CSUS
COLLEGE OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science

CSc 133 – Object-Oriented Computer Graphics Programming
Fall 2022 – Section 2
Dr. Muyan

Final Exam Study Guide

The Final Exam will be conducted in-person (in RVR 1013) on Dec 13, 2022 (Tuesday) starting at **12:45 pm (45 minutes later than the scheduled start time of the lecture)**. You will have 60 minutes to complete the exam. The exam will be *closed book* and *closed notes* and will consist of multi-choice and short-answer questions.

To be well prepared for the Final Exam, you should have *read all lecture notes, attended all lectures, completed all the reading and programming assignments*. In addition, you should be able to answer questions based on the numbered topics listed below.

(1) Answer questions based on the material listed in the Study Guide for the Midterm Exam. The emphasis on the Final Exam will be weighted toward material covered since the Midterm, but you should expect there will also be questions based on what is listed in the Midterm Exam Study Guide.

Animation

(2) Explain what is meant by frame-based animation. Be able to give a pseudo-code description of the basic structure of a frame-based animation program.

(3) Explain what collision detection and collision handling mean. Include a description of collision detection using bounding circles. Describe the interfaces associated with collision detection, and be able to write a code to perform collision detection and handling.

(4) Be able to write code using a CN1 **UITimer** object to perform simple animation.

Sound

(5) Explain what is meant by sampled audio. Include a description of the parameters which affect a sampled sound.

(6) List the names of several common sound file formats, and of several common sound APIs.

(7) Be able to write code that plays regular or background (looping) sounds using related build-in CN1 classes.

Transformations and Their Applications

(8) Explain and be able to work with common two-dimensional (2D) vector algebra notions.

(9) Explain and be able to work with common matrix operations such as addition, multiplication, transpose, etc.

(10) Be familiar with the matrix representation for common 2D affine transformations (translation, scaling, rotation), and with the concatenation of translations via matrix multiplication. Explain the advantages of using matrix representation and operations for transformations.

(11) Explain what is meant by the term *homogeneous coordinate*, and describe how to convert a homogeneous coordinate to a real coordinate and vice versa.

(12) Give a general description of the CN1 **Transform** class, and give an example showing what it can be used for.

(13) Explain how CN1 **Graphics** objects relate to the notion of a *Graphics Transform Stack*.

(14) Describe the overall organization of a GUI-based program which allows a user to draw various geometric shapes on the display by applying series of transformations to these shapes.

(15) Use trigonometry to derive the general formula for rotation of a point (X,Y) about the origin by an angle theta.

(16) Show how the formula for rotation (above) can be implemented by matrix multiplication.

(17) Be familiar with the difference between “multiplication from the left” and “multiplication from the right” with respect to matrix operations. Be able to write CN1 code which concatenates transformations in the proper order to achieve a specified result.

(18) List different coordinate systems used in a graphics program and explain the relationships between them, including how to convert from one to the next.

(19) Explain what is meant by the terms “window” (in the sense of a world coordinate system), “normalized device”, “viewing transformation”, and “display-mapping transformation”.

(20) Explain how a given hierarchical object can be composed of a collection of sub-objects and how dynamic transformations can be used to alter object’s transformations on-the-fly.

(21) Explain the relationship between “zoom” and “pan” operations and the world window. Be able to write CN1 code showing how these operations are performed and can be attached to pinching and pointer dragging.

(22) Explain how the introduction of a world coordinate system changes the way in which on-screen graphical object selection is done, and how this relates to the “inverse of viewing transformation” and “inverse of concatenation of local transformations”. Describe the sequence of steps required to determine if a screen point (such as from a pointer-pressed) lies within a given object described in local (object) coordinates.

(23) Describe the overall sequence of steps in the “graphics pipeline” – the steps involved in a **repaint()** operation, starting from a collection of objects each defined in its own local coordinate system and ending with a set of display drawings.

(24) Assuming a shape (e.g., a rectangle) contains a local transformation specifying how that shape should be transformed when it is drawn, provide CN1 code showing the correct way of adding the object’s local transformation as part of the drawing process.

Lines and Curves

(25) Explain what a parametric equation is and how it is used in a graphics system. Give the general form for a parametric point P(t) on a line between two points P0 and P1.

(26) Give a geometric description of a quadratic or cubic Bezier curve.

(27) Explain what is meant by the term *weight* (*blending function)* in relation to Bezier curves.

(28) Explain the relationship between the number of control points that define a Bezier curve and the degree of polynomial functions that define the weights of that Bezier curve.

(29) Given a set of control points, be able to sketch by hand a reasonable approximation of the corresponding Bezier curve.

(30) Give two distinct algorithms for rendering Bezier curves, and explain the characteristics and advantages of each.

(31) Describe a common use for Bezier-like curves.