МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Практичні роботи

Minor *«Розробник ігрових додатків»*

дисципліна *«Комп’ютерна графіка з OpenGL»*

(назва дисципліни)

Виконав: студент 3 курсу групи  *335а*

напряму підготовки (спеціальності):

*122 “Комп’ютерні науки”*

(шифр і назва напряму підготовки / спеціальності)

*Мінаков А.А.*

(прізвище й ініціали студента)

Прийняв: *доц. каф 603, к.т.н, Лучшев П.О.*

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:

Кількість балів:

Оцінка ECTS:

Зміст

[Практична робота 1. Основні принципи роботи з OpenGL 3](#_Toc177946558)

[Завдання, варіант № 14 3](#_Toc177946559)

[Системна інформація 3](#_Toc177946560)

[Теоретичні відомості 3](#_Toc177946561)

[Результати виконання практичної роботи](#_Toc177946562) 4

[Загальний перелік посилань 8](#_Toc177946571)

[Додаток А. Лістинг програми до практичної роботи №1 9](#_Toc177946572)

# Практична робота 1. Основні принципи роботи з OpenGL

## Завдання, варіант № 14

За допомогою інструментальних засобів, зазначених викладачем, створити простий програмний проєкт із підтримкою бібліотеки OpenGL. Розробити програму із застосуванням команд OpenGL, яка встановлює анізотропну систему координат, створює та виводить варіант зображення на екран/у вікно з урахуванням заданих примітивів та координат x1, y1 та x2, y2 . Для рисування координатної сітки необхідно використовувати пунктирні лінії. Контур фігури, осі та координатну сітку зобразити лініями різної товщини. Для парних варіантів точки повинні мати квадратну форму, а для непарних – круглу.

|  |  |  |
| --- | --- | --- |
| 14 | Примітиви:  GL\_POINTS, GL\_LINE\_STRIP  x1 = -2;  x2 = 7 y1 = -4;  y2 = 0 |  |

## Системна інформація

Для розробки та виконання практичних робіт використовувалися наступні апаратні та програмні засоби:

Processor AMD Ryzen 5 3600 6-Core Processor 3.60 GHz

RAM 16.0 GB

System type 64-bit operating system, x64-based processor

Edition Windows 10 Pro Version 22H2

IDE Microsoft Visual Studio Community 2022 (64-bit) version 17.8.6

## Теоретичні відомості

### Робота з графічними примітивами

Для відображення графічних елементів у OpenGL використовуються декілька команд, серед яких:

* glColor3d() — для задання кольору примітивів;
* glLineWidth() — для задання товщини ліній;
* glEnable() / glDisable() — для активації та деактивації різних функцій, наприклад, пунктирних ліній;
* glLineStipple() — для налаштування пунктирних ліній.

У нашому коді для малювання координатної сітки, осей та фігури використовувалися ці команди.

### Побудова координатної сітки

Для відображення сітки використовувалися пунктирні лінії, які створюються за допомогою функції glLineStipple(). Сітка дозволяє точно визначати позиції точок на площині та задавати контури фігур. Основні кроки для побудови сітки:

1. Встановлення кольору сітки командою glColor3d().
2. Використання функції glLineWidth() для задання товщини ліній.
3. Активація режиму пунктирних ліній командою glEnable(GL\_LINE\_STIPPLE).
4. Використання вектора для зберігання координат вершин сітки, який передається до OpenGL через функцію glVertexPointer().

### Малювання осей координат

Для побудови осей використовуються аналогічні методи з деякими змінами. Осі відображаються за допомогою товстих ліній, для яких задається інший колір і товщина. У нашому випадку осі зображаються жовтим кольором.

### Малювання полігонів та точок

В програмі передбачено малювання полігонів та окремих точок. Для цього були створені спеціальні класи, такі як Point та MyPolygon. Клас Point відповідає за відображення окремих точок на екрані, тоді як MyPolygon — за малювання полігонів, що складаються з кількох точок.

Для кожного примітиву використовується метод draw(), що викликає необхідні OpenGL-функції для відображення об’єктів.

## Результати виконання практичної роботи

### Розв'язання завдання

1. **Використання команд управління параметрами графічних примітивів:**

Для управління параметрами графічних примітивів були використані наступні команди (Додаток А):

* Колір: glColor3d(), рядки: 21, 49, 62, 112 та 127 у файлі Render.cpp.
* Тип: glLineStipple(), glEnable(), glDisable(), рядки: 23, 24 та 45 у файлі Render.cpp.
* Товщина: glLineWidth(), рядки: 22, 50, 63, 74, 87 та 128 у файлі Render.cpp.

1. **Коректне відображення завдання під час змінення розмірів/положення вікна:**

Коректне відображення завдання під час зміни розмірів вікна продемонстровано на рис. 1.1 та 1.2.

1. **Розроблення підпрограм для виключення дублювання коду:**

Підпрограми для уникнення дублювання коду були розроблені у рядках 20–28 файлу Render.cpp.

1. **Застосування циклів для створення зображень:**

Цикли для автоматичного малювання графічних елементів, таких як сітка, осі і відмітки, використовуються у рядках 35–46 та 86-93 файлу Render.cpp.

1. **Формування зображення векторними командами OpenGL:**

Векторні команди OpenGL, такі як glDrawArrays(), були використані для оптимізації рендерингу. У прикладі малювання сітки, осей та додаткових ліній використовується glDrawArrays() замість багатьох викликів окремих вершин. Це можна знайти у рядку 26 файлу Render.cpp.

1. **Використання ООП (розроблення власних класів):**

Використання об’єктно-орієнтованого підходу було реалізовано за допомогою класів Point та MyPolygon, які наведені у файлі Render.cpp (рядки 102 – 132). Клас Point відповідає за відображення точок, а клас MyPolygon — за побудову фігури.

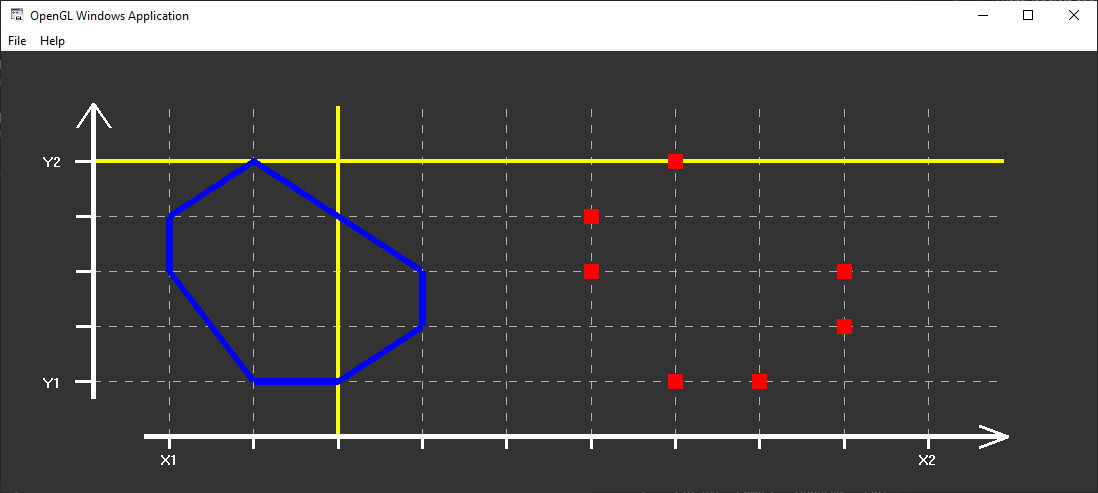


Рисунок 1.1 – Тестування програми при зміні ширини вікна

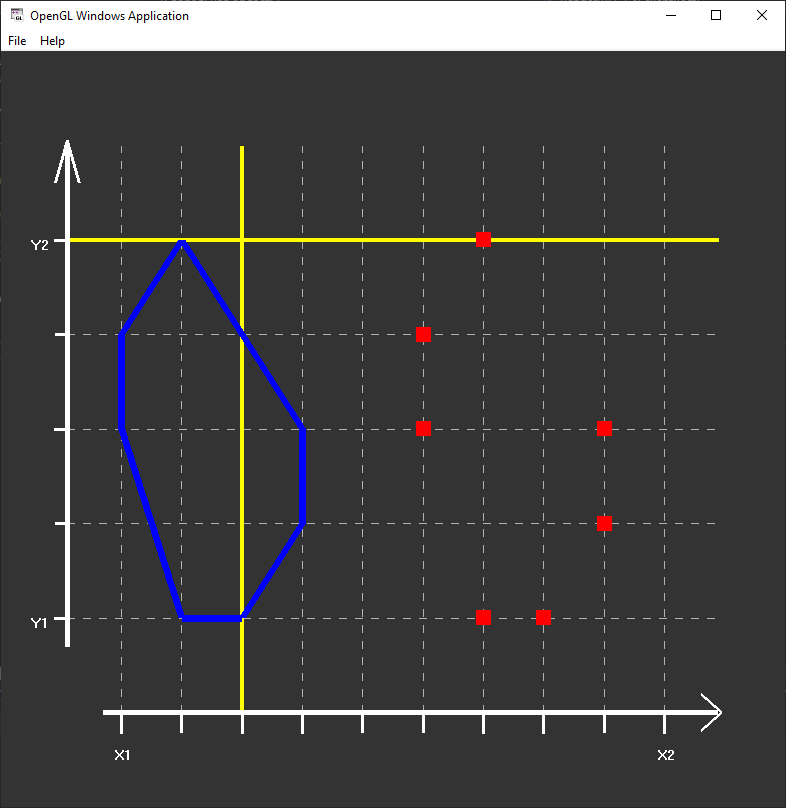


Рисунок 1.2 – Тестування програми при зміні висоти вікна

### Контроль виконання вимог та елементів завдання

В результаті виконання практичної роботи були повністю виконані елементи базового рівня та повністю підвищеного рівня складності, що відображено в таблиці 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблиця 1.1 | | | | |
| № з/п | Складність | Вимоги | Бали | Зроблено |
| 1 | Базовий рівень | Використання команд управління параметрами графічних примітивів (колір, тип, товщина) | 2 | **+** |
| 2 | Коректне відображення завдання під час змінення розмірів/положення вікна | 1 | **+** |
| 3 | Розроблення підпрограм для виключення дублювання коду | 1 | **+** |
| 4 | Застосування циклів для створення зображень | 1 | **+** |
| 5 | Підвищений рівень | Формування зображення векторними командами *OpenGL* (*glDrawArrays* и т.п.) | 1 | **+** |
| 6 | Використання ООП (розроблення власних класів) | 2 | **+** |

### Посилання на GitHub

<https://github.com/Mausipupsi/OpenGL>

# Загальний перелік посилань

1. Microsoft. glDrawArrays function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gldrawarrays>.
2. Microsoft. glLineStipple function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gllinestipple>
3. Microsoft. glEnable function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/glenable>
4. Microsoft. glDisable function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gldisable>
5. Microsoft. glColor3d function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/glcolor3d>
6. Microsoft. glLineWidth function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/gllinewidth>
7. Microsoft. glVertexPointer function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/glvertexpointer>
8. Microsoft. glVertex2d function [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/windows/win32/opengl/glvertex2d>

# Додаток А. Лістинг програми до практичної роботи №1

### Код файлу (Render.cpp)

1. #include "pch.h"
2. #include "glWinApp.h"
3. #include <vector>
4. using namespace std;
5. extern LPCSTR s1, s2, s3;
6. int LoadWindowDefaultFont() {
7. GLuint id = glGenLists(256);
8. wglUseFontBitmaps(wglGetCurrentDC(), 0, 256, id);
9. return id;
10. }
11. void OutText(LPCSTR str, double x, double y, double z = 0) {
12. glRasterPos3d(x, y, z);
13. glListBase(idFont);
14. glCallLists(static\_cast<GLsizei>(strlen(str)), GL\_UNSIGNED\_BYTE, str);
15. }
16. void SetupAndDrawLines(const vector<double>& vertices, float lineWidth, const double\* color) {
17. glColor3dv(color); // Встановлення кольору ліній
18. glLineWidth(lineWidth); // Встановлення ширини ліній
19. glEnableClientState(GL\_VERTEX\_ARRAY); // Включення масиву вершин
20. glVertexPointer(2, GL\_DOUBLE, 0, vertices.data()); // Встановлення вказівника на масив вершин
21. glDrawArrays(GL\_LINES, 0, static\_cast<GLsizei>(vertices.size() / 2)); // Малювання ліній
22. glDisableClientState(GL\_VERTEX\_ARRAY); // Вимкнення масиву вершин
23. }
24. void DrawGrid(double xMin, double xMax, double yMin, double yMax, double step) {
25. glEnable(GL\_LINE\_STIPPLE); // Включення режиму пунктирних ліній
26. glLineStipple(1, 0x00FF); // Встановлення шаблону пунктирних ліній
27. vector<double> vertices; // Створення вектора для зберігання вершин сітки
28. for (double x = xMin; x <= xMax; x += step) {
29. vertices.push\_back(x); // Додавання координат вершин сітки по осі X
30. vertices.push\_back(yMin - 1); // Додавання координат вершин сітки по осі Y
31. vertices.push\_back(x); // Додавання координат вершин сітки по осі X
32. vertices.push\_back(yMax + 1); // Додавання координат вершин сітки по осі Y
33. }
34. for (double y = yMin; y <= yMax; y += step) {
35. vertices.push\_back(xMin - 0.9); // Додавання координат вершин сітки по осі X
36. vertices.push\_back(y); // Додавання координат вершин сітки по осі Y
37. vertices.push\_back(xMax + 0.9); // Додавання координат вершин сітки по осі X
38. vertices.push\_back(y); // Додавання координат вершин сітки по осі Y
39. }
40. double gridColor[] = { 0.7, 0.7, 0.7 }; // Колір сітки
41. SetupAndDrawLines(vertices, 1, gridColor); // Налаштування та малювання ліній сітки
42. glDisable(GL\_LINE\_STIPPLE); // Вимкнення режиму пунктирних ліній
43. }
44. void DrawAxes(double xMin, double xMax, double yMin, double yMax) {
45. vector<double> vertices = {
46. xMin - 0.9, 0, xMax + 0.9, 0, // Координати вершин осі X
47. 0, yMin - 1, 0, yMax + 1 // Координати вершин осі Y
48. };
49. double axesColor[] = { 1, 1, 0 }; // Колір осей
50. SetupAndDrawLines(vertices, 4, axesColor); // Налаштування та малювання осей
51. vector<double> customLines = {
52. xMin - 0.9, yMin - 0.3, xMin - 0.9, yMax + 1, // Координати додаткових ліній
53. xMin - 0.3, yMin - 1, xMax + 0.9, yMin - 1 // Координати додаткових ліній
54. };
55. double customLinesColor[] = { 1, 1, 1 }; // Колір додаткових ліній
56. SetupAndDrawLines(customLines, 5, customLinesColor); // Налаштування та малювання додаткових ліній
57. glLineWidth(3); // Встановлення ширини ліній
58. glBegin(GL\_LINES); // Початок малювання ліній
59. glVertex2d(xMin - 0.9, yMax + 1.05); // Координати вершин стрілок осі Y
60. glVertex2d(xMin - 1.1, yMax + 0.6); // Координати вершин стрілок осі Y
61. glVertex2d(xMin - 0.9, yMax + 1.05); // Координати вершин стрілок осі Y
62. glVertex2d(xMin - 0.7, yMax + 0.6); // Координати вершин стрілок осі Y
63. glVertex2d(xMax + 0.95, yMin - 1); // Координати вершин стрілок осі X
64. glVertex2d(xMax + 0.6, yMin - 1.2); // Координати вершин стрілок осі X
65. glVertex2d(xMax + 0.95, yMin - 1); // Координати вершин стрілок осі X
66. glVertex2d(xMax + 0.6, yMin - 0.8); // Координати вершин стрілок осі X
67. glEnd(); // Кінець малювання ліній
68. glLineWidth(3); // Встановлення ширини ліній
69. glBegin(GL\_LINES); // Початок малювання ліній
70. for (double y = yMin; y <= yMax; y++) {
71. glVertex2d(xMin - 0.9, y); // Координати вершин поділок осі Y
72. glVertex2d(xMin - 1.12, y); // Координати вершин поділок осі Y
73. }
74. for (double x = xMin; x <= xMax; x++) {
75. glVertex2d(x, yMin - 1.22); // Координати вершин поділок осі X
76. glVertex2d(x, yMin - 1); // Координати вершин поділок осі X
77. }
78. glEnd(); // Кінець малювання ліній
79. OutText("Y1", xMin - 1.5, yMin - 0.1); // Відображення тексту "Y1" на осі Y
80. OutText("Y2", xMin - 1.5, yMax - 0.1); // Відображення тексту "Y2" на осі Y
81. OutText("X1", xMin - 0.1, yMin - 1.5); // Відображення тексту "X1" на осі X
82. OutText("X2", xMax - 0.1, yMin - 1.5); // Відображення тексту "X2" на осі X
83. }
84. class Point {
85. public:
86. double x, y; // Координати точки
87. Point(double x = 0, double y = 0) : x(x), y(y) {} // Конструктор з параметрами за замовчуванням
88. void draw() const {
89. glColor3d(1, 0, 0); // Встановлення червоного кольору для точки
90. glPointSize(15); // Встановлення розміру точки
91. glBegin(GL\_POINTS); // Початок малювання точки
92. glVertex2d(x, y); // Встановлення координат точки
93. glEnd(); // Кінець малювання точки
94. }
95. };
96. class MyPolygon {
97. public:
98. vector<Point> points; // Вектор точок, що складають полігон
99. MyPolygon(const vector<Point>& points) : points(points) {} // Конструктор з ініціалізацією вектора точок
100. void draw() const {
101. glColor3d(0, 0, 1); // Встановлення синього кольору для полігону
102. glLineWidth(7); // Встановлення ширини ліній полігону
103. glBegin(GL\_LINE\_STRIP); // Початок малювання полігону
104. for (const auto& point : points) {
105. glVertex2d(point.x, point.y); // Встановлення координат точок полігону
106. }
107. glEnd(); // Кінець малювання полігону
108. }
109. };
110. void Render(RECT& clientRect) {
111. glClearColor(0.2f, 0.2f, 0.2f, 1.0f); // Встановлення кольору фону
112. glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT); // Очищення буферів кольору, глибини та трафарету
113. glLoadIdentity(); // Скидання поточної матриці
114. int Height = clientRect.bottom - clientRect.top; // Визначення висоти вікна
115. int Width = clientRect.right - clientRect.left; // Визначення ширини вікна
116. double xMin = -2, xMax = 7, yMin = -4, yMax = 0; // Встановлення меж координатної системи
117. glViewport(0, 0, Width, Height); // Встановлення області перегляду
118. gluOrtho2D(xMin - 2.0, xMax + 2.0, yMin - 2.0, yMax + 2.0); // Встановлення ортогональної проекції
119. DrawGrid(xMin, xMax, yMin, yMax, 1.0); // Малювання сітки
120. DrawAxes(xMin, xMax, yMin, yMax); // Малювання осей
121. vector<Point> polygonPoints = {
122. Point(-1, -4), Point(-2, -2), Point(-2, -1), Point(-1, 0), Point(1, -2), Point(1, -3), Point(0, -4), Point(-1, -4)
123. }; // Визначення точок полігону
124. vector<Point> points = {
125. Point(4, -4), Point(3, -2), Point(3, -1), Point(4, 0), Point(6, -2), Point(6, -3), Point(5, -4), Point(4, -4)
126. }; // Визначення окремих точок
127. MyPolygon polygon(polygonPoints); // Створення об'єкта полігону
128. polygon.draw(); // Малювання полігону
129. for (const auto& point : points) {
130. point.draw(); // Малювання окремих точок
131. }
132. }