Tribhuvan University
Pulchowk Campus, IOE

PROJECT REPORT
ON
COMPUTER GRAPHICS

4 SEASONS

Submitted To:
Mr. BASANTA JOSHI
Department of Electronics and Computer Engineering

Submitted By:
Pratik Budhathoki(074BEX429)
Ranjan Shrestha(074BEX435)
Sailesh Shrestha(074BEX437)
Shiva Ram Godar(074BEX442)

March 05, 2020

ACKNOWLEDGEMENT

We are very grateful to our subject teacher/mentor Mr. Basanta Sir for this wonderful opportunity to bring our theoretical knowledge to life through the projects that involve the use of knowledge learned throughout the semester. His guidelines and suggestions proved very helpful in the accomplishment of our graphics project. We are extremely thankful to the Electronics and Communication Department for setting up the course that hones our skills and helps us to be solid as a rock in the competitive market fresh out of college.

We also would like to extend our gratitude to our friends who became a part of this project indirectly sharing the necessary concept they knew of the subject. In addition to that, we are also very pleased to have got a chance to give a shout out to the reference books and some youtube channels without which this project would have been incomplete.

Sincerely,

Pratik Budhathoki(074BEX429)
Ranjan Shrestha(074BEX435)
Sailesh Shrestha(074BEX437)
Shiva Ram Godar(074BEX442)

ABSTRACT

Computer Graphics has evolved significantly in recent years with the development of modern sophisticated graphics rendering, editing and programming softwares and libraries such as AutoCAD, Maya, Blender, Photoshop, OpenGL,etc. And, as a result, computer graphics has been an indispensable component for people from a diverse range of disciplines from artists to scientists, from engineers to businessmen, from animators to data analysts. Hence, it’s the urge of this modern age that we be able to represent our ideas fluently in the form of computer graphics to be able to visualize and present our data even more clearly for better products. So, we are attempting to create a 3D visualization of the four seasons of the year to apply our computer graphics theory into practice, meanwhile learning to use amazing softwares and libraries which will make our life easier in the future for sure.

TABLE OF CONTENTS

INTRODUCTION

1.1 Introduction

Our project has been titled as ‘4 Seasons’. 4 Seasons is a simulation model of how the features of a landscape change as the seasons(Summer, Autumn, Winter and Spring) pass one after another. The landscape features a normal settlement area with trees, house, road , street lamps. As the transition of the seasons occur one by one, we’ll be able to see how the whole landscape changes. It also features the visualization of seasonal phenomena such as rainfall, snowfall and wind. Users also have the ability to switch between seasons using key options and ,view and navigate the 3d model of each season using arrow keys and mouse(trackpad too).

Figure 1:- A visualization of 4 different seasons

1.2 Objectives of the project

- To apply the algorithms and techniques studied in computer graphics theory into a graphics simulation
- To develop our skills in the field of graphics modelling, programming and simulation
- To be able to visualize a given object in its 3D view adjusting natural parameters like shading and lighting

BACKGROUND THEORY

Four Seasons:

The Earth's axis is slightly tilted in relation to its orbit around the Sun. This is why we have seasons.In most cultures, including all western countries, the year is commonly divided into four seasons:

1. Spring
2. Summer
3. Fall or Autumn
4. Winter

Since the year has 12 months, each season lasts about three months.There are two major factors that influence the seasons you experience.They are –Tilt of the earth's axis as it orbits the sun and your location on earth.

Winter: Winter is the coldest period of the year. The time when animals hibernate, and vegetation decreases. There is a general lull in the environment and in areas where it snows, it is snowman time. The days are shortest in winter.

Autumn: Autumn marks the beginning of cool weather. It is the time when leaves turn yellow/brown, and animals begin preparations because "winter is coming". It is also harvest season, which is a cause for celebration in most cultures.

Spring: Spring is the beginning of the cycle. New vegetation grows, and the weather is generally warm and sunny. Sometimes it rains in spring. It's the time when the plants and animals come alive and out of hibernation.

Summer: Summer is the hot weather season. Temperatures run highest in summer, accompanied by, depending on the location, extreme humidity or dryness. It is also a time for heat waves and droughts, and even forest fires in some areas. Summer is a time for long days.

Now to the technical Computer Graphics part..

The first thing to be noted is that in 3D Graphics , we'll be mostly dealing with the models in the form of matrices and vectors. So, here's some theory about vectors and matrices.

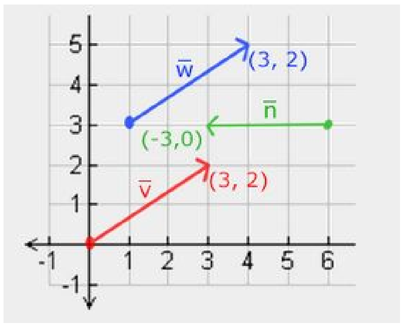
Vectors : Any physical quantity that has magnitude and direction

It is represented as:

$$\bar{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Scalar vector operations:

Addition(+),Subtraction(-), Multiplication(*), Division(/)



$Length/Magnitude : \sqrt{x^2 + y^2 + z^2}$

Unit vector: A vector of magnitude 1 (obtained by dividing a vector by its magnitude)

Converting a vector into unit vector is called **“Normalization”**

Dot product of two vectors:

$$\bar{v} \cdot \bar{k} = ||\bar{v}|| \cdot ||\bar{k}|| \cdot \cos\theta$$

Cross product of two vectors:

$$\begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} \times \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} = \begin{pmatrix} A_y \cdot B_z - A_z \cdot B_y \\ A_z \cdot B_x - A_x \cdot B_z \\ A_x \cdot B_y - A_y \cdot B_x \end{pmatrix}$$

<u>Matrices</u>

A rectangular array of numbers, symbols and/or mathematical expressions

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Matrix operations:

Addition/Subtraction

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Matrix-Scalar Multiplication

$$2 \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 2 \cdot 2 \\ 2 \cdot 3 & 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Identity Matrix

A matrix having all its major diagonal elements as ‘1’ and the rest ‘0’

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 \\ 1 \cdot 2 \\ 1 \cdot 3 \\ 1 \cdot 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Now the **Transformation Theory**

Scaling

Scaling simply means multiplying our original vector by some factor in x,y,z directions (individually or as a whole) to magnify or decrease its size.

$$\begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot y \\ S_3 \cdot z \\ 1 \end{pmatrix}$$

Translation

Translation means shifting the original vector to some other position in the space, i.e. adding a translation vector to the original vector.

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

Rotation

Rotating is simply changing the orientation angle of the original vector by some angle around the axes taking a reference point as the centre of rotation.

Rotation around the X-axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around the Y-axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x + \sin\theta \cdot z \\ y \\ -\sin\theta \cdot x + \cos\theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around the Z-axis:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y \\ \sin\theta \cdot x + \cos\theta \cdot y \\ z \\ 1 \end{pmatrix}$$

Cumulating the transformation vectors

Cumulating the transformation vectors simply means to multiply the individual transformation vectors into one

TransformedVector = TranslationMatrix * RotationMatrix * ScaleMatrix * OriginalVector; (GLM code, happens in the reverse order of operation!)

Coordinate Systems in Computer Graphics

5 coordinate systems that are important in CG:

- **Local space (or Object space)** - original coordinates of the object, relative to object’s origin
- **World space -** all coordinates relative to a global origin.
- **View space (or Eye space)-**all coordinates as viewed from a camera's perspective.
- **Clip space-** all coordinates as viewed from the camera's perspective but with projection applied
- **Screen space-** all coordinates as viewed from the screen. Coordinates range from 0 to screen width/height.

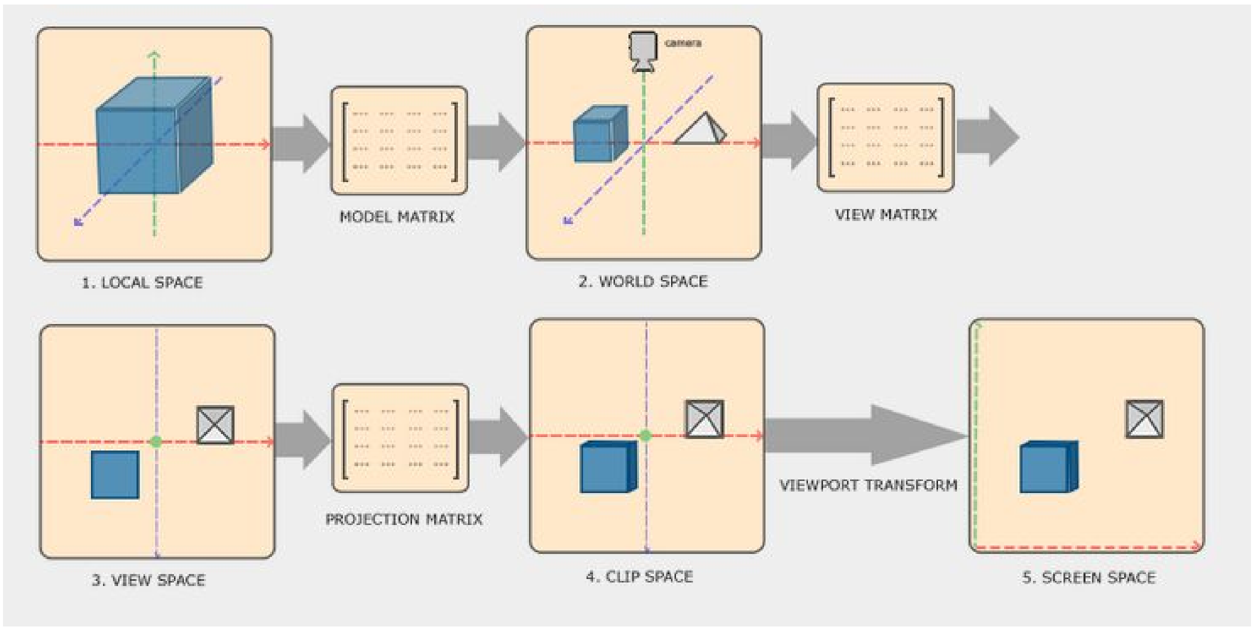
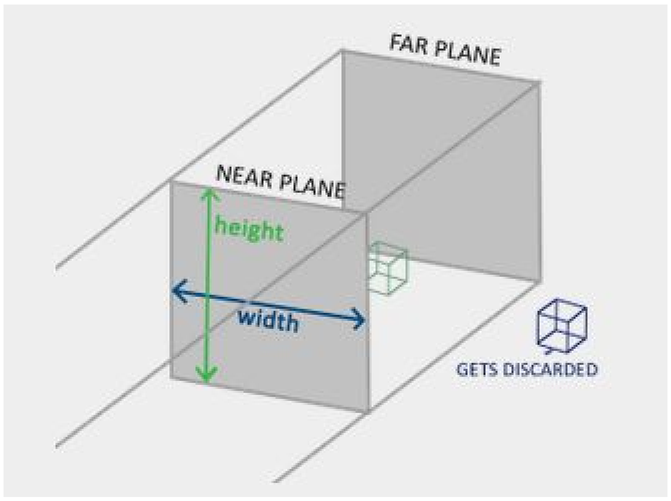


Fig. Overall process of converting objects from local space into view space coordinates

Thing about projection to be noted:

2 types of projection can be used: **Orthographic** and **Perspective**

An orthographic projection matrix directly maps coordinates to the 2D plane that is your screen, but in reality a direct projection produces unrealistic results since the projection doesn't take perspective into account. That is something the perspective projection matrix fixes for us. We'll be dealing with Perspective Projection in our program.



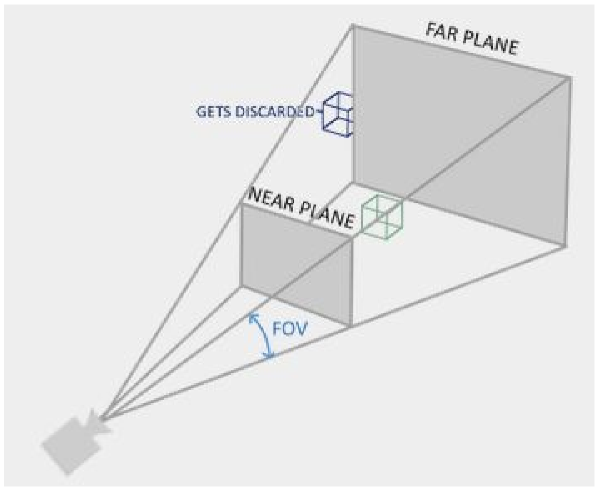


Fig. Orthographic Projection

Fig. Perspective Projection

Camera/View Space

Representing the objects from the perspective of camera

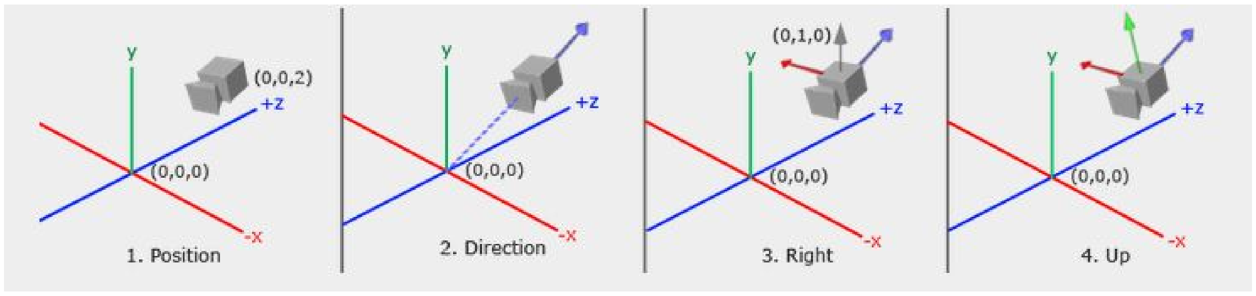
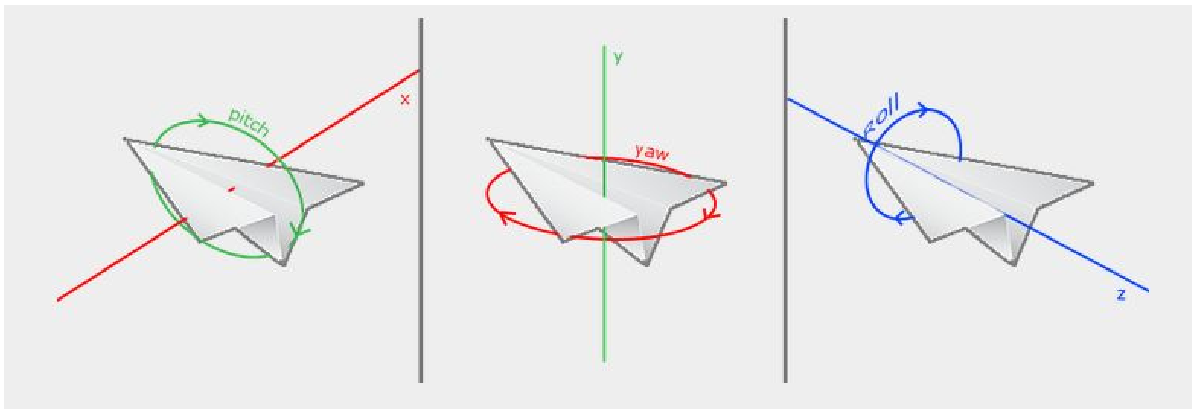


Fig. Camera space

Orientation Theory

Euler Angles

- Defined by Leonhard Euler somewhere in the 1700s.
- 3 angles: Pitch, Yaw and Roll
- Used to make movements of mouse



- Fig. Pitch, Yaw and Roll

Lighting Theory

For the Phong Lighting Model we’re using in our project, following components are required:

- **Ambient lighting**: effect of ambience on the color of the object
- **Diffuse lighting**: simulates the directional impact a light object has on an object. The more a part of an object faces the light source, the brighter it becomes.
- **Specular lighting**: simulates the bright spot of a light that appears on shiny objects. Specular highlights are more inclined to the color of the light than the color of the object.

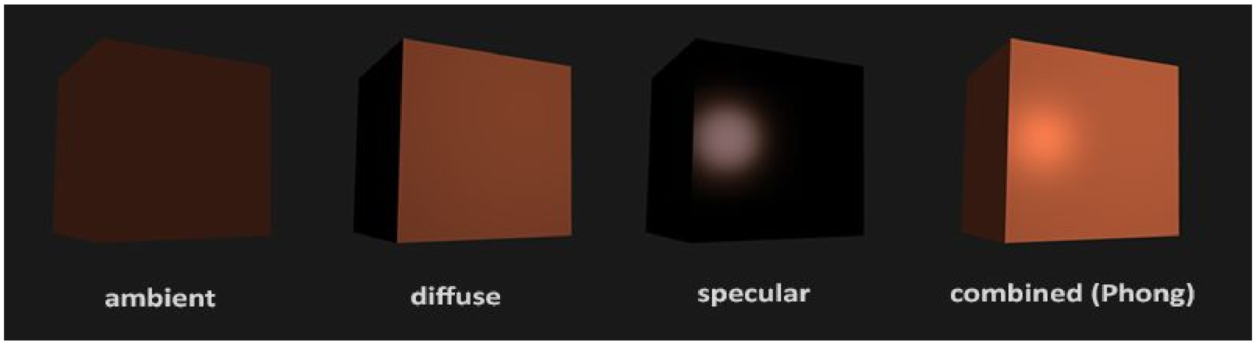
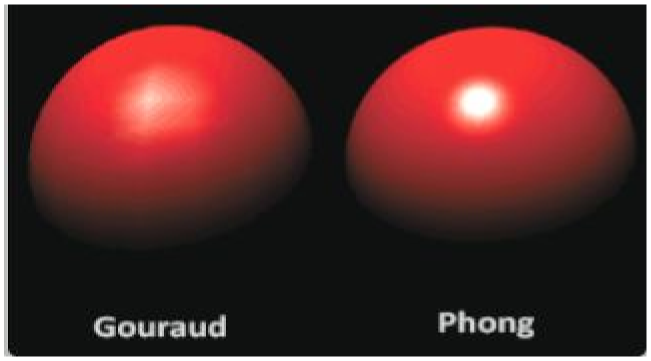


Fig. Phong Lighting Model

Finally, it’s implemented as: **result = (ambient + diffuse + specular) * objectColor;**

Gouraud:- lighting model in vertex shader(less vertices)->unrealistic

Phong:- lighting model in fragment shader(more vertices)->realistic



METHODOLOGY

For the implementation of our project, as required, we used the following resources.

- Blender (for our model designing and animation)
- OpenGL (for loading the Blender designed model into our program)

- Resources used along with OpenGL:
    - SOIL(Simple OpenGL Image Loader)
    - GLFW(Graphics Library Framework)
    - GLEW(OpenGL Extension Wrangler Library)
    - GLM(Graphics Library Mathematics)
    - Assimp(Open Asset Import Library)

Some hints about how the whole system works:

- **.blend** extension file generated by Blender is converted into .obj file and a .mtl file is generated along with it.
- **.obj** has the vertex positions, texture coordinates, normals and faces of the object
- **.mtl** has separate components of the Blender model with the texture file associated with it that helps in providing texture in OpenGL
- In OpenGL, the whole object and its attributes are stored as **Vertex Buffer Object(VBO)** and **Vertex Array Object(VAO)**
- The position , texture and lighting properties are mentioned in the **shader files** ,generally two types of shader files are used: **vertex_shader** and **fragment_shader**
- Vertex Shader contains the position of the object while fragment shader deals with properties like color and lighting , so that they can be processed in the GPU separately.

- GLM handles the mathematical operations and with the mathematics, modelling and graphics algorithms, finally a model can be viewed and navigated on the screen.

3.1 Workflow

Step 1: Start

Step 2: Models of spring season was first designed on a paper using pencils

Step 3: Different types of textures and planes were used to design the model in blender

Step 4: Spring season model was created at first and the same model was modified for different seasons

Step 5: All the 4 seasons were designed in Blender

Step 6: Codes for lighting, shading and loading the blender model in OpenGL was written

Step 7: The model was loaded in the OpenGL

Step 8: Necessary lightings and shading were adjusted furthermore to make the model more realistic

Step 9: The final model was rendered

Step 10: End

FINAL RESULTS

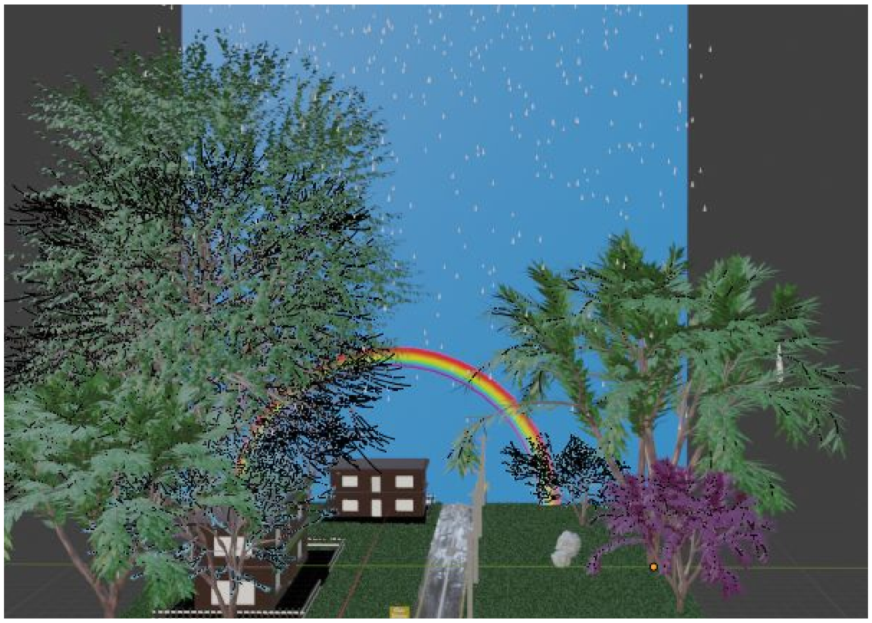


Fig. :- Summer Season

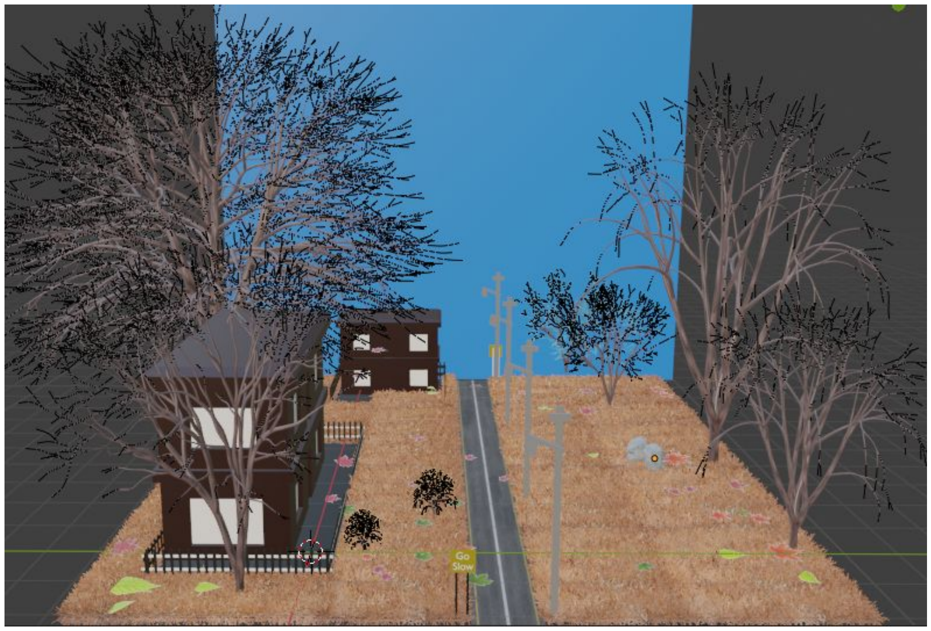


Fig:- Autumn Season

Fig: Winter Season

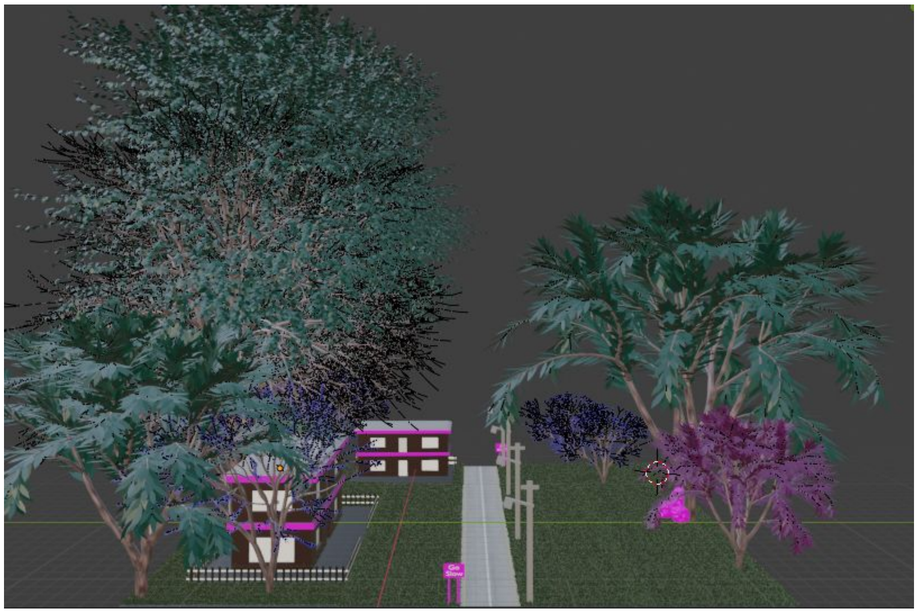


Fig: Spring Season

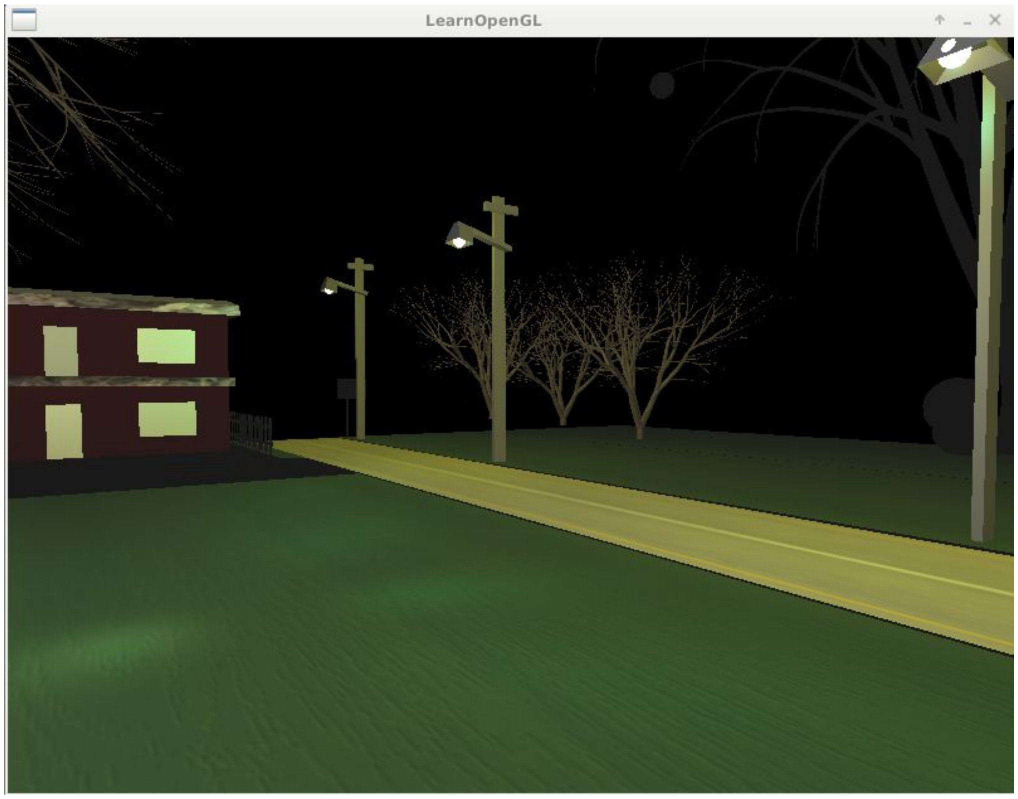


Fig. Night Mode

PROBLEMS FACED AND SOLUTIONS

While doing anything there comes the problems that have to be prioritised and should be taken care of for the successful accomplishment of the task. It will also help the reader not to go through the same problem learning from it.

During the project, we as a team had difficulties while collaborating in the coding stage as we were unfamiliar with the coding tools. We had to take a lot of support from the internet content to make it finally possible to load the blender-made objects into the OpenGL platform.

We also had to cope with the problem of applying theoretical concepts in programming. Designing some of the models took quite a lot of effort.A lot of problems had arose during the 3D modelling and texturing of the objects during the project and youtube tutorial became the savior at such a state. As the project went on, we got comfortable with the collaborating tools. We only learned the OpenGL as per required in the project rather than learning all the use of the whole library.

LIMITATIONS AND FUTURE ENHANCEMENTS

There is always room for improvement in everything we do. Our project certainly has a lot of limitations and improvements areas. Our project is not a complete embodiment of all the graphics concepts. The project lacks the perfect modelling which is the area that has to be focused on if one is thinking about taking it to the production level.

We haven't used much of the animations for our project. Only the animations can be seen in the rainy seasons for rain effect and the winter seasons for the snowfall effect, that’s limited to only in the Blender as we couldn’t implement the animation part in our OpenGL code. Shadows mapping is missing too. Advanced algorithms for generating finer filtered output are to be implemented as well. We’ll definitely add up on our project in the future.

CONCLUSION

Hence, with the completion of the project, a 3D environment of 4 seasons can be simulated. As a whole, the project was a good learning experience and we came face to face with practically applicable aspects of engineering which may guide us in developing professional projects in near future.

Meanwhile, it would provide a good basis for us, the programmers to work on big projects in near future. But most importantly, we got familiarized with concepts of computer graphics and its application in the technological and game development industry.

REFERENCES

- Hearb, Donal. " Computer Graphics C Version ". Pearson Education. 2008 Baral, Dayasagar.
- "The Secrets of Object Oriented Programming in C++". Bhudi puran Prakasha. November 2010.
- Deitel, Paul." C++ How to Program". Prentice Hall. 2012
- Bhatta, Ram. "A Text Book of Object Oriented Programming in C++", Alliance Publication. 2018.
- https://learnopengl.com/
- https://www.opengl-tutorial.org/
- https://www.youtube.com/channel/UCQ-W1KE9EYfdxhL6S4twUNw