

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
 НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання лабораторної роботи №1  
з дисципліни «Комп’ютерна графіка»**

Виконав:  
студент III курсу, групи ДА-22  
Жадько Микита Сергійович

Київ – 2024

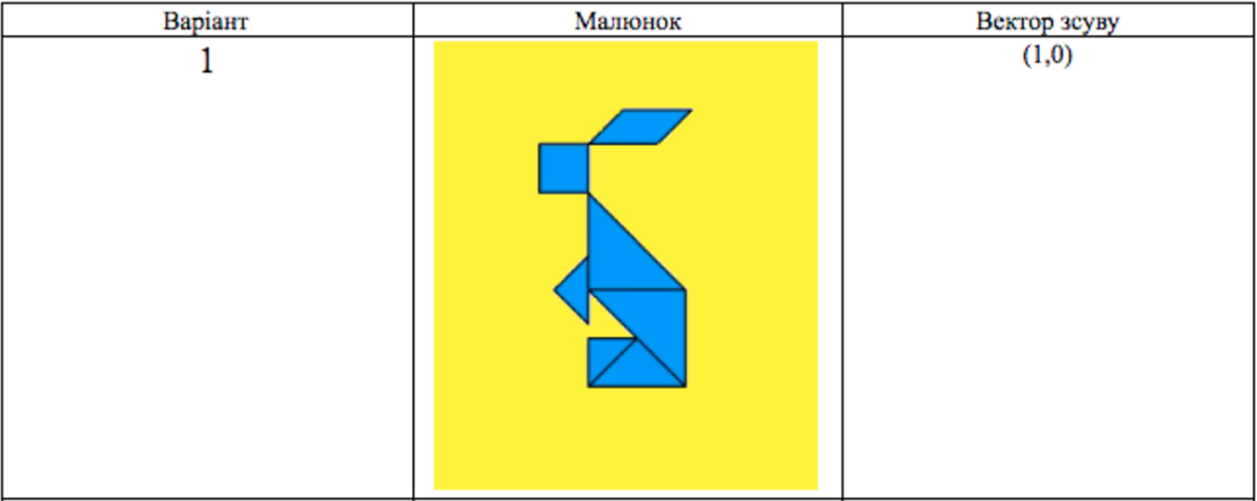
ПОСТАНОВКА ЗАДАЧІ

Мета роботи: отримати навички створення графічних програм. Ознайомитись з можливостями OpenGL або обрати іншу графічну бібліотеку.

Завдання:

1. Ознайомитися з принципами побудови двовимірної системи координат.
2. Використовуючи обрану графічну бібліотеку, на основі примітивів зобразити істоту за варіантом з таблиці
3. Використовуючи бібліотеку, що відповідає за системний рівень операцій вводу-виводу, - реалізувати рух істоти у заданому векторі (див. таблицю). Управляючі клавіші - ADWS

Завдання за варіантом:



ХІД РОБОТИ

1. **Чому обрано бібліотеку GLFW?**

GLFW є оптимальним вибором для цієї лабораторної роботи завдяки її простоті, легкій інтеграції з OpenGL і можливостям обробки подій вводу-виводу. Вона дозволяє швидко реалізувати необхідний функціонал, зосереджуючись на вивченні принципів OpenGL та реалізації руху істоти відповідно до завдання.

Вона має мінімалістичний інтерфейс і чітко структурований API, що дозволяє швидко розібратися з основними функціями. Для початківців це важливо, оскільки не потрібно витрачати час на вивчення складної документації. Підтримує основні операційні системи: Windows, macOS та Linux. Це дозволяє створювати програми, які будуть працювати на різних платформах без змін у коді.

Також, вона створена для роботи з OpenGL і надає зручний функціонал для налаштування контексту OpenGL, керування вікном і подіями. На відміну від GLUT, GLFW дає більше контролю над функціональністю OpenGL, що є перевагою для створення адаптивних і сучасних програм. Дозволяє створювати контексти OpenGL 3.x та новіші, що є важливим для роботи з сучасними графічними можливостями. Інші бібліотеки, наприклад GLUT, не завжди підтримують ці функції або працюють із ними менш ефективно.

Напевно, найбільша перевага в тому, що GLFW надає зручний функціонал для обробки подій клавіатури, миші та інших пристроїв вводу. Це спрощує реалізацію керування істотою за допомогою клавіш ADWS, що є обов'язковою вимогою завдання. Ну і ще вона надає розширені налаштування для роботи з вікнами, таких як роздільна здатність, режим повноекранного або віконного відображення, обробка декількох моніторів, що дозволяє адаптувати програму до будь-яких потреб.

1. **Блок-схема дерева проекту.**

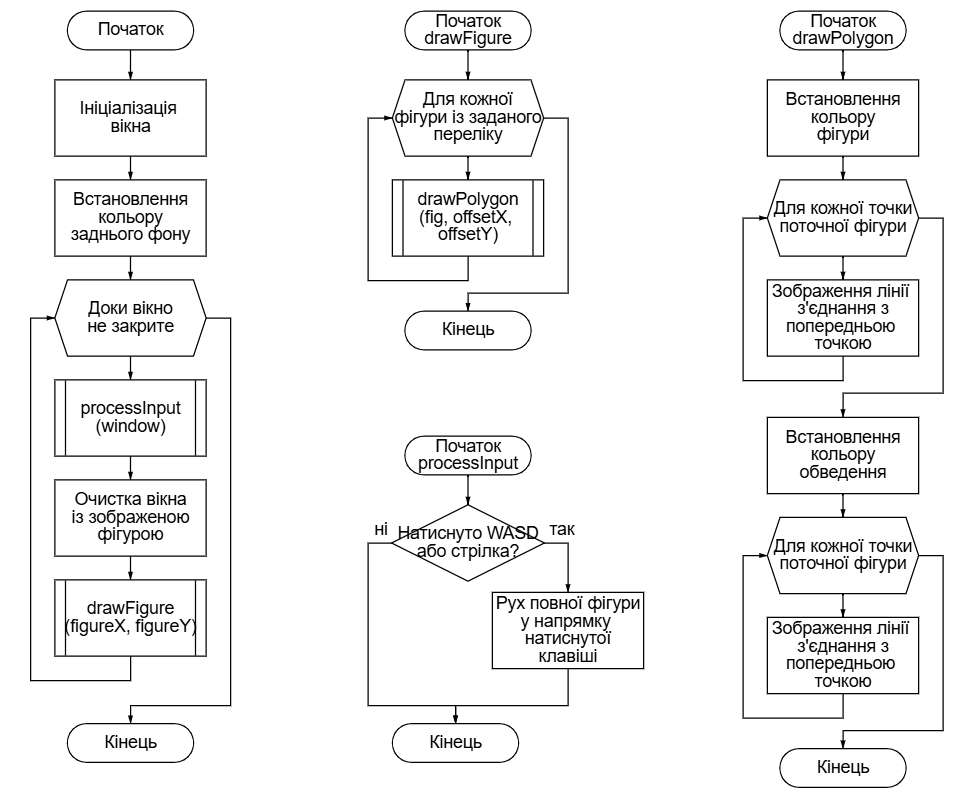


Рисунок 1 — Блок-схема дерева проекту.

1. **Результати виконання лабораторної роботи.**

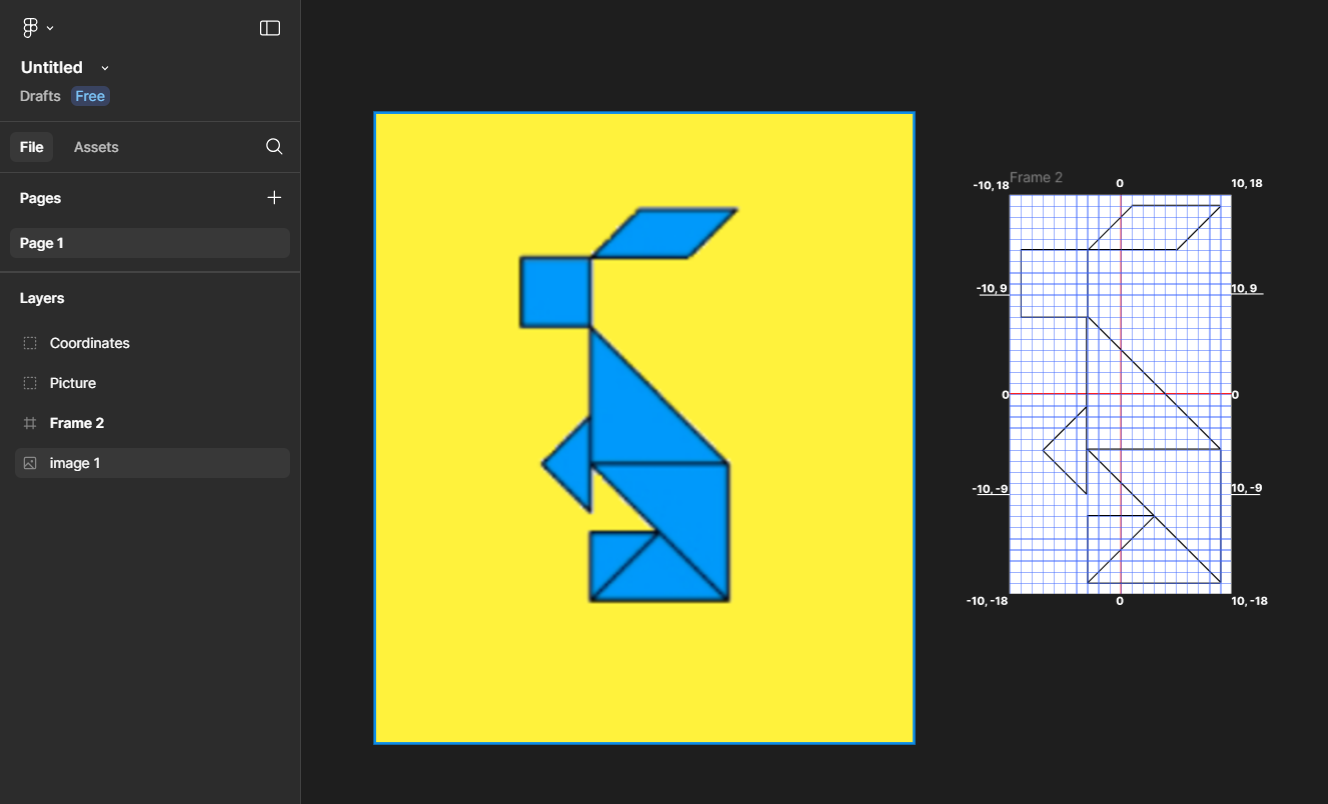


Рисунок 2 — Перенесення рисунку за варіантом на координатну площину для зручного програмного зображення.

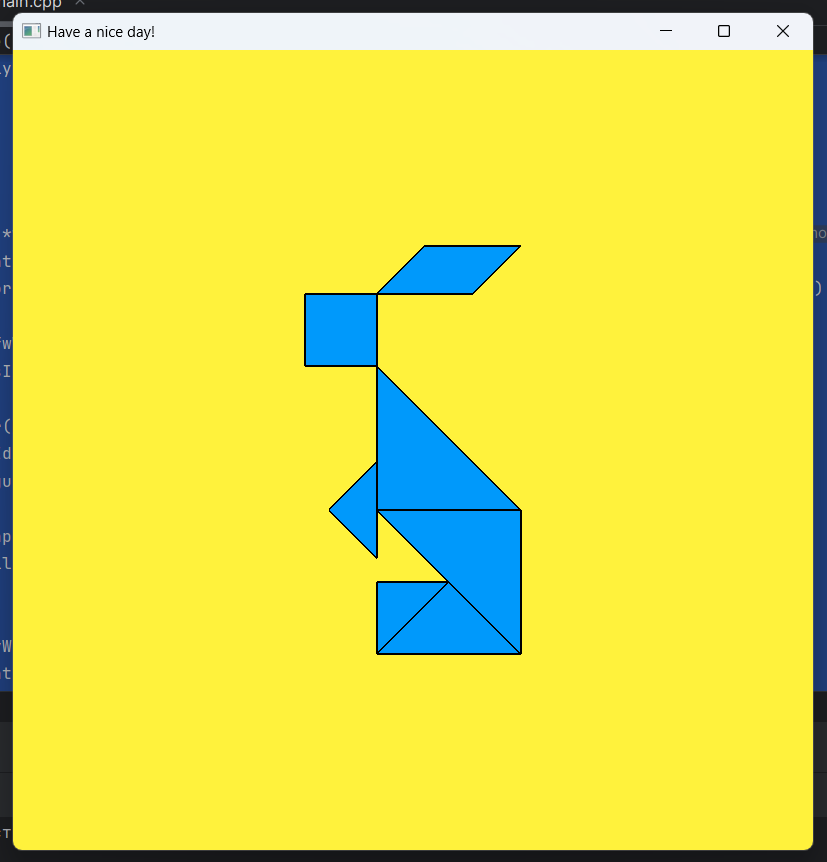


Рисунок 3 — Виконувана програма (фігурою можна керувати, чесно).

1. **Лістинг коду.**

#include <GLFW/glfw3.h>

#include <iostream>

#include <vector>

**using** **namespace** std**;**

const int WINDOW\_WIDTH **=** 800**;**

const int WINDOW\_HEIGHT **=** 800**;**

const float MOVE\_SPEED **=** 2.0f**;**

const float CELL\_SIZE **=** 0.03f**;**

float lastFrameTime **=** 0.0f**;**

float figureX **=** 0.0f**;**

float figureY **=** 0.0f**;**

vector**<**vector**<**vector**<**int**>** **>** **>** figure **=** **{**

**{**

**{-**9**,** 13**},**

**{-**3**,** 13**},**

**{-**3**,** 7**},**

**{-**9**,** 7**},**

**},**

**{**

**{**1**,** 17**},**

**{**9**,** 17**},**

**{**5**,** 13**},**

**{-**3**,** 13**},**

**},**

**{**

**{-**3**,** 7**},**

**{**9**,** **-**5**},**

**{-**3**,** **-**5**},**

**},**

**{**

**{**9**,** **-**5**},**

**{-**3**,** **-**5**},**

**{**9**,** **-**17**},**

**},**

**{**

**{-**3**,** **-**1**},**

**{-**3**,** **-**9**},**

**{-**7**,** **-**5**},**

**},**

**{**

**{**3**,** **-**11**},**

**{**9**,** **-**17**},**

**{-**3**,** **-**17**},**

**},**

**{**

**{-**3**,** **-**11**},**

**{**3**,** **-**11**},**

**{-**3**,** **-**17**},**

**},**

**};**

vector figureBounds **=** **{**

**-**9**,**

17**,**

9**,**

**-**17**,**

**};**

vector outlineColor **=** **{**

0**,**

0**,**

0**,**

**};**

vector figureColor **=** **{**

0**,**

153**,**

251**,**

**};**

vector backColor **=** **{**

255**,**

242**,**

60**,**

**};**

float calculateDeltaTime**()** **{**

float currentFrameTime **=** glfwGetTime**();**

float deltaTime **=** currentFrameTime **-** lastFrameTime**;**

lastFrameTime **=** currentFrameTime**;**

**return** deltaTime**;**

**}**

float normRGB**(**const int RGBvalue**)** **{**

**return** RGBvalue **/** 255.0f**;**

**}**

void drawPolygon**(**const vector**<**vector**<**int**>** **>** **&**pointsArray**,** const float xOffset**,** const float yOffset**)** **{**

glColor3f**(**normRGB**(**figureColor**[**0**]),** normRGB**(**figureColor**[**1**]),** normRGB**(**figureColor**[**2**]));**

glBegin**(**GL\_POLYGON**);**

**for** **(**int i **=** 0**;** i **<** pointsArray**.**size**();** i**++)** **{**

glVertex2f**(**xOffset **+** pointsArray**[**i**][**0**]** **\*** CELL\_SIZE**,**

yOffset **+** pointsArray**[**i**][**1**]** **\*** CELL\_SIZE**);**

**}**

glEnd**();**

glColor3f**(**normRGB**(**outlineColor**[**0**]),** normRGB**(**outlineColor**[**1**]),** normRGB**(**outlineColor**[**2**]));**

glLineWidth**(**2.0f**);**

glBegin**(**GL\_LINE\_LOOP**);**

**for** **(**int i **=** 0**;** i **<** pointsArray**.**size**();** i**++)** **{**

glVertex2f**(**xOffset **+** pointsArray**[**i**][**0**]** **\*** CELL\_SIZE**,**

yOffset **+** pointsArray**[**i**][**1**]** **\*** CELL\_SIZE**);**

**}**

glEnd**();**

**}**

void processInput**(**GLFWwindow **\***window**)** **{**

float deltaTime **=** calculateDeltaTime**();**

float moveSpeed **=** deltaTime **\*** MOVE\_SPEED**;**

**if** **(((**glfwGetKey**(**window**,** GLFW\_KEY\_A**)** **==** GLFW\_PRESS**)** **||**

**(**glfwGetKey**(**window**,** GLFW\_KEY\_LEFT**)** **==** GLFW\_PRESS**))**

**&&** figureX **+** figureBounds**[**0**]** **\*** CELL\_SIZE **>** **-**1.0f**)** **{**

figureX **-=** moveSpeed**;**

**}**

**if** **(((**glfwGetKey**(**window**,** GLFW\_KEY\_W**)** **==** GLFW\_PRESS**)** **||**

**(**glfwGetKey**(**window**,** GLFW\_KEY\_UP**)** **==** GLFW\_PRESS**))**

**&&** figureY **+** figureBounds**[**1**]** **\*** CELL\_SIZE **<** 1.0f**)** **{**

figureY **+=** moveSpeed**;**

**}**

**if** **(((**glfwGetKey**(**window**,** GLFW\_KEY\_D**)** **==** GLFW\_PRESS**)** **||**

**(**glfwGetKey**(**window**,** GLFW\_KEY\_RIGHT**)** **==** GLFW\_PRESS**))**

**&&** figureX **+** figureBounds**[**2**]** **\*** CELL\_SIZE **<** 1.0f**)** **{**

figureX **+=** moveSpeed**;**

**}**

**if** **(((**glfwGetKey**(**window**,** GLFW\_KEY\_S**)** **==** GLFW\_PRESS**)** **||**

**(**glfwGetKey**(**window**,** GLFW\_KEY\_DOWN**)** **==** GLFW\_PRESS**))**

**&&** figureY **+** figureBounds**[**3**]** **\*** CELL\_SIZE **>** **-**1.0f**)** **{**

figureY **-=** moveSpeed**;**

**}**

**}**

void drawFigure**(**const float offsetX**,** const float offsetY**)** **{**

**for** **(**const auto **&**fig**:** figure**)** **{**

drawPolygon**(**fig**,** offsetX**,** offsetY**);**

**}**

**}**

int main**()** **{**

glfwInit**();**

GLFWwindow **\***window **=** glfwCreateWindow**(**WINDOW\_WIDTH**,** WINDOW\_HEIGHT**,** "Have a nice day!"**,** **nullptr,** **nullptr);**

glfwMakeContextCurrent**(**window**);**

glClearColor**(**normRGB**(**backColor**[**0**]),** normRGB**(**backColor**[**1**]),** normRGB**(**backColor**[**2**]),** 1.0f**);**

**while** **(!**glfwWindowShouldClose**(**window**))** **{**

processInput**(**window**);**

glClear**(**GL\_COLOR\_BUFFER\_BIT**);**

glLoadIdentity**();**

drawFigure**(**figureX**,** figureY**);**

glfwSwapBuffers**(**window**);**

glfwPollEvents**();**

**}**

glfwDestroyWindow**(**window**);**

glfwTerminate**();**

**return** 0**;**

**}**