CS606: Computer Graphics / Term 1 (2025-26) / Programming Assignment 1 (A1)
International Institute of Information Technology Bangalore

Announcement Date: Aug 25, 2025 (Mon)
Submission Deadline: 11:59 pm IST, Sep 14, 2025 (Sun)

Summary: 2D crowd simulation renderer with 2D translation, rotation, scaling/zooming.
Learning Objectives:

- WebGL programming
- Creation and rendering of simple 2D geometric shapes as primitives
- Transformation of 2D objects and scene -- instance-wise implementation
- Key(board) and mouse events for changing control modes, and transformation implementation
- (Optional) Mouse events for picking objects

Individual/Personal Assignment: CrowdSimulationRenderer

The goal of this assignment is to develop a WebGL program that will render an application akin to a crowd simulator.

Figure 1: Layout in a Crowd Simulator Tool (https://doi.org/10.3390/buildings11100445)

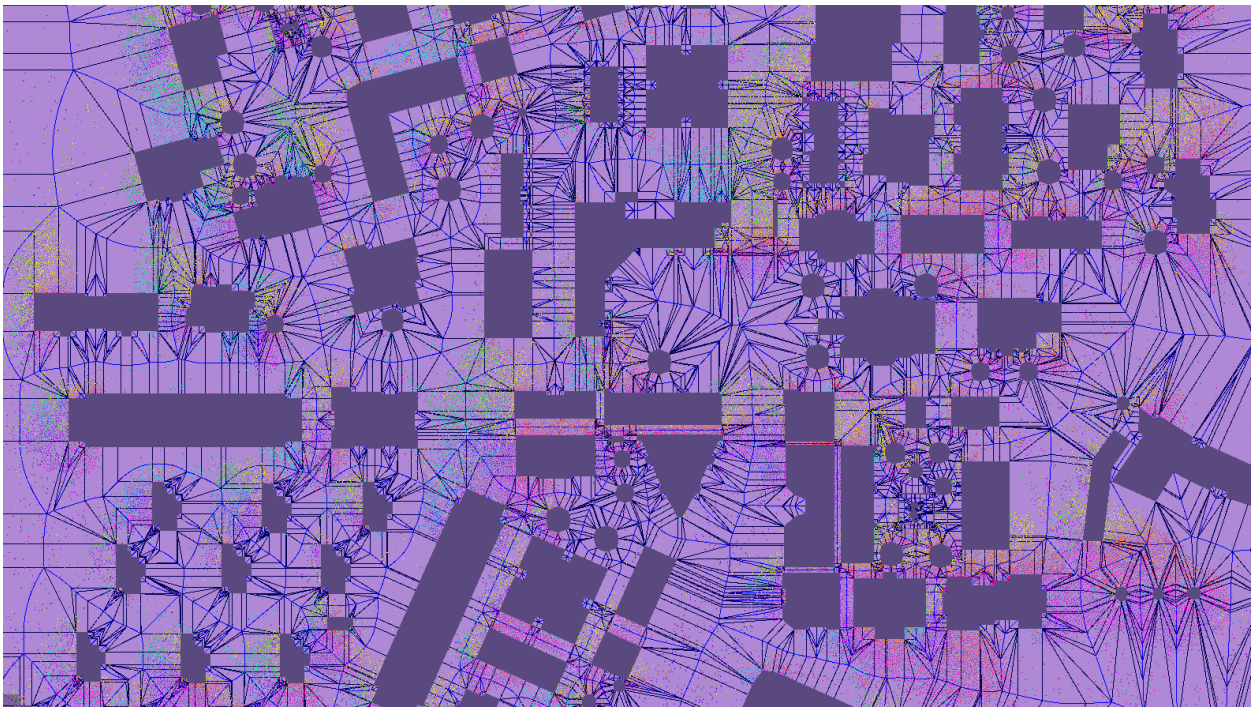

Figure 2: Another layout of deterministic crowd simulations using TerraCrowds® (https://www.ucrowds.com/integration/)

Desired functionalities:

1. Generate a random layout of points for triangulating a rectangle and the triangles for the spatial partitioning (as shown in Figures 1 and 2). *[Hint: keep your triangulation simplistic with fewer number of sufficiently far points, that is, the triangulation should not be as complex as the layouts in the Figures 1-2.]*
2. Place an obstacle in square or rectangle shape (e.g. the tan shapes in Figure 1 and the gray boxes in Figure 2), whose corners are also points mentioned in #1. The edges should be part of the triangulation (Figure 3).
3. With the static layout in place from #1 and #2, render black points representative of people in the space. #1-#3 provide the base state of the crowd (Figure 4).
4. Using mouse and keyboard functionalities:
    1. Rotate the obstacles, which should update the triangulation – the incoming edges to the obstacle corners may be updated to the nearest obstacle corner after rotation, so that there are no intersection of triangle edges (see Figure 3). There can be additional user interactions to delete extra edges which are not relevant for the new triangulation, and add new edges. (*Note: automating triangulation may be difficult, it would be easier to implement interactive addition and deletion of edges.*)
    2. Move the obstacles, which should again update the triangulation, as mentioned in #4.1.

3. The obstacles can be scaled up/down about its center, which should again update the triangulation, as given in #4.1.
4. Assign a population density (say 4 people per triangle), and color triangles with correct density as green, overpopulated as red, and under populated as blue. *[Hint: use lighter shades of the color, and color the primitives before rendering the points for people.]*
5. The mesh re-triangulation in #4.1-#4.3 should update the color scheme in #4.4.
6. A dot for person can be moved from one triangle to another, and the completion of this movement the source and destination triangles should be rendered with updated colors.

Note: the students can make their own strategy and choice of keyboard/mouse movements for #4.1-#4.6.

Instructions on submissions:

Submissions must be made on LMS. e-mail submissions will not be graded.

1. The deliverables in a zipper folder must include a source code folder, a folder with screenshots showing representation of the working of all desired functionalities, a demo video not longer than 5 minutes, a README file, and a report.
    a. The report in .pdf contains the answers to the questions given at the end of the assignment document. It also contains the sources that should be cited for your submission. If any source not cited is discovered to have been influential to the submission will be directly considered as plagiarism. The report can optionally contain necessary information about your implementation (any specific strategies you have used). More details on the submission are given in the course handbook on the LMS course page.
    b. The demo video should show the working of the app for all the desired functionalities including transitions between states.
    c. The README must have clear instructions on how to compile and use the app.
2. If the deliverables are larger than the maximum allowable size on LMS, submit a text document with a OneDrive URL. Care must be taken to not change this repository until grading is done.
    a. Submitting this link as a comment on LMS does not show your submission to be “submitted,” and hence, may get unnoticed. Hence, submitting the link as a comment is not recommended.
    b. The latest timestamp of all the contents of the submitted deliverables would be taken as the timestamp of the submission.

Questions to be answered in the report:

1. Explain your strategy for user interactions?
2. How is scaling about a corner of the quadrilateral different from that of the center?

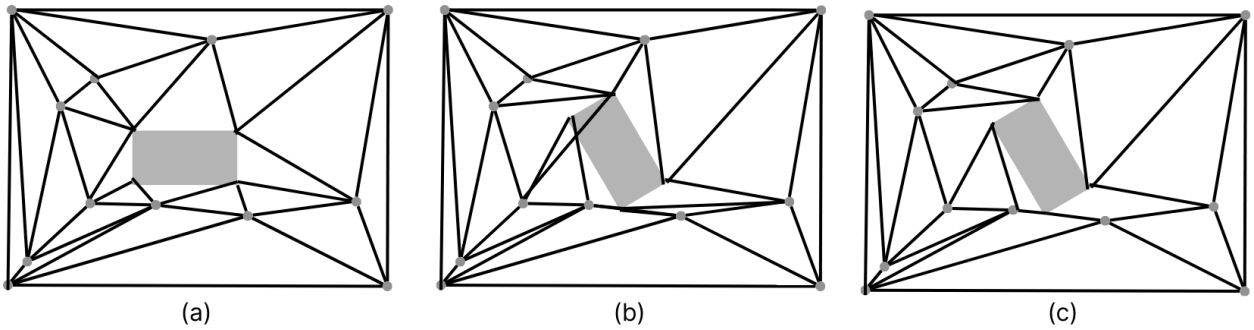

Figure 3: Layout of an obstacle with spatial partitioning using triangles in (a) is perturbed with the rotation of the obstacle, and the triangulation is locally corrected using edges to nearest points to the obstacle corners and elimination of intersecting edges.

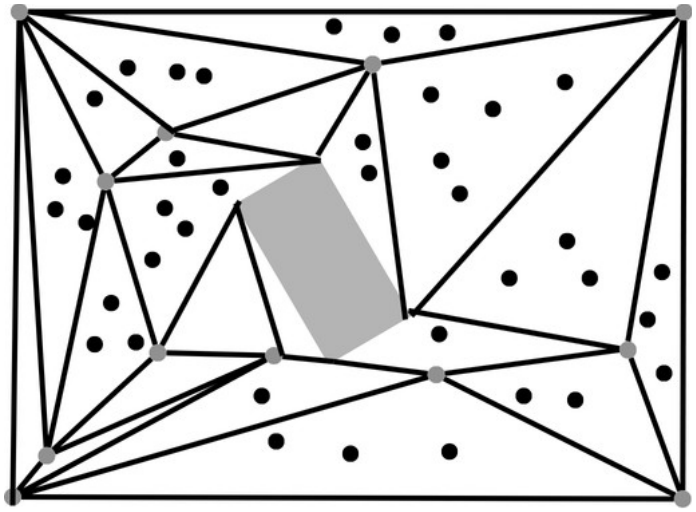

Figure 4: Layout of an obstacle with spatial partitioning using triangles and crowd rendered as black points.

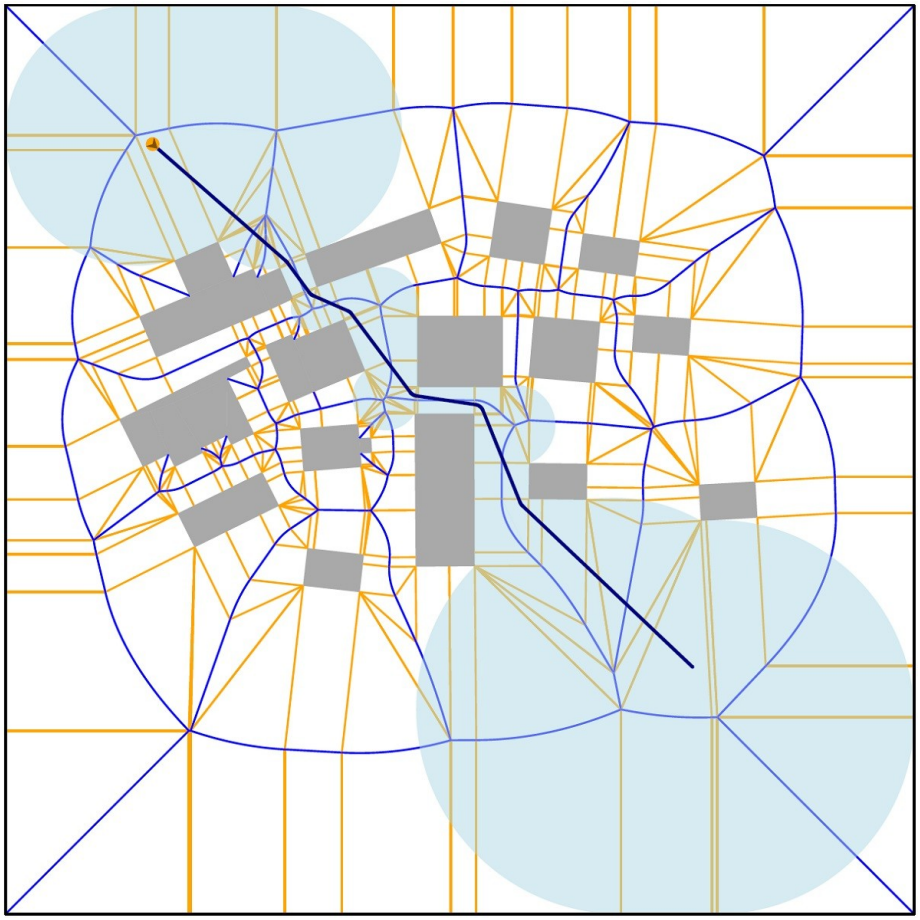

Figure 5: Another layout – the blue and black lines may be ignored.
(https://webspace.science.uu.nl/~gerae101/UU_crowd_simulation_case_studies.html)

Rubrics:

10% of final grade – Marks out of 20

- 6 marks for functionalities -- 2 marks each for correct and complete implementation of functionalities #4.1-#4.3
  - Correctness involves appropriate outcomes of choice of fixed point of rotation/scaling, distances for translation, angle of rotation, etc.
- 9 marks for user interactions – 1.5 marks for each point in #4.4
- 2 marks for report
- 2 marks for video
- 1 mark for complete and neat submission