SHANGHAITECH UNIVERSITY
School of Information Science and Technology
CS271: Computer Graphics II
Spring 2021
Final Course Project Topics
Released: Thursday, May 6th, 2021
Due: Monday, June 6th, 2021

Students are free to establish your own team with up to 3 team members in total. Each team need to select one topic from the following candidates while meeting the restriction of **the number of team members**, i.e., the number of your team members should be smaller or equal to the required team member indicated in each topic.
The division of labor in your team must be clear (need to be written down). It is not allowed that one person in your group handles everything while the rests do nothing. Everyone should be familiar with the selected project. **Please ensure academic integrity. Unless explicitly noted, work turned in should reflect your own/independent capabilities. No cheating (We will check very carefully!): 1) Don't share your codes. 2) No fake solutions! Otherwise, there will be serious consequences!**

Topic 1: Chordal Axis Transform (1)
Implement the Chordal Axis Transform (CAT) in 3D space and use it to divide the space for disjoint 3D objects. You need to show some examples.

Reference
1. Prasad L. *Rectification of the chordal axis transform and a new criterion for shape decomposition* [C]//International Conference on Discrete Geometry for Computer Imagery. Springer, Berlin, Heidelberg, 2005: 263-275.
2. Ma Y, Chen Z, Hu W, et al. *Packing irregular objects in 3D space via hybrid optimization*[C]//Computer Graphics Forum. 2018, 37(5): 49-59.

Topic 2: Media Axis Transformation and Its Application for 3D Vision and Shape Analysis (2)
Medial Axis Transform (MAT) is an important concept in computational geometry, which has been widely explored for shape approximation, shape recognition, shape retrieval, shape segmentation, etc. Please implement an 3D MAT algorithm and one relevant application. If you choose to use others' code as your baseline, improvement by your team is REQUIRED. Use sufficient experiments to demonstrate the superiority of your idea.

Reference
1. Li P, Wang B, Sun F, et al. *Q-mat: Computing medial axis transform by quadratic error minimization*[J]. ACM Transactions on Graphics (TOG), 2015, 35(1): 1-16.
2. Lin C, Li C, Liu Y, et al. *Point2Skeleton: Learning Skeletal Representations from Point Clouds*[J]. CVPR, 2021.
3. Lin C, Liu L, Li C, et al. *Seg-mat: 3d shape segmentation using medial axis transform*[J]. IEEE Transactions on Visualization and Computer Graphics, 2020.
4. Yang B, Yao J, Guo X. *DMAT: Deformable medial axis transform for animated mesh approximation*[C]//Computer Graphics Forum. 2018, 37(7): 301-311.

Topic 3: 3D Object Modeling from a Single or Multiple Image(s) (3)
Single camera is easy to acquire due to the popularity smart phone. 3D object modeling from a

www. shanghaitech. edu. cn
地址：上海市浦东新区华夏中路393号 邮编：201210
Add: 393 Middle Huaxia Road, Pudong, Shanghai 201210, China

single or multiple image(s) become more and more popular because of its wide applications in people's daily life. Please implement an algorithm to model 3D objects or humans from a single or multiple image(s). If you choose to use others' code as your baseline, improvement by your team is required. Use sufficient experiments to demonstrate the superiority of your idea.

Reference

1. Pavlakos G, Choutas V, Ghorbani N, et al. *Expressive body capture: 3d hands, face, and body from a single image[*C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 10975-10985.
2. Li C, Pan H, Liu Y, et al. *Bendsketch: Modeling freeform surfaces through 2d sketching*[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1-14.
3. Li C, Pan H, Liu Y, et al. *Robust flow-guided neural prediction for sketch-based freeform surface modeling*[J]. ACM Transactions on Graphics (TOG), 2018, 37(6): 1-12.

Topic 4: 3D Object Perception from Point Cloud (3)

Point cloud provides exact depth information of objects in 3D space, which is helpful for 3D object detection or segmentation. Please implement an algorithm of 3D perception for indoor or outdoor scenarios. If you choose to use others' code as your baseline, improvement by your team is REQUIRED. Use sufficient experiments to show the superiority of your idea.

Reference

Lecture 6 and Lecture 7

Topic 5: Implementation of Bilateral Normal Filtering and mesh simplification algorithm QEM (1)

The target of this topic is to obtain both the Bilateral Normal Filtering result of an input mesh and a QEM result after mesh simplification. Particularly, the input of the Bilateral Normal Filtering is a mesh with distinct disturbances and the output is a smoothed mesh. To achieve this, you first need to calculate the facet normal of each facet that is relatively smooth. Then, you update the position of vertices by facet normal. Both calculating the facet normal and updating the new vertices are done by iteration.

Reference

Lecture 10, 11

Topic 6: Variational Shape Approximation (1)

Implement the clustering part of the triangular mesh in the Variational Shape Approximation. Particularly, the input is the triangular mesh object and the number of clusters K, and the output is the cluster for each triangular facet and the normal of each cluster.

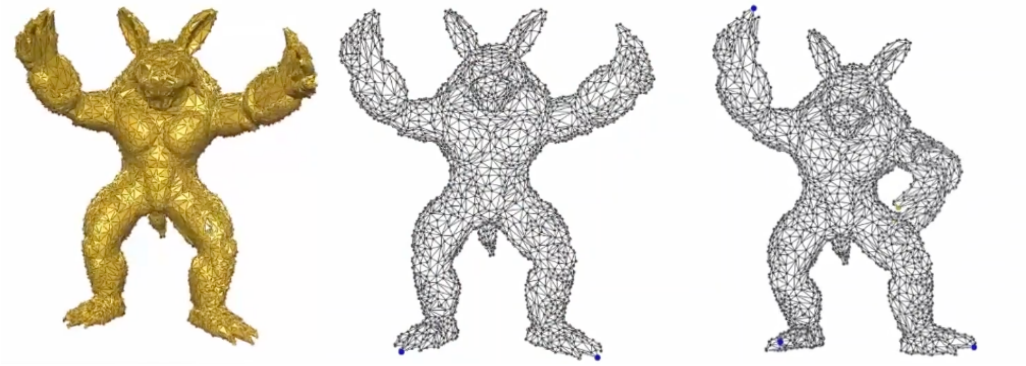
Reference

1. Lecture 11
2. David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. ACM Trans. Graph. 23, 3 (August 2004), 905–914. DOI:https://doi.org/10.1145/1015706.1015817

Topic 6: As-Rigid-As-Possible Surface Modeling (2)

Implement the as-rigid-as-possible surface modeling. Particularly, you should fix some vertices (e.g., 3) of a mesh object, and then drag another vertex to make a rigid deformation. Besides implementing the algorithm, you should provide a GUI interface for user interaction. For example,

www. shanghaitech. edu. cn
地址：上海市浦东新区华夏中路393号 邮编：201210
Add: 393 Middle Huaxia Road, Pudong, Shanghai 201210, China



Reference

1. Lecture 13
2. Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In Proceedings of the fifth Eurographics symposium on Geometry processing (SGP '07). Eurographics Association, Goslar, DEU, 109–116.

Topic 7: Programmable Shading and Shadows (2)

In this topic, your objective is to render meshes efficiently using programmable shading. We will provide the starting mesh viewer for this topic. The included mesh viewer allows you to visualize the mesh in several modes by pressing the keys 1 through 7 or using the popup menu (**The provided skeleton code is similar to what you have seen in Programming Assignment 4 and 5**). The active mode is displayed in the console window. All but the last two "Custom Shader" modes are already implemented in the skeleton code using the fixed function pipeline. You should complete the shaders in the *.glsl files in order to successfully render the mesh using the last two modes, which should employ programmable shaders.

Loading the meshes: You should specify the path to venus.obj or dragon.obj mesh files in the command line (e.g., ../models/venus.obj). For task 2 and 3, you must use the venus mesh because it is the only one that contains texture coordinates. For the remaining tasks, you may use either mesh.

1. Gouraud Shading

Implement the "Custom Shader – Gouraud Shading" mode. To do so, you must complete the vertex shader in blinn-phong-vert.glsl that performs Blinn-Phong shading at each vertex as described in the lecture notes (no attenuation term). You should mimic what is done by the fixed function pipeline solution for the "Smooth Shading" mode by looking at the cpp source code and the information provided in the glsl file. You should use the built-in uniform variables to retrieve this information in the vertex shader. Finally, you should fill in the pixel shader so that simply interpolates the colors computed at the vertices. If correct, the result should match that of the provided "Smooth Shading" mode.

2. Texture Mapping

Add a texture lookup in the pixel shader to read from the only supplied texture using the provided texture coordinates. After doing this, your "Custom Shader – Gouraud Shading" should match that of "Textured Smooth Shading".

3. Phong Shading

Implement the "Custom Shader – Phong Shading" mode in blinn-phong-frag.glsl. To do so, you must move your Blinn-Phong shading computation from the vertex shader to the pixel shader. The vertex Shader simply passes the vertex normal on to the rasterizer. The pixel shader should receive the interpolated normal and perform the Blinn-Phong shading computation for each pixel.

4. Vertex Animation

Finally, you should modify the vertex shader in blinn-phong-frag.glsl using an interesting

www. shanghaitech. edu. cn
地址：上海市浦东新区华夏中路393号 邮编：201210
Add: 393 Middle Huaxia Road, Pudong, Shanghai 201210, China

animation function. The function should modify the final vertex position based on the input vertex position and a provided frame-time variable.

When you arrive here, you should have become familiar with how to use basic OpenGL to process and render mesh, as well as to shade them using GLSL. Then, you are supposed to add shadows to the rendering application. The included mesh viewer allows you to visualize the mesh in several modes by pressing the keys 1 through 9 or using the popup menu. The active mode is displayed in the console window. You should finish the implementation of the modes "w/Shadow map" and "w/Shadow volume" in order to have a working version of the two techniques. A lot of the work has been already done for you in the skeleton code. Below are the parts that you should to complete (all of them are specified with a "TODO" comment).

5. Shadow Mapping

You should finish the implementation of the shadow mapping algorithm by completing the following parts of the code.

1) In the function SetupShadowMapPOVMatrices(), compute the transformation matrix that each vertex of the model needs to undergo in order to render the scene from the point of view of the light and generate the shadow map. Note that the transformation should compute the normalized device coordinates (i.e., screen coordinates).
2) In the function SetupShadowMapTextureMatrix(), compute the transformation matrix that each rendered pixel of the model needs to undergo in order to determine its position in the generated shadow map. Note that the transformation should compute the texture space coordinates. Again, refer to the code comments for more specific guidelines.
3) In the shader file render_perlight_shadow_map.glsl, complete the fragment shader so that it properly performs the depth test and determines whether the pixel is on shadow. The shader code comments describe the specific steps.

6. Shadow Volume

You should finish the implementation of the shadow volume algorithm. In the shader file shadow_volume.glsl, complete the geometry shader so that it detects silhouette edges. All three edges of each processed triangle need to be processed. The provided skeleton code calls the DetectAndProcessSilhouette() function for each of those edges, and your task is to determine if the edge is indeed a silhouette with respect to the light. This can be achieved by checking if the processed triangle is facing the light, and the adjacent triangle is facing away from the light. In case this is true, you should then extrude a quad (i.e., two triangles) to generate the shadow volume. Refer to the shader code comments for additional guidelines.

Topic 8: Volume Visualization with Raycasting and Transfer Function (3)

Scalar field visualization has always received a lot of attention in visualization domain, especially the 3D scalar field visualization, including scientific computing data visualization and engineering computing data visualization, etc. Meanwhile, it also includes visualizations of various measurement datasets. For example, computed tomography (CT) data used in the medical field and magnetic resonance (MRI) are domains that visualization is widely used.

Starting from the late 1980s, volume rendering is a very important method in visualization algorithms. Since volume rendering algorithm cannot only be used to display the surface information of volume dataset, it can also be applied to display the internal information of the dataset, thus it can display realistic 3D medical image dataset and help doctors conduct a comprehensive understanding and analysis, which plays a very important role in assisting doctors in diagnosis and treatment. Volume rendering transfers a large amount of abstract data into a visual image to enable people to get insight into the hiding information. The visibility of the voxel in the volume dataset drawn in the image depends on the opacity value assigned by the so-called transfer function. Those voxels with interest are assigned with a higher opacity value; on the contrary, it is assigned a lower opacity value. Voxel colors in the resulting image can also be

www. shanghaitech. edu. cn
地址：上海市浦东新区华夏中路393号 邮编：201210
Add: 393 Middle Huaxia Road, Pudong, Shanghai 201210, China

determined by the transfer function through assigning different colors to voxels and the features and the internal structure of volume dataset can be observed more clearly. The elaborately designed transfer functions can reveal important data structure and details.

Volume ray casting algorithm is a mature method for rendering a 3D volume dataset. And there are many resources available for us to follow. The main issue of volume rendering is how to efficiently and elaborately design a good transfer function for users. Therefore, a feasible interface is a must for implementing a volume rendering method.

Reference

1. https://www.uni-muenster.de/Voreen/
2. https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch39.html
3. Rost R J, Licea-Kane B M, Ginsburg D, et al. OpenGL shading language[M]. Pearson Education, 2009

Submission

Please submit your zipped file with a name “CS271_[Your group name]_[group member name and ID]_[Topic x]”. You should submit both your source code and a project report.

www. shanghaitech. edu. cn
地址：上海市浦东新区华夏中路393号 邮编：201210
Add: 393 Middle Huaxia Road, Pudong, Shanghai 201210, China