CAP-5701 Computer Graphics

Assignment 3

FIU Knight Foundation School of Comp. & Info. Sciences

1 Exercises (40 points)

1. (10 points) Using matrix multiplication and the matrix representation of $R(\theta)$, prove that the $R^{-1}(\theta) = R(-\theta)$ (Hint: you need to show that $R(\theta) \cdot R(-\theta) = R(-\theta) \cdot R(\theta) = I$ where I is the identity matrix.

2. (10 points) Find the 3×3 matrix representation of the 2D reflection transformation about line $2y + x - 4 = 0$.

3. (20 points) Assuming that the following list of points constructs a convex irregular pentagon, find the 3×3 matrix representation of a transformation that does the following:

   - scales the pentagon with respect to its centroid and with scale factor 0.5
   - shifts the shape to the left (in the opposite direction of x-axis) so that the leftmost vertex of the pentagon (V_2) stays on y-axis.

Vertex	x	y
V_1	1	1
V_2	0	2
V_3	0.5	3
V_4	7	2.5
V_5	6	1.5

2 Programming Assignment (60 points)

For this programming assignment, you need to complete the "Screen Saver" program posted on Canvas. This program is based on the one you implemented for the second programming assignment (the polygon Shader). This program creates an animation in which polygons move around, rotate, and bounce when hitting the borders of the display window. User can make the polygons on the screen start/stop moving by pressing the space bar on the keyboard. The Screen Saver program must provide a graphical UI for the user to:

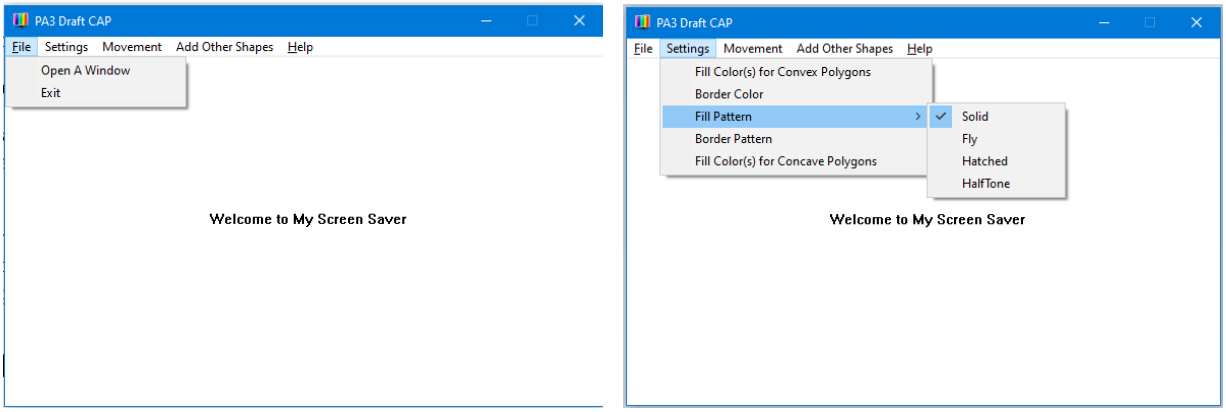

Figure 1: Menu items in the main form of the application for opening a display window and manipulating the draw/fill settings.

- Input convex polygons by moving mouse over the display window (like PA2)
- Specify the fill color(s) and fill pattern of the drawn polygons (like PA2)
- Specify the border color(s) and border pattern of the drawn polygons (like PA2)
- Specify the initial rotation direction of a polygon (clockwise or counter-clockwise)
- Specify how fast a polygon must rotate
- Specify the initial movement direction (right-up, left-up, right-down, left-down)
- Specify a polygon's constant movement speed (represented by two non-negative numbers v_x and v_y). Please note that the polygon keeps the same movement speed for its whole lifetime.
- Specify the whether or not a polygon shrinks when hitting the display window border and bounce.
- Specify the shrink ratio of a polygon in the case that it shrinks when bouncing.
- Specify how fast a polygon shrinks when bouncing.
- Input a circle and add it to the display window by entering the numerical values of its radius and the x-y coordinates of its center.

Figures 1 and 2 show the menus available in the main form of the program allowing the user to open a window, change the color settings, change the movement settings, and add a circle to the display window of program. Keep in mind that any change in the settings will affect only the polygons drawn after the change.

The user expects to see that an appropriate dialog opens up whenever one of the menu items is clicked. The user should be able to input relevant information through a dialog by typing some text and then clicking on the OK button of the dialog to confirm the input (just like the way PA2 was designed).

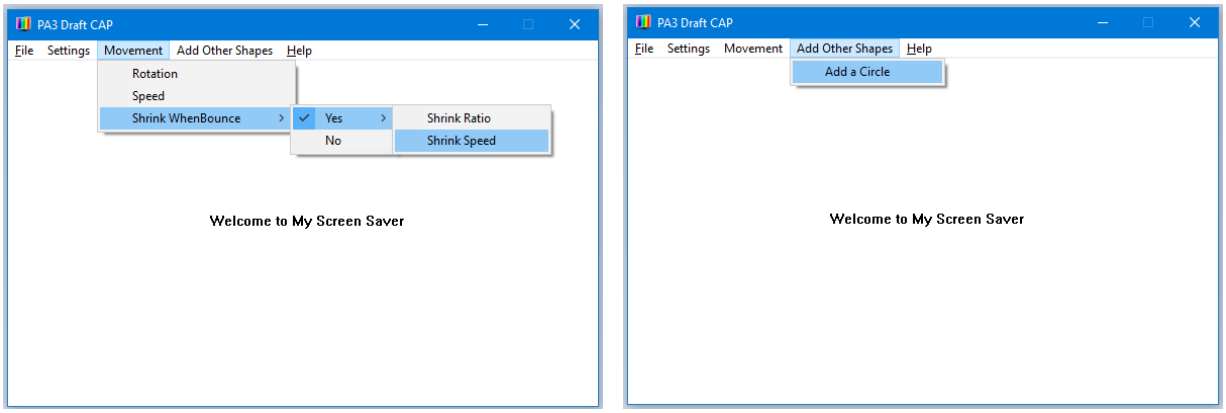

Figure 2: Menu items in the main form of the application for manipulating the movement settings and adding a circle.

2.1 Data Structures and Constants of PA3 Draft

In the initial draft of the program, the following structures and constants are defined to store the relevant information regarding the user-defined polygons:

```
#define VERTEX_NUM 101
#define POLYGON_NUM 100
#define X_SPEED 0.03//absolute value of polygon speed along x-axis
#define Y_SPEED 0.02//absolute value of polygon speed along y-axis
#define ROTATION_SPEED 0.02//degrees
#define BOUNCE_BACK_PERIOD 2000.0
#define SHRINK_WHEN_BOUNCE .15//ratio between 0 and 1
enum Movement { NONE = 0, UP = 1, DOWN = 2, RIGHT = 4, LEFT = 8,
CCW_ROTATE = 16, CW_ROTATE = 32, VERT_GROWTH = 64,
VERT_SHRINK = 128, HOR_GROWTH = 256, HOR_SHRINK = 512 };
enum Pattern{ SOLID = 0, FLY = 1, HATCHED = 2, HALF_TONE = 3 };
typedef struct { int x, y; GLfloat r, g, b, a;//rgba fill color
GLfloat br, bg, bb, ba;//rgba border color
} Vertex;
typedef struct { int centroidX, centroidY;//centroid coordinates
Pattern fillPattern; unsigned int borderPattern;
double tx, ty,//shift vector
rtheta, //rotation angle
sx, sy;//scale factor
Movement mode;//current movement
Movement pastMode;//movement before bouncing
Vertex *vertices;//vertex list
}ShadedPolygon;
```

3 Deliverables

Submit a zip file containing the following items:

- A pdf file containing your answer to the exercises.
- A readme.txt file containing the names, panther ids, and email addresses of students working on te assignment if you have done the assignment as a group of two.
- The source/resource files of your program.
- A short mp4 video (less than 3 minutes) illustrating how to interact with the program to draw and color polygons.