UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS344 - Computer Graphics

Practical 2 Specification: 2D Rendering

Release Date: 17-02-2024 at 06:00

Start By Date: 03-03-2024

Due Date: 20-03-2024 at 09:00

Total Marks: 92

Contents

1 General Instructions **3**

2 Overview **3**

3 Your Task: **3**

3.1 Floor Plan Requirements 4

3.2 Shape Requirements 4

3.3 Colour Requirements 5

3.4 Selection Requirements 5

3.5 Transformation Requirements 5

3.6 Wireframe 8

4 Marking rubric **8**

5 Bonus marks **11**

6 Implementation Details **11**

7 Submission **12**

8 Demo Instructions **12**

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually, no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as **no extension will be granted.**
- If your code does not compile, you will be awarded a mark of 0. The rendering output of your program will be primarily considered for marks, although internal structure may also be tested (eg. the presence/absence of certain functions or classes).
- Failure of your program to successfully exit will result in a mark of 0.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at http://www.ais.up.ac.za/plagiarism/index.htm.
- You are allowed to use any standard of C++.
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.
- No pre-build objects and textures may be used. All objects and textures that you need to use must be created by yourself.
- You must use OpenGL version 3.3 for this practical.

2 Overview

For this practical, you will need to render a scene populated with simple 2D objects and apply a set of transformations to the objects.

3 Your Task:

For this practical, you will need to render a simple 2D floor plan of the IT Kiosk. This will aid you in the design and layout of the IT Kiosk for the homework assignment. Each type of furniture/decor should be represented with a unique colour and shape combination. An illustrative example of a floor plan is given in Figure 1.

In the sections below, the different requirements are laid out. You are allowed to use any colour/shape combination to represent different types of objects in the floor plan. You may orientate the floor plan in any direction you like, as long as it is a realistic representation of a 2D square floor plan.

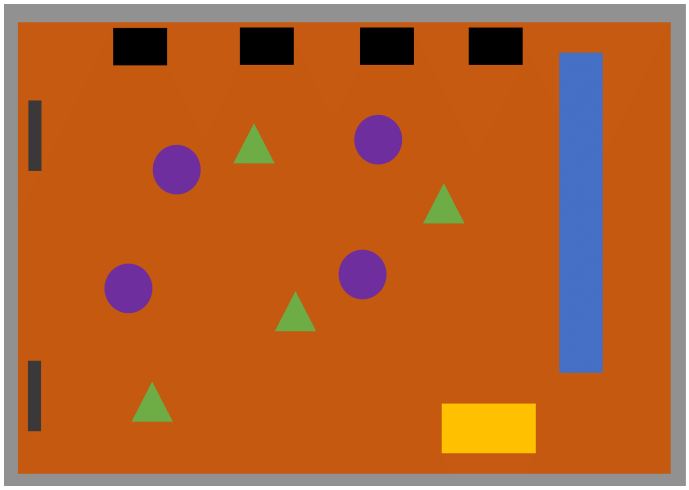

Figure 1: Layout Example

3.1 Floor Plan Requirements

The floor plan should have the following example visual components[1]:

- A shape indicating the floor of the IT Kiosk (represented in orange). Note the floor plan may not occupy the entire screen. There should be a border between the floor plan and the edges of the window.
- Entry doors for the IT Kiosk (represented by grey rectangles).
- The counter where orders are placed (represented by a blue rectangle).
- At least 3 different types of seating (one type is represented by the purple circles).
- At least 2 different types of decor (pot plants, dustbins, tables, etc.). The green triangles represent one example.

Your floor plan can contain more details if you wish to add it.

3.2 Shape Requirements

Your floor plan is required to at least contain two of the following distinct[2] shape types:

- Rectangle/Square
- Triangle
- High polygon circle
  - A circle that consists of at least 50 vertices
- Low polygon circle
  - A circle that consists of a vertex count of between 6 and 10.

[1] All shapes refer to examples in Figure 1

[2] Distinct implies that the shape as a whole is counted and not the internal polygons used to create the shape.

Note that you can use as many shapes as you like to build your floor plan, as long as it complies with the above requirements.

3.3 Colour Requirements

Your floor plan needs to fulfil the following requirements in terms of colour.

- At least 4 distinct colours to represent different types of objects.
- A floor colour that has to be a distinctive colour.
- The background colour has to also be a distinctive colour. *Hint: there is a opengl function that can assist with this.*

In conclusion, your floor plan needs to contain at least 5 different distinctive, solid colours with the background also being a distinctive colour.

3.4 Selection Requirements

In order to perform the transformations or animations listed in the following section, you need to be able to select an object to apply the transformation to.

- When the `1 key` is pressed, a seating object needs to be selected.
- When the `2 key` is pressed, a different seating object of a different type needs to be selected.
- When the `3 key` is pressed, a decor object needs to be selected.
- When the `4 key` is pressed, a different decor object of a different type needs to be selected.
- When the `0 key` is pressed, the selected object, if any, needs to be deselected.

The selected object should be highlighted by having the colour change to a pastel version of the original colour once the object has been selected. Note each repeated key press (i.e. the same key being pressed), does not need to select a different object. In other words, each key is bound to a single object of the required type.

3.5 Transformation Requirements

In this section are the transformation or animation requirements for your rendering. Note, keep the animation rates small such that it allows you to press the key multiple times before going out of camera view.

- When the `W key` is pressed, the selected object needs to move from its current position to the top of the screen.
- When the `S key` is pressed, the selected object needs to move from its current position to the bottom of the screen.

- When the `A key` is pressed, the selected object needs to move to the left of the screen.
- When the `D key` is pressed, the selected object needs to move to the right of the screen.
- When the `+ key` is pressed, the selected object needs to be scaled up, from the centre of the selected object.
- When the `- key` is pressed, the selected object needs to be scaled down, from the centre of the selected object.
- When the `E key` is pressed, the selected object needs to be rotated clockwise around the centre of the selected object.
- When the `Q key` is pressed, the selected object needs to be rotated counter-clockwise around the centre of the selected object.

Below are examples of the result after multiple key presses assuming the selected object is at the original rendering position.

Note that when a key is pressed the selected object should **not** return the original position. Your program should be able to apply the transformation in any arbitrary order with the selected object at any position on the screen.

The pink and white-green objects represent selected objects.

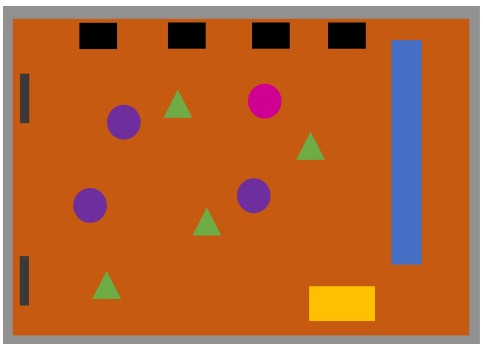

Figure 2: Original

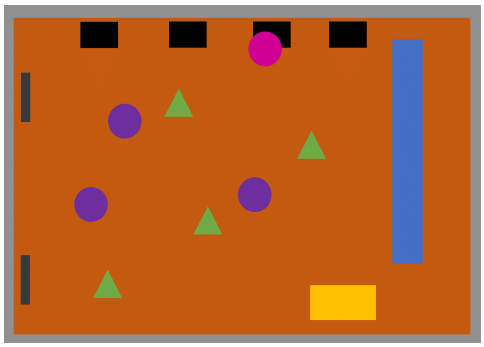

Figure 3: After a series of `W key` presses

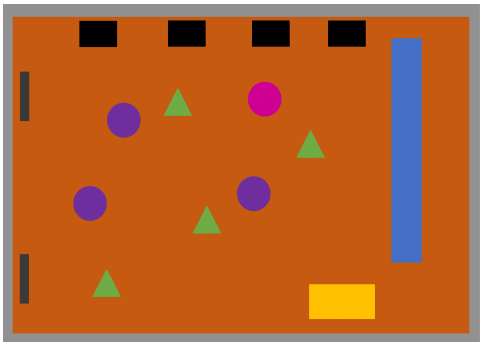

Figure 4: Original

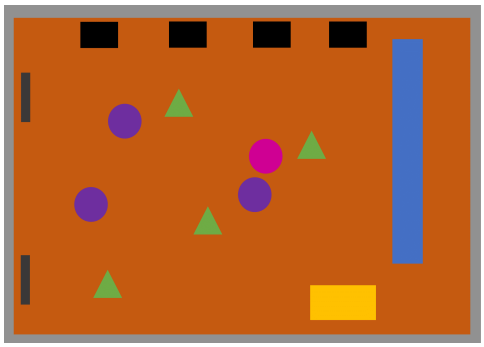

Figure 5: After a series of `S key` presses

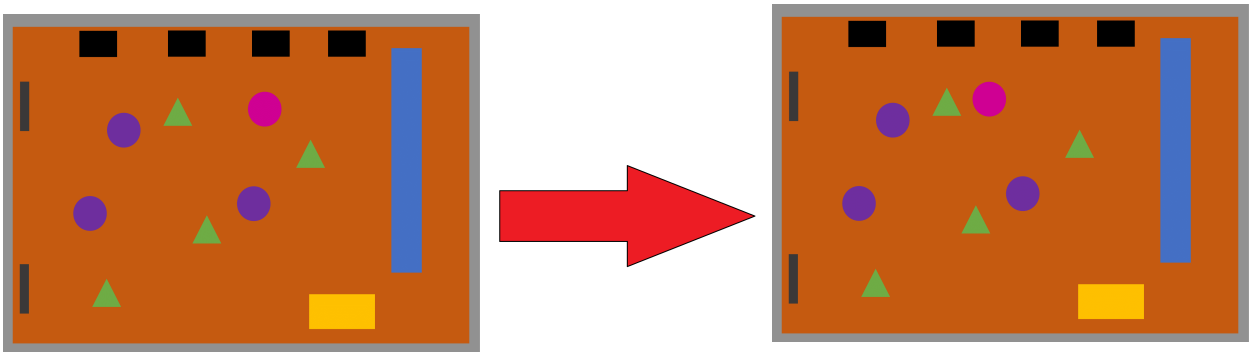

Figure 6: Original

Figure 7: After a series of `A key` presses

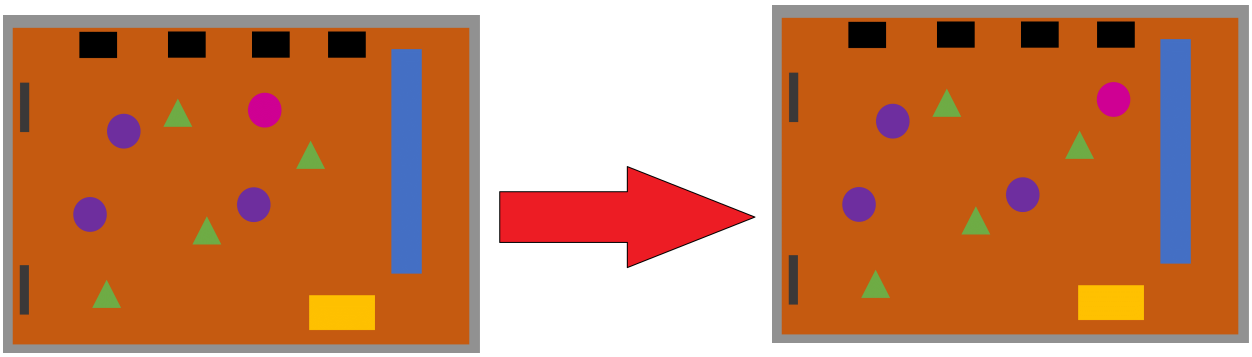

Figure 8: Original

Figure 9: After a series of `D key` presses

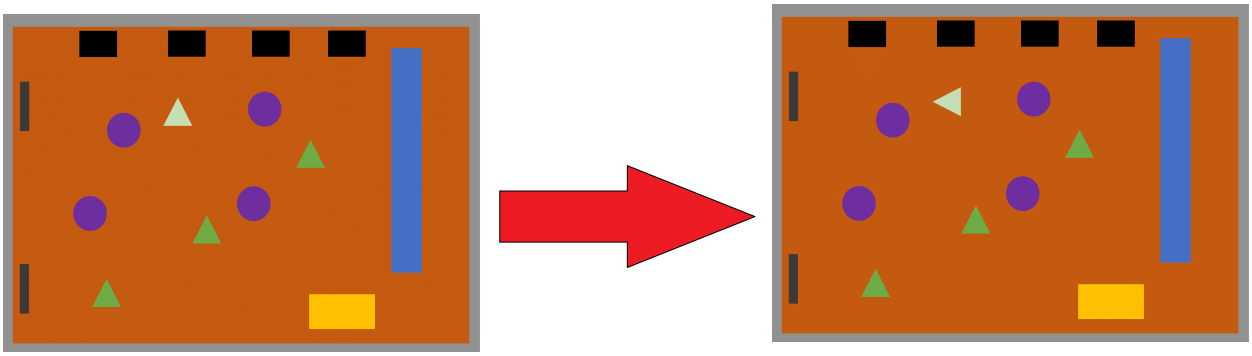

Figure 10: Original

Figure 11: After a series of `Q key` presses

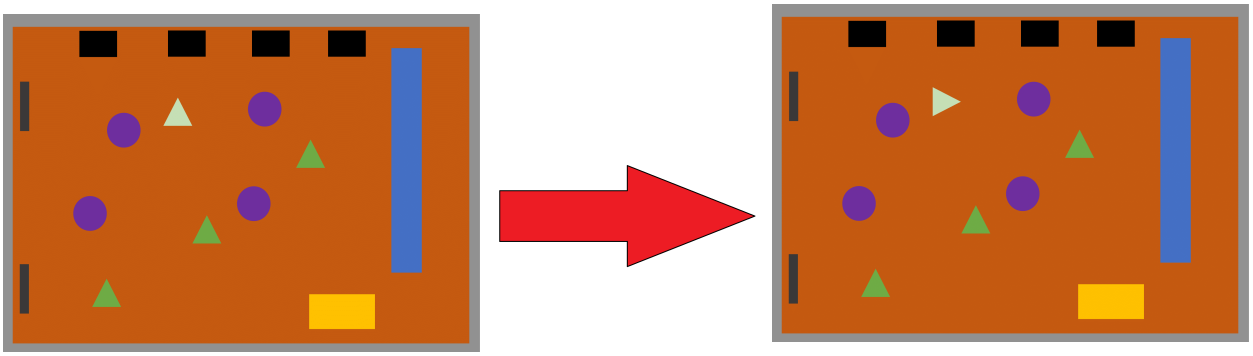

Figure 12: Original

Figure 13: After a series of `E key` presses

3.6 Wireframe

You are also required to implement a wireframe for your floor plan. The wireframe should maintain the colour scheme of your floor plan, such that the colours of the different shapes can be identified. The wireframe should also conform to all the transformations described earlier.

Your program should toggle between the wireframe and normal floor plan when the `Enter key` is pressed. *Hint: you may need to implement a time delay between key presses for the wireframe toggling such that the expected behaviour is achieved.*

Please note that you **must** use the `GL_LINES` to implement your wireframe. Using the `glPolygonMode` function will result in the forfeiting of your wireframe marks.

Please see the figure below for an example of a wireframe rendering of the floor plan.

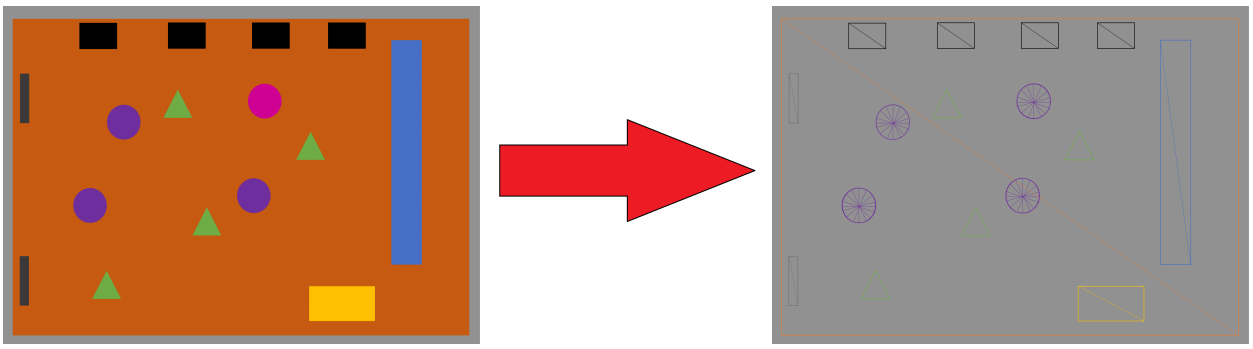

Figure 14: Original

Figure 15: Wireframe

4 Marking rubric

The following rubric will be used to mark your submitted assignment. Note you will be demo-ing the practical during the practical sessions on your own computer or a lab computer. Please see Table 1 for the rubric. Note: 1 mark will be subtracted for each transformation assessment criteria if the render is moved back to the centre before a new transformation is applied.

Assessment Criteria	0	1	2	3	4
Floor Plan Requirements [18 marks]					
Shape indicating floor plan	No shape		Shape occupies entire window		Correct floor plan shape
Entry doors to the IT Kiosk	No doors	Single Door			Two Doors
Order Counter	No counter		Counter Present		
Three different types of seating	No seating object types	Only one type of seating object	Only two types of seating objects		Three or more seating object types
Two different types of decor	No decor object types	Only one type of decor object			Two or more decor object types
Shape Requirement [16 marks]					
Two distinct rectangles or squares	No rectangles or squares		Only one distinct rectangle or square		Two or more distinct rectangles or squares
Two distinct triangles	No triangles		Only one distinct triangle		Two or more distinct triangles
Two distinct high polygon circles	No high polygon circles		One distinct high polygon circle		Two or more high polygon circles
Two distinct low polygon circles	No low polygon circles		One low polygon circle		Two or more low polygon circles
Colour Requirements [8 marks]					
Four distinct colours for objects	No colour	All objects have the same colour	Two distinct colours	Three distinct colours	Four distinct colours
Distinct floor colour	No floor colour	Floor colour same colour as objects	Distinct floor colour		
Distinct background colour	No background colour	Background colour not distinct	Background colour distinct		
Selection Requirements [18 marks]					
Seating object type 1	Not selectable		Selectable with incorrect key	Selectable with correct key but no colour change	Selectable with correct key press
Seating object type 2	Not selectable	Selectable with seating object 1	Selectable with incorrect key	Selectable with correct key but no colour change	Selectable with correct key press
Decor object type 1	Not selectable		Selectable with incorrect key	Selectable with correct key but no colour change	Selectable with correct key press
Decor object type 2	Not selectable	Selectable with decor object 1	Selectable with incorrect key	Selectable with correct key but no colour change	Selectable with correct key press
Unselecting object type	Not implemented	Implemented but no colour change	Correctly implemented		

Assessment Criteria	0	1	2	3	4
Transformation Requirements [20 marks]					
Horizontal translations	The selected shape is not able to move		The selected shape is able to move only in one direction		The selected shape is able to move in both directions
Vertical translations	The selected shape is not able to move		The selected shape is able to move only in one direction		The selected shape is able to move in both directions
Scaling	The selected shape is not able to scale		The selected shape is able to scale only in one direction		The selected shape is able to scale in both directions
Rotation	The selected shape is not able to rotate		The selected shape is able to rotate only in one direction		The selected shape is able to rotate in both directions
Transformations applied to all selected shapes	No shape is able to transform	Only one of the selectable shapes is able to perform all transforms	Only two of the selectable shapes is able to perform all transforms	Only three of the selectable shapes is able to perform all transforms	Four or more of the selectable shapes is able to perform all transforms
Wireframe [12 marks]					
Render	No wireframe		Partial wireframe		Correct wireframe
Colour	No wireframe		Partially coloured		Correctly coloured
Transformations	No wireframe		Partial transformations		Correct transformations

Table 1: Marking Rubric

5 Bonus marks

There are up to 10 marks available for bonus marks. The following will only be considered for bonus marks, and are worth 2 marks for each extra that you render or implement (max 10 marks).

- Allowing the selection of more than one object to apply a transformation to.
- Allowing objects to be selected by clicking on them.
- Allowing bounds checking so objects cannot be moved off-screen.
- Enabling the dynamic addition or removal of objects on the screen.
- Adding extra details to objects such as patterns. Note: you may **not** use in texture mapping for this.
- Being able to save the floor plan to a text file.
- Being able to load the floor plan from a text file.
- Being able to save a screenshot of the floor plan. Note this has to be done by the program using openGL.

6 Implementation Details

- You need to use OpenGL version 3.3 for this practical.
- You may **not** use any of the built-in mathematical libraries within the `glm` package. This included matrix arithmetic. *Hint: You may use your practical 1 in this practical.*
- You may **not** use any of the built-in OpenGL functions to generate the shapes for you. You need to create each shape from first principles.
- You may **not** use any of the built-in OpenGL functions to perform the transformations of the shapes. You need to transform each shape from first principles, either explicitly or by using the matrix arithmetic techniques discussed in class.
- You may only use the following C++ and OpenGL libraries:
  - stdio.h
  - stdlib.h
  - iostream
  - iomanip
  - cmath
  - sstream
  - GL/glew.h

  - GLFW/glfw3.h
  - glm/glm.hpp

  You may also use the `shader.hpp` and `glad.c` files that assist with compiling and linking of shaders. Your code should be able to be compiled without the assistance of an IDE, i.e. the project needs to be able to be compiled from terminal.

- All your helper classes and files needs to be in the same directory of the main.cpp.
- Ensure that the title of the window of your program is your correct student number.

7 Submission

You are required to submit on ClickUp under the appropriate submission link. In the archive that you submit, include a makefile and compiling instructions such that the program can be compiled and executed by the markers if needed. *Failure to upload to ClickUP will result in you forfeiting all marks for this practical.* No exceptions will be made on this matter.

8 Demo Instructions

1. You will first be required to download your submission from ClickUP.
2. You will then demo your practical to the tutor.
3. In the presence of the tutor, you will be required to upload the archive you downloaded from ClickUP to FitchFork. *Failure to upload to FitchFork will result in you forfeiting all marks for this practical.* No exceptions will be made on this matter.