3D Simulation game of Institution using UNITY3D and C#

MAJOR PROJECT REPORT

Submitted for the partial fulfillment of the requirement for the award of Degree

B.TECH

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

Aditya uikey -0101CS191010

Ayushi shakya-0101CS191035

Manish Chautele-0101CS191064

Guided By:

Dr. Rajeev Pandey

(Associate Professor) -DoCSE

Dr. Shikha Agrawal

(Associate Professor)- DoCSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDALAYA

BHOPAL - 462033

SESSION 2019 - 2023

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Ayushi shakya, Aditya Uikey, Manish Choutele** of B.TECH. Fourth Year, Computer Science & Engineering have completed their Major Project entitled “3D SIMULATION GAME OF INSTITUION USING UNITY3D AND C#” during the year 2022-23 under our guidance and supervision.

We approve the project for the submission for the partial fulfillment of the requirement for the award of degree of B.TECH in Computer Science & Engineering.

Dr.Rajeev Pandey
Project guide

Dr. Shikha Agrawal
Project guide

DECLARATION BY CANDIDATE

We, hereby declare that the work which is presented in the major project, entitled “3D SIMULATION GAME OF INSTITUION USING UNITY3D AND C#” submitted in partial fulfillment of the requirement for the award of Bachelor degree in Computer Science and Engineering has been carried out at University Institute of Technology RGPV , Bhopal and is an authentic record of our work carried out under the guidance of **Dr. Rajeev Pandey** (Project Guide) **and Dr.Shikha Agrawal** (Project Guide) ,Department of Computer Science and Engineering, UIT RGPV, Bhopal.

The matter in this project has not been submitted by us for the award of any other degree

Ayushi Shakya [0101CS191035]

Aditya Uikey [0101CS191010]

Manish Choutele [0101CS191064]

ACKNOWLEDGEMENT

After the completion of major project work, words are not enough to express our feelings about all those who helped us to reach our goal, feeling above all this is our indebtedness to the almighty for providing us this moment in life.

First and foremost, we take this opportunity to express our deep regards and heartfelt gratitude to our project guide **Dr.Rajeev Pandey and Dr.Shikha Agrawal of Computer Science and Engineering Department, RGPV Bhopal** for their inspiring guidance and timely suggestions in carrying out our project successfully. They have also been a constant source of inspiration for us.

We are extremely thankful to **, Prof. Uday Chaurasia Head, Computer Science and Engineering Department, RGPV Bhopal** for his cooperation and motivation during the project. We would also like to thank all the teachers of our department for providing invaluable support and motivation. We are also grateful to our friends and colleagues for their help and cooperation throughout this work.

Ayushi Shakya [0101CS191035]

Aditya Uikey [0101CS191010]

Manish Choutele [0101CS191064]

Abstract

Our project is based on the concept of virtual gaming and is inspired by AR VR technologies where one person can visit any place they want and have a real world experience through virtual world.

We are making a 3D adventure and simulation game of our institutes campus with the use of one of the most popular gaming platform UNITY3D and language C# where a student character can visit institution through the adventures game and can get a virtual tour of the RGPV.

VR provides a unique and immersive way to interact with others. VR is also more affordable than ever before and has been embraced by the entertainment industry,. It allows viewers to see directly through the 'eyes' of a real or imagined subject and more specifically, VR can create 'embodied cognition'.It can transform the way of education which is provided nowadays. It can be also used in industries, medical, security, military, Anti-terrorism, Traffic accidents awareness, Weather forecasting,

Which is why we decided to present our institute in the virtual world for the future students who wishes to see campus before admission.

With the rise of popularity of VR gaming and existence of virtual world like metaverse every sector is stepping up their game in the virtual industry which is estimated to worth in trillions by 2023 .With RGPV being fairly a popular and prestigious university , having such a vast campus does not have any virtual presence and being located in the outskirts of the bhopal it is fairly far and difficult for students to visit the campus.We have proposed the solution as to make the 3D Virtual model of the UIT RGPV campus which would make it easy for an outsider to visit our college virtually.

3D visualization usually allows users to relate what is displayed to what they know more easily. For example, when a doctor sees the visualization of a heart (in a 3D-to-2D projection), he/she can reconstruct the 3D shape/structure (including the occluded parts) without much effort.
With the use of 3D visualization we have added the touch of simulation and adventure gaming

Simulation is used in many contexts, such as simulation of technology for performance tuning or optimizing, safety engineering, testing, training, education, and video games. Simulation is also used with scientific modelling of natural systems or human systems to gain insight into their functioning, as in economics

Table of Contents

CHAPTER 5

CHAPTER 6

CHAPTER 1

INTRODUCTION

C#

C# is a modern, general-purpose programming language that can be used to perform a wide range of tasks and objectives that span over a variety of professions. C# is primarily used on the Windows .NET framework, although it can be applied to an open source platform. This highly versatile programming language is an object-oriented programming language (OOP) and comparably new to the game, yet a reliable crowd pleaser.

<u>When was C# created-</u>

When compared to long-standing languages like Python and PHP, C# is a young addition to the programming family at nearly twenty years old. The language was developed in the year 2000 by Microsoft's Anders Hejlsberg, a Danish software engineer with a history for popular creations. Anders has taken part in the creation of a handful of dependable programming tools and languages, including Microsoft's TypeScript and Delphi, a suitable replacement for Turbo Pascal.

As of November 2022, C# ranked #4 on the PYPL Popularity of Programming Language Index, right behind Java and JavaScript. The data used to compile this index is based on how often people search for a tutorial on different programming languages in Google.

C# has also made a consistent appearance in the top ten programming languages in the TIOBE Index, a report that pulls its data from a compilation of popular search engines including Google, YouTube and Bing.

<u>Where did C# get its name-</u>

In the beginning, C# was originally titled COOL, a clever acronym that stood for "C-like Object Oriented Language." Unfortunately, Microsoft was unable to hang onto the fun name for reasons having to do with trademark law.

C# was originally designed to rival Java. Judging by the quick rise to popularity and the positive response from both new and seasoned developers, it's safe to say that goal has been achieved.

Advantages of C# Programming

- C# programming can save you time

Perhaps the greatest advantage is how much time you can save by using C# instead of a different programming language. Being that C# is statically typed and easy to read, users can expect to spend less time scouring their scripts for tiny errors that disrupt the function of the application.

C# also emphasizes simplicity and efficiency, so programmers can spend less time writing complicated stacks of code that are repeatedly used throughout the project. Top it all off with an extensive memory bank and you've got a time-effective language that can easily reduce labor hours and help you meet tight deadlines without tossing back that third cup of coffee at 2:00am.

- C# is easy to learn

In addition to the time you can save during project development, you'll also spend less time learning C# as opposed to the more difficult programming languages out there. Thanks to its simplicity and easy-to-use features, C# offers a fairly low learning curve for beginners.

This language makes for a great first step into the field and provides aspiring developers with a comfortable way to become familiar with programming without becoming discouraged and overwhelmed.

- C# is scalable and easy to maintain

C# is a programming language that is remarkably scalable and easy to maintain. Because of the strict nature of how static codes must be written, C# programs are reliably consistent, which makes them much easier to adjust and maintain than programs that are written using other languages.

If you ever need to return to an old project written in C#, you'll be pleased to find that while your processes may have changed over the years, your C# stack has remained the same across the board. There is a place for everything and everything is in its place.

- C# has a great community

In the world of coding and programming, the importance of a helpful community on which you can depend simply can't be overstated. Programming languages aren't a platform or service with a dedicated help line or convenient IT support. Programmers must rely on the support of others in the same field who have experienced the same roadblocks and frustrations.

One such community of helpful programming experts can be found on StackOverflow. Because this Q&A site was constructed in C#, it's no surprise that C# developers make up a massive portion of the community where you can go to ask, answer, brainstorm, or vent.

If you prefer to collaborate with like-minded individuals face-to-face, C# also has an extensive community on Meetup.com, where members can join both online and IRL discussions that are scheduled at random or on a consistent basis.

- C# is object-oriented

C# is completely object-oriented, which is a rare characteristic for a programming language. Many of the most common languages incorporate object orientation to an extent, but very few have accomplished the magnitude of C# without losing favor from the people.

There are many different advantages to object-oriented programming (or OOP), such as efficiency and flexibility to name a few. Some developers who are unfamiliar with OOP may feel a little reluctant to choose a new language with such a heavy emphasis on it, but don't worry: understanding object-oriented programming isn't all that difficult.

Applications of C#

- Web Application Development

Regardless of the platform, you can still use the **C# programming language**. You can build dynamic websites and web apps using the .NET platform or other open-source platforms.

- Windows Applications

Microsoft created C# for Microsoft. So, there is no question why it is popular in building Windows applications. It makes your development process smooth, and functionalities such as **C# garbage collection** performs excellently.

Also, developers can count on the community support and documentation of developing applications and programs that are specific to the Microsoft platform's architecture.

- Games

In the gaming world, developers prefer C# programming language. This language is particularly robust in building fan-favorite games. Also, the Unity game engine—one of the most popular game engines today—is built using C++ and C#.

Game developers use game engines like Unity to build video games. It has more than a billion active users. It is also a trusted platform for 1.5 million active users worldwide. These active users continue to develop award-winning games and AR/VR experiences.

C# integrates seamlessly with the Unity engine. For mobile developers, it can be used virtually on any modern mobile device or console using cross-platform technology like Xamarin. So, if you want to break into the game development industry, or join the bandwagon of Virtual Reality, then you can leverage the programming language's potential.

UNITY 3D

Unity is, simply put, the world's most popular game engine. It packs a ton of features together and is flexible enough to make almost any game you can imagine.

With unrivaled cross-platform features, Unity is popular with both hobby developers and AAA studios. It's been used to create games like Pokemon Go, Heathstone, Rimworld, Cuphead, and plenty more.

While 3D is in the name, Unity 3D also packs tools for 2D game development.

Programmers love it because of the C# scripting API and built-in Visual Studio integration. Unity also offers JavaScript as a scripting language and MonoDevelop as an IDE to those who want an alternative to Visual Studio.

But artists love it as well since it comes with **powerful animation tools** that make it simple to create your own 3D cutscenes or build 2D animations from scratch. Nearly anything can be animated in Unity.

Also Unity 3D offers a free version so developers can release games made with Unity Personal without paying for the software, so long as they make less than $100,000 from games made with Unity.

For those willing to pay, Unity offers some extra features and a flexible licensing plan under a tiered subscription model. Premium users will have access to Unity's source code and developer support as well.

Because Unity has been around since 2005 it has developed a **massive following of users** and an amazing library of resources. Not only does Unity have fantastic documentation, but the sheer wealth of videos & tutorials online is staggering.

Beginners are wise to start with Unity for this reason alone. Among a laundry list of video game engines Unity serves as a portal of knowledge and resources built solely on their incredible community.

Unity Features

- Unity 3D comes loaded with a ton of professional tools for both programmers and artists.

- Unity provides a workspace that combines artist-friendly tools with a component-driven design that makes game development pretty darn intuitive.

- Both 2D and 3D development is possible in Unity, with 2D physics handled by the popular Box2D engine.

- Unity uses a component-based approach to game dev revolving around prefabs. With prefabs, game designers can build objects and environments more efficiently and scale faster.

- With powerful shaders, physics-based materials, post-processing, and high-resolution lighting systems, Unity can deliver impressive graphics across the board.

- For the platform itself: Unity was built in C++ and optimized over the years for performance. Premium users will have access to Unity’s source code for even greater opportunities.

- Cross-platform deployment is a major draw for today’s developers and Unity shines in this area. With support for **every major console and operating system**, games developed in Unity can be deployed to absolutely any platform.

- With Unity’s editor tools you can simultaneously handle inputs for mice, keyboards, and game controllers.

- There’s also some pretty strong support for cloud-based solutions for multiplayer games with server hosting and scalable matchmaking, making it an all-in-one solution to multiplier experiences.

- Team collaboration has been greatly improved in the newer versions of Unity. Built-in version control and cloud integration make working with others easier than ever before.

What Is Game Development

Game development is essentially the process of building and designing a game from the concept phase to the launch phase. The process involves multiple steps, each requiring a unique set of skills and a team of highly motivated developers. The main stages of game development from the technical perspective include:

The Concept Phase

In this phase, developers and concept creators measure the viability of the game's overall concept and its features. They decide what the game should fundamentally include and the end motive. The concept phase involves creating the game's story, missions, levels, and objectives.

The Design Phase

This phase involves planning and designing the game environment. Developers and game designers leverage powerful code to bring virtual design to reality. The design phase also involves planning several features within the game to make it unique and gripping, including the characters, the game landscape, player rewards, and more.

The Development Phase

The development phase involves writing innumerable lines of code to bring the game to life based on the pre-planned design. Developers implement every single feature into the game and bring concepts to life.

The Production Phase

This phase is a continuation of the development phase and mostly involves adding audio and visual effects to the game environment. It also involves integrating game levels into the gameplay.

Testing Phase

This is a crucial phase in the process that involves the acumen of testers and developers alike. The objective of this phase is to identify bugs in-game levels, characters, the game environment, and more. Bugs identified are then fixed before the master release happens.

The Launch Phase

The game finally hits the market after alpha/beta releases and is made available to gamers. Various other non-technical aspects like marketing and advertising are planned during this phase

CHAPTER 2

LITERATURE SURVEY

Existing problem

With the rise of popularity of VR gaming and existence of virtual world like metaverse every sector is stepping up their game in the virtual industry which is estimated to worth in trillions by 2023 .With RGPV being fairly a popular and prestigious university , having such a vast campus does not have any virtual presence and being located in the outskirts of the bhopal it is fairly far and difficult for students to visit the campus

Which is why we decided to present our institute in the virtual world for the future students who wishes to see campus before admission.

Our approach

We have proposed the solution as to make the 3D Virtual model of the UIT RGPV campus which would make it easy for an outsider to visit our college virtually.

3D visualization usually allows users to relate what is displayed to what they know more easily. For example, when a doctor sees the visualization of a heart (in a 3D-to-2D projection), he/she can reconstruct the 3D shape/structure (including the occluded parts) without much effort.
With the use of 3D visualization we have added the touch of simulation and adventure gaming

Simulation is used in many contexts, such as simulation of technology for performance tuning or optimizing, safety engineering, testing, training, education, and video games. Simulation is also used with scientific modelling of natural systems or human systems to gain insight into their functioning, as in economics

CHAPTER 3

PROBLEM DESCRIPTION

Lack of 3D Virtualization

With the rise of popularity of VR gaming and existence of virtual world like metaverse every sector is stepping up their game in the virtual industry which is estimated to worth in trillions by 2023 .With RGPV being fairly a popular and prestigious university , having such a vast campus does not have any virtual presence and being located in the outskirts of the bhopal it is fairly far and difficult for students to visit the campus.We have proposed the solution as to make the 3D Virtual model of the UIT RGPV campus which would make it easy for an outsider to visit our college virtually.

3D visualization usually allows users to relate what is displayed to what they know more easily. For example, when a doctor sees the visualization of a heart (in a 3D-to-2D projection), he/she can reconstruct the 3D shape/structure (including the occluded parts) without much effort.
With the use of 3D visualization we have added the touch of simulation and adventure gaming

Effects of 3D Visualization.

Here are some of the effects **of 3D Visualization and Animation**:

1. Visualize and analyse before you build

A 3D design is far more versatile than a traditional blueprint or sketch. Architects can use 3D rendering to create designs and walk through experiences that can be viewed from multiple angles, giving the client a clear perspective of the architect’s vision. These designs can provide the client with a virtual experience that allows them to visualize and comprehend the building’s features. By providing a virtual experience of how and what should be constructed, visualization promotes greater awareness and understanding of the project among management and site personnel. Thereby, Decision-makers can make sound decisions by visualizing future construction operations rather than relying on their intuition, instincts, and opinions.

2. Accurate & Sustainable Design

Precise dimension input makes designs vivid and accurate, saving time that was previously used to draw accurate lines. With the new 3D interior visualization tools, you can see views from various angles before laying a single brick of the foundation.

When sustainability is the norm, buildings that combine well with their surroundings contribute the most to a better future. Computer-aided design (CAD) software includes features that allow architects to test how environmental conditions in the construction area will affect the building. This enables architects to design buildings that take advantage of natural conditions such as sunlight, wind, rain, and so on, reducing their reliance on electricity or other artificial resources.

3. Reduced Clashes & Rework, Easy Re-modelling, and Amendments

3D visualization makes it simple to identify loopholes and design errors from the first stage of the design that are immediately fixed or an appropriate alternative are provided. The building or structural designs can be reviewed before they are materializing with 3D rendering and architectural visualization. Such concepts can be used for cost estimation/optimization, finalization of the to-be utilized materials, eliminating design faults, upgrading or downgrading a design, avoiding clashes or conflicts in the design, assessing its sustainability, etc.

4. Saves on Cost with Enhanced Efficiency

During the initial design process, the 3D visualization technology identifies errors and design failures. There is less money to fix errors and thus save costs. It also speeds up operational efficiency, because developers, contractors, investors, etc. have better coordination since they have seen a 3D image of the project on the same page. Construction activities can easily be viewed on the computer screen using 3-D CAD models, thus eliminating the cumbersome and costly physical models that might be needed to reconstruct certain scenarios. Also, Sequencing of interrelated activities in an operation can be better planned by organizing the resources for maximum productivity.

5. Safety

Problems with equipment positioning and manpower congestion in certain areas can be visualized before the actual operation, thus preventing accidents and safety problems such as the collision between two machines and losses in productivity.

CHAPTER 4

PROPOSED WORK

Game Proposed

We have proposed a 3D simulation and visualization game where we are building a virtual Model of the UIT-RGPV campus in a game format where a student character or new person who wishes to see RGPV can visit and take a tour of the college through an interesting game.We are doing all this with a simulation game using softwares like UNITY 3D ,blender and other necessary softwares

Simulation is used in many contexts, such as simulation of technology for performance tuning or optimizing, safety engineering, testing, training, education, and video games. Simulation is also used with scientific modelling of natural systems or human systems to gain insight into their functioning, as in economics

Levels of Game proposed-

1st - where student character will be able to take a tour of the college campus

2nd - where student character will be able to unlock and discover new college hangout spots which are popular among student and will be introduced to many other characters as well.

3rd- Where every character of the game will be able to attend college fests virtually and will be given knowledge about different fests.

About UIT-RGPV structure

University Institute of Technology Rajiv Gandhi Proudyogiki Vishwavidyalay

UIT-RGPV, Bhopal is a community of teachers, staff & students, who are dedicated to the achievement of a shared vision of excellence. The community believes in renewing and transferring knowledge & skills and the development of positive & purposeful attitudes and values. Being in the campus of the State Technological University, the UIT-RGPV is in a position to offer excellent and high quality academic services to the students.

History and Location

UIT-RGPV is one of the premier engineering institutions in Central India. The Institution was established in the year 1986, by the Government of Madhya Pradesh, as the Government Engineering College (GEC), Bhopal. In 1998 the Government of Madhya Pradesh declared this Institute autonomous and named it as Bhopal Engineering College. This Institute was renamed University Institute of Technology-RGPV (Technological University of the State of Madhya Pradesh) w.e.f.

from July 2002 by an order of Govt. of Madhya Pradesh. It is currently known as University Institute of Technology-RGPV, Bhopal or UIT-RGPV.

The Institution is situated in Bhopal, a city well known for its natural beauty & lakes. The city is well connected to the other parts of the country, by air, rail, and road. The large number of industries situated in and around Bhopal, provide excellent training opportunities to the students and staff.

Campus Layout and Environment

The institution is situated near Gandhi Nagar on the outskirts of the city, about 3 k.m. from the airport on the Bhopal by pass road. The campus is spread on a contiguous 241.14 acres. The campus has Instructional blocks, Workshop, Library Block and administrative block. Apart from this the campus has residential facilities for the staff and separate hostels for both girls and boys. The institution provides a very healthy academic atmosphere and has large open space, playgrounds and academics parks.

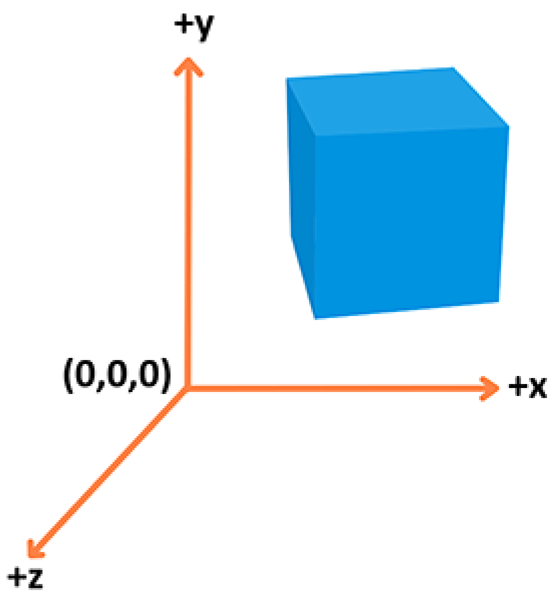
Management and Faculty

The UIT-RGPV is governed by the rules and regulations of RGPV, and its councils of governance and management. The Vice Chancellor of RGPV is the nominal head of all management committees of the UIT-RGPV. The Institute has academic autonomy and therefore has freedom to adopt its own scheme and syllabus. Freedom is available to the UIT-RGPV faculty to adopt a variety of students learning strategies and develop learning resources. The RGPV controls the final certification of the students, though continuous assessment is managed by the faculty of the UIT RGPV.

UIT RGPV teaching faculty forms, an effective combination of experienced, well qualified, and talented, teachers. Most faculty members have done their Ph.D./M.E./M.Tech. from reputed institutions like IITs, NITs and University departments. Senior faculty members have held various Government and non-Government portfolios. Some have been eminent educationists and have contributed to leading international & national development agencies. The institution draws heavily on faculty expertise for planning, executing and monitoring diverse activities.

Use of 3D Modelling

3D essentially is all about representations of shapes in a 3D space, with a coordinate system used to calculate their position.



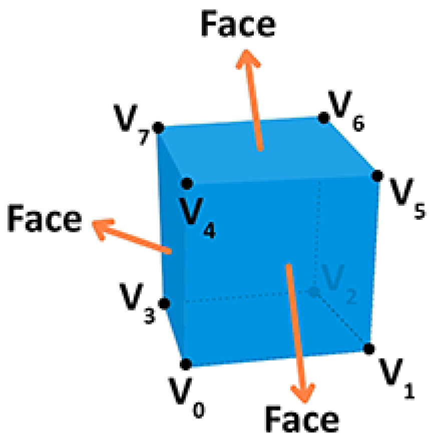
WebGL uses the right-hand coordinate system — the x axis points to the right, the y axis points up, and the z axis points out of the screen, as seen in the above diagram.

Objects

Different types of objects are built using vertices. A **Vertex** is a point in space having its own 3D position in the coordinate system and usually some additional information that defines it. Every vertex is described by these attributes:

- **Position**: Identifies it in a 3D space (x, y, z).
- **Color**: Holds an RGBA value (R, G and B for the red, green, and blue channels, alpha for transparency — all values range from 0.0 to 1.0).
- **Normal:** A way to describe the direction the vertex is facing.
- **Texture**: A 2D image that the vertex can use to decorate the surface it is part of instead of a simple color.

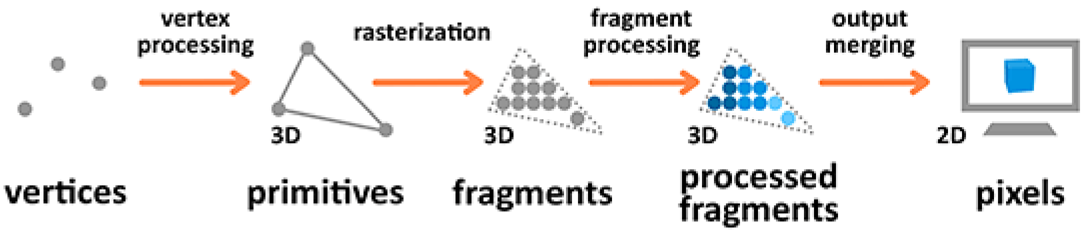
You can build geometry using this information — here is an example of a cube:



A face of the given shape is a plane between vertices. For example, a cube has 8 different vertices (points in space) and 6 different faces, each constructed out of 4 vertices. A normal defines which way the face is directed in. Also, by connecting the points we're creating the edges of the cube. The geometry is built from a vertex and the face, while material is a texture, which uses a color or an image. If we connect the geometry with the material we will get a mesh.

Rendering pipeline

The rendering pipeline is the process by which images are prepared and output onto the screen. The graphics rendering pipeline takes the 3D objects built from **primitives** described using **vertices**, applies processing, calculates the **fragments** and renders them on the 2D screen as **pixels**.



Terminology used in the diagram above is as follows:

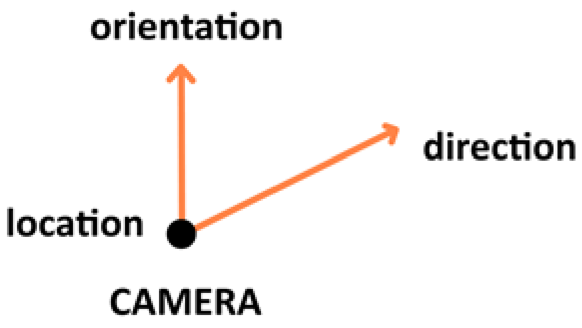
- A **Primitive**: An input to the pipeline — it's built from vertices and can be a triangle, point or line.
- A **Fragment**: A 3D projection of a pixel, which has all the same attributes as a pixel.

- A **Pixel**: A point on the screen arranged in the 2D grid, which holds an RGBA color.

Vertex processing

Vertex processing is about combining the information about individual vertices into primitives and setting their coordinates in the 3D space for the viewer to see. It's like taking a photo of the given scenery you have prepared — you have to place the objects first, configure the camera, and then take the shot.

There are four stages to this processing: the first one involves arranging the objects in the world, and is called **model transformation**. Then there's **view transformation** which takes care of positioning and setting the orientation of the camera in the 3D space. The camera has three parameters — location, direction, and orientation — which have to be defined for the newly created scene.



Projection transformation (also called perspective transformation) then defines the camera settings. It sets up what can be seen by the camera — the configuration includes *field of view*, *aspect ratio* and optional *near* and *far planes*. read the Camera paragraph in the Three.js article to learn about those.

The last step is **viewport transformation**, which involves outputting everything for the next step in the rendering pipeline.

Rasterization

Rasterization converts primitives (which are connected vertices) to a set of fragments.

Those fragments — which are 3D projections of the 2D pixels — are aligned to the pixel grid, so eventually they can be printed out as pixels on a 2D screen display during the output merging stage.

Fragment processing

Fragment processing focuses on textures and lighting — it calculates final colors based on the given parameters.

Textures

Textures are 2D images used in the 3D space to make the objects look better and more realistic. Textures are combined from single texture elements called texels the same way picture elements are combined from pixels. Applying textures onto objects during the fragment processing stage of the rendering pipeline allows us to adjust it by wrapping and filtering it if necessary.

Texture wrapping allows us to repeat the 2D image around the 3D object. Texture filtering is applied when the original resolution or the texture image is different from the displayed fragment — it will be minified or magnified accordingly.

Lighting

The colors we see on the screen is a result of the light source interacting with the surface colors of the object's material. Light might be absorbed or reflected. The standard **Phong Lighting Model** implemented in WebGL has four basic types of lighting:

- **Diffuse**: A distant directional light, like the sun.
- **Specular**: A point of light, just like a light bulb in a room or a flashlight.
- **Ambient**: The constant light applied to everything on the scene.
- **Emissive**: The light emitted directly by the object.

Output merging

During the output manipulation stage all the fragments of the primitives from the 3D space are transformed into a 2D grid of pixels that are then printed out on the screen display.

What are simulation games?

Simulation games are a genre of games that are designed to mimic activities you'd see in the real world. The purpose of the game may be to teach you something. For example, you could learn how to fish. Others simulation games take on operating a business such as a farm or a theme park.

How do simulation games work?

Usually, there are no defined goals in a simulation game. Players are given the freedom to roam and explore the environment freely. Some types of simulation games may include war games, role play simulation, or games related to business.

CHAPTER 5

IMPLEMENTATION

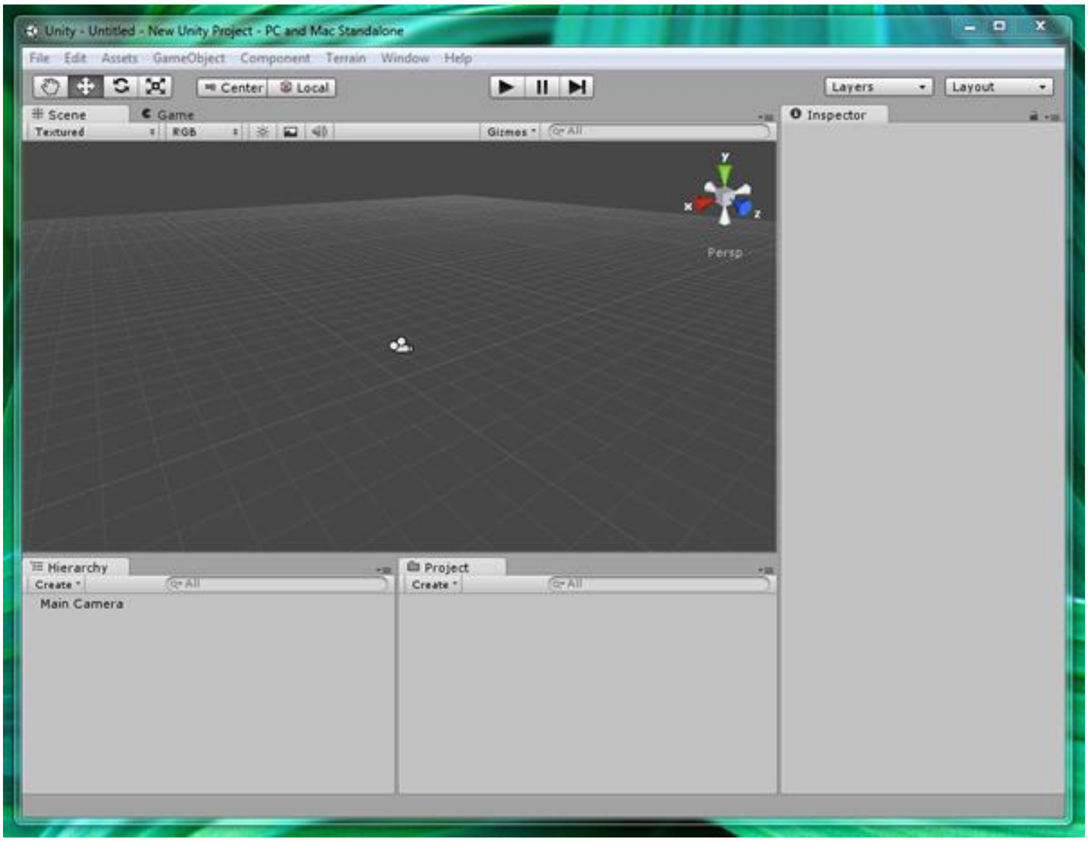
Use of Unity3D

What is Unity3D?

Unity3D is a powerful cross-platform 3D engine and a user friendly development environment. Easy enough for the beginner and powerful enough for the expert; Unity should interest anybody who wants to easily create 3D games and applications for mobile, desktop, the web, and consoles

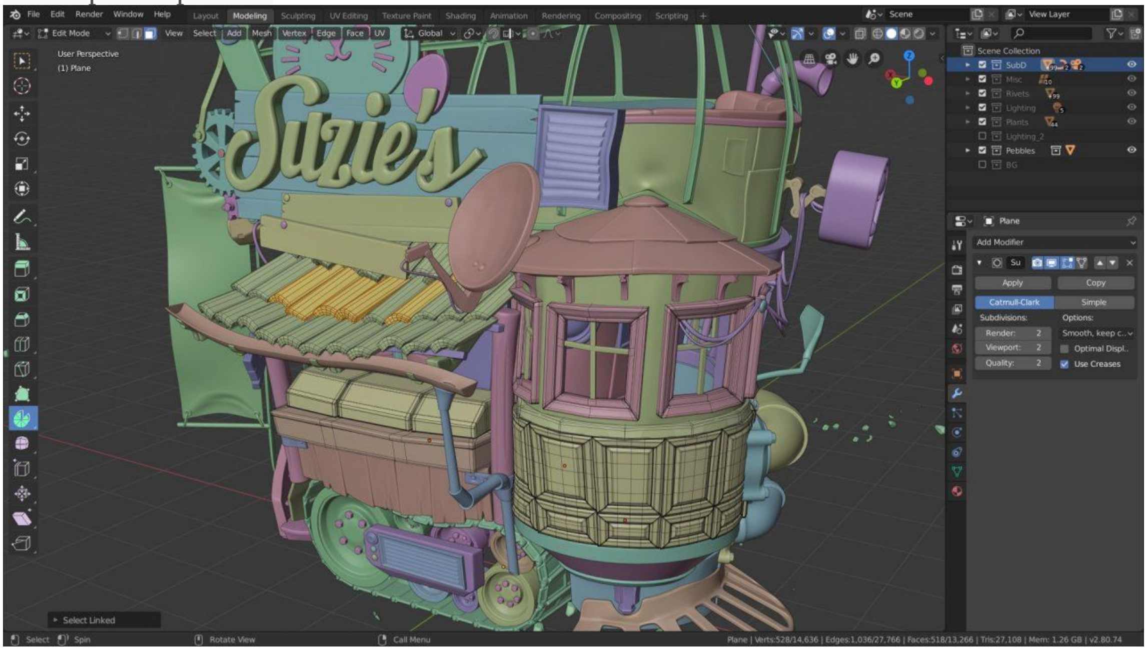
The Application

The Unity application is a complete 3D environment, suitable for laying out levels, creating menus, doing animation, writing scripts, and organizing projects. The user interface is well organized and the panels can be fully customized by dragging and dropping.



Use of Blender

Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender’s API for Python scripting to customize the application and write specialized tools; often these are included in Blender’s future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process



Catmull-Clark

Subdivisions:
Render: 2
Viewport: 2
Quality: 2

Options:
Smooth, keep c...
□ Optimal Displ..
☑ Use Creases

Plane | Verts:528/14,636 | Edges:1,036/27,766 | Faces:518/13,266 | Tris:27,108 | Mem: 1.26 GB | v2.80.74

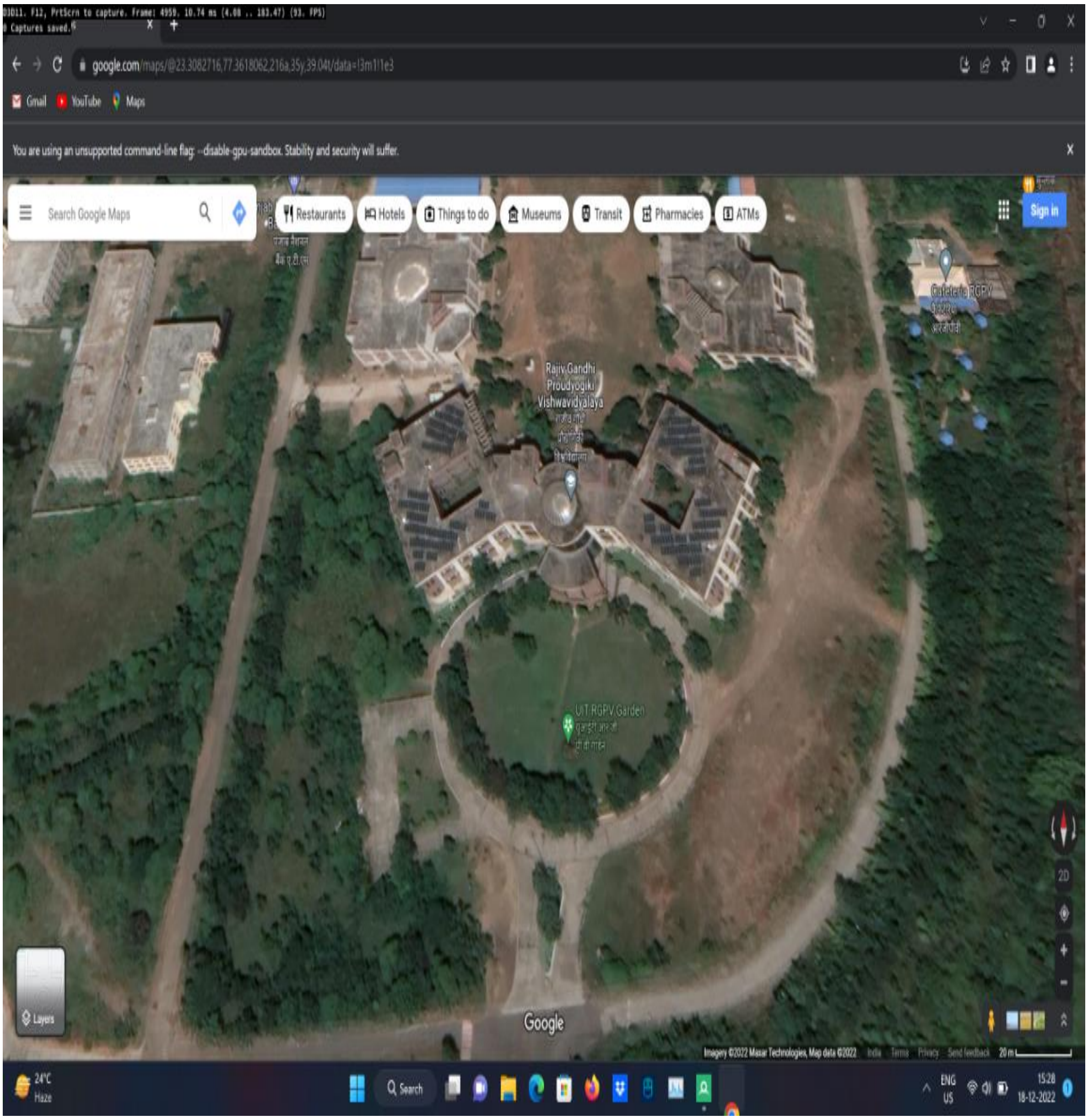
Using 3D Models

Depending on their purpose, there are different types of 3D models. To give just a few examples, some are intended for 3D printing, some for animation, and some for video games. The main difference between these model types is the way they're optimized to serve their purpose. Although the software itself that's used for their creation can sometimes be the same, it's the stuff behind the scenes of the models that heavily dictates their intended use.

Uses of Map

Google maps

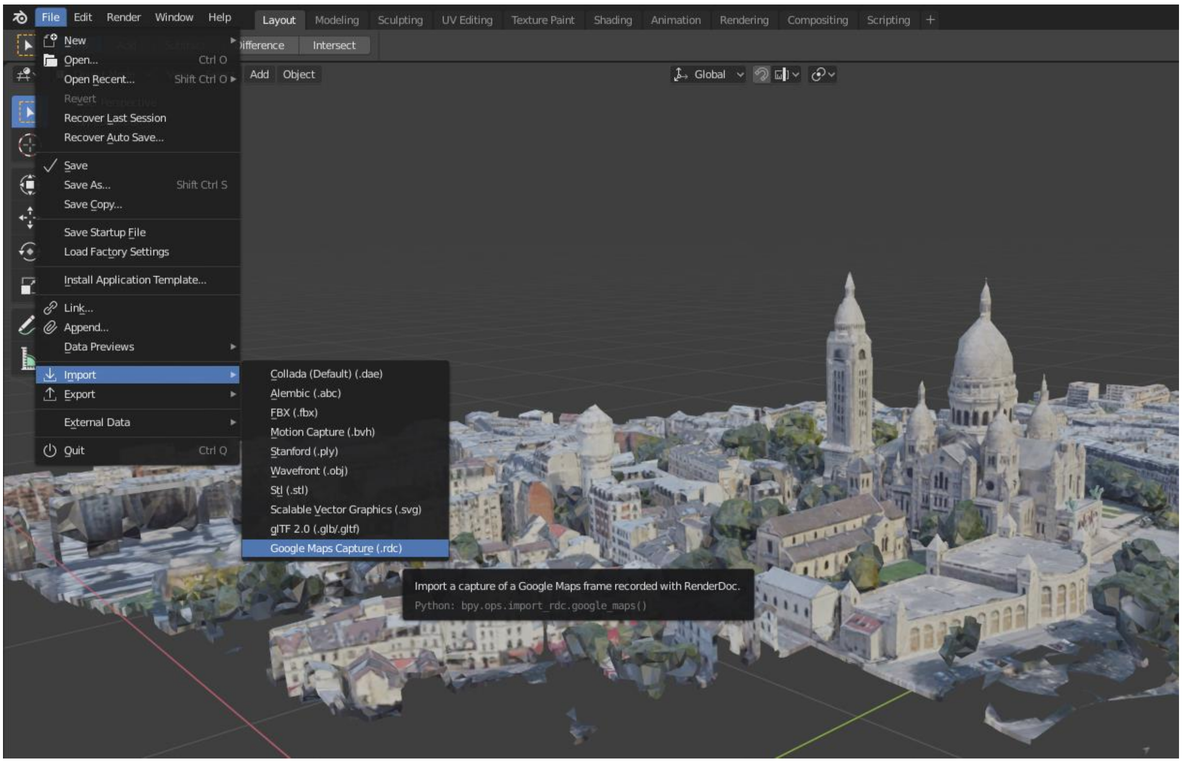
To create the 3d environment of the map for the game it needed to be reference from the google maps and the data to analysis and build the as close as possible to the real world.



```
D3D11. F12, PrtScrn to capture. Frame: 4959. 10.74 ms (4.08 .. 183.47) (93. FPS)
0 Captures saved.
```

You are using an unsupported command-line flag: --disable-gpu-sandbox. Stability and security will suffer.

To make a realistic environment google maps can be used to set to surrounding according to the location. Google maps are of great use in game making and blender.



Import a capture of a Google Maps frame recorded with RenderDoc.

```
Python: bpy.ops.import_rdc.google_maps()
```

Use of HMTL & CSS

HTML and CSS are both markup languages; they define how and where content on a web page is displayed.

In this project they needed to displayed detailed portfolio of the game all necessary things are include that be displayed on that web page.

Visual Studio Code

Visual Studio Code is a free, lightweight but powerful source code editor that runs on your desktop and on the web and is available for Windows, macOS, Linux, and Raspberry Pi OS. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages (such as C++, C#, Java, Python, PHP, and Go), runtimes (such as .NET and Unity), environments (such as Docker and Kubernetes), and clouds (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

The need of visual studio code in this game is to write the code (c#) for the intract with design , models and environment.

All the physics that build the game and adding more depth it use to define with the help of c# and for writing that coding it needed the industry level code editor “Visual Stuido Code”.

CHAPTER 6

Expected Outcome and future work

Expected outcome of the project must be the virtualization of the institute in form of a game that will include all mechanism of simulation and the interactive models.Models that show in the below fig. is the rough for the reference of the game that will help us to reach the better final outcome.

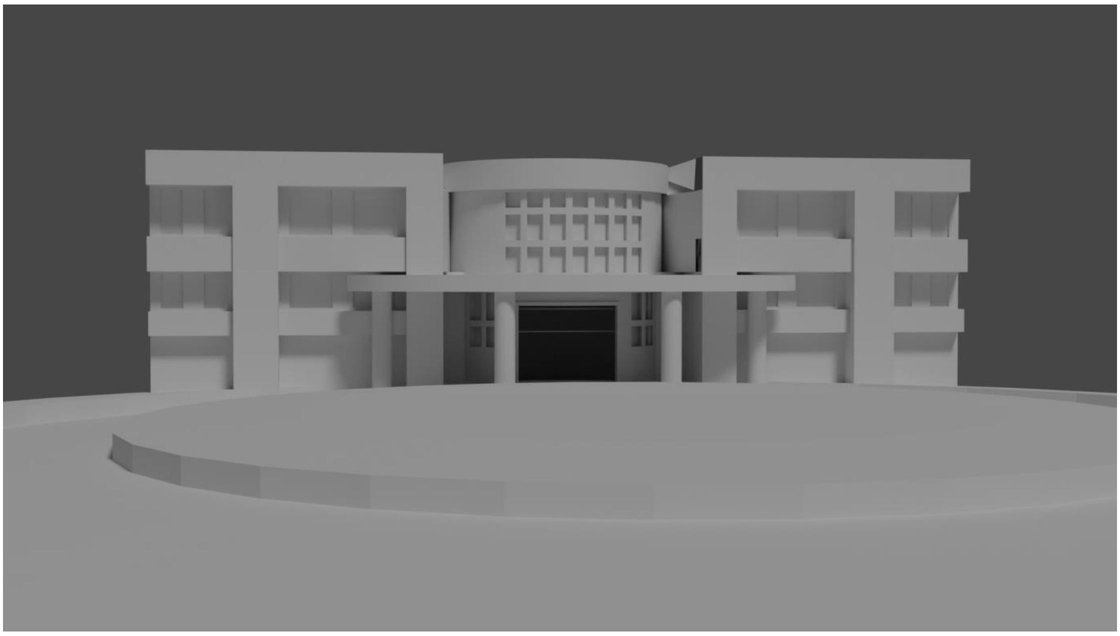
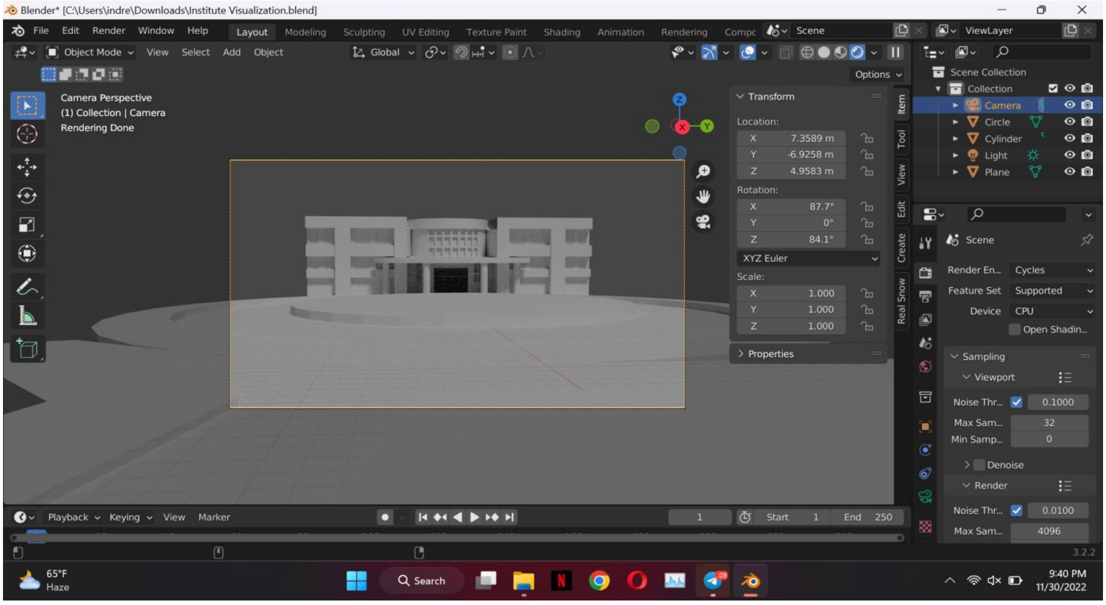


Fig.1 Rough Structure of Main Building



Blender* [C:\Users\indre\Downloads\Institute Visualization.blend]

Camera Perspective
(1) Collection | Camera
Rendering Done

Transform
Location:
X 7.3589 m
Y -6.9258 m
Z 4.9583 m
Rotation:
X 87.7°
Y 0°
Z 84.1°
XYZ Euler
Scale:
X 1.000
Y 1.000
Z 1.000

Scene Collection
Collection
Camera
Circle
Cylinder
Light
Plane

Scene
Render En... Cycles
Feature Set Supported
Device CPU
□ Open Shadin...
Sampling
Viewport
Noise Thr... ☑ 0.1000
Max Sam... 32
Min Samp... 0
□ Denoise
Render
Noise Thr... ☑ 0.0100
Max Sam... 4096

Start 1 End 250

Fig.2 Interface of Blender