Assignment 1 short report

Implemented functions:

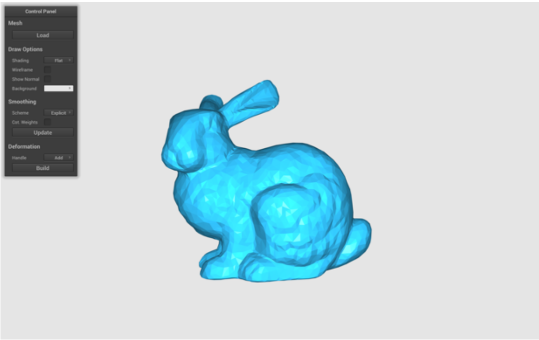
- void Mesh::umbrellaSmooth(bool)
- void Mesh::implicitUmbrellaSmooth(bool)

[Test case A]

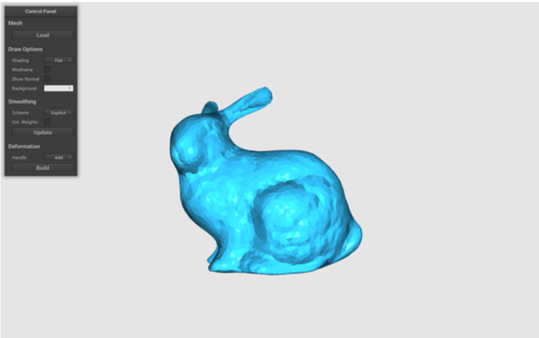
Implicit vs Explicit

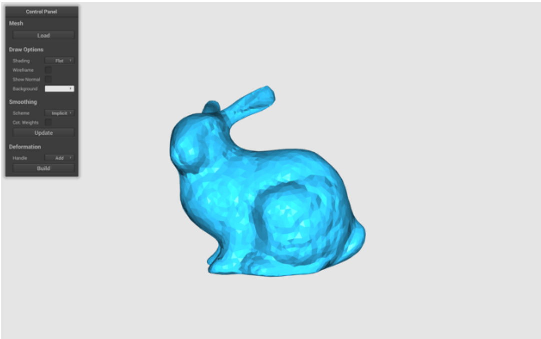
1. Fixed hyper-parameters: lambda=0.5, iterations=20, using uniform smoothing
2. Meshes: data/bunny.obj, data/cube_bumpy.obj, data/feline.obj.
3. Results:

Bunny Original:

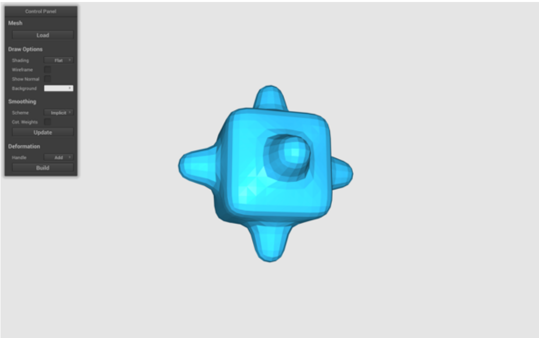


Bunny Smoothing (explicit: left; implicit: right)

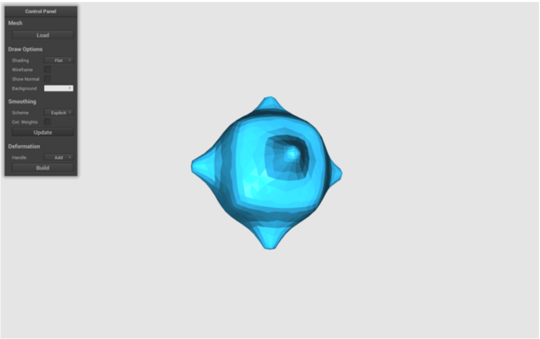


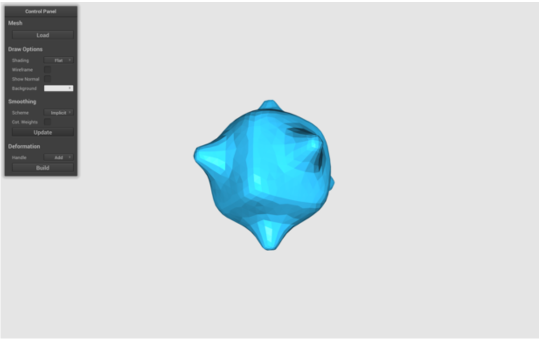


Cube_Bumpy Original:

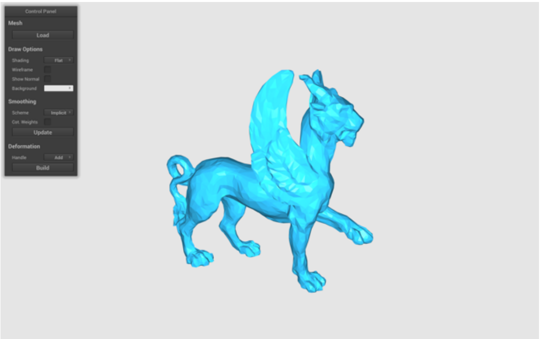


Cube_Bumpy Smoothing (explicit: left; implicit: right)

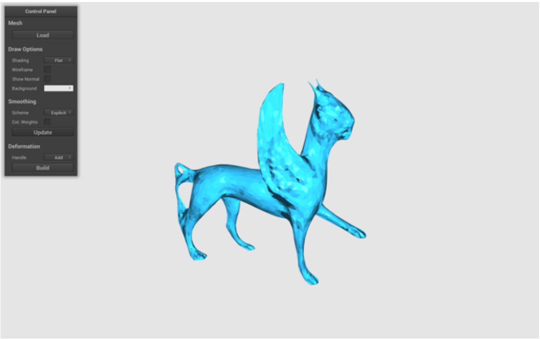


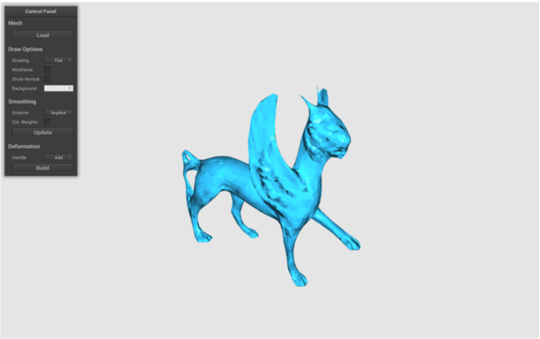


Feline Original:



Feline Smoothing (explicit: left; implicit: right)



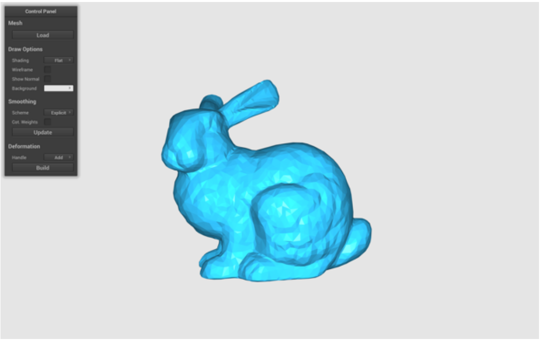


Conclusion A: From the Feline example, we can see that implicit preserves more details from the shape and pattern of the feline's ears and legs. From the Bunny example, the implicit smoothed mesh preserves a curvier border compared to the explicit. The result from the Cube_Bumpy is not obvious though.

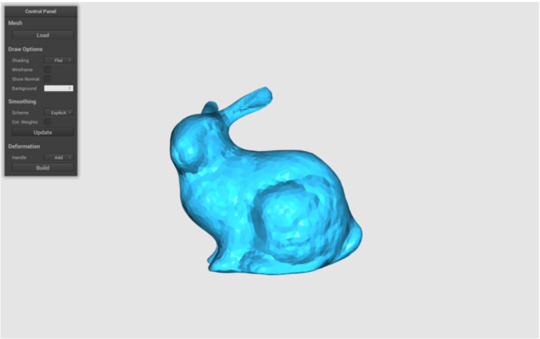
[Test case B]
Cotangent weights vs Uniform smoothing

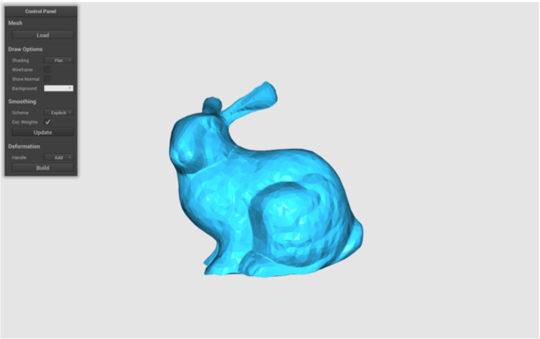
1. Fixed hyper-parameters: lambda=0.5, iterations=20, using explicit smoothing
2. Meshes: data/bunny.obj, data/cube_bumpy.obj, data/feline.obj.
3. Results:

Bunny Original:

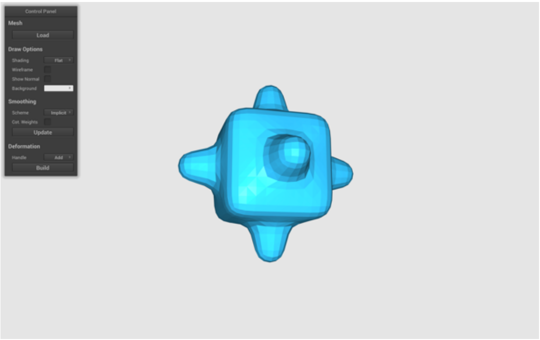


Bunny Smoothing (uniform: left; contangent: right)

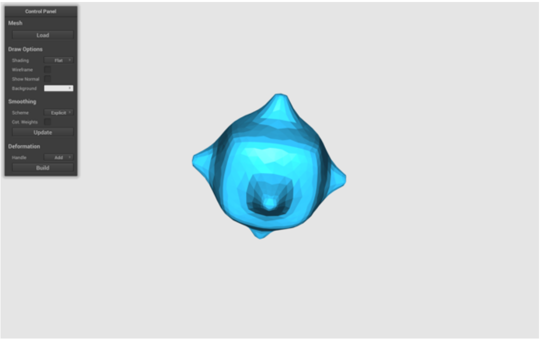


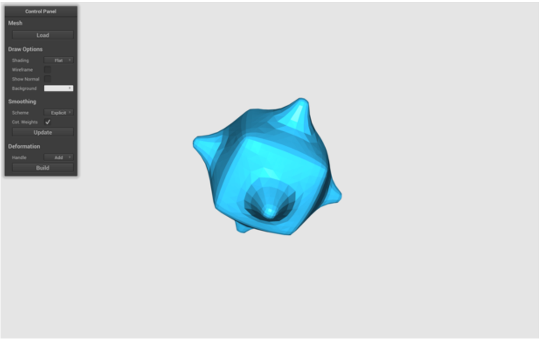


Cube_Bumpy Original:

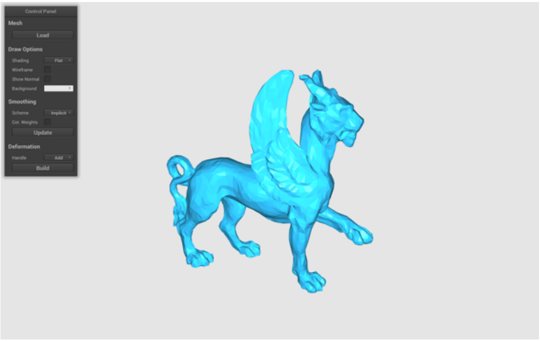


Cube_Bumpy Smoothing (uniform: left; contangent: right)

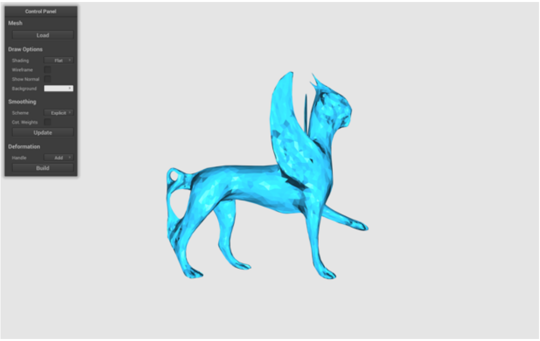


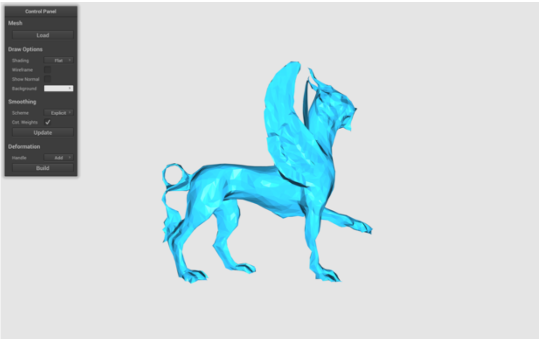


Feline Original:



Feline Smoothing (uniform: left; contangent: right)





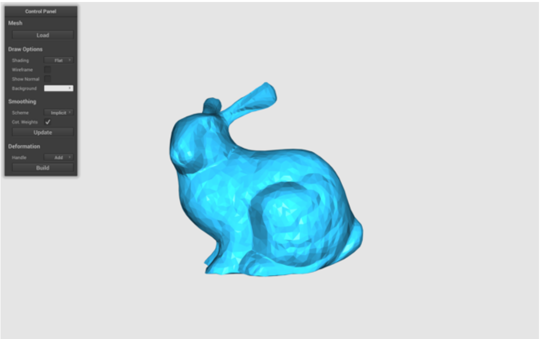
Conclusion B: From the Feline example, we can see that contangent is more similar to the original shape compared to the uniform. This can be shown by the feline's legs, body, wings, etc. From the Bunny example, the contangent smoothed mesh is a bit fatter than the uniform one. The result from the Cube_Bumpy is not obvious again.

[Test case C]

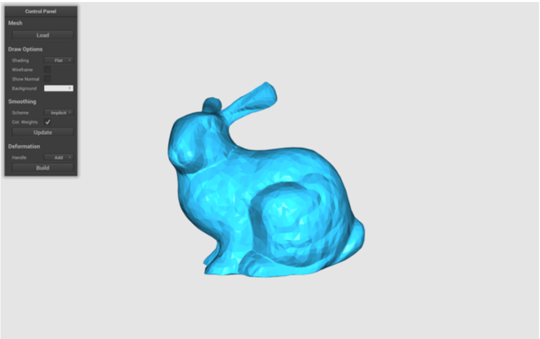
Lambda (high=0.7, middle=0.5, low=0.3)

1. Fixed hyper-parameters: iterations=20, using implicit, contangent smoothing
2. Meshes: data/bunny.obj, data/cube_bumpy.obj, data/feline.obj.
3. Results:

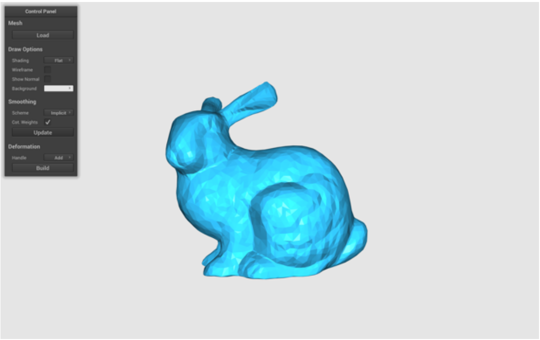
Bunny Smoothing (high to low; left to right to bottom)



Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

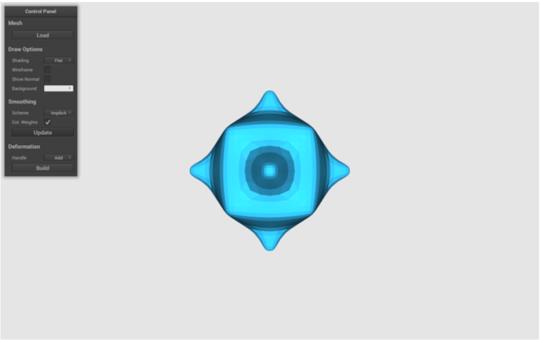


Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

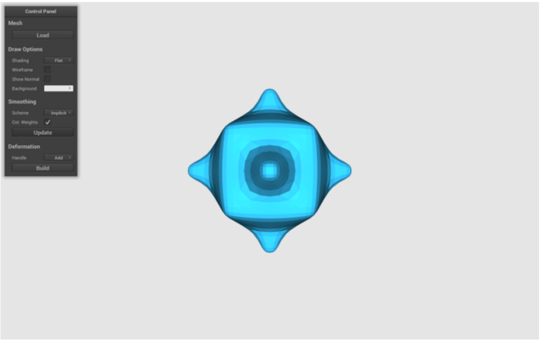


Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

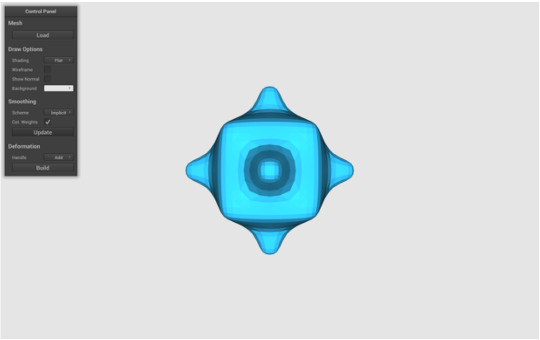
Cube_Bumpy Smoothing (high to low; left to right to bottom)



Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

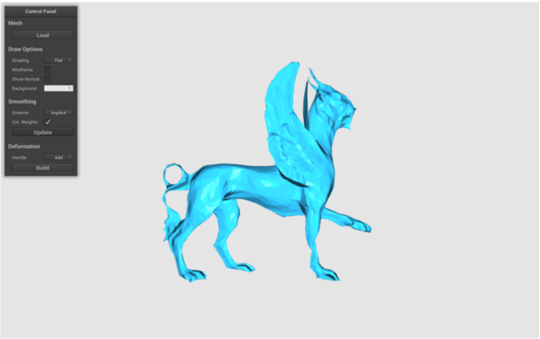


Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

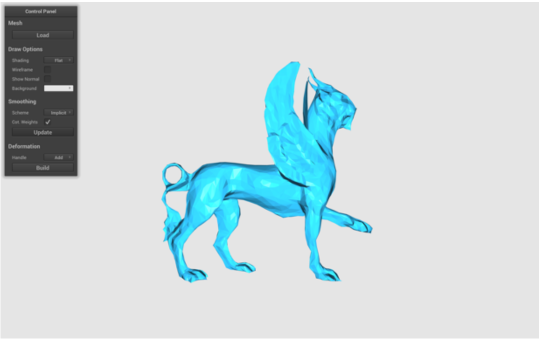


Shading Flat
Wireframe □
Show Normal □
Background
Scheme Implicit
Cot. Weights ✓
Handle Add

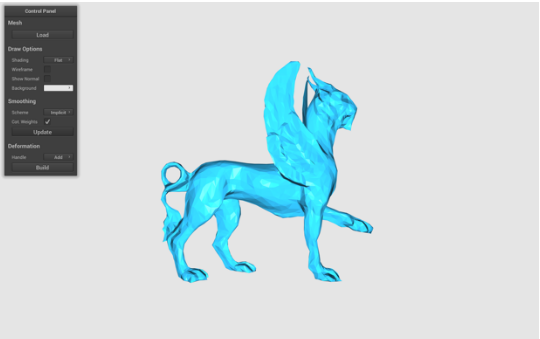
Feline Smoothing (high to low; left to right to bottom)



Shading Flat
Wireframe □
Show Normal □
Scheme Implicit
Cot. Weights ✓
Handle Add



Shading Flat
Wireframe □
Show Normal □
Scheme Implicit
Cot. Weights ✓
Handle Add



Shading Flat
Wireframe □
Show Normal □
Scheme Implicit
Cot. Weights ✓
Handle Add

Conclusion C: From the Feline example, we can see that the thickness of the tail decreases as the lambda value increases. From the Bunny example, the border shape is more preserved when lambda is low. The result from the Cube_Bumpy is not obvious though, but we can still observe the squareness change of the center cube from the bird view. This implies that larger lambda has more significant smoothing over iterations.