advanced part of radiosity is the recursive bouncing of light rays forth and back between surfaces in the model, until a limited recursion reached. In this process colouring of the surfaces are influenced, one influence the colouring of there neighbour and vice versa. The results for illuminated model(including empty spaces) can be used for the additional inputs as carrying out calculations in a ray-tracing model or ray-casting .

Iterative/recursive technique makes the emulation process of complex objects slow. Before rapid radiosity calculation came to standardized, some graphic artists were to a technique of referring loose as false radiosity by simply darkening areas in the texturing scene corresponding to joints, corners and recesses, and implication by self-illumination or diffuse mapping for use in scanline rendering. Even presently, advanced radiosity calculations are reserved for calculating the ambiance of the room, from the light reflecting off ceiling, walls and floor without considering the facts that contribute like the complex objects. The complex objects or otherwise be be replaced with similar simpler objects for radiosity calculation .

Radiosity calculations, inspite of being independent of viewpoints which tends to increase involved computations ,makes all viewpoints useful. Reuse of same radiosity data are implemented in a number of frames, making radiosity effective to improve on the flatness condition of ray casting, without impingement the overall rendering time-per-frame.

This makes radiosity a prime component in leading the real-time rendering, and has been used in the full length creation of well-known animated 3D-cartoon films.

3.5 Shading

It is the phenomenon of colouring and setting up varying intensities of brightness to a surface in accordance with the amount of lighting.

3.5.1 Ambient Occlusion

Ambient occlusion is a shading method used in 3D computer graphics which helps add realism to local reflection model by taking into account attenuation of light due to occlusion. Ambient occlusion attempts to approximate the way light radiates in real life, especially off what are normally considered non-reflective surfaces.

Unlike local method Phong Shading, amient occlusion is a global method, meaning the illumination at each point is function of thoer geometry in the scene. However, if is a very crude approximation to full global illumination. The soft appearance achived by ambient occlusion alone is similar to the way an object appear on in over cast day.

3.5.2 Phong Shading

Phong shading are techniques used in 3D computer graphics for shading purposes to produce certain degree of realism.Here three elements come to play- specular, diffuse and ambient lighting for each surface point considered of a model.it encloses the method of interpolating surface normals into rasterized polygons to estimate pixel colours.

This reflection from the surface points of the model may be referred as the Phong lighting, Phong illumination or namely Phong reflection model. In the linguistic context of pixel shaders or circumstances where calculation of lighting comes into play in “shading” techniques, it is called as Phong shading. Phong interpolation refers to the method of interpolation with each surface points which is also referred as per-pixel lighting.

3.6 Subsurf

A subdivision surface populary known as subsurf, is a method of representing a smooth surface via the specification of a coarser piecewise linear polygon mesh. The smooth surface can be calculated from the coarse mesh as the limit of a recursive process of subdividing each polygonal face into smaller faces that better approximate the smooth surface.

3.6.1 Refinement schemes

Subdivision surface refinement schemes can be broadly classified into two categories: interpolating and approximating.

Interpolating schemes are required to match the original position of vertices in the original mesh. Approximating schemes are not; they can and will adjust these positions as needed. In general, approximating schemes have greater smoothness, but editing applications that allow users to set exact surface constraints require an optimization step. This is analogous to spline surfaces and curves, where Bezier splines are required to interpolate certain control points, while B-splines are not.

There is another division in subdivision surface schemes as well, the type of polygon that they operate on. Some function for quadrilaterals(quads), while others operate on triangles.

Scheme	Type	Technique
Catmull–Clark (1978)	Approx.	Generalized bi-cubic uniform B-spline to produce
Doo–Sabin (1978)	Approx.	Extended Chaikin's corner-cutting method for curves to surfaces Analytical expression of bi-quadratic uniform B-spline surface
Loop (1987)	Approx.	quartic box-spline of six direction vectors
Mid-Edge (1999)	Approx.	four-directional box spline
$\sqrt{3}$ (2000)	Approx.	
Butterfly (1990)	Inter.	extended the four-point interpolatory
Kobbelt (1996)	Inter.	Variational subdivision method that tries to overcome uniform subdivision drawbacks

Table 3.1: Subsurf Techniques

3.7 Realtime Rendering

Rendering for interactive media, such as games and simulations, is calculated and displayed in real time, at rates of approximately 20 to 120 frames per second. In real-time rendering, the goal is to show as much information as possible as the eye can process in a 30th of a second (or one frame, in the case of 30 frame-per-second animation). The goal here is primarily speed and not photo-realism. In fact, exploitations can be applied in the way the eye 'perceives' the world, and as a result the final image presented is not necessarily that of the real-world, but one close enough for the human eye to tolerate. Rendering software may simulate such visual effects as lens flares, depth of field or motion blur. These are attempts to simulate visual phenomena resulting from the optical characteristics of cameras and of the human eye. These effects can lend an element of realism to a scene, even if the effect is merely a simulated artifact of a camera. This is the basic method employed in games, interactive worlds and VRML. The rapid increase in computer processing power has allowed a progressively higher degree of realism even for real-time rendering, including techniques such as HDR rendering. Real-time rendering is often polygonal and aided by the computer's GPU.

3.8 Scene

A scene contains more than the 3d object a light source (point light) and virtual camera. A camera determines the viewpoint of the scene where as the without like it impossible see anything i.e. whole rendered output will be pitch black. Table 3.2 list different types of light modes as given in out modeling software.

3.8.1 Virtual Camera

The virtual camera is the main link between the areas of visualization and vision of 3D scenes. It is the element that maps 3D objects in the scene into 2D information in the image, and it is used in a dual manner in visualization and vision processes. [8]

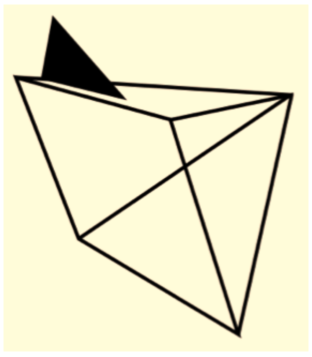


Fig. 3.1: Blender's virtual Camera [5]

The simplest model of a virtual camera is the pinhole camera, which has 7 degrees of freedom representing the parameters **position** (3 degrees of freedom), **orientation** (3 degrees of freedom) and **focal length** (1 degree of freedom).

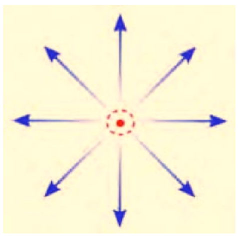
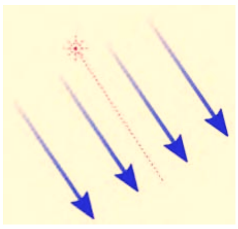
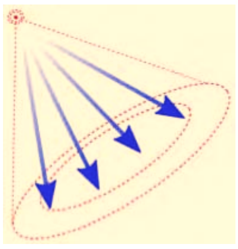
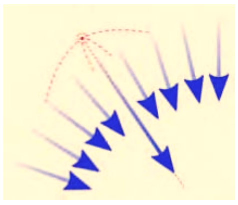
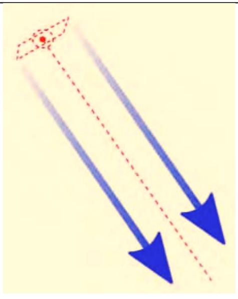
Light Effect	Type	Description
	Point	This light emits in all directions from a single point. With appropriate falloff, it can resemble a candle or a small lightbulb. It is very useful for rim light effects, where parts of an object need to be lit in order to stand out from the background.
	Sun	Otherwise known as a directional light, this is light that floods a scene from a given angle. It gets its name because it is similar to how the sky lights the world: flooding the scene from a given direction, not from a single point. Location does not affect sun lights; it is the rotation that is important. Whichever way a sun light is rotated, the whole scene gets light from that particular angle with parallel light rays.
	Spot	This is similar to a point lamp, but within a restricted V-shape direction. This light works very much like a theater spotlight. It casts a circle on a surface it is aimed at, and has settings to control the softness of the circular edges.
	Hemi	A hemi light produces an ambiance similar to a sun light, except that instead of creating light from a single direction, it acts as though the light is emitted from a sky dome. It is like having a single sun from the dominant direction, accompanied with smaller lights to illuminate the sides of objects in the scene.
	Area	Type An area light is like having a cluster of lights over an area of a specified size. It is useful for creating light emitted from a surface, such as a TV or the back of a fridge.

Table 3.2: Different Types of Light modes in Blender [5]

3.9 Implementation

We Tried testing with the different render option available with in the blender. We gave the try to some of the external render engines like LuxRender and YafaRay which was quite unsuccessful. We made the short video of 360° bird eye view.

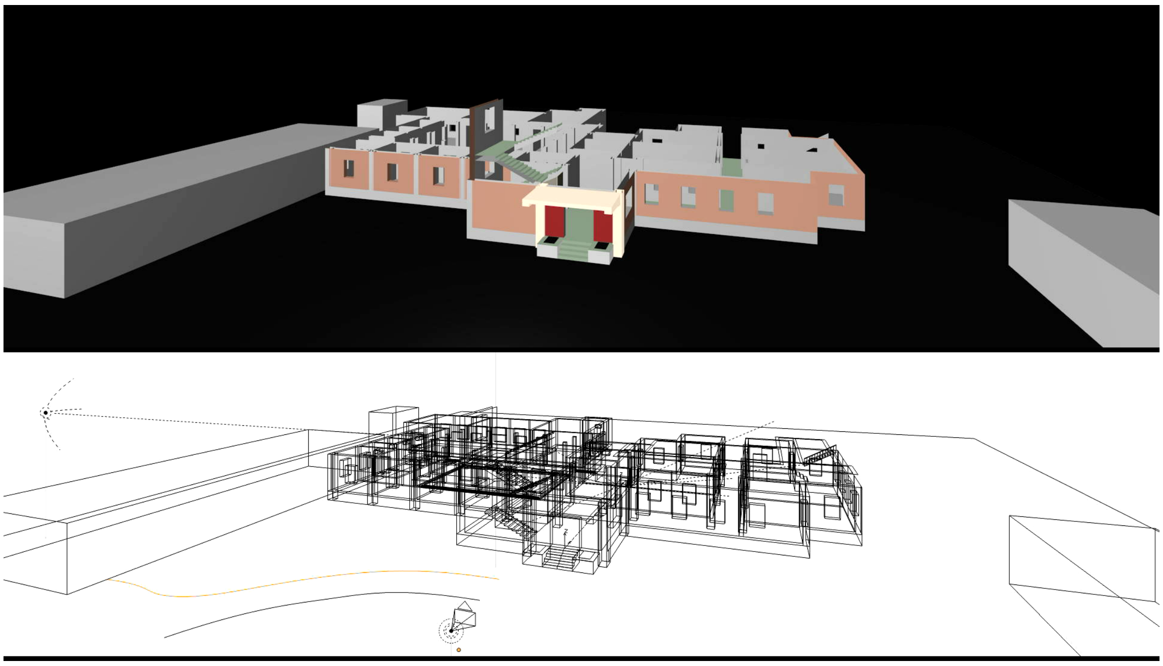


Fig. 3.2: Blender internal Render Video Output

3.9.1 Conclusion

Cameras and lights setup plays major role in visualization. Emphasis on the importance of setting up key lights and fill lights to illuminate a scene for creating renders.

Chapter 4

Texture and Materials

Natural Surfaces can be quite complex in their appearance. Color, specularity, and reflection can organically change across a surface as a result of location, climate interaction, and variations in the natural substance. Learning how to create believable natural surface materials help in development of many other material types. After all, most manufactured objects are created from, or based on, natural materials. [9]

4.1 Texture Mapping

In the visual arts, texture is the perceived surface quality of a work of art. It is an element which is distinguished by its perceived visual and physical properties. Texture mapping is a method of adding texture(a bitmap or raster image) detail or color to the surface of 3D-object.

A texture map is applied to the polygon surfaces. Every polygon is assigned with the texture information either by procedural or explicitly assignment. Sampling Image coordinates are then interpolated across the surface to produce a rich visual result. Mapping is set of parameters that describe how a texture should be applied to an object i.e. scale, offset, rotate, and so on. [9, 10]

The way the resulting pixels on the screen are calculated from the texels (texture pixels) is governed by texture filtering. The fastest method is to use the nearest-neighbor interpolation, but bilinear interpolation or trilinear interpolation between mipmaps are two commonly used alternatives which reduce aliasing or jaggies. In the event of a texture coordinate being outside the texture, it is either clamped or wrapped. [7]

4.1.1 Projection

Texture can be mapped to the surface with the various orientation, as if it was a slide projection. It is achieved computing texture coordinates or coordinate generation.

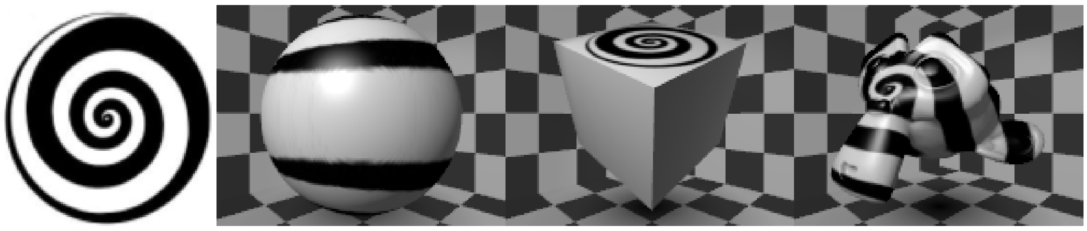

(a) Texture (b) Sphere (c) Cube (d) Non Uniform

Fig. 4.1: Flat Projection on different Surfaces

Four main types of projection that are available in the texture mapping are Flat, Cube, Sphere and Tube which is show in the Fig 4.2.

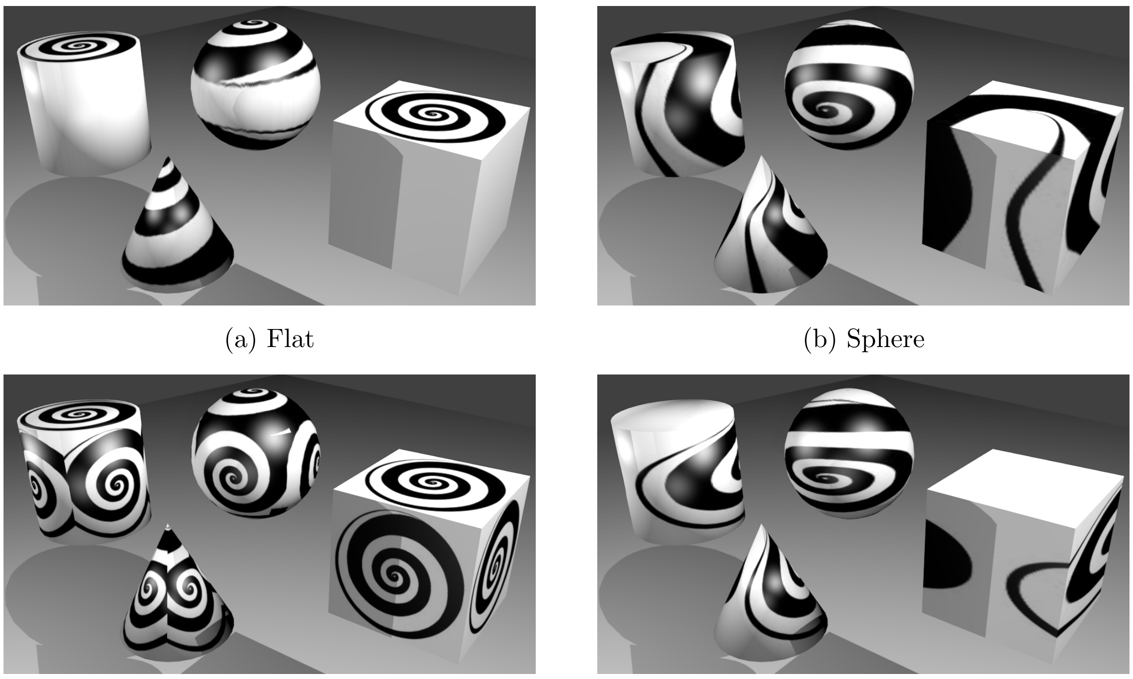

(a) Flat (b) Sphere (c) Cube (d) Tube

Fig. 4.2: Types of Projection

4.1.2 Perspective Correctness

Each vertex of a given polygon are specified by texture coordinates are interpolated using an extended Bresenham's line algorithm which determines plotting of the straight line between two given points in an n-dimensional raster forming a close approximation. If these texture coordinates are linearly interpolated across the screen, the result is affine texture mapping.

Bresenham's gives considerably fast calculation, but when polygons are at the angle to the plane of screen noticeable discontinuity between adjacent triangles is found. Perspective correct texturing accounts for the positions of vertices's in 3D space, but it is slower to calculate. Instead of interpolating the texture coordinates directly, the coordinates are divided by their depth, and the reciprocal of the depth value is interpolated and used to recover the perspective-correct coordinate. This correction makes it so that in parts of the polygon that are closer to the viewer the difference from pixel to pixel between texture coordinates is smaller (stretching the texture wider), and in parts that are farther away this difference is larger (compressing the texture). [7]

4.2 Procedural Texture

Getting multiple 2D texture to form a visually consistent appearance without looking tiled is a difficult and tedious task. As an alternative for the traditional methods procedural texturing are created using an algorithm and have realistic representation of natural elements such as wood, stone, granite, marble etc. Unlike traditional method of using 2D surface position, 3D position are used to evaluate all visible. [5]

Realistic looking render is usually achieved by using function like turbulence, fractal noise or some simple functions like sum of sinusoidal functions. These functions are used of the "randomness" found in nature. These are called solid texturing. Other Different types of procedural texturing exist like Generic and Cellular which we won't be using.

4.3 Bump Map

Simulating bumps and wrinkles on the surface of an object is done using technique called bump mapping. It allows the texture to directly control the surface and gives a very good appearance of a complex surface, such as tree bark or rough concrete. This is achieved by perturbing the surface normals of the object and using the perturbed normal during illumination calculations. The result is an apparently bumpy surface rather than a perfectly smooth surface although the surface of the underlying object is not actually changed. Normal and parallax mapping are the most commonly used ways of making bumps. [7, 11].

4.4 UV Mapping

Making a 2D image representation of a 3D model is know as UV mapping. UV texturing also permits 3D object to be painted with color from an image which is knowns as UV texture map, just an ordinary texture.

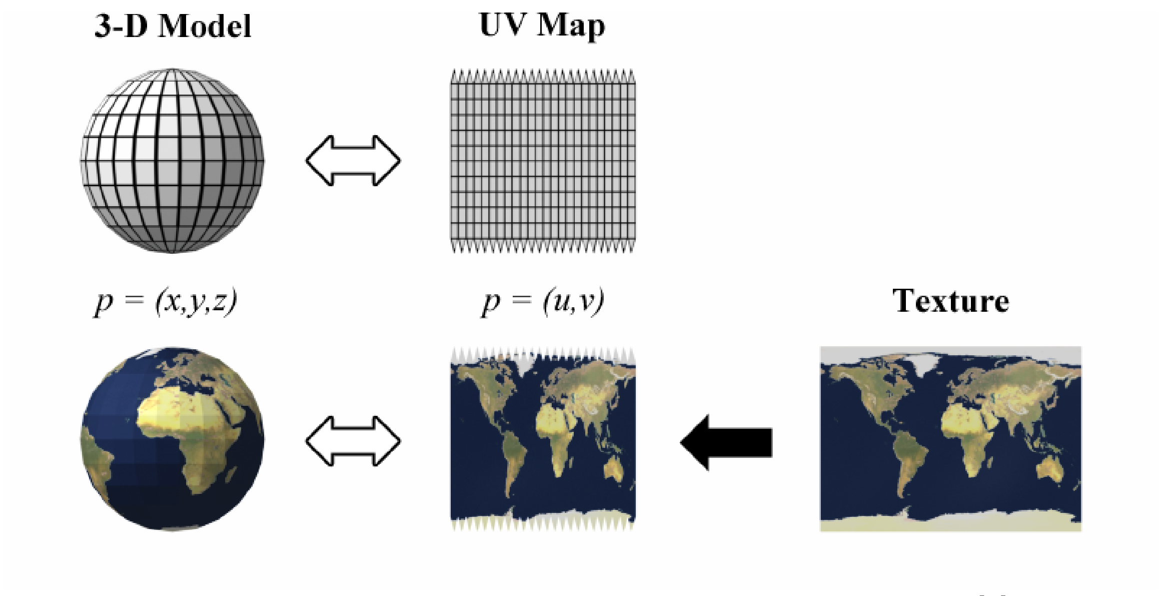


Fig. 4.3: Working of UV-Mapping. Illustrative by Tschmits [7]

The letters “U” & “V” are describe the 2D mesh because “X”, “Y” and “Z” are already used to describe the 3D object in space and “W” when using quaternions. UV texture coordinates to determine how to paint the 3D-surface, which reduce the computationally intensive rendering like procedural, multilevel texturing. This technique enhance the visual richness of texture immensely with the relatively low computation. [12]

The UV Mapping process at its simplest requires three steps: unwrapping the mesh, creating the texture, and applying the texture. Often a UV map is be generated, and then the artist adjusts and optimize it to minimize seams and overlaps. If the model is symmetric, the artist might overlap opposite triangles to allow painting both sides simultaneously. UV coordinates are applied per face, not per vertex. This means a shared vertex can have different UV coordinates in each of its triangles, so adjacent triangles can be cut apart and positioned on different areas of the texture map. [12]

UV in the Sphere is given by:

$$u = \sin\theta\cos\phi = \frac{x}{\sqrt{x^2+y^2+z^2}} \qquad v = \sin\theta\cos\phi = \frac{y}{\sqrt{x^2+y^2+z^2}}$$

4.5 Multi-Texturing

Multi-Texturing is the use of more than one texture at a time on a polygon. For instance, a light map texture may be used to light a surface as an alternative to recalculating that lighting every time the surface is rendered. It can give a very good appearance of a complex surface, such as tree bark or rough concrete, that takes on lighting detail in addition to the usual detailed coloring. [9]

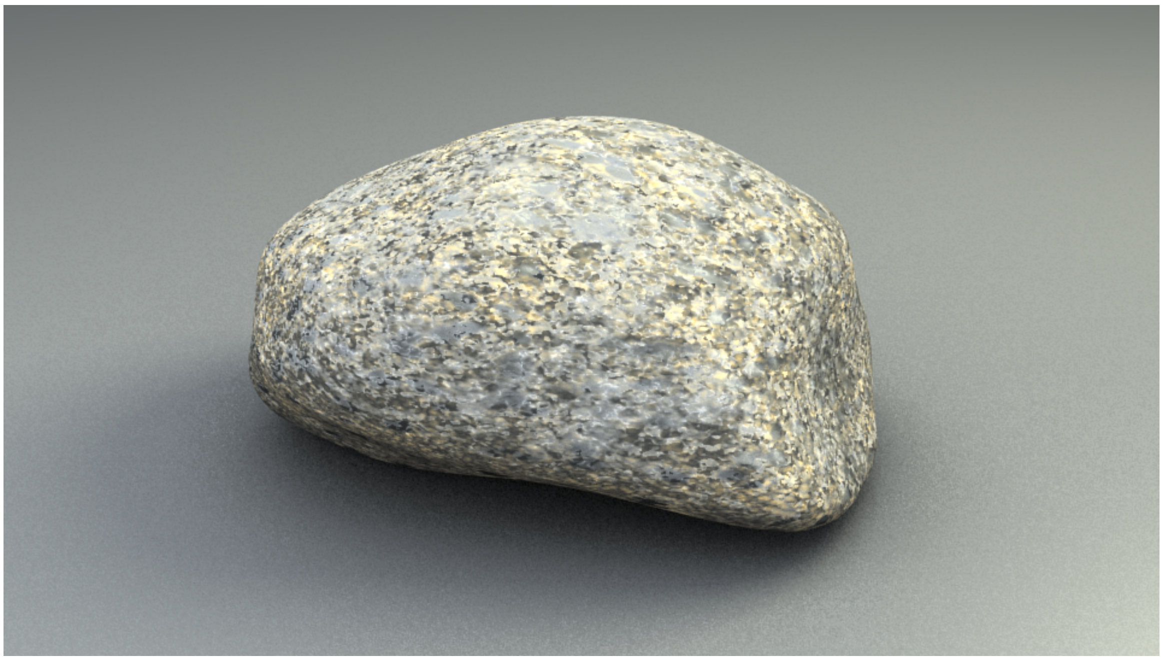


Fig. 4.4: Multiple Procedural Textures with Bump mapping [9]

4.6 Materials

Materials system is an advanced type of texture mapping. It allows for 3D objects to simulate different types of materials in real life. This makes it so that the texture not only contains graphical data, but references for sound data and physics data (such as density). For example, if a texture makes an object look like wood, it will sound like wood(if something hits it or its scraped along a surface), break like wood, and even float like wood. If it was made of metal, it will sound like metal, dent like metal, and sink like metal. This allows more flexibility when making objects in games. [7]

A materials system allows a designer to think about objects in a different way. Instead of the object just being a model with a texture applied to it, the object, or part of the object, is made up of a material. Currently there are these major materials: wood, concrete (or stone), metal, glass, dirt, water, and cloth (such as carpeting). [7]

4.7 Implementation

The surface of a natural material may seem one of the easiest to reproduce in a 3D suite such as Blender. In many ways simulating natural objects in Blender will require more complex materials and textures than man-made objects to make them look convincing. Blender offers a vast array of material and texture tools to aid you in the creation of natural-looking surfaces. Because of this there are many ways to produce similar, and equally pleasing, results. However, there are approaches that will speed material creation and make the process easier, adaptable, and more enjoyable. [9]

We tested the steps involved in applying a simple procedural texture to a mesh and other surface qualities such as bumpiness. We tried to bring out the photo realistic effect as possible. Yet much of the areas were not explored due to time restriction and the goal of our project.

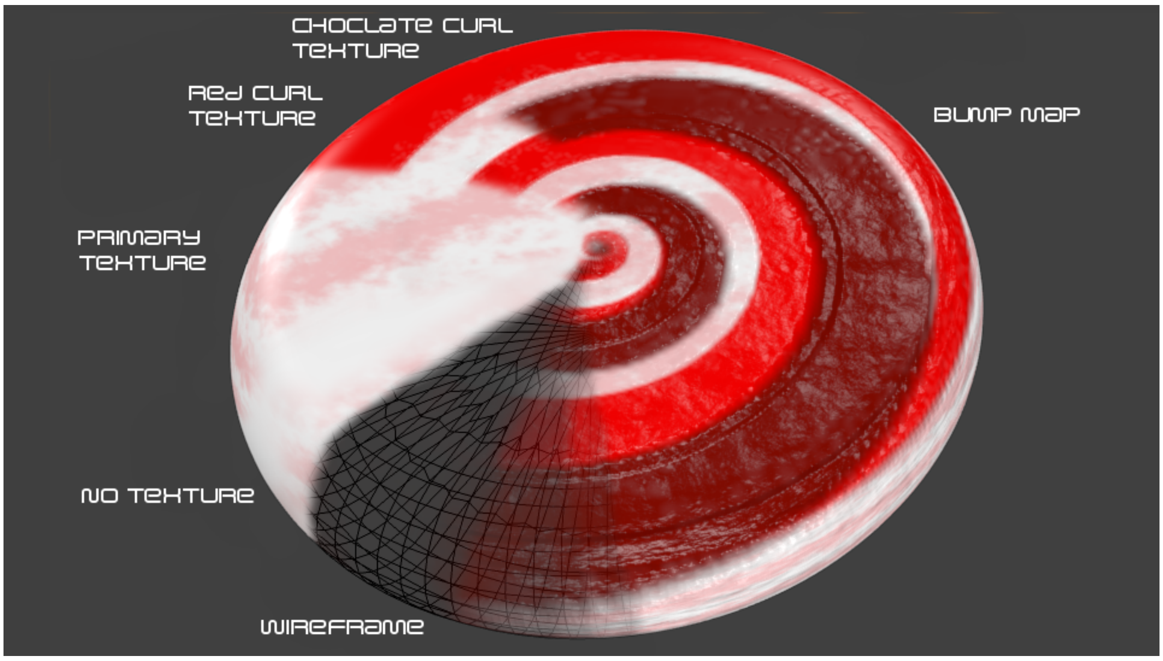


Fig. 4.5: Texturing Process

4.7.1 Conclusion

Materials and texturing are very in-depth subjects, and these examples have only scratched the surface. Yet further learning have to be done understanding. The optimization of the rendering computation is the main aim of our project for the virtual tour would be requiring the lots of texturing techniques like Procedural Texturing, UV-mapping etc

Chapter 5

Physic Engine

A physics engine is computer software that provides an simulation of certain physical systems, such as rigid body dynamics, soft body dynamics, and fluid dynamics, of use in the domains of computer graphics. Their main uses are for games engines, films and for scientific purposes. [7]

Generally, there are two classes of physics engines: real-time and high-precision. **High-precision** engines require raw processing power and used in computer animated movies and scientific purposes. We will be focusing mainly in the **Real-time engines** for this project which provide simplified approximate calculation for real-time response. As its used in video games and other forms of interactive computing.

5.1 Collision Detection

Interaction between objects, environment and the player makes a game realistic. Meshes are used to represent the 3D objects in a game, which are basically of two types. One of them is used to represent highly complex and detailed shapes i.e. the smooth surfaces of an object in a game. The second one which represents a simplified version of the same object (invisible wire frame mode) to the physics engine as a purpose of speed. The physics processing of the mesh objects which may be a bounding sphere, box or a convex hull is often referred to as the collision geometry. Use of bounding boxes or spheres is used in simple collision detection use to reduce the cost of computation.

Precision in discrete collision can be achieved by calculation of framerate. Each frame is treated as separate entity. Situations where there is a small-fast moving object with low framerate, it appears as if it teleported another place, instead of the smooth visible motion. Projectiles moving at sufficiently high speeds will miss targets, if the target is small enough to fit in the gap between the calculated frames of the fast moving projectile. Continuous collision detection such as in Bullet or Havok physics engine does not suffer this problem.

A finite based element-based system is an alternative to this, where a volumetric tessellation is created. Aspects such as plasticity, volume preservation and toughness of the object are results of tessellation, after which solver are used to model the stress within the 3D object. Degree of fracture, deformation and other physical effects with realism and uniqueness are derived using stress control. The engine's ability to modeling physical behavior increases as the content of modeled elements. Changes are confirmed to visual representation of the 3D object by the finite element system in application to deformation shader run on the GPU or CPU. As a result of performance overhead and lack in creation of finite element representations for 3D art objects, which is impractical for games.

5.2 Brownian Motion

Physics is always seen active, in the real world. All particles in the universe experience the constant Brownian motion jitter as the forces push forth and back against each other. Such constant activities unnecessarily waste the CPU power of a game physics engine which causes problem like decreased framerate. Thus, putting of objects to "sleep" by disabling the computation of physics on objects that have been inactive for a certain period of time. i.e. freezes in the place until it is reactivated by a collision with some other actively physical object, only than physical processing starts again.

Earlier use of rigid body dynamics in physics-based character animation had prevailed as they were easier to calculate and faster, whereas modern games and movies have started using soft body physics for particle effects, liquids and cloth. For stimulating characteristic of water and other fluid like properties Fluid dynamics simulation are deployed as well as the flow of fire and explosions through the air.

5.3 Paradigms

Two core components of Physics engines games as collision response or detection system and the dynamics simulation. These are responsible to solve the forces affect on the simulated objects. Modern physics engines have been introducing many physics like fluid simulations, control animated systems and asset integration tools. The three major paradigms for the physical simulation of solids are as:

Penalty method: Mainly for soft-body physics, popular for deformable. Here interactions are modeled as mass-spring systems.

Constraint based method: where constraint equations are solved that estimate physical laws.

Impulse based method: where impulses are applied to object interactions.

Finally, hybrid methods are possible that combine aspects of the above paradigms.

5.4 Simulation

Simulation is also knows as Procedural Animation, all natural processes are governed by the laws of physics, like a collapsing wall or water splashing into a glass. With a greater or lesser degree of success, be simulated by a computer program. Often, a computer can do a much better job of animation than a human being can, because it can actually simulate the physics of the situation.

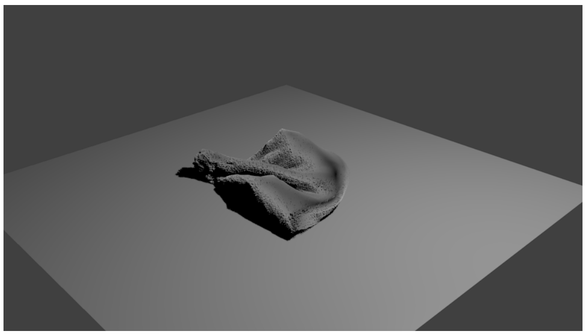


Fig. 5.1: Testing Cloth and Fur Simulation

5.5 Game Engine

A game engine a different beast, designed for the creating games. A game engine offers much more than the physics engine, it a rendering engine, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support etc. Same game engines are used for creating many different games to economized the cost of development. [7]

5.6 Implementation

We used Game Engine for creating interactive demo of Virtual Tool. We used the Blender 2.56 version for this purpose.

The Blender Game Engine (BGE) has several ways in which interactivity can be programmed into a game environment.

5.6.1 Blender Game Engine

Blender comes with the game engine support. built-in game design tools with Bullet physics support. Blender's game design tools like logic blocks, have the advantage of GUI implement functionality. For beginners, might be easiest way to get started without prior programming knowledge. Although for the greater scalability and versatility, Python can be used with the combination with logic blocks.

5.6.2 Conclusion

As far as the Implementation point of the view project seems to be working good enough. But its inadequate in the field of professionalism. We made this project as for our exploration in 3D Graphics and use open-source for creating the working 3D model. As our project time constrains, we had to highly neglect visualization and atheistic part of the project.

Bibliography

[1] Ton Roosendaal, Roland Hess, and Blender Foundation. *The Essential Blender: Guide to 3D creation with the open source suite Blender*. No Starch Press, Sep 2007.

[2] How to Make Model Town. http://www.ehow.co.uk/how_5074512_make-model-town.html.

[3] Virtual Tours of Dudley Castle archive. http://www.exrenda.net/dudley/dudley.htm.

[4] Donald Hearn and M. Pauline Baker. *Computer graphics: C version*. Prentice Hall, Apr 1997.

[5] Lance Flavell. *Beginning Blender*. Apress, Sep 2010.

[6] Calculate distance, bearing and more between latitude/longitude points. http://www.movable-type.co.uk/scripts/latlong.html.

[7] Online Free Encyclopedia. http://en.wikipedia.org.

[8] Luiz Velho, Paulo Cezar Pinto Carvalho, Jonas Gomes, and Luiz Velho. *Mathematical Optimization in Computer Graphics and Vision*. Morgan Kaufmann, Apr 2008.

[9] Colin Litster. *Blender 2.5 Materials and Textures Cookbook*. Packt Publishing, Jan 2011.

[10] Luke Ahearn. *3D Game Textures*. Focal Press, Oct 2010.

[11] James F. Blinn. Simulation of Wrinkled Surfaces. volume 12, pages 286–292. SIGGRAPH '78, SIGGRAPH-ACM, Aug 1978.

[12] Tony Mullen. *Mastering Blender*. John Wiley and Sons, Mar 2009.

[13] Andrew Price. The Complete Architecture Series. http://www.blenderguru.com/the-complete-architecture-series/.

[14] Allan Brito. *Blender 3D 2.49: Architecture, Buildings, and Scenery*. Packt Publishing, Aug 2010.

[15] David Franson. *2D Artwork and 3D Modeling for Game Artist*. Cengage Learning, Oct 2003.

[16] Dave Shreiner. *OpenGL Programming Guide*. Addison-Wesley Professional, Aug 2009.

[17] Thomas Larsson. *Code snippets Intro to Scripting in Blender 25x*.

[18] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.5*. Focal Press, Oct 2010.

[19] David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. *Level of detail for 3D graphics*. Morgan Kaufmann, Mar 2009.