LITM: Layer-Interpolating Tet Mesh

Final project for CS6491

ABSTRACT

The goal of this project is to help the student practice inventing, justifying, implementing, debugging, and teaching advanced algorithms that construct, display, process, and encode triangle (tri) meshes and tetrahedron (tet) meshes.

1 Due date and deliverables

The **final version of the project is due on November 28** before class. It must be submitted online onto T-Square. One submission per team of one or two members.

The submission must include the following:

- A zip of the **sketch**, including data and image files needed to run the sketch. Please, do not include in the zipped folder images or videos that are not necessary to run it. Make sure that the code of the sketch has a clear structure and enough comments to understand where the specific parts/modules are and what they compute.
- A brief **report** in PDF
- A short **video** (<5mns) stating the problem, showing your solution and the animation of its construction, and explaining the nature of your approach (using graphics and animation as appropriate).

About the report:

- Give your project an original title, such as "SpaceBallRoller"
- Include a proper header with that title, names of the authors, reference to "CS6491 Fall 2017, Project 5", and the date
- Include the headshots of the team members above their names
- A clear, concise, complete, and concrete **problem statement** formulated in your own words using mathematical concepts learned in class and assisted by images produced by your program
- A precise statement of your contributions and the level of completion of your solution: Have you solved the problem? If not, what is missing and why? If yes, characterize your results (asymptotic complexity, measured time average per ball, assumed restrictions on the input (such as for example that balls must be pairwise disjoint or that you need at least 3 balls on each plane, or that you assume a 'general configuration' where specific spatial alignments are forbidden), and comments on how reliable your solution is (does it crash and if so how often).
- A clear, but high level **outline** of the nature of your solution/approach. This should target a senior researcher in this field.
- A brief review of the most relevant prior art that you have found. Do not merely cite the work or include a section of their abstract. Instead, show that you have read the paper and that you were able to extract the parts that are relevant to this project. Discuss whether you have followed any of the ideas suggested in prior art or how your solution relates to them.
- A step-by-step guideline for a junior developer charged to implement this from scratch.

I encourage you to also create a personal web page where you will post this report and the video and to email the TA and me the link to these with the names of the team members and cc'ing them… in case we have comments. In fact, if you do it in advance of the deadline, we may try to give you early feedback.

NOTE: Each team must submit on T-Square an early **draft of your report** by **November 14**. This should include the most of the above, including the problem statement, the outline of your approach, the discussion of prior art, and at least some of the implementation details and results.

2 Goal of the project

Consider two horizontal **planes**, called the **floor** and the **ceiling**. Let 'h' be the spacing between them.

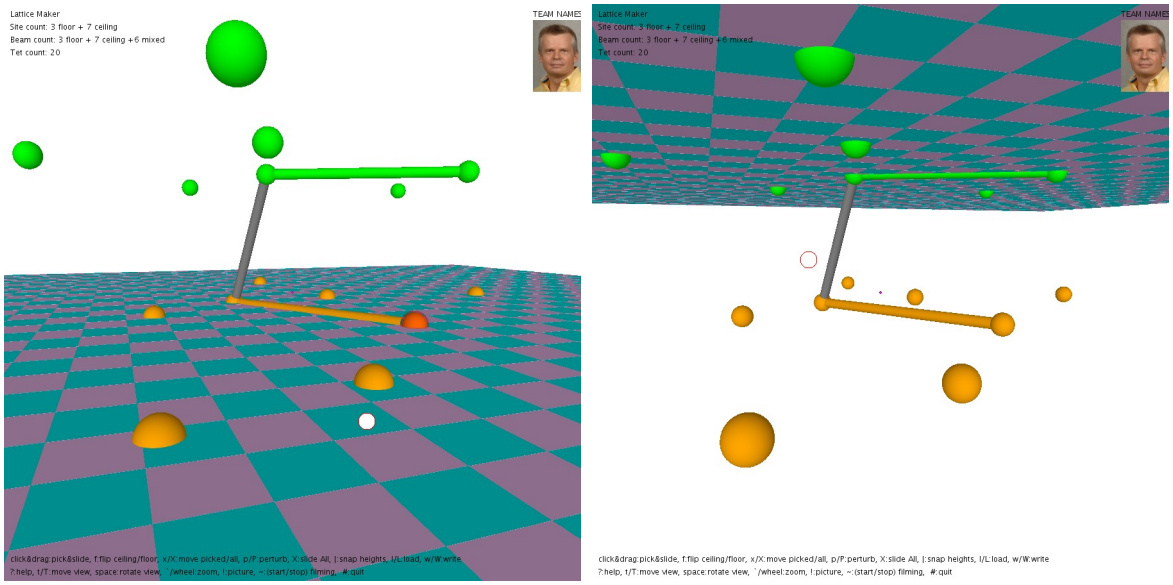
On each plane, there is a **cloud** of **balls** with centers (called 'sites') on that plane. All balls have the same radius 'r'.

Your solution should accomplish two things:

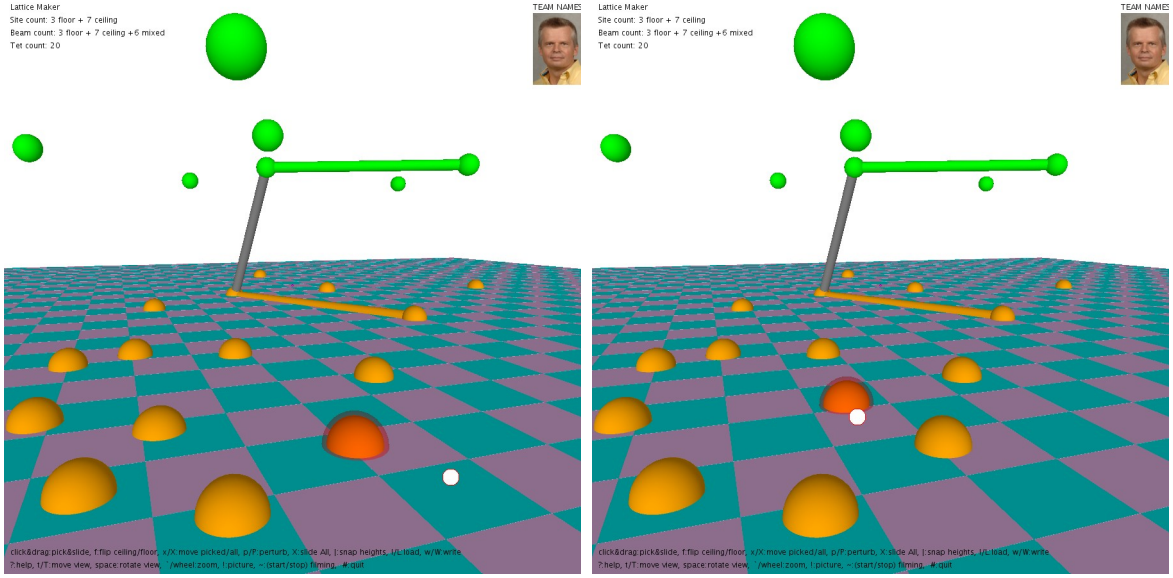
1) Compute the **edges** ("tubes") of the Delaunay Tetrahedralization of the union of the sites on both planes
2) Compute a high-resolution water-tight triangle mesh that approximates the boundary of the union of all balls and tubes and render this mesh using smooth shading with visible and hidden silhouettes drawn.

3 Code provided

The code provided lets you switch which plane is active by pressing ‘f’.

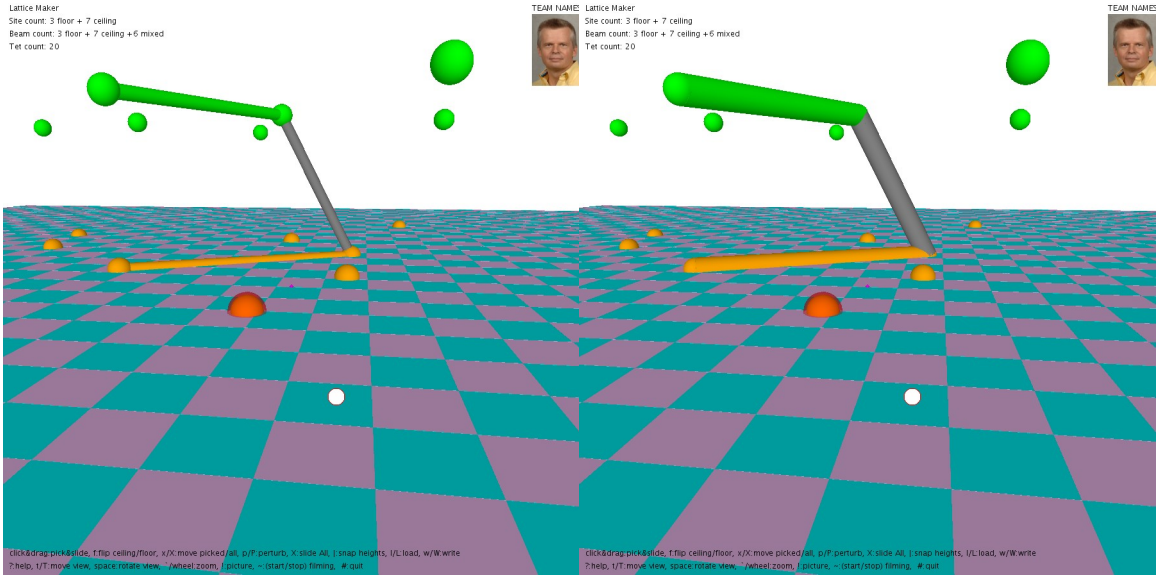


You can click any of the balls in the active plane, slide it by dragging the mouse, add new balls (keep ‘a’ pressed and click where you want the new ball) or delete a ball (point and press ‘d’).



Add an action and two key presses: ‘p’ that perturbs the location of each ball on the active plane by a very small random amount and ‘P’ that perturbs it by a large random amount in x and y (cap the permutation to 100 or so, but avoid sites that fall outside of the drawn rectangle). These keys will be used to (1) avoid singular configurations and (2) keep you honest and make it easier to test your algorithms for a variety of input configurations.

You can switch between showing tubes with half-radius (‘h’), for clarity, and full radius (‘H’) when computing the bounding triangle mesh.



You can manipulate the view by sliding the red focus point (‘t’ + mouse drag) or dragging it up/down (‘T’), by zooming in/out (‘`’ + mouse move, not drag), and by rotating around the focus point (SAPCE + mouse move, not drag)

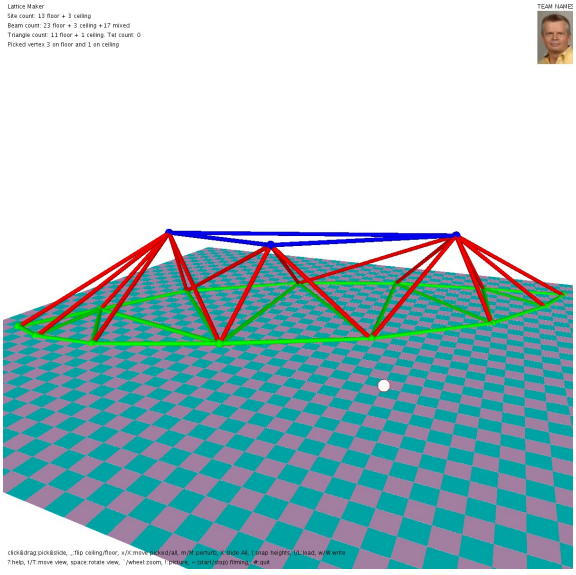
4 Suggestions for making progress

Figure out the math, implement, visualize, test, debug the computation of centers of circumscribing circles and spheres.

Extend the above to compute and show the bulge of such a sphere from one of the triangles.

Implement the most naïve Delaunay tetrahedralization in 3 phases: 3-1 up, 1-3 down, and 2-2 mixed. Visualize and debug each phase one at a time.

Test and do not add duplicate beams. Count the floor, ceiling, and mixed edges and write the counts on the canvas. Also count the floor and ceiling triangles and the tets created by each phase. Make sure that all is correct. Use different colors, additional graphics, and key selections to show these.



Make n randomly generated points on the floor and ceiling and record timing as a function of n to motivate and report benefit of future improvements.

Work in parallel on (1) improving tetrahedralization and on (2) using ball pivoting to compute a triangular mesh that bounds the union of all balls and beams.

5 Resources

Circumspheres

http://mathworld.wolfram.com/Circumsphere.html
http://www2.washjeff.edu/users/mwoltermann/Dorrie/70.pdf
https://www.jstor.org/stable/2973351?seq=1#page_scan_tab_contents

Delaunay triangulations
https://en.wikipedia.org/wiki/Delaunay_triangulation
https://cs.stackexchange.com/questions/2400/brute-force-delaunay-triangulation-algorithm-complexity
https://graphics.stanford.edu/courses/cs468-02-winter/Papers/linearDT.pdf

Delaunay tetrahedralization
https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2002/tr-2002-02.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.438&rep=rep1&type=pdf
http://wias-berlin.de/software/tetgen/examples.dragon.html

Triangulations from slices
https://pdfs.semanticscholar.org/781b/58f08864a8839aa50b61902a77746bda4bce.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.242&rep=rep1&type=pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.5208&rep=rep1&type=pdf
https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2204&context=cstech

Delaunay tetrahedralization from slices
https://hal.inria.fr/inria-00076008/document
https://www.cs.jhu.edu/~misha/Fall13b/Papers/Cazals06.pdf
https://www.researchgate.net/publication/224147241_Delaunay_Triangulation_Based_Three_Dimensional_Anatomical_Facial_Reconstruction_from_2D_CT_Slices
http://liris.cnrs.fr/~rchaine/DOCUMENTS/sgp07_pre.pdf

Surface reconstruction from point clouds:
https://www.csun.edu/~ctoth/Handbook/chap35.pdf
https://hal.inria.fr/inria-00072024/document
https://www.cs.purdue.edu/homes/aliaga/cs334-15fall/lectures/lec-voronoi-and-triangulation.pdf

Ball Pivoting:
http://www.research.ibm.com/vistechnology/pdf/bpa_tvcg.pdf
https://www.cs.jhu.edu/~misha/Fall13b/Notes/Bernardini99.reviews.pdf
http://www.ipol.im/pub/art/2014/81/article_lr.pdf
http://rodschulz.github.io/BPA/
http://web.cs.ucdavis.edu/~amenta/w11/DTapp.pdf