HuBMAP (Human BioMolecular Atlas Program)

Jyotiprakash Panda
Pritish Thombare

Spring ‘24

Previous Work

- In the proposed project, our objective was to streamline the process of image simplification using the CGAL library.
- To achieve this, we initially converted the images from the GLB format to the OFF format. This conversion was essential because the CGAL library requires images in the OFF format to perform the simplification process effectively.
- By doing so, we ensured compatibility with CGAL's tools and algorithms, enabling us to efficiently simplify the images as intended using CGAL's triangulated mesh simplification algorithm.

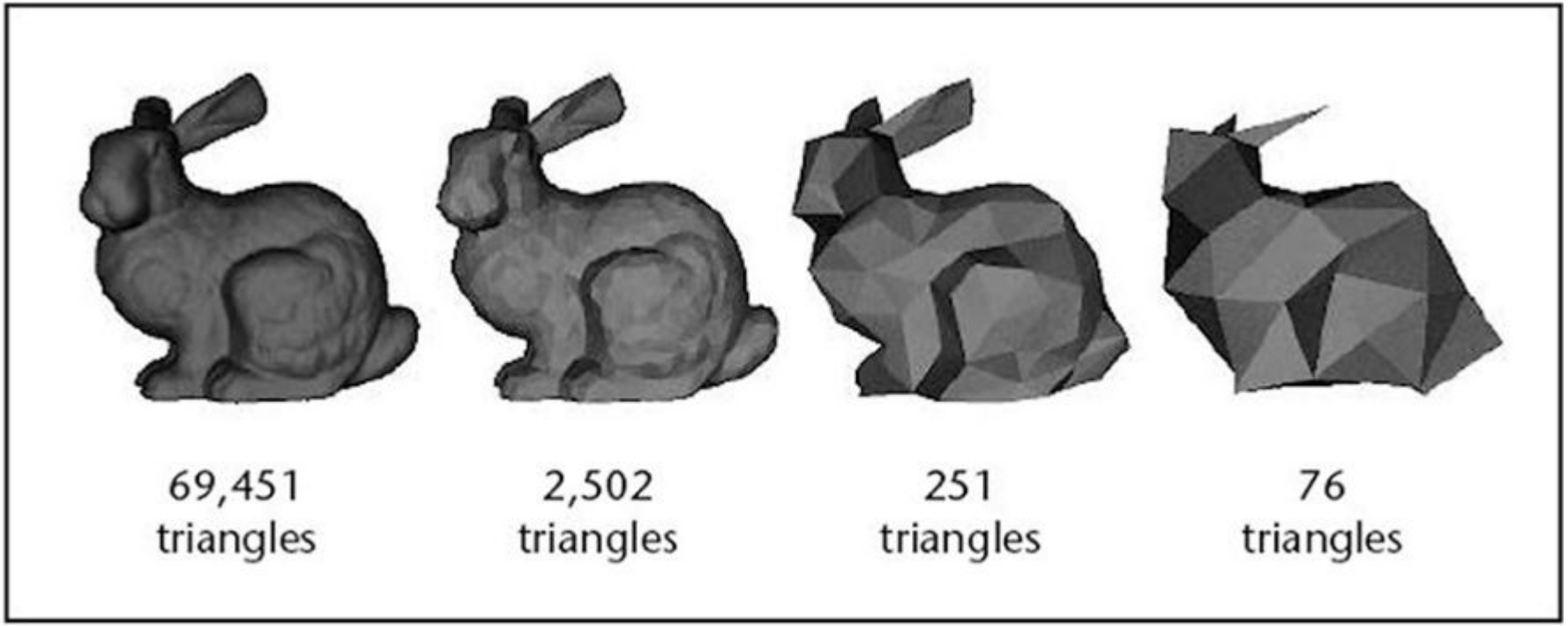
Outline of the Current work

Adopting Python for Mesh Simplification
Transition: Shift from C++ CGAL to Blender's Python libraries for image processing and mesh simplification, aiming for increased flexibility and ease of use.

Workflow Setup:
Install and configure Blender to integrate with our existing pipeline, leveraging its advanced capabilities for improved processing efficiency.

Augment Blender's script:
Add features to the existing Blender script to generate LODs of glb files conveniently.

Level of Detail



Exploring Blender

- Local Blender setup.
- Installation of third party libraries in Blender e.g. Pandas
- Running the Blender python script in background.
- Adding command line arguments to the script.
- Adding function imports from another Python script.
- Implicit path handling for generating outputs.
- Running the script on sample glb files and confirming the outputs.

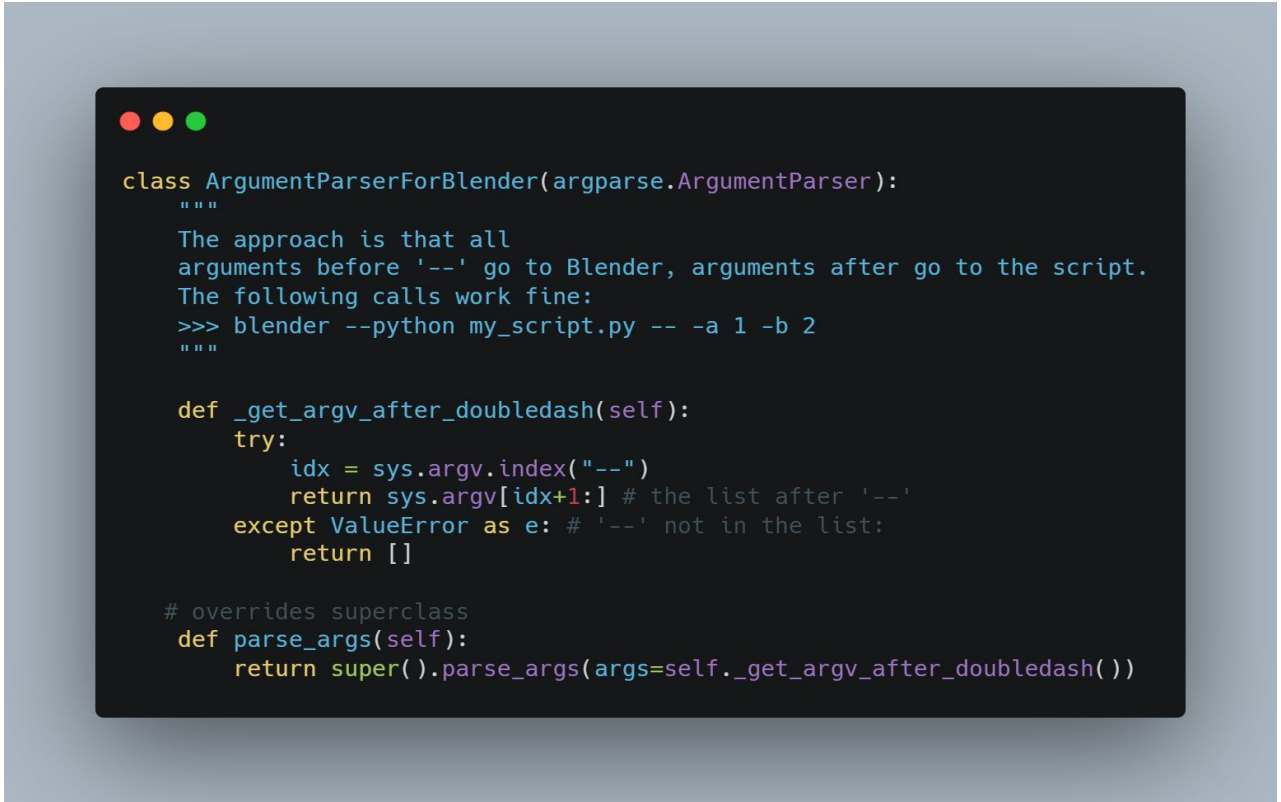
Run script in background

```
.\blender.exe --background --python 'C:\Users\pthombare\Documents\Blender\hubmap_copy.py' --
"C:\Users\pthombare\Documents\Blender\downloaded_organs\3d-vh-f-blood-vasculature.glb"
"C:\Users\pthombare\Documents\Blender\studio_small_01_4k.exr" -lod 20 100 -max_traingles 146556
```

Argument format

```
    parser = ArgumentParserForBlender()
    parser.add_argument('input_file', type=str, help='Input file path')
    parser.add_argument('hdr_file', type=str, help='HDR file path')
    parser.add_argument('-lod', nargs='+', type=int, help='List of LOD values')
    parser.add_argument('-max_triangles', nargs='*', type=int, default=[], help='List of Max triangle
values')
```

Argparse



```python
class ArgumentParserForBlender(argparse.ArgumentParser):
    """
    The approach is that all
    arguments before '--' go to Blender, arguments after go to the script.
    The following calls work fine:
    >>> blender --python my_script.py -- -a 1 -b 2
    """

    def _get_argv_after_doubledash(self):
        try:
            idx = sys.argv.index("--")
            return sys.argv[idx+1:] # the list after '--'
        except ValueError as e: # '--' not in the list:
            return []

   # overrides superclass
    def parse_args(self):
        return super().parse_args(args=self._get_argv_after_doubledash())
```

Code walkthrough

Thank you!