GROUP - 15

Indian Institute of Technology Mandi

DESIGN PRACTICUM
Final Project Report

CINEMATIC VISUALIZATION USING OPENCASCADE

IC-102P

Mentor name	Dr. Jinesh Machchhar
Team members	Yash Chaudhary b22146@students.iitmandi.ac.in Computer Science Satyam Kumar b22236@students.iitmandi.ac.in Computer Science Anusha Tiwari b22089@students.iitmandi.ac.in Computer Science Dheeravath Madhu b22100@students.iitmandi.ac.in Computer Science Ankit Yadav b22195@students.iitmandi.ac.in Data Science & Engineering Prisha Singh b22168@students.iitmandi.ac.in Data Science & Engineering

Table of Contents

CONTENTS

INTRODUCTION

Opencascade is an open-source , full scale 3D geometry library. It consists of C++ classes grouped into packages. It is cross platform and easily accessible.

The field of computer-aided design (CAD) has revolutionized the way of designers and engineers conceptualize and create complex 3D models and parts. However, effectively communicating these intricate designs to stakeholders, clients or colleagues can often be challenging, especially when relying solely on static 2D renderings or technical documentation.

Through this project, we aimed to bridge this gap by writing a robust library for the OpenCASCADE geometric kernel which can be used for creating automated cinematic visualization of any input parts or CAD models.

The library we developed is comprehensive and is broadly divided into two parts :
The first part is for creating and saving 3D models and parts (along with a hierarchy if required) to the commonly used formats such as OBJ, STEP and STL.

The second part of the library is meant to be used for creating automated visualizations of the 3D parts generated by OpenCASCADE using advanced rendering techniques, all possible through a single script. As per the user's requirements, several kinds of adjustments can be made to the scene before rendering, and the output can be saved in the commonly used image and video formats such as PNG, JPG and MP4.

PROGRESS: WEEK 1

Learning about the Opencascade geometric kernel, compilation from source and setup on IDE.

The first week was spent in reading parts of the official Opencascade documentation, and then compiling the source code on our laptops using C++ compilers such as CMake and Microsoft's Visual C++ . This was followed by the setting up of an Opencascade project on our IDEs (Visual Studio / CLion) as instructed by our faculty mentor.

This was done successfully on all three major dektop operating systems: Windows, macOS and Linux.

Opencascade Testimonial -

BMW
Quick Mesh Application
«Thanks to our new application based on Open CASCADE Technology, BMW hopes to shorten this phase of the development process from several days to several hours.»

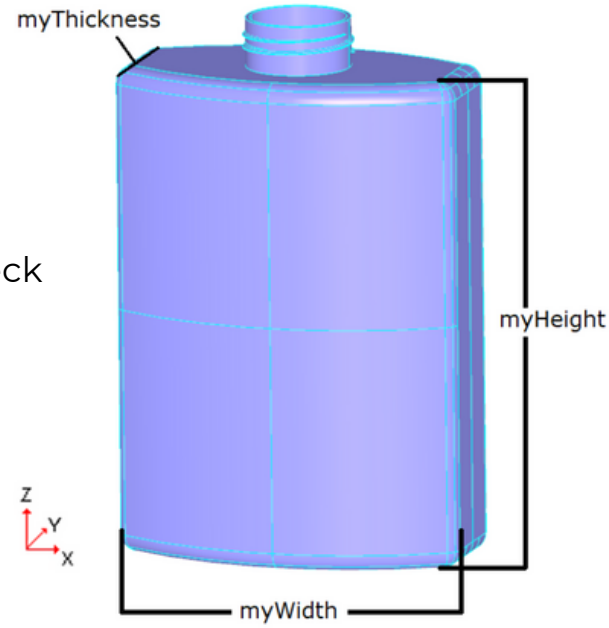
Peter Hoff
Project Manager

PROGRESS: WEEK 2

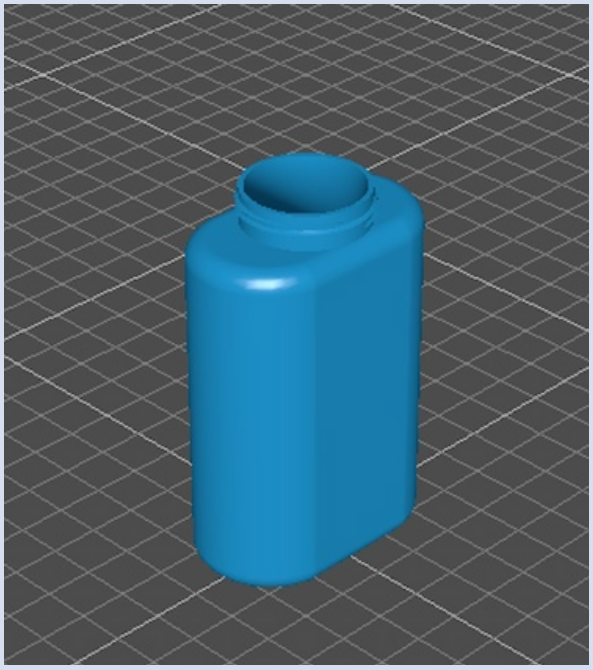
Visualization of shapes on Opencascade using Application Interactive Services (AIS)

The modeling requires four steps:

- build the bottle's Profile
- build the bottle's Body
- build the Threading on the bottle's neck
- build the result compound



The result we got after running the executable file.



PROGRESS: WEEK 3

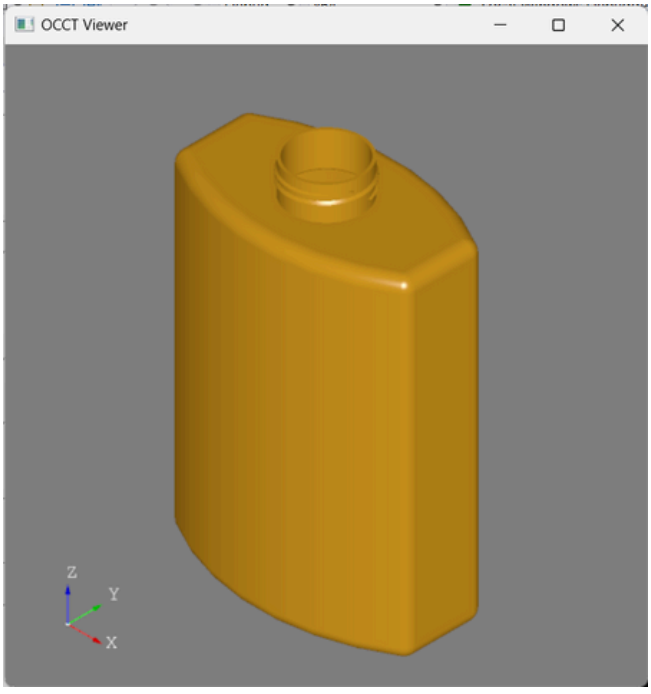
Reading and Writing STEP Files in Opencascade

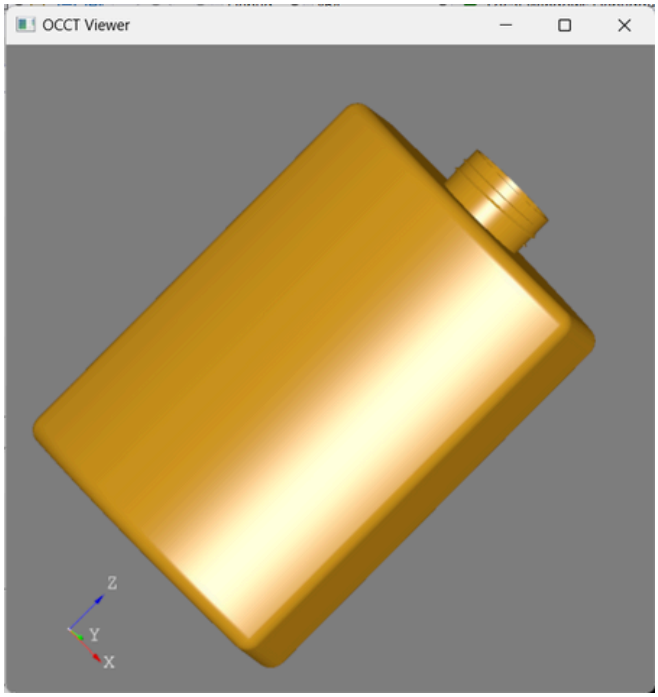
```cpp
int main(int argc, char** argv)
{
    // Define dimensions for the bottle shape
    Standard_Real width = 250.0;
    Standard_Real height = 420.0;
    Standard_Real thickness = 90.0;
    TopoDS_Shape bottleShape = MakeBottle(width, height, thickness);
    STEPControl_Writer stepWriter;
    // Transfer the solid shape to the STEP writer
    stepWriter.Transfer(bottleShape, STEPControl_ManifoldSolidBrep);

    // Specify the filename for the STEP file
    std::string fileName = "bottle.stp";

    // Write the STEP file
    stepWriter.Write(fileName.c_str());
    std::cout << "STEP file saved successfully.\n";

    // Read the STEP file
    STEPControl_Reader reader;
    reader.ReadFile(fileName.c_str());
    reader.TransferRoots();
    TopoDS_Shape result = reader.OneShape();
    std::cout << "STEP file read successfully.\n";
    GlfwOcctView anApp;

  try
  {
    anApp.run(result);
  }
  catch (const std::runtime_error& theError)
  {
    std::cerr << theError.what() << std::endl;
    return EXIT_FAILURE;
  }

    return 0;
}
```

PROGRESS: WEEK 4

Rotation and Translation of shapes

```
    aWindow->Map();
    myView->Redraw();
}

void rotateAfterDelay(int delay) {
    std::this_thread::sleep_for(std::chrono::milliseconds(delay));
    myView->Rotate(V3d_Z, M_PI/2.0, Standard_True);
}

void SaveSnapshot(std::string filename) {
    if (myView.IsNull())
        return;

    // Provide dimensions for the image
    int width = 1000;
    int height = 1000;

    // Create a new pixmap with the specified dimensions and RGB format
    Image_PixMap aPixmap;
    aPixmap.InitZero3D(Image_Format_RGB, NCollection_Vec3<Standard_Size>(width, height, 1));

    // Render the view onto the pixmap
    V3d_ImageDumpOptions aOptions;
    myView->ToPixMap(aPixmap, aOptions);

    // Convert Image_PixMap to OpenCV Mat
    cv::Mat img(height, width, CV_8UC3, (void*)aPixmap.Data(), cv::Mat::AUTO_STEP);

    // OpenCV saves the image
    cv::imwrite(filename, img);
}
```

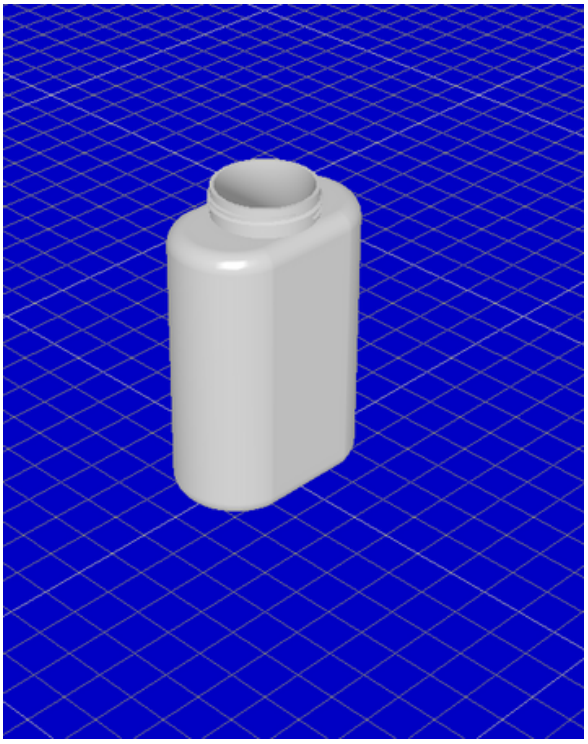


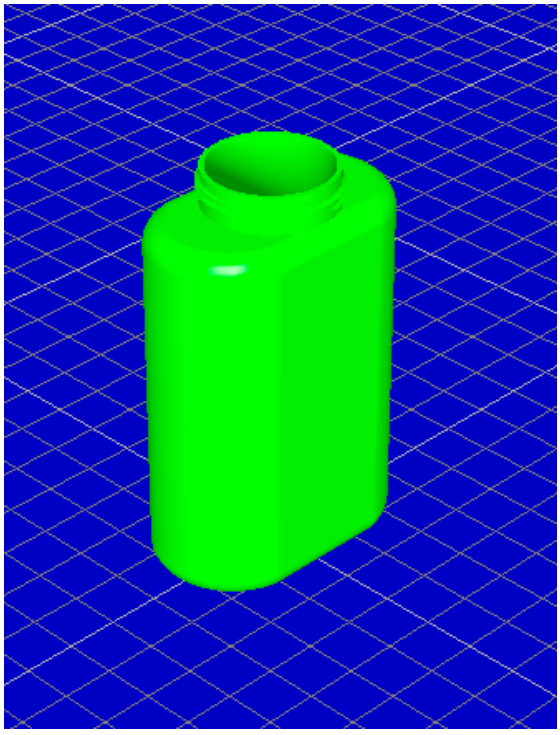


PROGRESS: WEEK 5

Rendering in Image and Video Formats (using Opencascade's renderer)

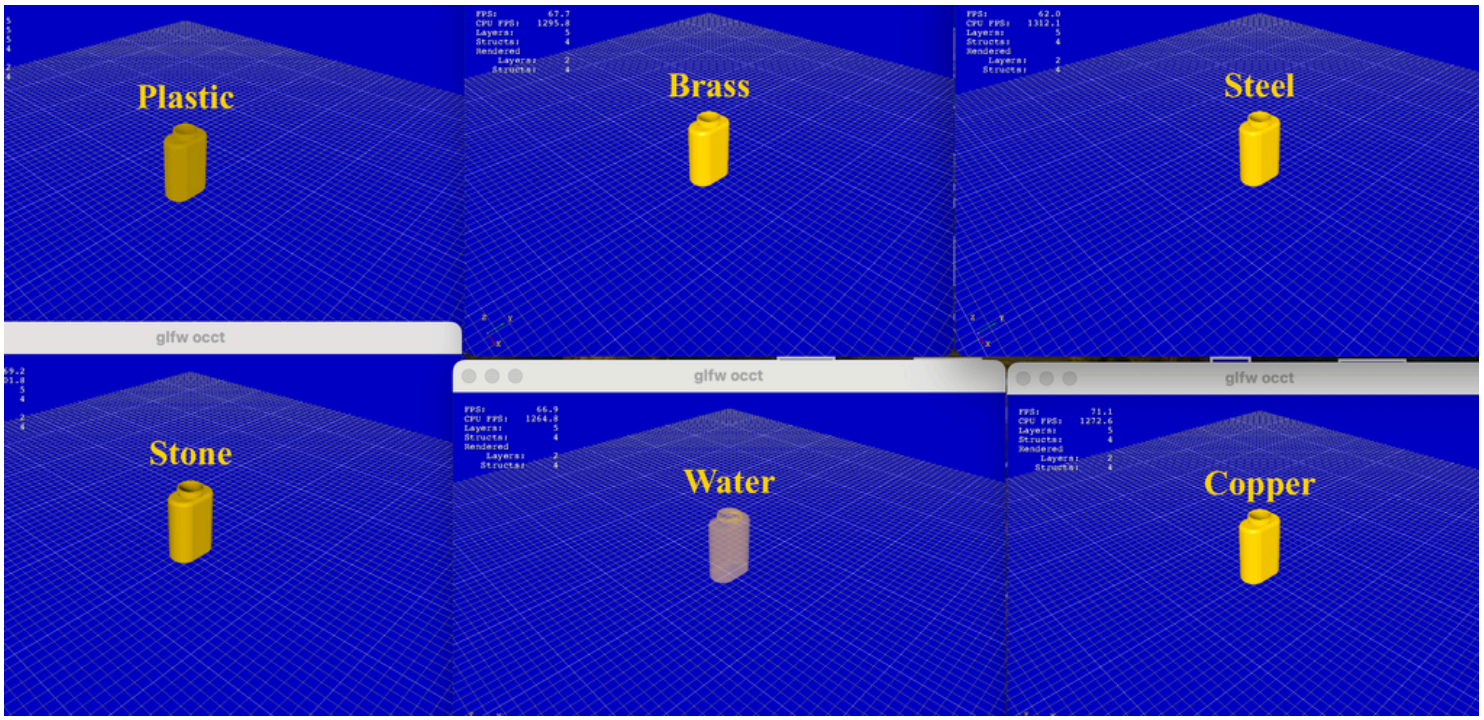
```
void SaveSnapshot(std::string filename) {
    if (myView.IsNull())
        return;

    // Provide dimensions for the image
    int width = 1000;
    int height = 1000;

    // Create a new pixmap with the specified dimensions and RGB format
    Image_PixMap aPixmap;
    aPixmap.InitZero3D(Image_Format_RGB, NCollection_Vec3<Standard_Size>(width, height, 1));

    // Render the view onto the pixmap
    V3d_ImageDumpOptions aOptions;
    myView->ToPixMap(aPixmap, aOptions);

    // Convert Image_PixMap to OpenCV Mat
    cv::Mat img(height, width, CV_8UC3, (void*)aPixmap.Data(), cv::Mat::AUTO_STEP);

    // OpenCV saves the image
    cv::imwrite(filename, img);
}
```





PROGRESS: WEEK 6 (AFTER MIDSEM)

Applying Various Materials to the Objects and controlling Transparency

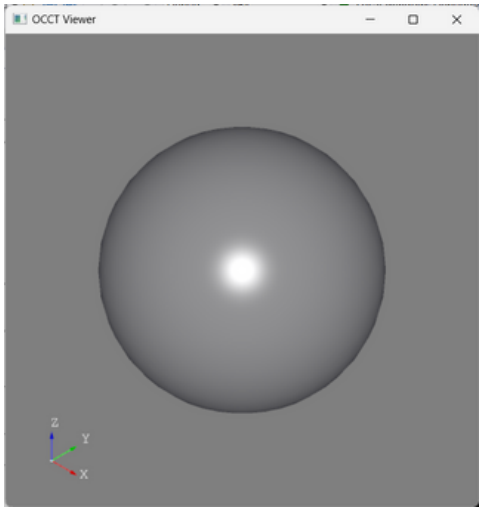


	A	B	C	D	E
1	Time (in ms)	**Shading**			
2	**Material**	**Phong**	**Gouraud**	**Pbr**	**Unlit**
3	None	432	472	388	415
4	GLASS	461	472	434	489
5	PLASTIC	395	444	484	403
6	SHINY_PLASTIC	447	412	461	446
7	CHROME	404	411	476	435
8	TRANSPARENT	396	419	499	433
9					

Analysis of time taken to apply different shadings to selected materials

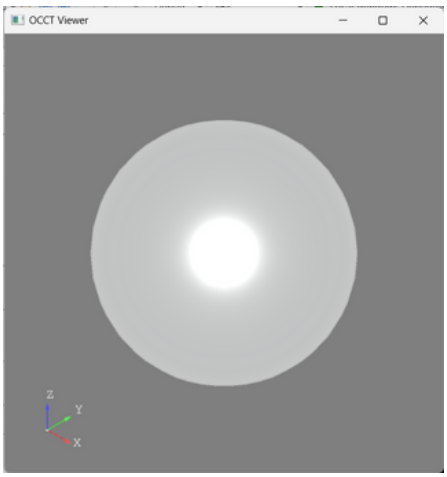
PROGRESS: WEEK 7

Applying Different Types of Shading (Phong, Gouraud, etc.)

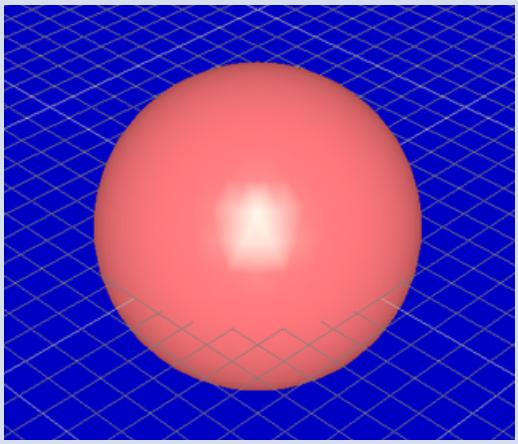


Phong

```
void SetShading(Graphic3d_TypeOfShadingModel shading) {
    myView->SetShadingModel(shading);
}

void SetShapeMaterial(Graphic3d_MaterialAspect aMaterialAspect, Standard_Real aTransparency)
{
    // Retrieve the displayed shape
    AIS_ListOfInteractive aList;
    myContext->DisplayedObjects(aList);

    // Iterate through displayed objects and set material for each shape
    for (AIS_ListIteratorOfListOfInteractive it(aList); it.More(); it.Next())
    {
        Handle(AIS_InteractiveObject) anObj = it.Value();
        if (anObj->IsKind(STANDARD_TYPE(AIS_Shape)))
        {
            Handle(AIS_Shape) aShape = Handle(AIS_Shape)::DownCast(anObj);
            if (!aShape.IsNull())
            {
                // Set material aspect
                aShape->SetMaterial(aMaterialAspect);

                // Set transparency
                aShape->SetTransparency(aTransparency);
            }
        }
    }

    // Redraw the view
    myView->Redraw();
}
```



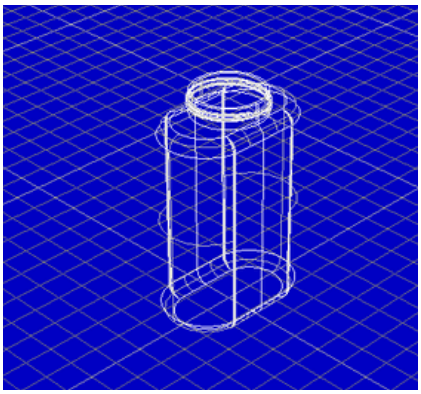
PBR

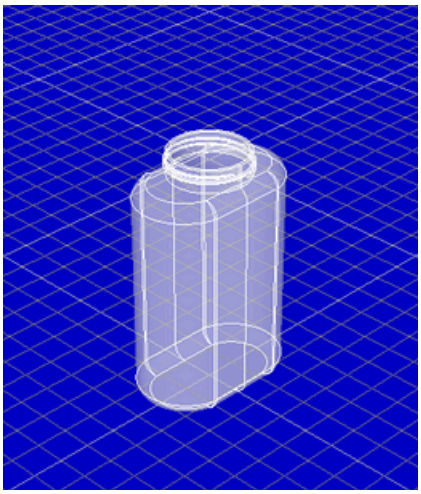


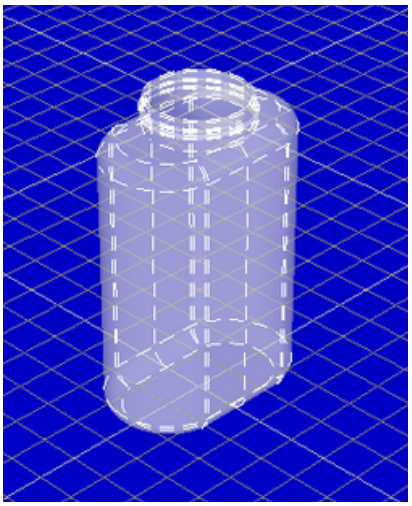
Gouraud

PROGRESS: WEEK 8

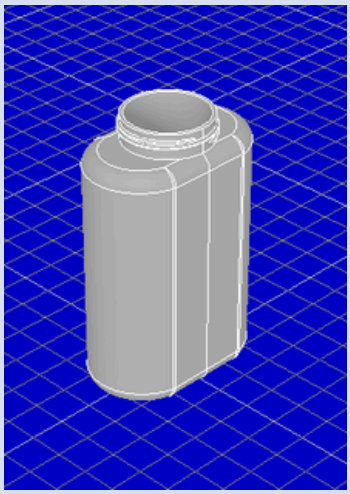
Controlling the edge thickness of each edge of an object and trying different types of edges

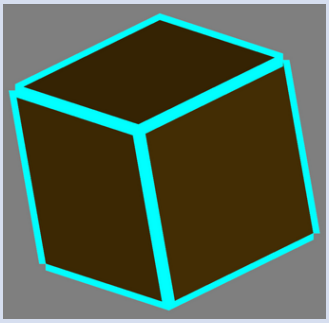






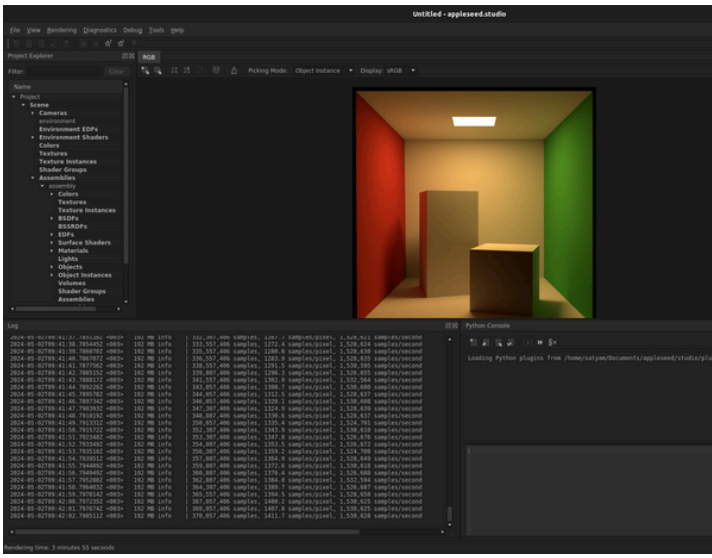
```
void GlfwOcctView::initDemoScene() {
    if (myContext.IsNull()) {
        return;
    }

    myView->TriedronDisplay(Aspect_TOTP_LEFT_LOWER, Quantity_NOC_GOLD, 0.08, V3d_WIREFRAME);


    TopoDS_Shape aBottleShape = MakeBottle(30, 70, 50);
    Handle(AIS_Shape) aBottle = new AIS_Shape(aBottleShape);

    // Ensure the bottle is displayed shaded
    myContext->Display(aBottle, AIS_Shaded, 0, false);

    // Set the display mode to shaded for the bottle
    myContext->SetDisplayMode(aBottle, AIS_Shaded, Standard_True);


    Handle(Prs3d_Drawer) aDrawer = aBottle->Attributes();
    Handle(Prs3d_LineAspect) lineAspect = new Prs3d_LineAspect(Quantity_NOC_BLACK, Aspect_TOL_DASH, 9.0);
    aDrawer->SetLineAspect(lineAspect);


    aDrawer->SetFaceBoundaryDraw(true);
    aDrawer->SetFaceBoundaryAspect(new Prs3d_LineAspect(Quantity_NOC_BLACK, Aspect_TOL_DASH, 16.0));


    Graphic3d_MaterialAspect aMaterialAspect(Graphic3d_NOM_GLASS);
    aBottle->SetMaterial(aMaterialAspect);


    myContext->Redisplay(aBottle, true);
}
```

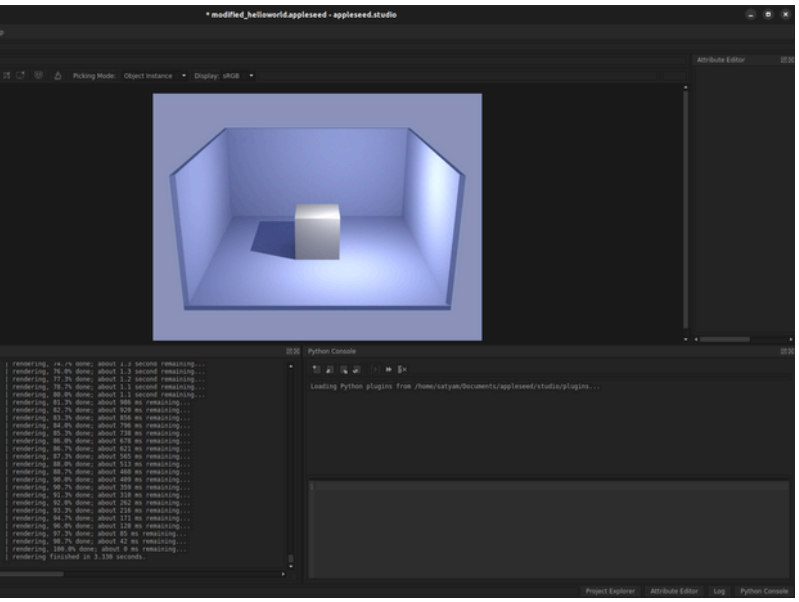


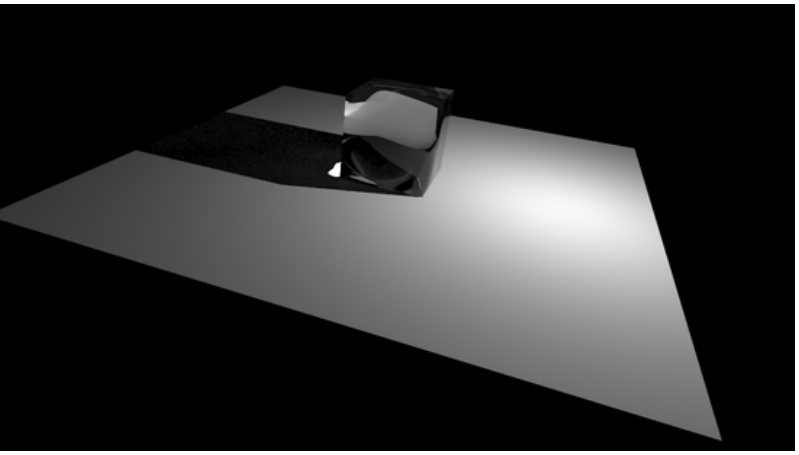


PROGRESS: WEEK 9

Rendering using Appleseed/Cycles renderer



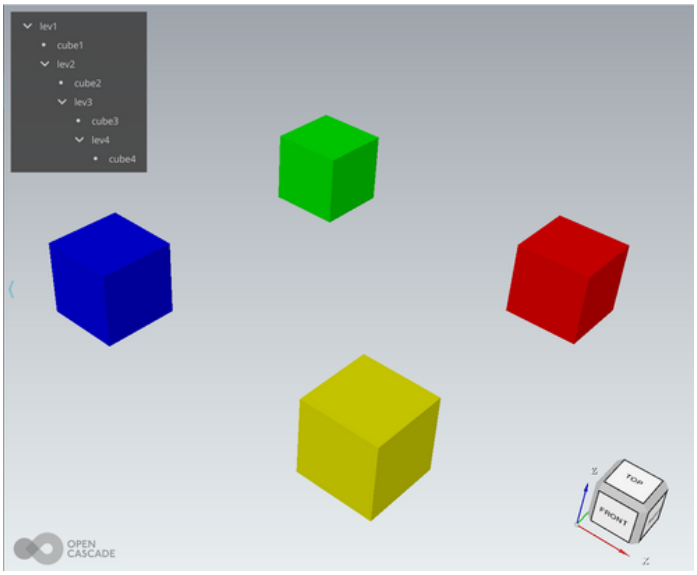




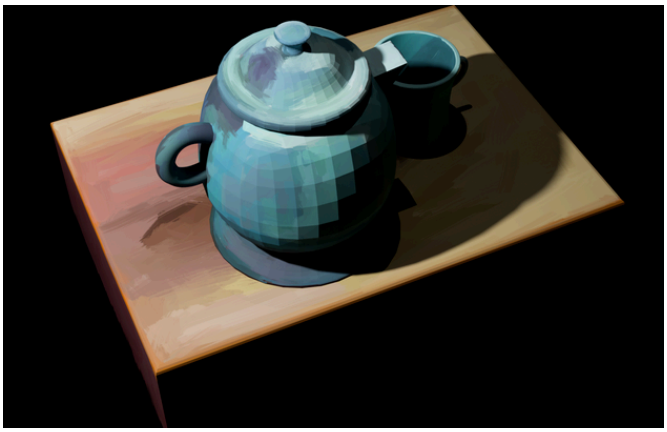
> Cycles only supports .xml files and Appleseed only supports .obj and .appleseed project files for rendering.

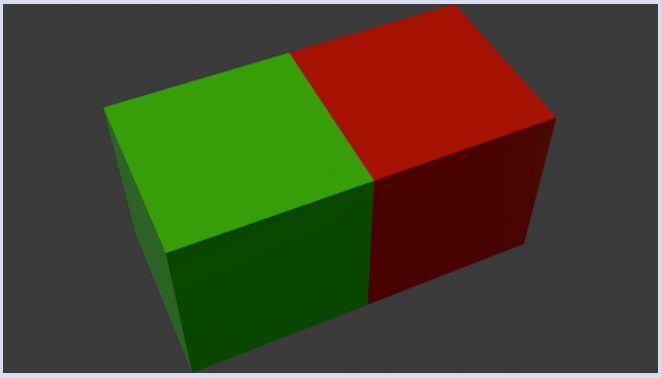
PROGRESS: WEEK 10

Rendering using the Blender Python API and hierarchy creation for objects.




```python
# Clear existing objects and materials
bpy.ops.object.select_all(action='DESELECT')
bpy.ops.object.select_by_type(type='MESH')
bpy.ops.object.delete(use_global=False)

# Import OBJ file with materials
bpy.ops.wm.obj_import(filepath=obj_file)

# Get the imported objects
imported_objects = bpy.context.selected_objects

# Set Cycles as the rendering engine
bpy.context.scene.render.engine = 'CYCLES'
```





VISION

For academicians, engineers, and other mathematics enthusiasts working in the field of topology and computer graphics, our project would act as an effective tool to solve various visualization and simulation-related problems. In addition to this, the library would be of great use in a variety of diverse fields like architecture, healthcare , defence and interior-designing.

01

Architecture

To review design proposals before the actual construction begins, we can first create real-life like simulations of different architectural spaces, and then start building. The ability to experiment with different materials and dimensions will give an apt preview of the final result. It will not only save time post construction but also help architects take better and more informed decisions.

02

Healthcare

Study of anatomy can be simplified using our library.We can use the 3D visualisation for surgery planning. We can also this technology for minimizing errors in any healthcare instrument that is developed. Such innovation has become crucial in healthcare.

03

Defence

Defence can be greatly benefitted as our library is capable of creating and rendering 3D objects. We can therefore simulate real life like evironments and visualize the battlefield strategies. It can give us an idea about the terrain, like the objects around , their shape and material , etc.

SAMPLE CODE

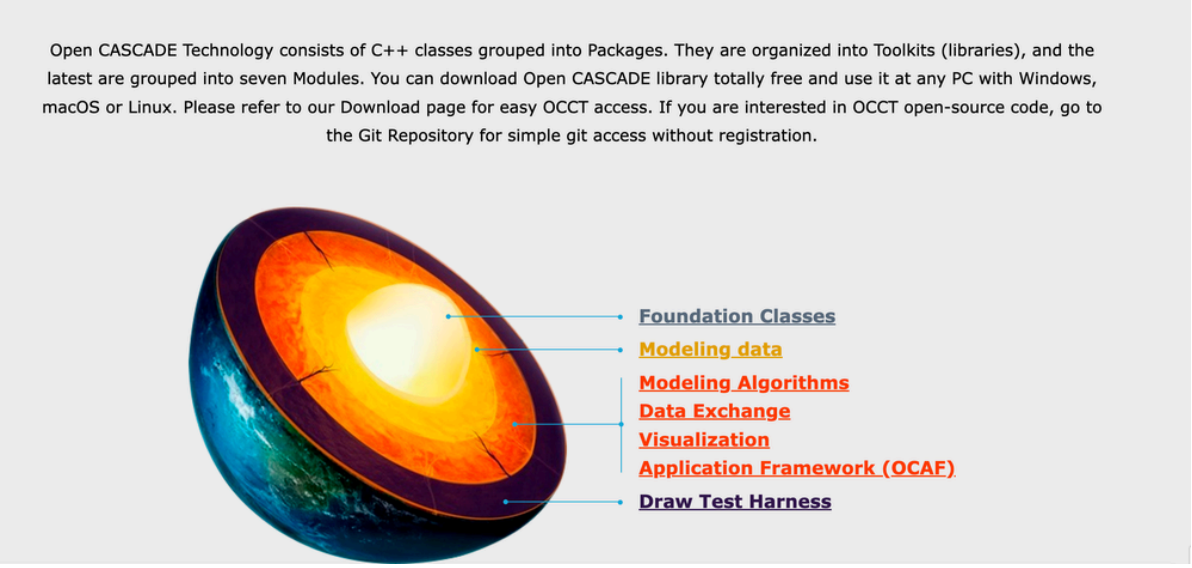
code for saving files ...

```
int main() {
    gp_Pnt points[] = {
            gp_Pnt(400, 400, 0),
            gp_Pnt(-400, 400, 0),
            gp_Pnt(-400, -400, 0),
            gp_Pnt(400, -400, 0)
    };
    BRepPrimAPI_MakeBox b1(gp_Pnt(0, 0, 0), 20, 20, 20);
    TopoDS_Shape box1 = b1.Shape();
    BRepPrimAPI_MakeBox b2(gp_Pnt(20, 0, 0), 20, 20, 20);
    TopoDS_Shape box2 = b2.Shape();
    BRepPrimAPI_MakeBox b3(points[2], 200, 200, 200);
    TopoDS_Shape box3 = b3.Shape();
    Assembly a;
    Part* p1 = a.addChild("main", "c1");
    p1->addShape(b1);
    Part* p2 = a.addChild("main", "c2");
    p2->addShape(b2);
    Part* p3 = a.addChild("main:c1", "c3");
    p3->addShape(b3);
    Part* main = a.getPart("main");
    main->addColor(Quantity_Color(1, 0, 0, Quantity_TOC_RGB));
    p2->addColor(Quantity_Color(0, 1, 0, Quantity_TOC_RGB));
    a.saveAsSTEP("boxes.step");
    a.saveAsSTEP("boxes.obj");
    return 0;
}
```

Class	Functionality	Remarks
Class Part	To handle individual objects in the complete structure	• It can store colors, names, geometry and any required property. • It also handle its position in the structure.
Class Assembly	To handle the structure of the Part objects	• It handles the components in a hierarchy. • It can also save files based on the hierarchy.

REFERENCES

1. https://www.opencascade.com/
2. https://dev.opencascade.org/
3. https://wiki.freecad.org/OpenCASCADE
4. https://docs.blender.org/api/current/index.html
5. https://pypi.org/project/bpy/



ACKNOWLEDGEMENTS

The successful completion of this project would not have been possible without the constant support and guidance of our Faculty Mentor, Dr. Jinesh Machchhar. He ensured that we made consistent progress throughout the semester and provided us with useful references and insights whenever required.

We also thank Dr. Gajendra Singh for giving us this wonderful opportunity.