Final Project Report: Design and 3D Modeling

Abdelmounaim Salouani

April 30, 2025

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Course: Design and 3D Modeling
Instructor: Vendrell Vidal, Eduardo
Institution: ETSINF-UPV

Email: abdelmounaim.chakib.salouani@gmail.com
asaloua@teleco.upv.es

1 Introduction

The primary goal of this project was to design and develop a system for 3D modeling and object creation using Python and Blender. The project focused on generating `.obj` files for static shapes, positioning these shapes dynamically in space, and creating Blender Python scripts to automate the modeling process.

The theoretical foundation of the project revolved around computational geometry, 3D modeling principles, and the use of Python libraries for automation and visualization. The project aimed to provide a user-friendly interface and tools for generating and manipulating 3D objects programmatically.

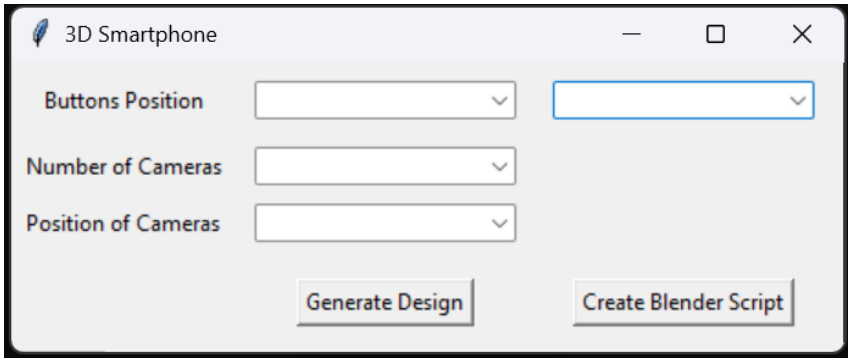


Figure 1: example of gui used to get user's input

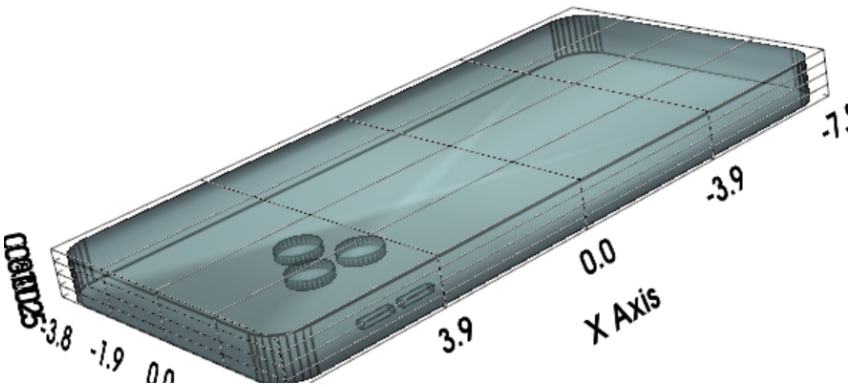


Figure 2: example of pre-visualization

2 Development

The project followed an agile development methodology, progressing through several iterative phases:

2.1 Planning

The main idea was to generate appropriate `.obj` components while addressing face orientation issues. The program was designed to be exposed for external use through an API, inspired by Python's library structure.

2.2 Resources

The project utilized various tools and libraries:

- **Obj file creation**: Math and NumPy for computational geometry.
- **Visualization**: PyVista, MeshLab, and Blender for rendering and editing.
- **Blender scripting**: The `bpy` library for automating tasks in Blender.
- **GUI**: Tkinter for creating a user interface.
- **Execution automation**: Shell and batch scripting for seamless execution.

2.3 Phases of Development

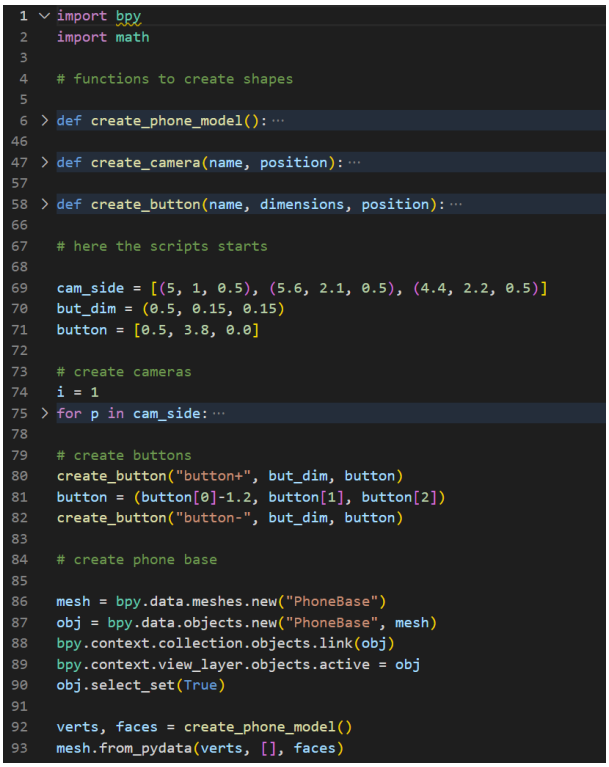
The project was developed in four main phases:

1. **Version 1**: Basic `.obj` file production and GUI template creation.
2. **Version 2**: Addition of curves and merging of shapes.
3. **Version 3**: Development of the Blender script for dynamic modeling.
4. **Version 4**: Integration of all code components and incorporation of dynamic aspects.

3 Results

The final deliverables of the project included:

- A live demonstration of the application.
- A user manual detailing the usage of the system (included in the final presentation and down below).
- Execution scripts (`execute.sh` for Linux and `execute.bat` for Windows) to run the application. These scripts allowed users to:
  - Generate `.obj` files.
  - Create Blender Python scripts.
  - Pre-visualize the output in the command line.
- The generated `.obj` files and Blender scripts could then be used for further 3D modeling.



```python
import bpy
import math

# functions to create shapes

def create_phone_model():

def create_camera(name, position):

def create_button(name, dimensions, position):

# here the scripts starts

cam_side = [(5, 1, 0.5), (5.6, 2.1, 0.5), (4.4, 2.2, 0.5)]
but_dim = (0.5, 0.15, 0.15)
button = [0.5, 3.8, 0.0]

# create cameras
i = 1
for p in cam_side:

# create buttons
create_button("button+", but_dim, button)
button = (button[0]-1.2, button[1], button[2])
create_button("button-", but_dim, button)

# create phone base

mesh = bpy.data.meshes.new("PhoneBase")
obj = bpy.data.objects.new("PhoneBase", mesh)
bpy.context.collection.objects.link(obj)
bpy.context.view_layer.objects.active = obj
obj.select_set(True)

verts, faces = create_phone_model()
mesh.from_pydata(verts, [], faces)
```

Figure 3: blender's template

4 Performance

The project was primarily developed by Abdelmounaim Salouani. The tasks were divided as follows:

- **Obj file creation**: 30%
- **Visualization and rendering**: 25%
- **Blender scripting**: 25%
- **GUI and automation**: 20%

As the sole contributor, Abdelmounaim completed 100% of the work, demonstrating a comprehensive understanding of the tools and methodologies used.

From a technical perspective, the application demonstrated robust performance in generating and handling 3D objects. The use of Python libraries such as NumPy and PyVista ensured efficient computation and visualization of 3D shapes. The modular design of the code allowed for seamless integration of new features, such as dynamic shape positioning and Blender script generation.

The execution scripts (`execute.sh` and `execute.bat`) provided a smooth user experience, automating the process of object creation and script generation. The application was tested on multiple platforms such as meshlab, and its performance was consistent, with minimal resource consumption and fast processing times for complex 3D models.

5 References

The following resources and tools were consulted and utilized during the project:

- Python libraries: NumPy, PyVista, Tkinter, and `bpy`.
- Visualization tools: MeshLab and Blender.
- Documentation and tutorials for Blender scripting and Python-based 3D modeling.
- Shell and batch scripting guides for shell scripting and batch scripting.