CS606: Computer Graphics / Term 2 (2020-21) / Programming Assignment 1
International Institute of Information Technology Bangalore

Announcement Date: Jan 15, 2021
Submission Deadline: 11:59 pm IST, Jan 31, 2021

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 points and objects

Assignment: AbstractArtistApplication

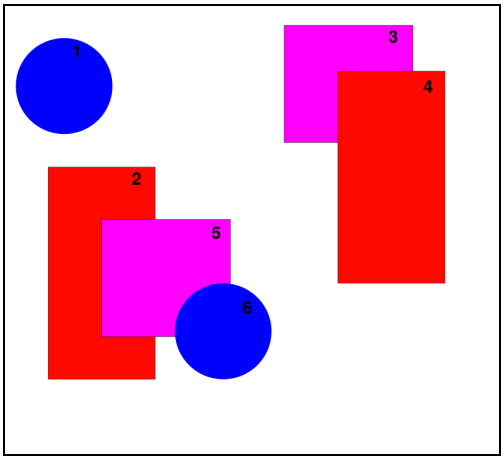

Figure 1: A sample artwork created using the application, showing the outcome of the drawing-mode. The labels on primitives are only representational for demonstrating the modifications due to change of modes, with no requirement for implementation.

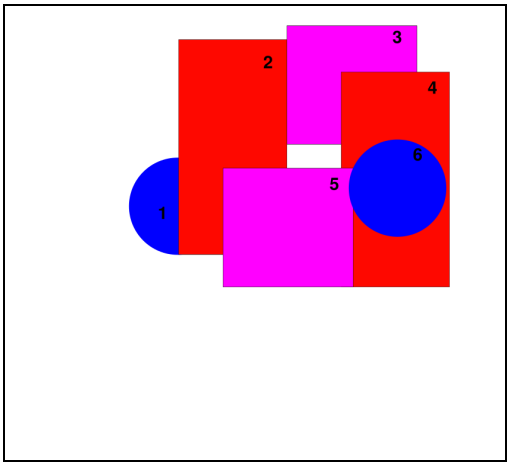

Figure 2: Outcome of the instance-transformation-mode of the sample artwork in Figure-1. The labels on primitives are only representational for demonstrating the modifications due to change of modes, with no requirement for implementation.

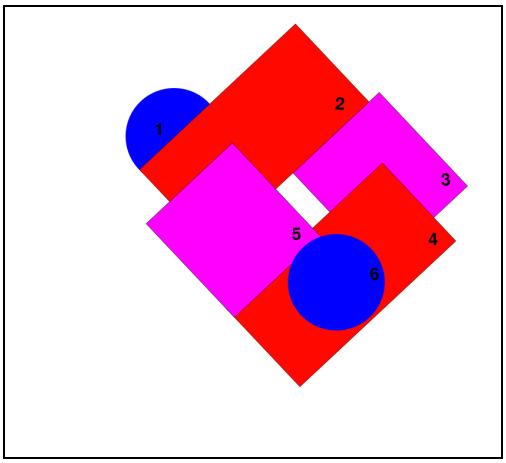

Figure 3: Outcome of the scene-transformation-mode of the sample artwork in Figure-2. The labels on primitives are only representational for demonstrating the modifications due to change of modes, with no requirement for implementation.

Deliverables:

Submissions must be made on LMS.

1. The deliverables in a zipper folder must include a source code folder, a folder with screenshots, and a demo video not longer than 5 minutes. More details on the submission are given in the course handbook on the LMS course page.
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.

Assignment-1 Specifications:

1. 3 control modes: drawing-mode (mode-value=0), instance-transformation-mode (mode-value=1), and scene-transformation-mode (mode-value=2).
2. Primitives: pertaining to unit-size; square, and rectangle of side ratio 1:2; optionally, circle, for bonus points on the assignment.
    a. Color: magenta (RGB=[1,0,1]) square, red (RGB=[1,0,0]) rectangle, blue (RGB=[0,0,1]) circle.
3. User-interactions:
    a. Keys to select control mode (Hint: clicking on the key "m" for iterating through 3 values for mode-value)
        i. Clicking "m" should increment the mode-value through 0, 1, 2, and then back to 0, and then repeat the increment. (Hint: use the mod function.)
        ii. If and when the mode-value changes from 2 to 0, the transformation matrix applied to the scene must be reset to the identity matrix.
        iii. One could also exit the program after the scene transformation is completed at mode-value=2.
        iv. Clicking on the "Esc" key should terminate/end the application.
    b. When the state of the application is at mode-value=0:
        i. Keys to select the current-shape (Hint: use "r", "s", "c" to select the current-shape to be rectangle, square, and circle).
        ii. After setting the shape, click a position in the canvas using the left mouse click. An object instance (primitive) of the current-shape must be rendered at the position of the mouse in such a way that the centroid of the primitive is placed at the position of the mouse click. (*Note: picking using mouse click will be discussed in class.*)
    c. When the state of the application is at mode-value=1:
        i. Use the left mouse button to pick an object instance (primitive). The position of the left mouse button upon clicking is saved, and the primitive P closest to the mouse click position is selected. (*Note: picking using mouse click will be discussed in class.*)
            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.
        ii. Once the primitive is selected, the primitive can be transformed using translation, scaling, or deletion.
            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 world space) in the specified direction.

2. Use the +/- keys for increasing or reducing the size of the object. Each key press could increase/reduce the object dimensions by a fixed amount, say 10%. The scaling should preserve the centroid of the primitive, and hence must be done about the primitive. The same scaling factor has to be applied to both x- and y-directions, to ensure uniform scaling.
3. Use the "x" key or optionally, the "Delete" key to delete the primitive.

d. When the state of the application is at mode-value=2:
   i. The "scene" is the bounding box of all objects in the scene, i.e., a rectangle which tightly fits all the objects. The bounding box can be aligned with the principal axes (x and y). Compute the center/centroid of this bounding box, O_S.
   ii. Use the left/right arrow keys for rotation about the z-axis in clockwise/counterclockwise directions, respectively. This is the rotation of all objects in the scene as a group. Hence, the rotation must be about O_S.

Points to note about the change of mode-value:

1. When we are in mode-value=2, 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 retained when moving from mode-value=2 to mode-value=3.
2. Now, for mode-value=3, the primitives are considered as a group, for which we have a (rectangular) bounding box 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. Now, for mode-value=1, new primitives get added. It's going to be hard to add primitives when the group is rotated in a different orientation. Hence, we will need to undo the rotations done in mode-value=3, but retain the transformations done in mode-value=2. If no more changes are needed after mode-value=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 artpiece created, and exit.

Questions to be answered in the report:

1. How did you program separate transformation matrices for all the object instances (primitives), and the scene?
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? (Hint: Consider it with respect to rotation and scaling.)
