CS606: Computer Graphics / Term 2 (2021-22) / Programming Assignment 1 (A1)
International Institute of Information Technology Bangalore

Announcement Date: Jan 25, 2022
Submission Deadline: 11:59 pm IST, Feb 07, 2022

Summary: 2D interactive planar rendering 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) events for changing control modes, and transformation implementation
- Mouse events for picking objects

Assignment: TangramApplication

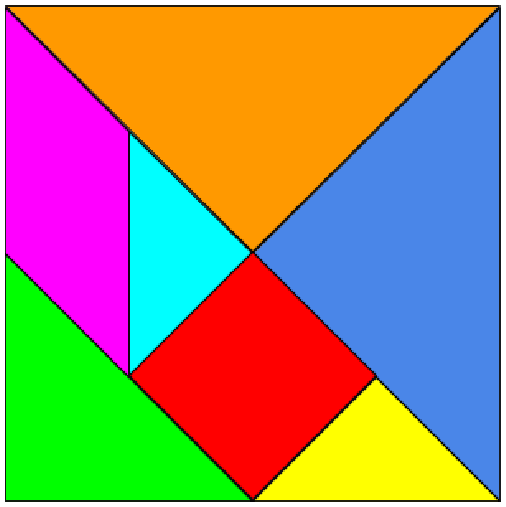

Figure 1: Tangram Puzzle pieces in initial (square) configuration C0 (http://www.controlaltachieve.com/tangrams)

The goal of this assignment is to develop a WebGL program that will replicate the user interface in the Google Drawing link given in http://www.controlaltachieve.com/tangrams

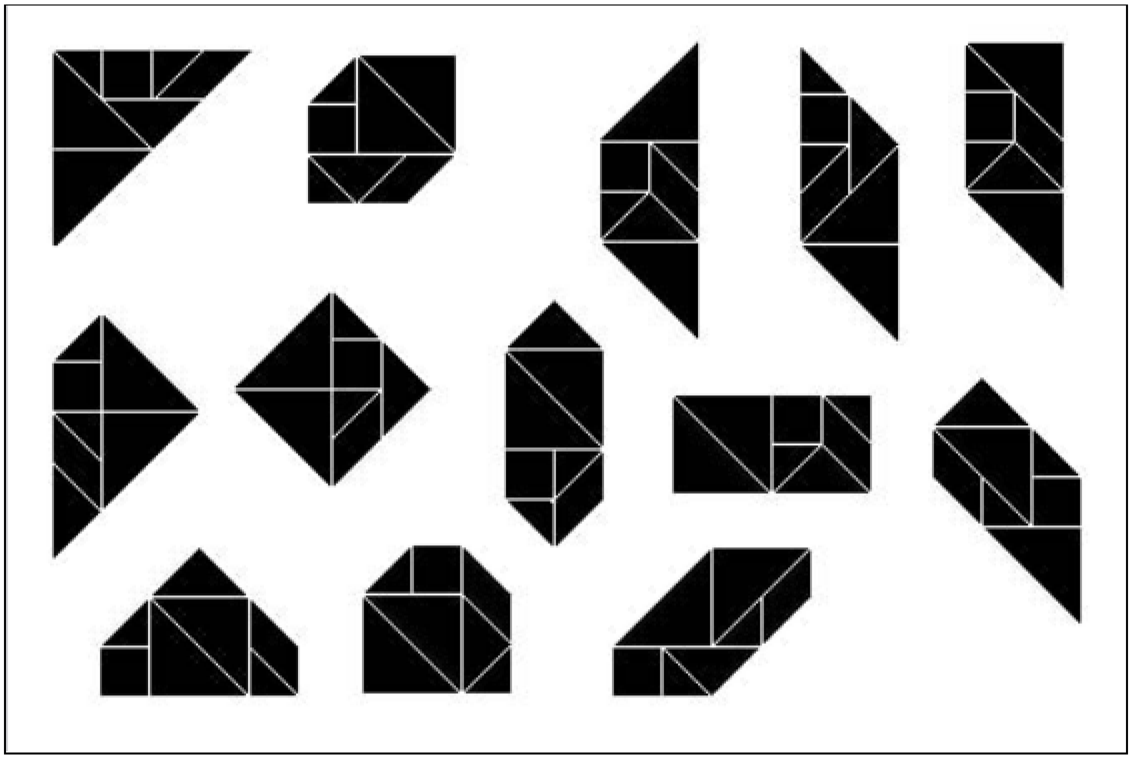

Figure 2: Sample of geometric shapes that can be made with the tangram puzzle pieces, each referred to as final configuration C1.

<u>A. Deliverables:</u>

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 (mentioned in C.4), 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 through mode-value=0,1,2,3 for 2 or 3 final configurations given in Figure 2.
    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 repository URL (Google Drive, OneDrive, etc.). 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.

<u>B. Assignment-1 Specifications</u>:

1. Primitives: the triangles, square, parallelogram - 7 pieces given in Fig.1. Use the same color scheme.
2. There are following control modes: replicate-C0-mode (Mode-0), primitive-transformation-mode (Mode-1), C1-transformation-mode (Mode-2), clear-C1 (Mode-3). *Mode-i implies the value of a variable for control mode (mode-value) is the integer i.*
3. User-interactions:
    a. Keys to select control mode:
        i. Clicking "m" (or any other key) should increment the mode-value through 0, 1, 2, 3, and then back to 0, and then repeat the increment. (Hint: use the mod function.)
        ii. If and when the mode-value changes from 3 to 0,
            1. the transformation matrix applied to C1 must be reset to the identity matrix.
            2. It can be observed as a fresh run, equivalent to restarting the application.
    b. When the state of the application is Mode-0:
        i. The canvas should have a reference-area R0 on the left, and a drawing-area R1 on the right, as shown in Figure 3. R0 and R1 can be of the same size. Hence, R0 can be chosen to be not a tight fit of the square, i.e., the C0 configuration (ref. Figure 1).
        ii. Draw C0 in the R0 and replicate it in the R1. Draw R0 and R1, as well, for reference.
        iii. This mode must draw C0 as-is (the square format) in R0, and its copy in R1, in an *exploded state* (Figure 3). The exploded state can be randomized every time mode-value=1 is visited. The positions of the primitives in the

exploded state should be such that they are entirely contained within R1.

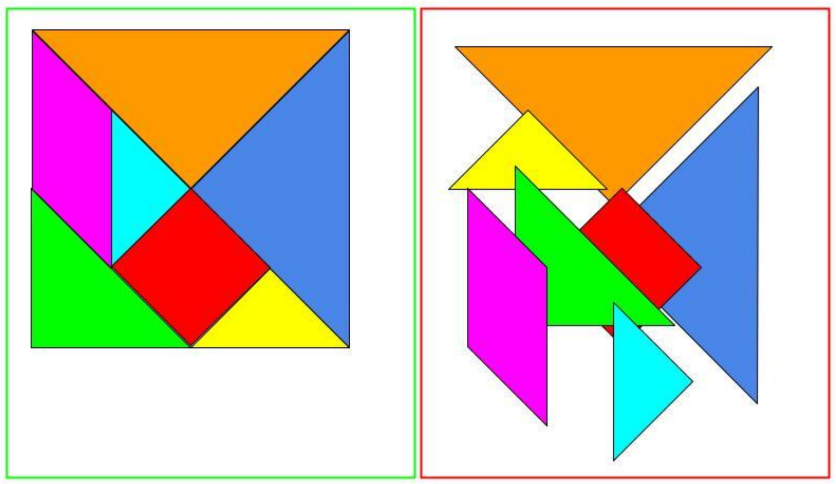

Figure 3: Mode-0 accomplishes replication of square C0 in the reference- area R0 (green box) in the left, where the copy of the pieces (primitives) are, in a random exploded state (as an example) in the drawing-area, R1 (red box)

c. When the state of the application is Mode-1:
   The primitives of R1 can be moved to form one of the configurations in C1 (Figure 2). The primitives can spill out of R1 in this mode (Figure 4), where Mode-2 is where they are contained once again in R1. Primitives in R0 do not move in this application at all and serve as a reference alone.
   i. In this mode, only objects in R1 can move, and the centroids of the primitives must stay within R1.
   ii. Use the left mouse button to pick a primitive. The position of the left mouse button upon clicking is saved, and the primitive P whose centroid is the closest to the mouse click position is selected.
      1. Once the primitive is selected, its centroid has to be updated based on the transformation, e.g. translation would require a change in centroid.
   iii. Once the primitive is selected, the primitive can be transformed using translation, rotation, scaling.
      1. Use the up/down arrow keys for translation along the +/- y-axis, and right/left arrow keys for translation along the +/- x-axis, respectively. Each such key-press should translate the object by a fixed distance (in R1) in the specified direction.
      2. Use the (/) parentheses keys for rotation about the z-axis in clockwise/counterclockwise directions, respectively. This is the

rotation of the primitive. Hence, the rotation must be about the centroid of the primitive. The quantity of rotation (i.e. the angular increment) for each key-press may be determined based on trial and error, to accomplish satisfactory user interactions.

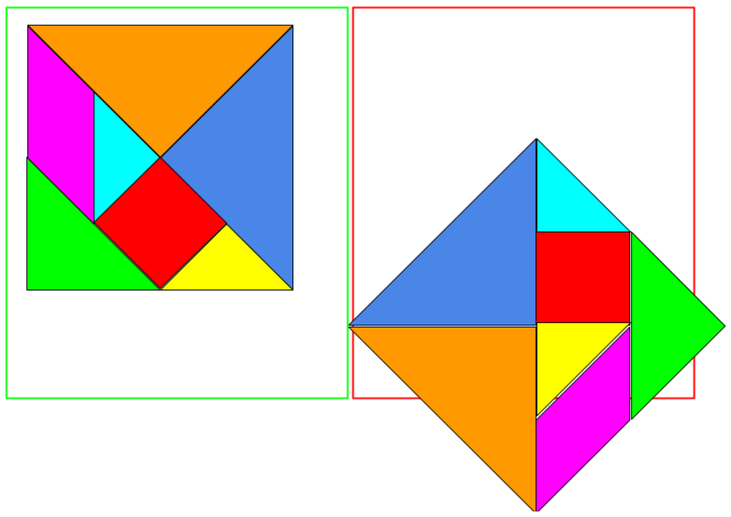

Figure 4: Mode-1 accomplishes creating one of the configurations C1 (Figure 2) in R1.

d. When the state of the application is Mode-2:
The purpose of this mode is to ensure that the contents of an imaginary box R2, which is a tight bounding box of the primitives in R1, fit well into the rectangular box R1, as shown in Figure 5
    i. Use the +/- key to scale R2 along with the contents, about the center of R2 to expand/reduce uniformly, respectively.
    ii. Use the up/down/left/right arrow keys for translation of R2 along with its contents, where the motion is as explained c.iii.1.
    iii. Use (/) keys to rotate R2 along with its contents about the center of R2, where the motion is as explained in c.iii.2.

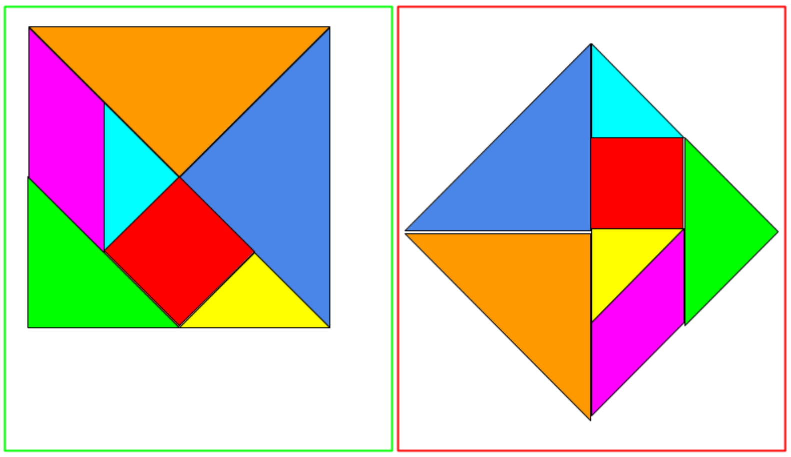

Figure 5: The primitives with C1 configuration are made to fit into R1 in Mode-2.

e. When the state of the application is Mode-3:
   The primitives in R1 must be deleted, thus clearing the drawing-area, as shown in Figure 6.

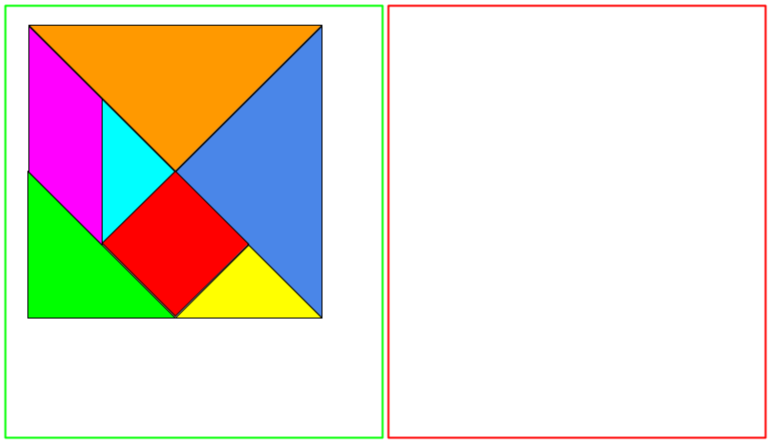

Figure 6: The primitives with C1 configuration in R1 are cleared in Mode-3.

C. Points to note about the change of mode-value and primitive-based transformations:

1. When the application is at Mode-1, the primitives have to change positions if they have been selected by clicking, and, as and when transformed by translation and/or scaling. These transformed positions have to be preserved when moving from Mode-1 to Mode-2.
2. All transformations during Mode-1 have to be tracked, to update the modelview matrix. Two alternative methodologies can be adopted to accomplish tracking the changes during Mode-1.

   a. You could work with one primitive at a time. Pick a primitive and transform its orientation and position in the exploded position to the final position in the chosen C1 configuration, and then work with the next primitive. The primitive already worked with should not be visited again.
   b. Alternatively, you could flexibly transform multiple primitives one after another until the completion of selected C1.
3. In Mode-2, the primitives are considered as a group, for which we have a (rectangular) bounding box R2, and thus, a centroid for the bounding box. Then, we rotate this bounding box essentially, and thus, its contents. Thus, we exploit this hierarchy of organization of primitives to perform the transformation. For visual debugging, you could render R2 and its centroid by pressing a specific key of your choice.
4. If no more changes are needed after Mode-2, the app should ideally have a provision to save the image that is created. But for now, one can just take a screenshot of the tangram puzzle with C0 and C1.

Questions to be answered in the report:

1. What is the difference in the implementation of the two methodologies mentioned in C.2. a. and C.2.b.? (*Hint: how would the transformation matrices for the primitives be managed?)*
2. What API is critical in the implementation of "picking" using mouse button click?
3. What would be a good alternative to minimize the number of key click events used in this application? Your solution should include how the mode-value changes are incorporated.
4. Why is the use of centroid important in transforming a primitive or a group of primitives? *(Hint: transformations such as rotation and scaling.)*
