Computer Graphics, 2018

Assignment 1: Moving Circles
Submission Due: Oct. 14 (Sun.), 23:59, at i-campus

First, refer to the general instruction for computer graphics assignments, provided separately. If somethings do not meet the general requirements, you will lose corresponding points or the whole scores for this assignment.

1. Objective

In computer graphics, the majority of drawing primitives are triangles. The goal of this assignment is to learn how to draw and animate such primitives (here, 2D). Precisely, you need to draw and move 2D colored circles on the screen.

2. Triangular Approximation of Circle

In general, a circle is represented by its center position and radius. To draw the circle, we first need to convert it to a finite set of triangle primitives. A common way of doing this is to approximate the circle by dividing it to a number of equal-size triangles. This process is called the "tessellation" or "subdivision."

Figure 1 illustrates how to approximate a circle using 8 triangles resulting in an octagon. For your implementation, use more triangles (e.g., 36 or 72 triangles) to obtain a smoother boundary.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

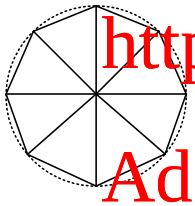

Figure 1: *An octagonal approximation of a circle.*

Given a center position $\mathbf{p}_0 = (x_0, y_0)$ and a radius r, you can compute the coordinates of each boundary vertex as

$$\mathbf{p}_0 + r(\cos\theta, \sin\theta),$$

where θ indicates the angle at the xy-plane. In your shader program, you will need to use 4D coordinates for `gl_Position`; in that case, use `vec4(`$x_0 + r\cos\theta$`, `$y_0 + r\sin\theta$`, 0, 1)`, where 0 indicate the z coordinate. For the moment, do not care about the last component 1 (this will be covered later in the course).

Also, recall that the default viewing volume is provided with $[-1, 1]$ for the x, y, and z axes.

3. Requirements

You may start from "cgbase" project, already distributed on the class web.

- If ESCAPE or 'q' keys are pressed, the program is terminated.
- Initialize the window size to 1280×720 (10 pt).
- The program has more than 20 solid circles (Figure 2) (10 pt).
- Your circle should be a circle (*not an ellipse*), even when the window is being resized (10 pt).
- The circles are initialized with random position, radius, color, and speed (10 pt).
- The circles' initial positions require to avoid collision (20 pt).
- The circles are moving in random directions (10 pt).
- When a circle meet other circles or side walls, its direction and speed are randomly changed to avoid overlaps (20 pt).

When you animate circles, it is better to compute the amounts of movements based on the *time* difference between the current and previous frames rather than simply using a frame counter. This ensures that your application runs at the same speed for different machines. It is also good to test your application in a different (e.g., your friend's) machine in advance.

Figure 2: *An example screenshot of a single animation frame.*

4. Mandatory Requirements

What is listed below is mandatory. So, if anything is missing, you lose 50 pt for each.

- Instancing should be applied for a static vertex buffer. Your vertex buffer should stay constant at run time, which means positioning and coloring of the circles should be implemented using *uniform* variables.

5. (Optional) Elastic Collision

It would be better to apply physics to the collision among the circles rather than to rely solely on the random control over velocities; refer to the following link for the elastic collision:

https://en.wikipedia.org/wiki/Elastic_collision

When you apply a physically-based collision to your assignment, you get an extra credit of 20 pt. Please note that in your report.

6. What to Submit

- Source, project, and executable files (90 pt)
- A PDF report file: YOURID-YOURNAME-A1.pdf (10 pt)
- Compress all the files into a single archive and rename it as YOURID-YOURNAME-A1.7z.