Підготовка:

Робота на машині:

Захист: Лабораторна робота №6

Тема: Алгоритми видалення невидимих ліній та поверхонь

Завдання

# 1. Створити програму, що матиме зручний інтерфейс (вікно, головне меню, рядок статусу та ін.). У вікні програми повинно створюватися та відображатись зображення фігури, створеної відповідно завдань лабораторних робіт №4,5.

# 2. За допомогою методів та алгоритмів видалення невидимих ліній та поверхонь створити більш реалістичне зображення тривимірної фігури Програма повинна надавати можливість виконання над фігурою трьох основних афінних перетворень, як у лабораторній роботі №5, а також вмикати та вимикати режим видалення невидимих ліній.

**Теоретичні відомості**

Задача видалення невидимих ліній та поверхонь є однією з найбільш цікавих та складних у комп'ютерній графіці. Алгоритми видалення полягають у визначенні ліній ребер, поверхонь або об'ємів, які є видимими або невидимими для спостерігача, що знаходиться в указаній точці простору. Необхідність видалення невидимих ліній, ребер, поверхонь або об'ємів проілюстрована на рис.18. Рисунок наочно демонструє, що зображення без видалення невидимих ліній сприймається неоднозначно.

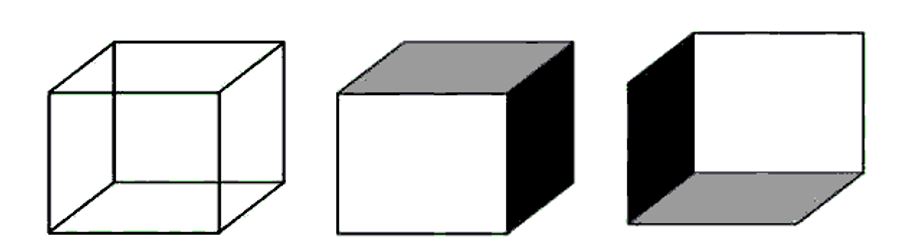


Рис. 18 – Неоднозначність сприйняття зображення куба

Складність задачі видалення невидимих ліній та поверхонь призвела до появи великої кількості різноманітних способів її розв'язання. Багато з них орієнтовані на спеціалізовані додатки. Єдиного (загального) розв'язку цієї задачі, який був би придатним для усіх різноманітних випадків, природно, не існує. Для кожного випадку обирається метод, що найбільш підходить. Наприклад, для моделювання процесів у реальному часі потрібні швидкі алгоритми, в той час як для формування складного реалістичного зображення, в якому представлені тіні, прозорість і фактура, що враховують ефекти віддзеркалення і заломлення кольору у найменших відтінках, фактор часу виконання вже не такий важливий. Подібні алгоритми працюють повільно, і часто на обчислення потрібно декілька хвилин або навіть годин. Існує тісний взаємозв'язок між швидкістю роботи алгоритму і детальністю його результату. Жоден з алгоритмів не може досягнути гарних оцінок для цих двох показників одночасно. По мірі створення все більш швидких алгоритмів можна будувати все більш детальні зображення. Реальні завдання, однак, завжди будуть вимагати враховувати ще більшу кількість деталей. Всі алгоритми такого роду так чи інакше включають в себе сортування, причому головне сортування ведеться по геометричній відстані від тіла, поверхні, ребра чи точки до точки спостерігання або картинної площини. Основна ідея, що покладена в основу сортування по відстані, полягає в тому, що чим далі розташований об'єкт від точки спостерігання, тим більше ймовірність того, що він буде повністю або частково закритий одним з об'єктів, більш близьких до точки спостерігання. Після визначення відстаней або пріоритетів за глибиною залишається провести сортування по горизонталі та по вертикалі, щоб з'ясувати, чи буде об'єкт, що розглядається, дійсно закритий об'єктом, який розташований ближче до точки спостерігання. Ефективність будь-якого алгоритму видалення в значній мірі залежить від ефективності процесу сортування.

Докладно більшість цих алгоритмів описані у роботі 1. Тут розглянемо лише основні. Більшість з них можна значно поліпшити, якщо виконати попередні обчислення щодо видалення нелицевих граней, тобто тих граней, які повністю перекриваються іншими.

Алгоритм видалення нелицевих граней спочатку видаляє з кожного багатогранника ті ребра або грані, які екрануються самим тілом. Для цього можна використати простий тест: якщо одна або обидві суміжні грані обернені своєю зовнішньою поверхнею до спостерігача, то ребро є видимим. Цей тест виконується обчисленням скалярного добутку координат вектора, що направлений від багатогранника до спостерігача, на вектор зовнішньої нормалі грані: якщо результат додатний, то грань видима.

Код програми представлений на лістингу 1

**public class** Main **extends** Application  
{  
 **private** PerspectiveCamera **camera** = **new** PerspectiveCamera(**false**);  
 **private** BorderPane **borderPane** = **new** BorderPane();  
 **private** VBox **vBox** = **new** VBox();  
 **private** Slider **sliderX** = **new** Slider(-200, 0, -100);  
 **private** Slider **sliderY** = **new** Slider(0, 500, 250);  
 **private** Slider **sliderZ** = **new** Slider(0, 800, 300);  
 **private** Slider **sliderR** = **new** Slider(0, 360, 0);  
 **private** Slider **sliderS** = **new** Slider(0.1, 10, 1);  
 **private** MeshView **meshView** = **this**.createMeshView();  
 **private** PointLight **light** = **new** PointLight();  
 **private** AmbientLight **ambientLight** = **new** AmbientLight();  
 **private** RadioButton **xPivot** = **new** RadioButton();  
 **private** RadioButton **yPivot** = **new** RadioButton();  
 **private** RadioButton **zPivot** = **new** RadioButton();  
 **private** Label **a1** = **new** Label();  
 **private** Label **a2** = **new** Label();  
 **private** Label **a3** = **new** Label();  
 **private** Label **a4** = **new** Label();  
 **private** Label **a5** = **new** Label();  
 **private** Label **a6** = **new** Label();  
 **private** Label **a7** = **new** Label();  
 **private** RadioButton **normalDraw** = **new** RadioButton();  
 **private** RadioButton **cullFaceDraw** = **new** RadioButton();  
 **private** RadioButton **lineDraw** = **new** RadioButton();  
  
 **public static void** main(String[] args)  
 {  
  
 Application.*launch*(args);  
 }  
  
 @Override  
 **public void** start(Stage stage)  
 {  
 Group root = **new** Group(**borderPane**, **meshView**, **light**, **ambientLight**);  
 Scene scene = **new** Scene(root, 800, 700, **true**);  
 stage.setScene(scene);  
  
 **borderPane**.setCenter(**camera**);  
  
 ToggleGroup ivan = **new** ToggleGroup();  
 **xPivot**.setToggleGroup(ivan);  
 **yPivot**.setToggleGroup(ivan);  
 **zPivot**.setToggleGroup(ivan);  
 **xPivot**.setText(**"по ох"**);  
 **yPivot**.setText(**"по оy"**);  
 **zPivot**.setText(**"по оz"**);  
 **xPivot**.setSelected(**true**);  
 **meshView**.setRotationAxis(Rotate.***X\_AXIS***);  
 ivan.selectedToggleProperty().addListener((ov, old\_toggle, new\_toggle) -> {  
 **if** (ivan.getSelectedToggle() != **null**) {  
 **if**(ivan.getSelectedToggle() == **xPivot**) {  
 **meshView**.setRotationAxis(Rotate.***X\_AXIS***);  
 **sliderR**.setValue(**meshView**.getRotate());  
 }  
 **else if**(ivan.getSelectedToggle() == **yPivot**){  
 **meshView**.setRotationAxis(Rotate.***Y\_AXIS***);  
 **sliderR**.setValue(**meshView**.getRotate());  
  
 }  
 **else if**(ivan.getSelectedToggle() == **zPivot**) {  
 **meshView**.setRotationAxis(Rotate.***Z\_AXIS***);  
 **sliderR**.setValue(**meshView**.getRotate());  
 }  
 }  
 });  
  
 ToggleGroup alek = **new** ToggleGroup();  
 **normalDraw**.setToggleGroup(alek);  
 **cullFaceDraw**.setToggleGroup(alek);  
 **lineDraw**.setToggleGroup(alek);  
 **normalDraw**.setText(**"Нормальная отрисовка"**);  
 **cullFaceDraw**.setText(**"Отрисовка задних полигонов"**);  
 **lineDraw**.setText(**"Линейная отрисовка"**);  
 **normalDraw**.setSelected(**true**);  
  
 alek.selectedToggleProperty().addListener((ov, old\_toggle, new\_toggle) -> {  
 **if** (alek.getSelectedToggle() != **null**) {  
 **if**(alek.getSelectedToggle() == **normalDraw**) {  
 **meshView**.setCullFace(CullFace.***NONE***);  
 **meshView**.setDrawMode(DrawMode.***FILL***);  
 }  
 **else if**(alek.getSelectedToggle() == **cullFaceDraw**){  
 **meshView**.setCullFace(CullFace.***FRONT***);  
 **meshView**.setDrawMode(DrawMode.***FILL***);  
 }  
 **else if**(alek.getSelectedToggle() == **lineDraw**) {  
 **meshView**.setCullFace(CullFace.***NONE***);  
 **meshView**.setDrawMode(DrawMode.***LINE***);  
 }  
 }  
 });  
  
 **sliderX**.valueProperty().addListener((ov, old\_val, new\_val) -> {**meshView**.setTranslateX(new\_val.intValue());  
 });  
 **sliderY**.valueProperty().addListener((ov, old\_val, new\_val) -> **meshView**.setTranslateY(new\_val.intValue()));  
 **sliderZ**.valueProperty().addListener((ov, old\_val, new\_val) -> **meshView**.setTranslateZ(new\_val.intValue()));  
 **sliderR**.valueProperty().addListener((ov, old\_val, new\_val) -> {**meshView**.setRotate(new\_val.doubleValue());});  
 **sliderS**.valueProperty().addListener((ov, old\_val, new\_val) -> {**meshView**.setScaleX(new\_val.doubleValue());  
 **meshView**.setScaleY(new\_val.doubleValue());  
 **meshView**.setScaleZ(new\_val.doubleValue());});  
 **vBox**.setPadding(**new** Insets(10,10,10,10));  
 **vBox**.setSpacing(10);  
 **a1**.setText(**"Движение по х"**);  
 **a2**.setText(**"Движение по у"**);  
 **a3**.setText(**"Движение по z"**);  
 **a4**.setText(**"Вращение"**);  
 **a5**.setText(**"Выберите ось вращения"**);  
 **a6**.setText(**"Маштабирование"**);  
 **a7**.setText(**"Выберите тип отрисовки"**);  
 **vBox**.getChildren().add(**a1**);  
 **vBox**.getChildren().add(**sliderX**);  
 **vBox**.getChildren().add(**a2**);  
 **vBox**.getChildren().add(**sliderY**);  
 **vBox**.getChildren().add(**a3**);  
 **vBox**.getChildren().add(**sliderZ**);  
 **vBox**.getChildren().add(**a4**);  
 **vBox**.getChildren().add(**sliderR**);  
 **vBox**.getChildren().add(**a5**);  
 **vBox**.getChildren().add(**xPivot**);  
 **vBox**.getChildren().add(**yPivot**);  
 **vBox**.getChildren().add(**zPivot**);  
 **vBox**.getChildren().add(**a6**);  
 **vBox**.getChildren().add(**sliderS**);  
 **vBox**.getChildren().add(**a7**);  
 **vBox**.getChildren().add(**normalDraw**);  
 **vBox**.getChildren().add(**cullFaceDraw**);  
 **vBox**.getChildren().add(**lineDraw**);  
 **borderPane**.setLeft(**vBox**);  
  
 **meshView**.setTranslateX(-100);  
 **meshView**.setTranslateY(250);  
 **meshView**.setTranslateZ(300);  
  
 **ambientLight**.setTranslateX(0);  
 **ambientLight**.setTranslateY(250);  
 **ambientLight**.setTranslateZ(-200);  
  
 stage.setTitle(**"Fool"**);  
 stage.show();  
  
 }  
  
 **public** MeshView createMeshView()  
 {  
 **float**[] points =  
 {  
 420, 0, 20, *// A* 500, 130, 0, *// D* 400, 130, 0, *// C* 480, 0, 20, *// B* 420, 0, 80, *// J* 500, 130, 100, *// o* 400, 130, 100, *// K* 480, 0, 80, *// L* 430, -20, 30,  
 430, -20, 70,  
 470, -20, 70,  
 470, -20, 30,  
  
 415, -50, 15,  
 415, -50, 85,  
 485, -50, 85,  
 485, -50, 15,  
  
 415, -80, 15,  
 415, -80, 85,  
 485, -80, 85,  
 485, -80, 15,  
  
 410, 160, 10,  
 410, 160, 90,  
 490, 160, 90,  
 490, 160, 10,  
  
 410, 180, 10,  
 410, 180, 90,  
 490, 180, 90,  
 490, 180, 10,  
  
 410, 180, 130,  
 490, 180, 130,  
  
 386, 87, 37,  
 386, 87, 64,  
 406, 90, 64,  
 406, 90, 37,  
  
 400, 24, 39,  
 400, 24, 61,  
 415, 27, 61,  
 415, 27, 39,  
  
 386, 87, 64,  
 386, 87, 104,  
 407, 90, 104,  
 407, 90, 64,  
  
 390, 67, 64,  
 390, 67, 104,  
 407, 70, 104,  
 407, 70, 64,  
  
 514, 87, 63,  
 514, 87, 36,  
 494, 90, 36,  
 494, 90, 63,  
  
 504, 24, 61,  
 504, 24, 39,  
 484, 27, 39,  
 484, 27, 61,  
  
 514, 87, 104,  
 514, 87, 63,  
 494, 90, 63,  
 494, 90, 104,  
  
 510, 67, 104,  
 510, 67, 62,  
 490, 70, 62,  
 490, 70, 104  
 };  
  
 **float**[] texCoords =  
 {  
 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f,  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
 1.0f, 1.0f, 0.5f, 1.0f, 0.0f, 5.0f, 0.0f, 0.0f,  
 0.0f, 5.0f, 0.0f, 0.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f, 0.5f, 1.0f, 0.0f, 5.0f,  
 0.0f, 0.0f, 0.0f, 5.0f  
 };  
  
 **int**[] faces =  
 {  
 0, 0, 2, 2, 1, 1, *// front* 1, 1, 3, 3, 0, 0,  
 0, 0, 6, 6, 2, 2,  
 6, 6, 0, 0, 4, 4,  
 5, 5, 6, 6, 4, 4,  
 5, 5, 4, 4, 7, 7,  
 1, 1, 5, 5, 7, 7,  
 1, 1, 7, 7, 3, 3,  
  
 0, 0, 8, 8, 9, 9,  
 4, 4, 0, 0, 9, 9,  
 4, 4, 9, 9, 10, 10,  
 7, 7, 4, 4, 10, 10,  
 3, 3, 11, 11, 8, 8,  
 0, 0, 3, 3, 8, 8,  
 7, 7, 10, 10, 11, 11,  
 3, 3, 7, 7, 11, 11,  
  
 8, 8, 12, 12, 13, 13,  
 9, 9, 8, 8, 13, 13,  
 9, 9, 13, 13, 14, 14,  
 10, 10, 9, 9, 14, 14,  
 10, 10, 14, 14, 15, 15,  
 11, 11, 10, 10, 15, 15,  
 11, 11, 15, 15, 12, 12,  
 8, 8, 11, 11, 12, 12,  
  
 12, 12, 16, 16, 17, 17,  
 13, 13, 12, 12, 17, 17,  
 13, 13, 17, 17, 18, 18,  
 14, 14, 13, 13, 18, 18,  
 14, 14, 18, 18, 19, 19,  
 15, 15, 14, 14, 19, 19,  
 15, 15, 19, 19, 16, 16,  
 12, 12, 15, 15, 16, 16,  
 16, 16, 19, 19, 18, 18,  
 17, 17, 16, 16, 18, 18, *//32* 20, 20, 1, 1, 2, 2,  
 21, 21, 20, 20, 2, 2,  
 21, 21, 2, 2, 6, 6,  
 22, 22, 21, 21, 6, 6,  
 22, 22, 6, 6, 5, 5,  
 23, 23, 22, 22, 5, 5,  
 23, 23, 5, 5, 1, 1,  
 20, 20, 23, 23, 1, 1,  
  
 24, 24, 20, 20, 21, 21,  
 25, 25, 24, 24, 21, 21,  
 25, 25, 21, 21, 22, 22,  
 26, 26, 25, 25, 22, 22,  
 26, 26, 22, 22, 23, 23,  
 27, 27, 26, 26, 23, 23,  
 27, 27, 23, 23, 20, 20,  
 24, 24, 27, 27, 20, 20,  
  
 25, 25, 21, 21, 28, 28,  
 26, 26, 29, 29, 22, 22,  
 21, 21, 22, 22, 28, 28,  
 22, 22, 29, 29, 28, 28,  
 24, 24, 26, 26, 27, 27,  
 25, 25, 26, 26, 24, 24,  
  
 30, 30, 34, 34, 35, 35,  
 31, 31, 30, 30, 35, 35,  
 31, 31, 35, 35, 36, 36,  
 32, 32, 31, 31, 36, 36,  
 32, 32, 36, 36, 37, 37,  
 33, 33, 32, 32, 37, 37,  
 33, 33, 37, 37, 34, 34,  
 30, 30, 33, 33, 34, 34,  
  
 34, 34, 37, 37, 36, 36,  
 35, 35, 34, 34, 36, 36,  
 30, 30, 32, 32, 33, 33,  
 31, 31, 32, 32, 30 ,30,  
  
 38, 38, 42, 42, 43, 43, *//front* 39, 39, 38, 38, 43, 43,  
 39, 39, 43, 43, 44, 44,  
 40, 40, 39, 39, 44, 44,  
 40, 40, 44, 44, 45, 45,  
 41, 41, 40, 40, 45, 45,  
 41, 41, 45, 45, 42, 42,  
 38, 38, 41, 41, 42, 42,  
 42, 42, 45, 45, 44, 44,  
 43, 43, 42, 42, 44, 44,  
 38, 38, 40, 40, 41, 41,  
 39, 39, 40, 40, 38, 38,  
  
  
 46, 46, 50, 50, 51, 51,  
 47, 47, 46, 46, 51, 51,  
 47, 47, 51, 51, 52, 52,  
 48, 48, 47, 47, 52, 52,  
 48, 48, 52, 52, 53, 53,  
 49, 49, 48, 48, 53, 53,  
 49, 49, 53, 53, 50, 50,  
 46, 46, 49, 49, 50, 50,  
 50, 50, 53, 53, 52, 52,  
 51, 51, 50, 50, 52, 52,  
 46, 46, 48, 48, 49, 49,  
 47, 47, 48, 48, 46, 46,  
  
 54, 54, 58, 58, 59, 59, *// front* 55, 55, 54, 54, 59, 59,  
 55, 55, 59, 59, 60, 60,  
 56, 56, 55, 55, 60, 60,  
 56, 56, 60, 60, 61, 61,  
 57, 57, 56, 56, 61, 61,  
 57, 57, 61, 61, 58, 58,  
 54, 54, 57, 57, 58, 58,  
 58, 58, 61, 61, 60, 60,  
 59, 59, 58, 58, 60, 60,  
 54, 54, 56, 56, 57, 57,  
 55, 55, 56, 56, 54, 54,  
 };  
  
 **int**[] facesmoothes =  
 {  
 0,0, 1,1, 0,0, 1,1, 0,0, 1,1, 1,1, 0,0,  
 1,1, 0,0, 1,1, 0,0, 0,0, 1,1, 0,0, 1,1,  
 0,0, 1,1,  
 2,3, 3,3, 0,0, 3,3, 2,3, 3,3, 2,2,  
 1,1, 0,0, 1,1, 0,0, 1,1, 0,0, 1,1, 0,0,  
 9,9, 0,0, 1,1, 2,2, 3,3, 4,4, 5,5,  
 0,0, 1,1, 2,2, 3,3, 4,4, 5,5, 6,6, 7,7, 10,10, 9,9,  
 8,8, 8,8  
 };  
  
 TriangleMesh mesh = **new** TriangleMesh();  
 mesh.getPoints().addAll(points);  
 mesh.getTexCoords().addAll(texCoords);  
 mesh.getFaces().addAll(faces);  
 PhongMaterial material = **new** PhongMaterial();  
 material.setDiffuseColor(Color.***GREEN***);  
 MeshView meshView = **new** MeshView();  
 meshView.setMaterial(material);  
 mesh.getFaceSmoothingGroups().addAll(facesmoothes); *//108* meshView.setMesh(mesh);  
 **return** meshView;  
 }  
}

Результат роботи

Режим нормальної отрисовки фігури зображено на рисунку 6.1.

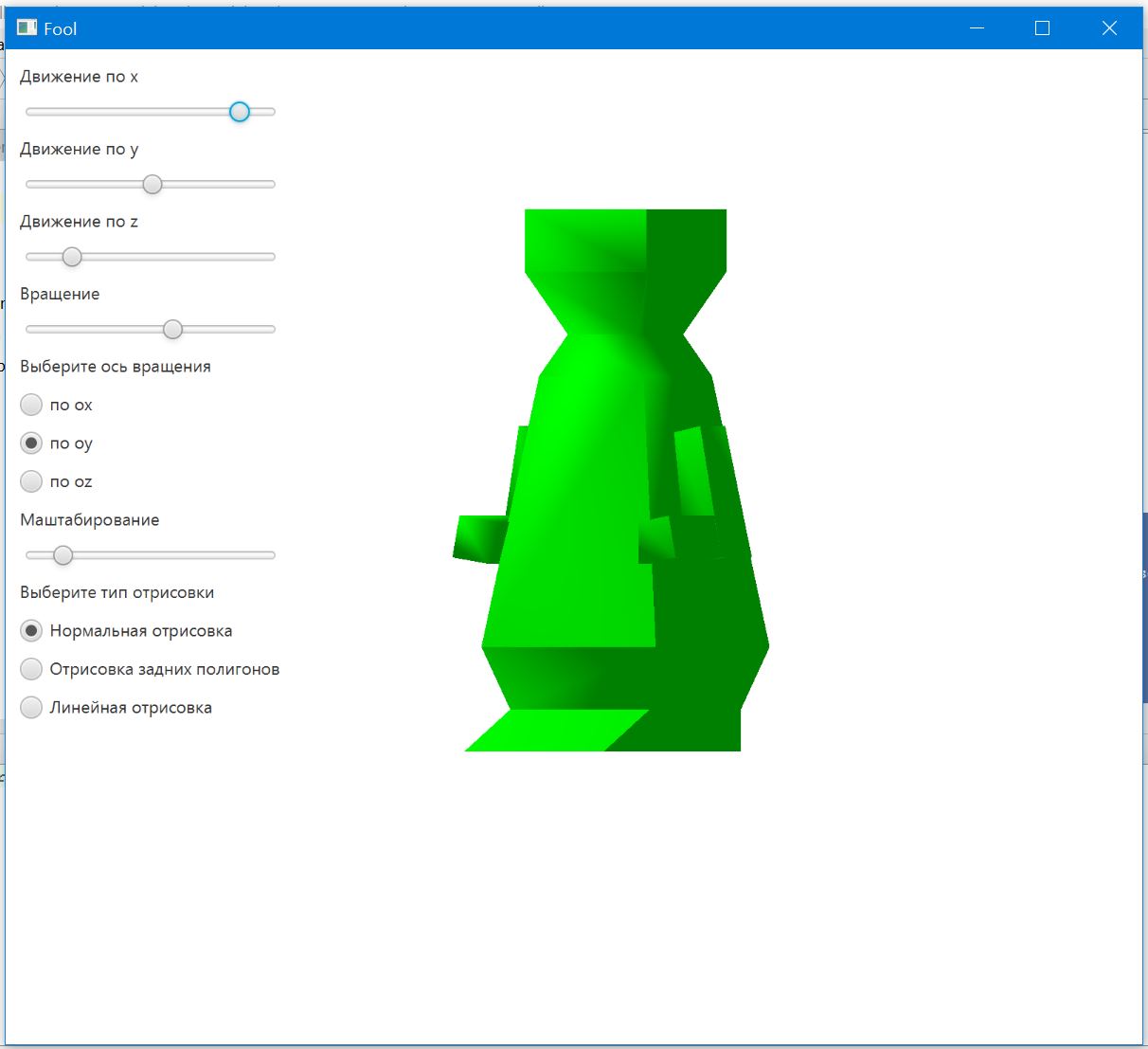


Рисунок 6.1 – Нормальна отрисовка фігури

Режим отрисовки задніх полігонів зображено на рисунку 6.2.

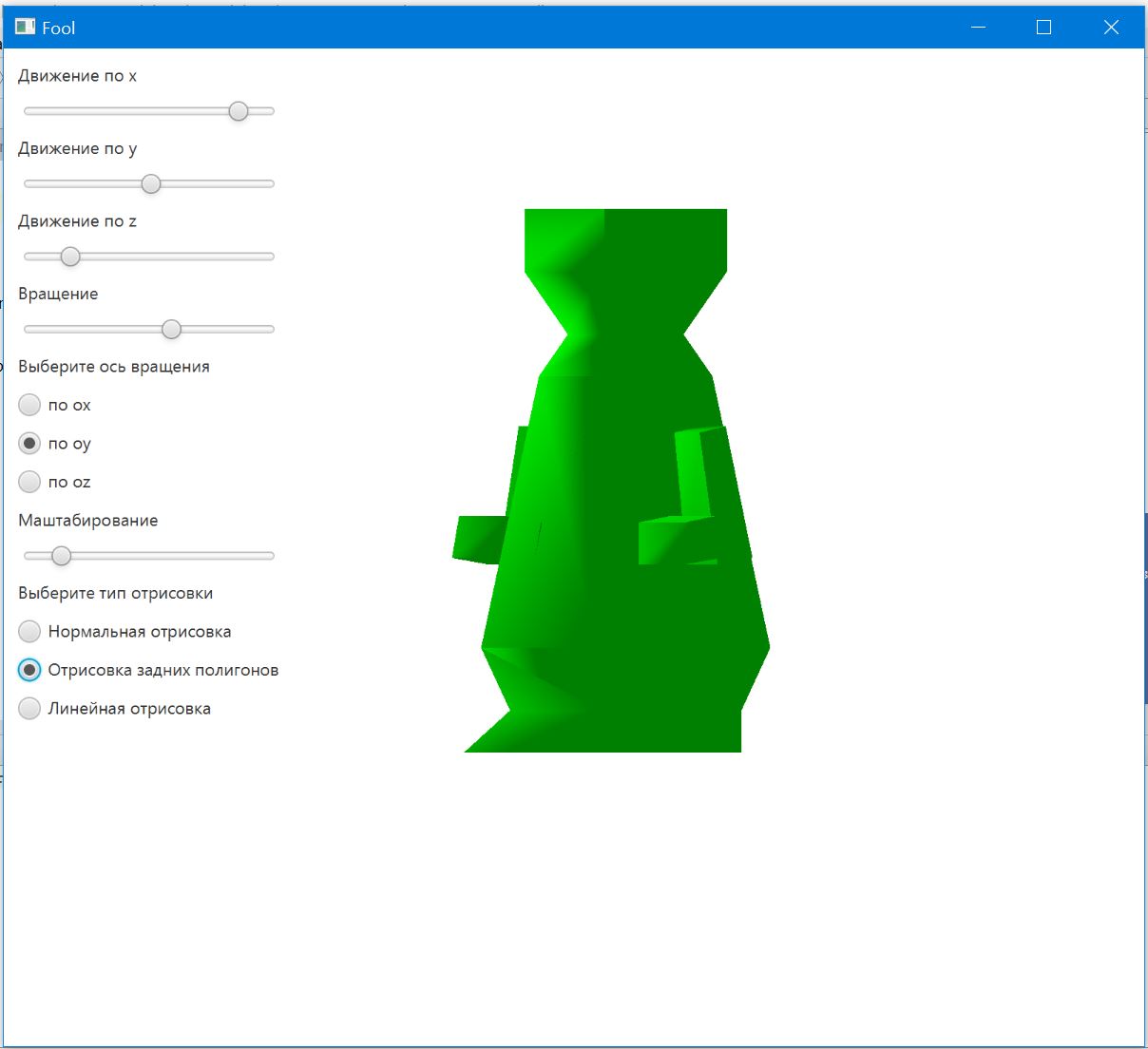


Рисунок 6.2 – Режим отрисовки задніх полігонів

Режим отрисовки ліній зображено на рисунку 6.3.

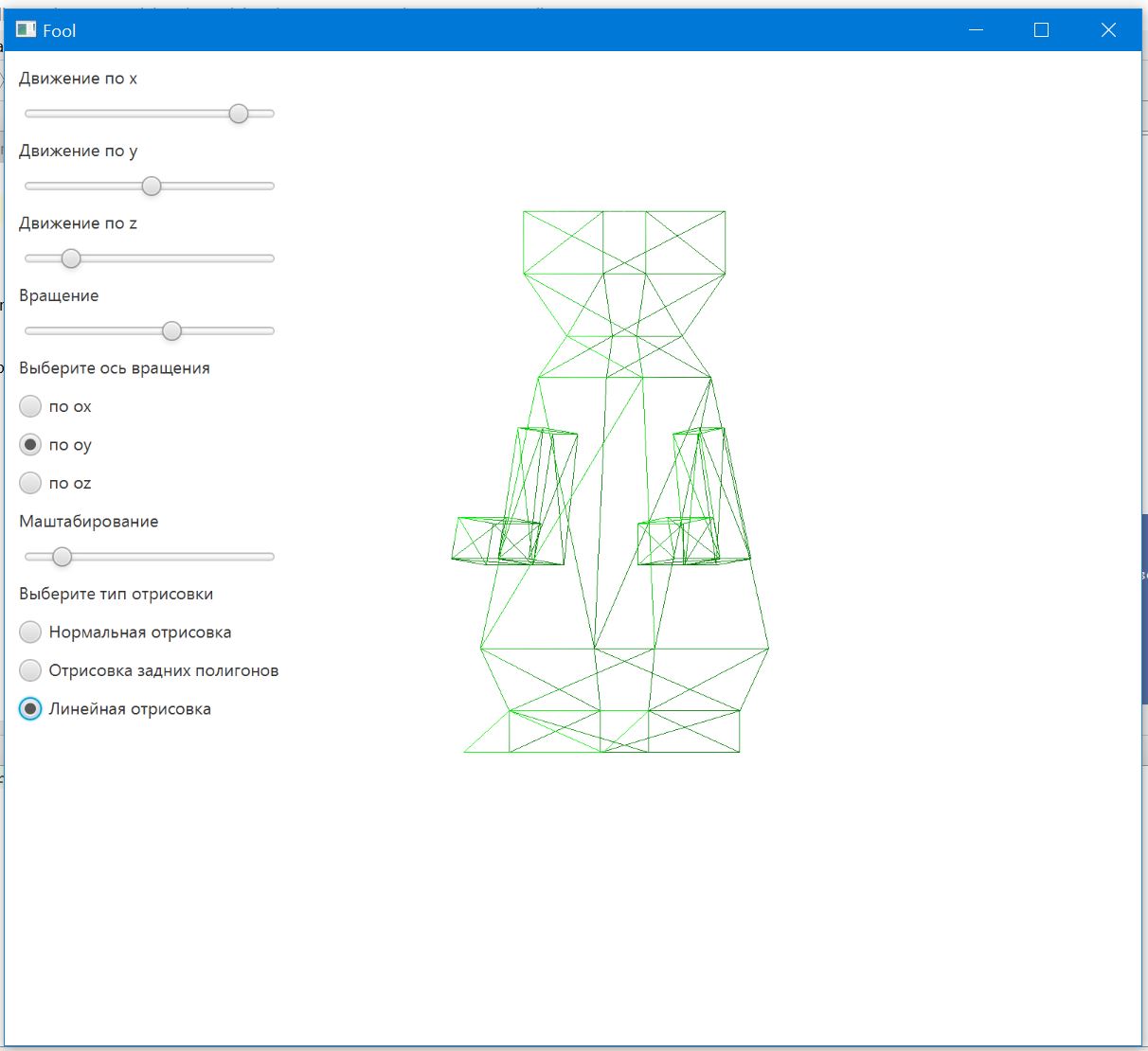


Рисунок 6.3 – Режим отрисовки лінії