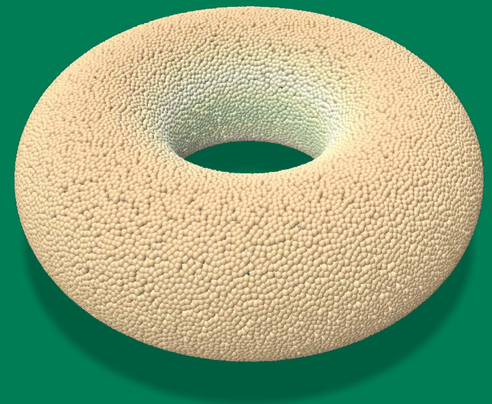
Exploring the Laplacian in Computer Graphics

Week 2

Crane He Chen

The Johns Hopkins University



2023 Fall

Logistic

- Next week is critical hands-on part. Make sure you bring a laptop.
- I will post videos about how to install CMake on MacOS/Windows/Linux.
- I want to ask if you are comfortable or not if I record this lecture and post it online.

    A. If yes, can you fill in this Student Information Release Authorization form for me?

    https://registrar.jhu.edu/guidelines-for-recording-class-meetings/

    B. If not, I will edit the video and delete the part with your voice.

C++ or Python?

My recommendation (in the context of computer graphics):

- Management/quick check of data

simpler

- Algorithms

flexibility to modify libraries

Visual Data

90% of the information processed by the brain is visual.

Image credits: Adobe image stock

Images

How are images captured?

Photo-realistic images

- Captured by camera
- Rendered by computer from 3D scene
- Generated by AI

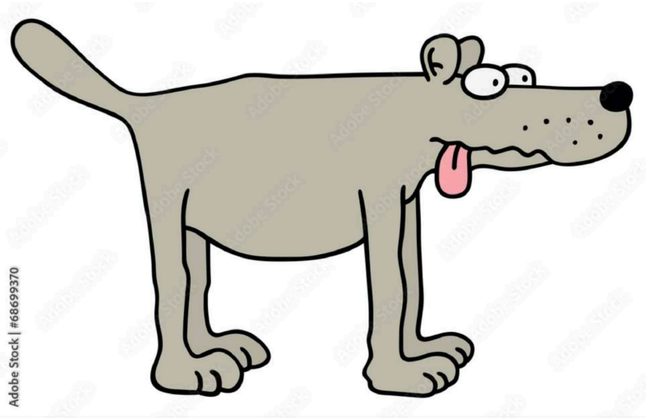
Image credits: my photography, Pixar's movie Pipper, Adobe image stock

Images

How are images captured?

Non photo-realistic images

- Created by Artist

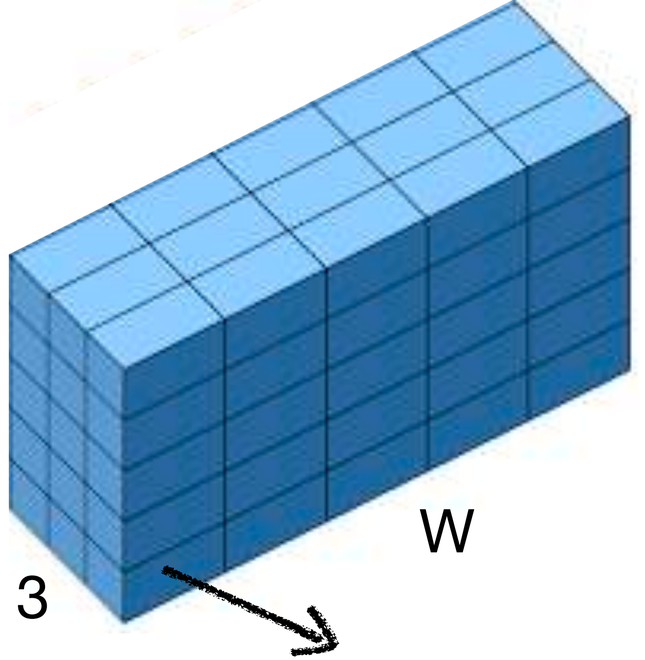


- Generated by AI

Image credits: Adobe image stock

Images

How are images stored in a computer?



- Imagine you have a cuboid, whose Wcm x Hcm x Dcm
- Now, stop thinking about cm, replace it with different unit called “pixel” for W and H, while D has no unit.
- This cuboid has an official name in the context of visual data, we call it “regular grid”
- WxH is the resolution of your image, 3 represents 3 channels, red, green, blue

Images

How do I access and visualize images from my code?

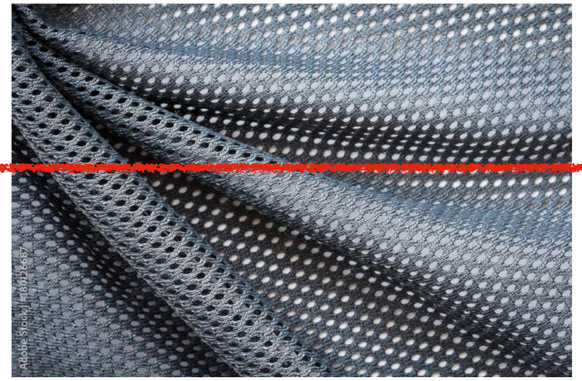
- The IO of image processing libraries, (e.g. pillow)


```
from PIL import Image

# Open an image file
image_path = "example.jpg"  # Replace with the path to your image file
image = Image.open(image_path)

# Get dimensions
width, height = image.size
print(f"Image Dimensions: Width = {width}, Height = {height}")

# Show the image
image.show()
```

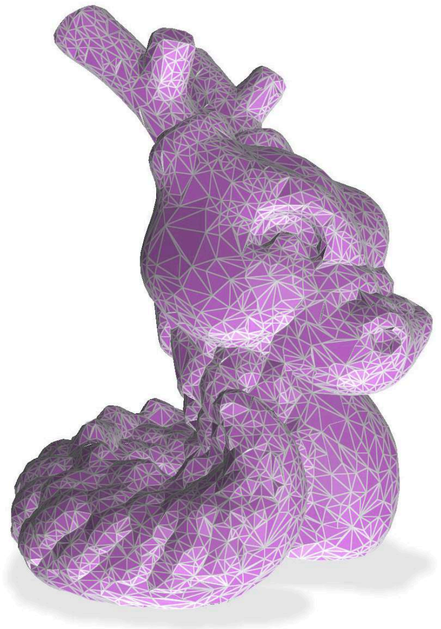
3D Models — Mesh

What's a mesh?

- ~~some sort of fabric with holes?~~



- a representation of 3D models



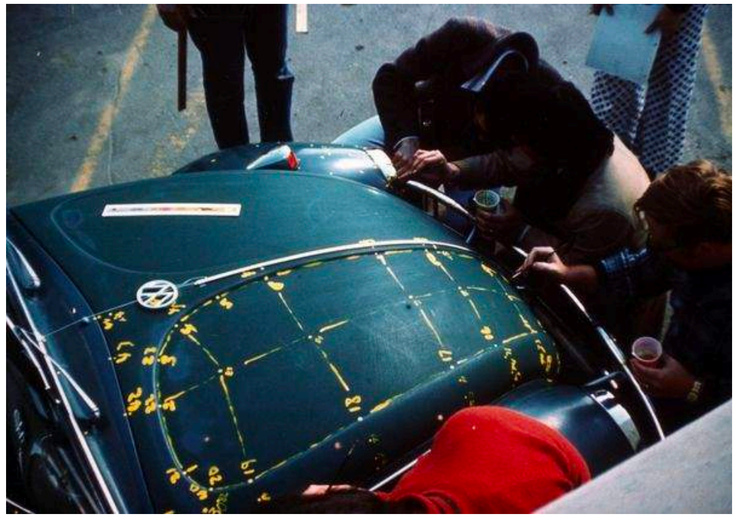
National Museum of Mathematics, NYC

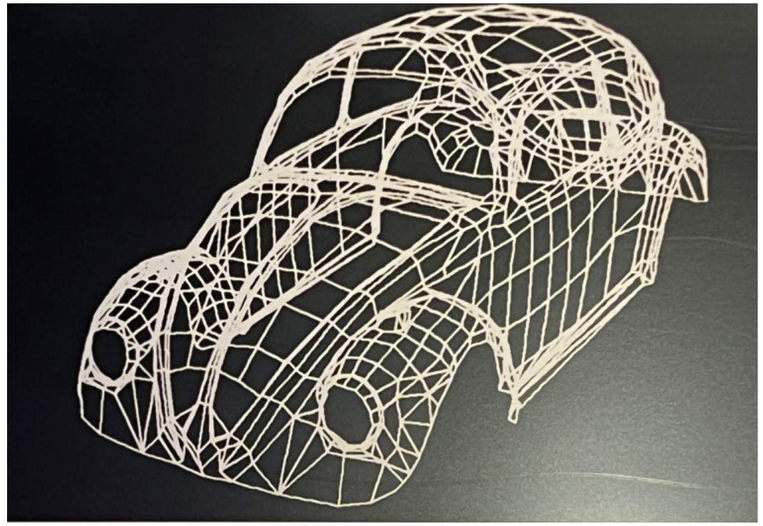
3D Models — Mesh

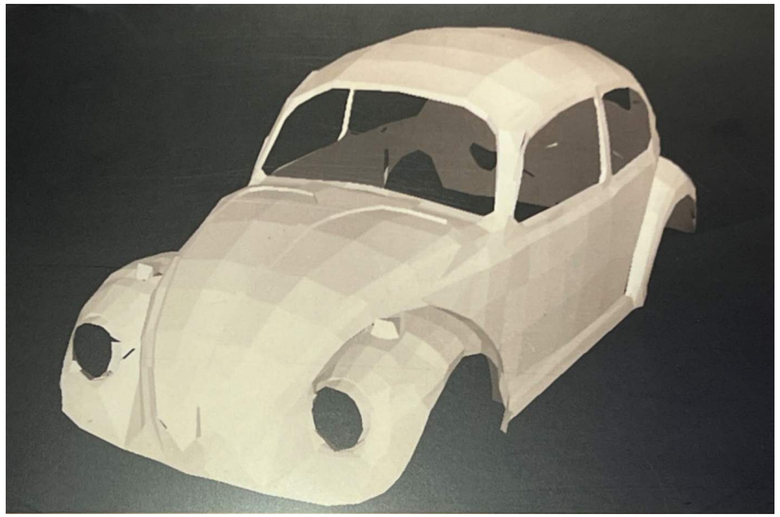
How are meshes captured?

- Created by Artists
- Reconstructed by algorithms
- What about the old days?

In 1972, University of Utah……







Sutherland's Volkswagen, Computer History Museum, Mountain View, CA

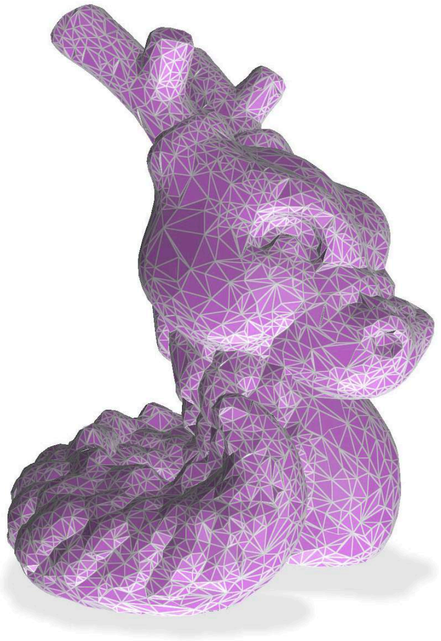
3D Models — Mesh

How are meshes stored in a computer?

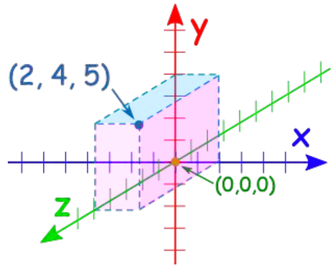
- For simplicity, let's think about triangle meshes now. Meaning, you are covering arbitrary surface with triangles, no seam, no overlapping

- For each triangle, how many vertices, edges, faces do you have?

- Now, think about this entire dragon you just tiled with many triangles, if you have "n" triangles, "f" faces. You are allowed to use Microsoft Excel to create tables. Can you think of a way to store this triangle on your computer?



Hint: you may have a coordinate

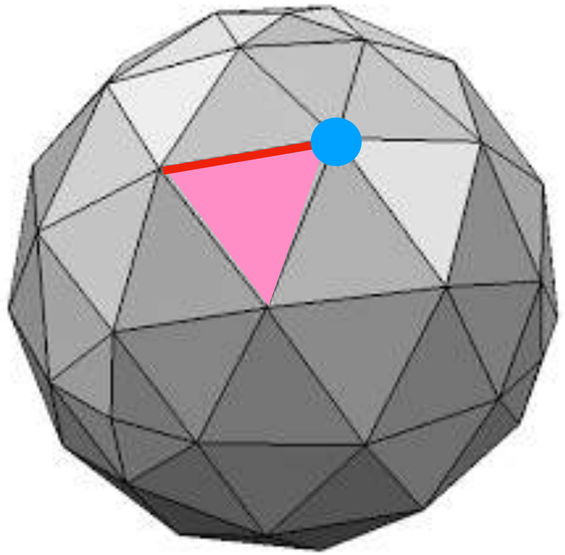


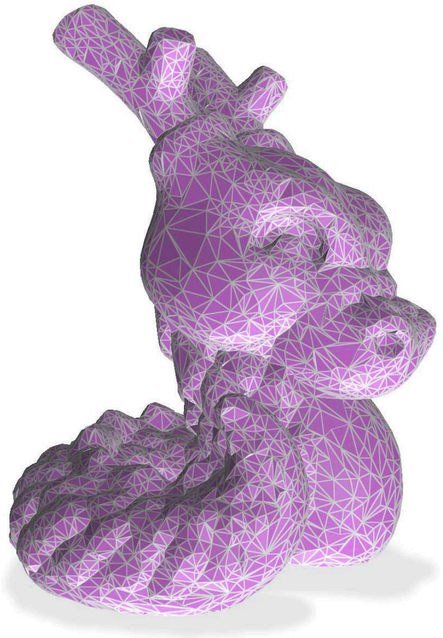
3D Models — Mesh

How are meshes stored in a computer?

Fundamental Components of a Mesh

- Vertices
- Faces
- Edges





Other data structures such as half-edge and winged-edge are out of the scope of this course.
But you can look up if interested.

3D Models — Mesh

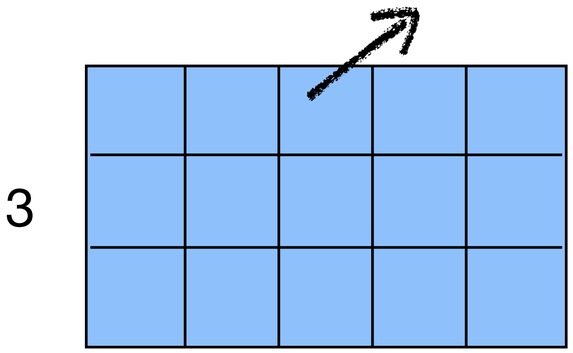
How are meshes stored in a computer?

There exists many different ways mesh can be stored. For now, let's learn one of the simplest (and most popular) way:

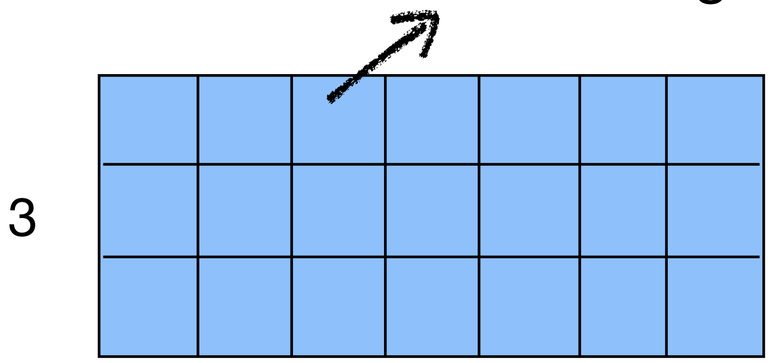
Face-Vertex Meshes

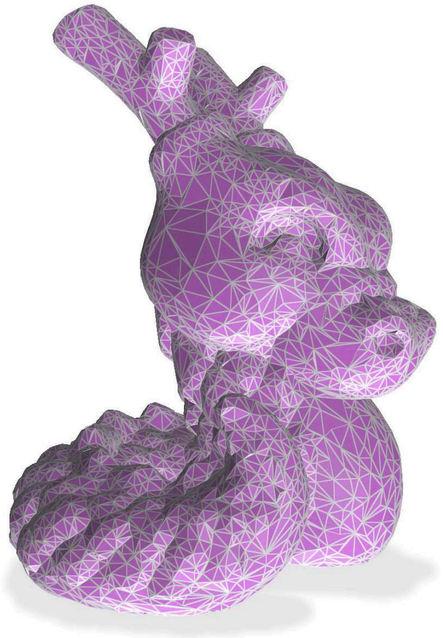
- Vertices = [(x1, y1, z1), (x2, y2, z2), …]
- Faces = [(i1, i2, i3), (i4, i5, i6), …]

vertex coordinates



indices of vertices constructing a face

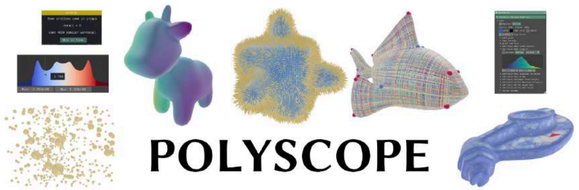




3D Models — Mesh

How do I access and visualize meshes from my code?

- The IO of geometry processing libraries (e.g. libigl), GUI of 3D data (e.g. polyscope)




```
import igl  # Import the libigl library
import polyscope as ps  # Import the polyscope library
import numpy as np

# Read the mesh from a file
v, f = igl.read_triangle_mesh("HappyDragon.ply")

# Create a rotation matrix for 90 degrees rotation around x-axis
angle = np.radians(90)
rotation_matrix = np.array([[1, 0, 0],
                            [0, np.cos(angle), -np.sin(angle)],
                            [0, np.sin(angle), np.cos(angle)]])
# Rotate vertices with the matrix
v = np.dot(v, rotation_matrix)

# Print the dimensions of V (vertices) and F (faces)
print("Vertices shape:", v.shape)
print("Faces shape:", f.shape)

# Initialize Polyscope
ps.init()
ps.set_ground_plane_mode("shadow_only")  # set +Z as up direction
ps.set_shadow_darkness(0.1)              # lighter shadows
# Register the mesh in Polyscope
ps_mesh = ps.register_surface_mesh("my_mesh", v, f)
ps_mesh.set_color((68/255,254/255,157/255))
ps_mesh.set_edge_color((0.36,0.36,0.36))  # white edges
ps_mesh.set_edge_width(1.5)      # adjust as needed
# Show the Polyscope GUI
ps.show()
```

Now, your turn!

Go to the course webpage to download data!

We’ll wok on visualizing these data together!

Pair-Coding

```
pip install numpy
```

```
pip install Pillow
```

```
python -m pip install libigl
```

```
pip install polyscope
```

Pair-Coding

```
from PIL import Image

# Open an image file
image_path = "example.jpg"  # Replace with the path to your image file
image = Image.open(image_path)

# Get dimensions
width, height = image.size
print(f"Image Dimensions: Width = {width}, Height = {height}")

# Show the image
image.show()
```

Pair-Coding

```
import igl  # Import the libigl library
import polyscope as ps  # Import the polyscope library
import numpy as np

# Read the mesh from a file
v, f = igl.read_triangle_mesh("HappyDragon.ply")

# Create a rotation matrix for 90 degrees rotation around x-axis
angle = np.radians(90)
rotation_matrix = np.array([[1, 0, 0],
                            [0, np.cos(angle), -np.sin(angle)],
                            [0, np.sin(angle), np.cos(angle)]])
# Rotate vertices with the matrix
v = np.dot(v, rotation_matrix)

# Print the dimensions of V (vertices) and F (faces)
print("Vertices shape:", v.shape)
print("Faces shape:", f.shape)

# Initialize Polyscope
ps.init()
ps.set_ground_plane_mode("shadow_only")  # set +Z as up direction
ps.set_shadow_darkness(0.1)              # lighter shadows
# Register the mesh in Polyscope
ps_mesh = ps.register_surface_mesh("my_mesh", v, f)
ps_mesh.set_color((68/255,254/255,157/255))
ps_mesh.set_edge_color((0.36,0.36,0.36))  # white edges
ps_mesh.set_edge_width(1.5)      # adjust as needed
# Show the Polyscope GUI
ps.show()
```

Take-aways from Today's Lecture

- You learned a new terminology in computer graphics, "mesh"
- Conceptually, you understood what's visual data, how they are captured and stored in computers
- You succeeded in checking visual data with Python
- You just got your hands on Libigl and Polyscope, two of the most popular libraries in the research world of computer graphics

Are There Any Questions?