CS3241 Computer Graphics (2019/2020 Semester 1)

Lab Assignment 1

Release Date: 30 August 2019, Friday

Submission Deadline: 13 September 2019, Friday, 11:59 PM

LEARNING OBJECTIVES

Basic OpenGL, input & interaction, and animation. After completing the programming assignment, you should have learned

- the basic structure of an OpenGL program,
- how to use some basic OpenGL functions,
- how to use the GLUT (FreeGLUT) callback to get user input and enable interaction,
- how to use double-buffering to make animation look smoother, and
- how to use the GLUT (FreeGLUT) timer callback to control the speed of animation.

TASKS

You are provided with an **incomplete OpenGL program**, and your job is to complete it according to the following requirements. The program is supposed to do the followings:

- Opens up a blank window in the beginning.
- Lets user click the left mouse button anywhere in the window to add a disc centered at the position of the mouse cursor. The disc is given a random size (there is a lower and an upper limits), a random speed (there is an upper limit), and a random color. There is a limit to the total number of discs that the user can add.
- Once added, every disc continues to fly with a constant speed until it hits the window boundary. Then it is reflected (simple reflection with no energy loss).
- User can press the 'w' key to toggle between wireframe polygon mode and filled polygon mode.

Please download the ZIP file **Lab1_todo.zip** from the **Lab Assignments** folder in LumiNUS Files. A Visual Studio 2017 solution file **main.sln** is provided for you to build your program.

You can try the **completed Win32 executable** program **main_done.exe** found in the same ZIP file. Please read the instructions shown in the console window to learn how to operate the program. Try to resize the window and see what happens to the flying discs. Press the 'w' key to switch between wireframe polygon mode and filled polygon mode.

Follow the following instructions to complete **main.cpp** as required. You must not add or change any other files.

1) Study the source program very carefully.

2) Complete the **`DrawDisc()`** function. The function must draw the input disc using **`GL_TRIANGLE_FAN`** in its color. You can refer to
   a. https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glBegin.xml
   b. http://www.glprogramming.com/red/chapter02.html#name2

for more information on `GL_TRIANGLE_FAN`. Note that the vertices on the triangle fan must be provided in counter-clockwise direction. Since the trigonometric functions (e.g. sine and cosine) are quite expensive to compute, you should pre-compute the vertices once (for a unit-radius disc) and re-use them for all the discs later.

3) Complete the **`MyMouse()`** function. If the left mouse button is pressed, and if the maximum limit on the number of discs has not been reached, a new disc is generated centered at the position of the mouse cursor. The disc is given
   - a random radius (value is between `MIN_RADIUS` and `MAX_RADIUS`),
   - a random speed in the *x* direction (value is between `-MAX_X_SPEED` and `-MIN_X_SPEED` and between `MIN_X_SPEED` and `MAX_X_SPEED`),
   - a random speed in the *y* direction (value is between `-MAX_Y_SPEED` and `-MIN_Y_SPEED` and between `MIN_Y_SPEED` and `MAX_Y_SPEED`), and
   - a random RGB color.

4) Setting up the correct viewing in the **`MyReshape()`** function. You should use the **`glOrtho()`** or **`gluOrtho2D()`** function to set the viewing volume. The viewing volume should be set up in such a way that when the window is resized, the discs do not change their sizes or get distorted on the screen. Instead, the discs can move in the whole window interior, and get reflected by the boundaries of the new window.

5) Complete the **`UpdateAllDiscPos()`** function. This function updates the position of each disc by its speed in each of the *x* and *y* directions. At its new position, if the disc is entirely or partially outside the left window boundary, then shift it right so that it is inside the window and just touches the left window boundary. Its speed in the *x* direction must now be reversed (negated). Similar update is applied for the cases of the right, top, and bottom window boundaries.

6) Change the program to use **double-buffering**.

7) If you have a fast computer, you will notice that the animation is too fast to show anything clearly. You can see this when you run the given executable **assign1_bad.exe**. You can use the GLUT timer callback to control the speed of the animation by maintaining a constant frame rate (`DESIRED_FPS`). Refer to https://www.opengl.org/resources/libraries/glut/spec3/node64.html to find out more about the GLUT function **glutTimerFunc()**.

You may add additional constants, global variables and functions to the given program.

Beside GLUT (or FreeGLUT), you should not use any other third-party libraries. You must make sure that your code is compilable in Microsoft Visual Studio 2017 and later, and executable on a Win32 or Win64 machine.

GRADING

The maximum marks for this programming assignment is **100**, and it constitutes **6%** of your total marks for the course. The marks are allocated as follows:

- 20 marks — drawing of each disc using GL_TRIANGLE_FAN.

- 10 marks — pre-computing and storing vertex positions of disc, and correctly re-using them for drawing the discs using GL_TRIANGLE_FAN.
- 15 marks — adding a new disc when left mouse button is clicked.
- 20 marks — setting up the orthographic projection (the viewing).
- 20 marks — updating the positions of the discs.
- 5 marks — using double-buffering.
- 10 marks — using GLUT timer callback to control speed of animation.

Note that marks will be deducted for bad coding style. If your program cannot be compiled and linked, you get 0 (zero) mark.

Good coding style. Comment your code adequately, use meaningful names for functions and variables, and indent your code properly. You must fill in your **Name**, **Student Number**, and **NUS email address** in the **header comment**.

SUBMISSION

For this assignment, you need to **submit only** your completed **main.cpp**.

You must put it/them in a ZIP file and name your ZIP file ***your-student-number*_lab1.zip**. For example, **A0123456X_lab1.zip**. All letters in your student number must be capitalized.

Submit your ZIP file to the **Lab 1 Submissions / Group T0*x*** folder in LumiNUS Files, where **T0*x*** is your officially allocated Tutorial group number. Before the submission deadline, you may upload your ZIP file as many times as you want to the correct folder. **We will take only your latest submission.** Once you have uploaded a new version to the folder, you **must delete the old versions**. Note that when your file is uploaded to the folder, the filename may be automatically appended with a number. This is fine, and there is no need to worry about it.

DEADLINE

Late submissions will NOT be accepted. The submission folder will automatically close at the deadline.

——— End of Document ———