Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “Об’єктно-орієнтоване програмування”

«БІБЛІОТЕКА КЛАСІВ ДЛЯ РОБОТИ З ГРАФІЧНИМИ ПРИМІТИВАМИ»

Керівник, проф. Бондарєв В. М.

Студент гр. ПЗПІ-17-1 Кириченко О. В.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В. М.

ст.викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В.М.

ст.викл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю.

Харків 2018

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Напрям: ***Програмна інженерія***

Курс 1 .Група *ПЗПІ-17 -1* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Кириченка Олега Володимировича*

(Прізвище, Ім'я, По батькові)

1 Тема проекту: *Бібліотека класів для роботи з графічними примітивами*

2 Термін здачі студентом закінченого проекту: ***“7” - червня - 2018 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи,*

4 Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація бібліотеки, проектна специфікація бібліотеки, опис програми-демонстраціі, інструкція користувача, висновки*

5 Перелік графічного матеріалу:

*Діаграма класів бібліотеки, алгоритми, приклади екранних форм*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 15-02-2017 р. |
| 2 | Формулювання вимог до програми | 16-02-2017 – 26-02-2017 р. |
| 3 | Розробка архітектури системи | 01-03-2017 – 15-03-2017 р. |
| 4 | Розробка бібліотки класів, модульне тестування | 16-03-2017 – 01-05-2017 р. |
| 5 | Функціональне тестування доопрацювання розробленої програмної системи | 01-05-2017 – 07-05-2017 р. |
| 6 | Розробка програми-демонстрації, що містить графічний інтерфейс користувача | 08-05-2017 – 15-05-2017 р. |
| 7 | Тестування системи з використанням програми-демонстрації. | 16-05-2017 – 20-05-2017 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 21-05-2017 – 04-06-2017 р. |
| 9 | Захист | 04-06-2017 – 7-06-2017 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кириченко Олег Володимирович

(Прізвище, Ім'я, По-батькові)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв Володимир Михайлович

(Прізвище, Ім'я, По батькові)

« 15 » лютого 2017 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 30с., 28 рис., 2 додатки, 5 джерел.

Метою роботи є розробка бібліотеки класів для роботи з графічними примітивами на базі графічної бібліотеки OpenGL на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2012, платформи .NET Framework 4.0, мови програмування C#, бібліотеки OpenGL та Tao Framework.

В результаті отримана програма дозволяє створювати наступні геометричні фігури: точка, пряма, відрізок, овал, круг, парабола, гіпербола, багатокутник, прямокутник, трикутник, квадрат; Наявна можливість об’єднувати фігури в єдину систему, яка буде також сприймається як фігура. Є можливість намалювати будь-яку з перелічених фігур із використанням різних кольорів, товщин лінії, режимів малювання, можливістю накладання текстур. Над фігурами можна виконувати трансформаційні дії: переміщення, поворот, масштабування. Бібліотека дозволяє знаходити множини точок перетину фігур, для кривих – знаходити дотичну, проведену із деякої її точки, знаходити окружність кривизни у деякій точці кривої.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, OPENGL, TAO FRAMEWORK, БІБЛІОТЕКА КЛАСІВ ДЛЯ РОБОТИ З ГРАФІЧНИМИ ПРИМІТИВАМИ

ЗМІСТ

Вступ 6

1 Специфікація бібліотеки класів 7

1.1 Огляд структури проекту 7

1.1.1 Огляд класу GlFigure 8

1.1.2 Огляд класу GlCurve 10

1.1.3 Огляд допоміжних геометричних класів 11

1.1.4 Огляд особливостей класів-наслідників GlCurve 13

1.1.5 Огляд класу GlPointR2 15

1.1.6 Огляд класу GlLineR2 17

1.1.7 Огляд класу GlPolygon 18

1.1.8 Огляд класу GlFigureSystem 19

2 Специфікація до програми-демонстрації 22

2.1 Огляд функціоналу програми-демонстрації 22

Висновки 29

Перелік посилань.. 30

Додаток А Алгоритм пошуку перетину двох кривих 31

Додаток Б Діаграма класів 32

ВСТУП

На сьогоднішній день можливості відображення графічної інформації значно покращилися за допомогою наявності великої кількості графічних бібліотек, але більшість з них є досить складними для використання, а їх використання для досить простих задач не є виправданим. Використання ж, наприклад, великих двигунів для таких цілей не виправдане також. Для розв’язання цієї проблеми було розробленого бібліотеку класів на базі OpenGL.

Актуальність такої системи полягає в тому, що вона значно полегшує взаємодію розробника із деякими функціями OpenGL. Також важливо, що бібліотека не прив’язана до конкретної технології платформи .NET.

Мета курсової роботи – розробка бібліотеки класів для роботи з графічними примітивами на базі графічної бібліотеки OpenGL та програми-демонстрації. На фінальному етапі розробки проект повинен відповідати всім встановленим вимогам, а функціонал програми повинен містити такі основні пункти:

* створення та відображення різних двомірних геометричних фігур;
* трансформації фігур;
* знаходження точок перетину фігур, дотичних, окружностей кривизни;
* створення допоміжних класів, наприклад, вектор;
* створення класів, шо реалізують OpenGL-контекст для різних технологій платформи .NET.

Програма має бути надійною, задовольняти користувача та не повинна мати ніяких непередбачуванних аспектів при будь-яких діях користувача.

Завданням роботи є закріплення навичок використання стандартних методів та інструментів розробки програмного забезпечення з використанням технології Windows Forms платформи .NET для реалізації описаного програмного продукту.

СПЕЦИФІКАЦІЯ БІБЛІОТЕКИ КЛАСІВ

* 1. Огляд структури проекту

На етапі розробки архітектури системи було використано уніфіковану мову моделювання UML, за допомогою якої було розроблено діаграму класів (дод. Б), що відображує залежності між класами та декларацію їх методів та полів. Найабстрактнішим класом є клас GlFigure (рис. 1.1), від якого наслідуються класи GlPointR2, GlLineR2, GlPolygon, GlFigureSystem та абстрактний клас GlCurve (рис. 1.2).



Рис 1.1 – Структура класу GlFigure



Рис 1.2 – Структура класу GlCurve

1.1.1 Огляд класу GlFigure

Клас GlFigure декларує велику кількість абстрактних методів, які повинні бути перевизначені у його нащадках. Також наявні й не є абстрактні методи, деякі з яких викликають перевизначені методи нащадків. Клас також має абстрактні властивості та набір подій.

Властивості класу:

* BOX – властивість тільки для читання, що повертає прямокутник із сторонами, паралельними осям координат, що обмежує фігуру;
* Center – властивість тільки для читання, що повертає копію точки фігури, що вважається центральною;
* CountOfPoints - властивість тільки для читання, що повертає кількість точок фігури, на яких вона будується.

Методи класу:

* Draw() – малює фігуру поточним кольором із поточною товщиною лінії;
* Draw(Texture T) – малює фігуру, та накладає на неї текстуру T;
* Draw(GlRectangle Border) – малює ту частину фігури, що знаходиться прямокутника Border;
* Draw(GlColor drawColor) – малює фігуру кольором drawColor;
* Draw(float lineWidth) – малює фігуру товщиною лінії linewidth;
* DrawFill() – малює заповнену фігуру;

Інші методи групи Draw працюють за схожим принципом.

* getCopy() – повертає копію фігури;
* getCurrentColor() – повертає поточний колір;
* getCurrentGlWidth() – повертає поточну товщину лінії;
* Методи групи getIntersection повертають масив точок перетину двох фігур;
* getScaled(float Scale) – повертає масштабовану фігуру (оригінал не змінюється);
* isPointBelongs(GlPointR2 P) – повертає true або false в залежності від того, чи належить точка P контуру фігури;
* moveTo(float x, float y) – переміщує фігуру на координати (x; y) (якщо фігура складена – розраховується вектор переносу для її центра, за яким переносяться усі точки фігури);
* Rotate(float angle) та Rotate(float SIN, float COS) – повертає фігуру на кут angle.

1.1.2 Огляд класу GlCurve

Клас GlCurve – також абстрактний. Він потрібен тому, що криві, класи для яких наявні у бібліотеці, мають багато схожого. Також у цьому класі перевизначені деякі з абстрактних методів та властивостей класа GlFigure, наявні власні абстрактні методи та властивості.

Властивості класу:

* CenterX, CenterY – властивості, що позначають координати центра кривої, а також початка координат системи, у якій рівняння кривої має канонічний вигляд;
* COS, SIN – тригонометричні функції кута між горизонтом та віссю Ox системи, у якій рівняння кривої має канонічний вигляд;
* DirectVector – об’єкт класу GlVectorR2, що позначає напрям осі Ox системи, у якій рівняння кривої має канонічний вигляд;
* RealBox - властивість тільки для читання, що повертає прямокутник, що обмежує фігуру;
* curvePoints[] – масив точок кривої. Наявний індексатор тільки для читання, що повертає копію точки кривої за її індексом;
* getCurvatureCircle(GlPointR2 P) – повертає окружність кривизни для кривої у точці P;
* getFDiff(float X), getSDiff(float X) – повертає значення першої та другої похідної відповідно у точці з абсцисою X;
* getTangentFromBelongs(GlPointR2 P) – повертає об’єкт класу GlLineR2, що є дотичною у до кривої у точці P, що належить кривій;
* isPointInside(GlPointR2 P) – повертає true або false в залежності від того, чи знаходиться точка P усередині частини площини, обмеженої кривою;
* Методи ActivateDrawStart(), ActivateDrawing(), ActivateDrawed() – допоміжні private-методи, що ініціюють відповідні події;
* Методи DrawPoints та updatePointsPosition – допоміжні private-методи.

1.1.3 Огляд допоміжних геометричних класів

У бібліотеці наявні наступні допоміжні класи: GlVectorR2 (рис. 1.3) – вектор, GlLineSegment (рис. 1.4) – відрізок. Клас GlLineSegment наслідується від класа GlVectorR2.



Рис 1.3 – Структура класу GlVectorR2



Рис 1.4 – Структура класу GlLineSegment

Клас GlVectorR2 містить:

* protected-поля dx та dy, що позначають його координати, та відповідні властивості;
* Властивість Length, що позначає довжину вектора;
* Константу FAULT, що позначає припустиму похибку при розрахунках;
* Перевантажені бінарні оператори: “+” – сума векторів, “-” – різниця векторів, “\*” – скалярний добуток векторів або добуток вектора та числа, “^” – векторний добуток векторів;
* Метод fromPointToPoint(GlPointR2 P) – повертає точку, на яку буде вказувати вектор, якщо розмістити його початок у точці P;
* Метод getReversedVector() – повертає протилежний вектор (оригінал не змінюється);
* Методи getRotatedVector(float angle) та getRotatedVector(float SIN, float COS) – повертають вектор, повернутий на кут angle (оригінал не змінюється);
* Методи для перевірки колінеарності та перпендикулярності векторів.

Клас GlLineSegment містить:

* private-поля startPoint та endpoint, що позначають точки, якими обмежений відрізок, та відповідні властивості;
* Константу FAULT, що позначає припустиму похибку при розрахунках;
* Метод для графічного відображення Draw;
* Метод isPointBelongs(GlPointR2 P), який повертає true або false, в залежності від того, чи належить точка P відрізку.

1.1.4 Огляд особливостей класів-наслідників GlCurve

Деякі з класів-наслідників GlCurve, звичайно, мають свої особливості. Наприклад, клас GlOval (рис. 1.5) має властивість Focuses, що позначає масив з двох точок, що є фокусами овала. Також клас GlOval містить метод getTangentFromPoint(GlPointR2 P), що повертає масив з двох прямих, які є дотичними до овала, проведеними із точки P, що не належить овалу. Клас GlOval має наслідника – GlCircle.

Клас GlHyperbola (рис. 1.6) не має додаткових методів та властивостей.

Клас GlParabola (рис. 1.7) має властивість Focus, що позначає фокус параболи, властивість Directriss, що позначає пряму-директрису параболи. Також наявна властивість A, яку можна змінювати зовні і яка позначає старший коефіцієнт у канонічному рівнянні параболи.



Рис 1.5 – Структура класу GlOval



Рис 1.6 – Структура класу GlHyperbola



Рис 1.7 – Структура класу GlParabola

1.1.5 Огляд класу GlPointR2

Клас GlPointR2 (рис. 1.8) має дуже важливу роль через те, що усі фігури в бібліотеці будуються на деякій множині точок.

Клас GlPointR2 містить:

* private-поля x та y, що позначають координати точки, та відповідні властивості;
* Метод Distance(GlPointR2 P), що повертає відстань до точки P;
* Метод getPointTranslatedToRotatedSystem(float SIN, float COS, GlPointR2 systemCenter), що повертає копію точки, перенесену до системи координат, обернутої на деякий кут, із центром в точці systemCenter;
* Метод getTranslatedBackPoint(float SIN, float COS, GlPointR2 systemCenter), що повертає копію точки, перенесену до початкової системи із системи, оберненої на деякий кут, із центром в точці systemCenter.



Рис 1.8 – Структура класу GlPointR2

1.1.6 Огляд класу GlLineR2

Клас GlLineR2 (рис. 1.9) також є дуже важливим через те, що усі методи пошуку перетину перетину фігур, які не використовують чіткі формули, шукають перетин через багатократний пошук перетин прямих та відрізків. Такий пошук перетину працює для пар кривих.



Рис 1.9 – Структура класу GlLineR2

Клас GlLineR2 містить:

* private-поля pointOfLine та directVector, що позначають точку прямої та напрямний вектор відповідно, та відповідні властивості, що повертають їх копії;
* Перевантаження метода Draw для відображення частини прямої конкретної довжини;
* Методи для пошуку паралельних та перпендикулярних прямих, що проходять через задану точку;
* Метод getProjection(GlPointR2 P), що повертає проекцію точки P на пряму.

1.1.7 Огляд класу GlPolygon

Клас GlPolygon (рис. 1.10) позначає набір точок, що при відображенні послідовно з’єднуються.

Клас GlPolygon містить:

* private-масив вершин багатокутника та індексатор для нього;
* Властивості P та S, що позначають периметр та площу багатокутника відповідно;
* Метод isPointInside(GlPointR2 P), що повертає true або false в залежності від того, чи знаходиться точка P всередини багатокутника;
* Метод AddVertex(GlPointR2 P), що додає точку P до вершин багатокутника та викликає метод перерахування координат центра багатокутника;
* Події OnVertexAddingStart, OnVertexAdding, OnVertexAdded. Підписник події OnVertexAddingStart повинен повернути true або false. При поверненні false додання вершини не буде проведене.



Рис 1.10 – Структура класу GlPolygon

1.1.8 Огляд класу GlFigureSystem

Клас GlFigureSystem (рис. 1.11) позначає набір фігур, що сприймаються як єдине ціле.



Рис 1.11 – Структура класу GlFigureSystem

Клас GlFigureSystem містить:

* private-масив фігур, що містить система та індексатор для нього;
* Властивості CountOfPoints та CountOfFigures, що позначають кількість фігур в системі та загальну кількість точок;
* Метод AddFigure(GlFigure F), що додає до системи фігуру F;
* Події OnFigureAddingStart, OnFigureAdding, OnFigureAdded. Підписник події OnFigureAddingStart повинен повернути true або false. При поверненні false додання вершини не буде проведене.

СПЕЦИФІКАЦІЯ ДО ПРОГРАМИ-ДЕМОНСТРАЦІЇ

Для демонстрації роботи бібліотеки та тестування в реальних умовах було розроблено спеціалізовану програму (рис. 2.1).

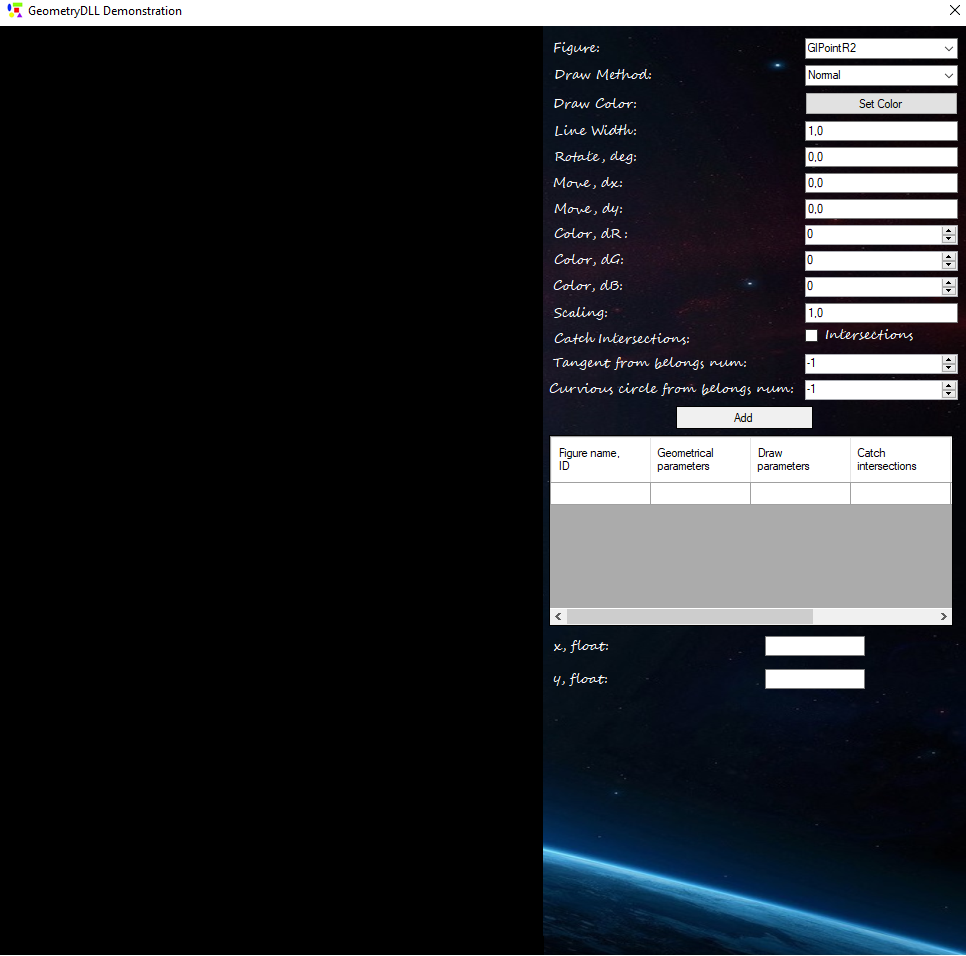


Рис 2.1 – Головна форма програми

У програмі-демонстрації наявні можливості наступні можливості:

* Вибір фігури, що буде додана (рис. 2.2)
* Вибір режиму відображення фігури (рис. 2.3)
* Вибір кольору для малювання (рис. 2.4)
* Вибір товщини пензлика (рис. 2.5)
* Вибір кута обернення фігури за один кадр (рис. 2.6)
* Вибір переміщення фігури за один кадр (рис. 2.7)
* Вибір зміни кольору за один кадр (рис. 2.8)
* Вибір зміни масштабу за один кадр (рис. 2.9)
* Вибір режиму відображення перетинів фігур (рис. 2.10)
* Вибір номера точки, що належить кривій, для побудови дотичної (рис. 2.11)
* Вибір номера точки, що належить кривій, для побудови окружності кривизни (рис. 2.12)
* Вибір режиму відображення обмежуючого прямокутника (рис. 2.13)
* Вибір геометричних параметрів фігури (рис. 2.14)
* Перегляд доданих фігур (рис. 2.15)
* Редагування фігури (рис. 2.16)
* Перегляд результату у режимі реального часу (рис. 2.17)

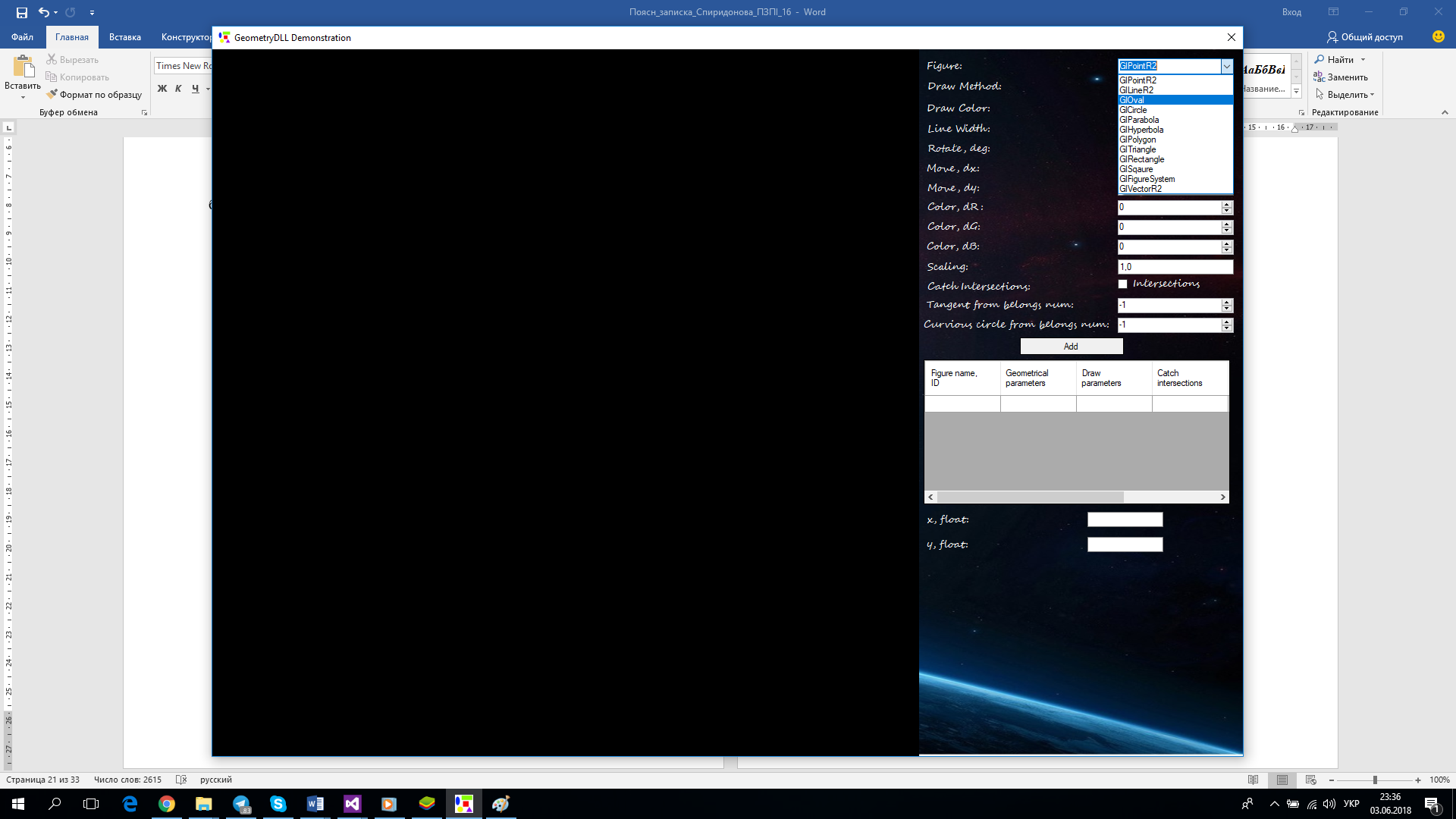


Рис 2.2 – Вибір виду фігури, що буде додана



Рис 2.3 – Вибір режиму відображення фігури

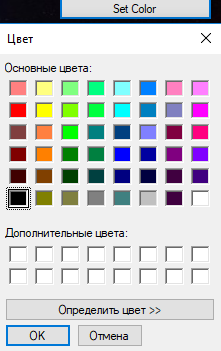


Рис 2.4 – Вибір кольору для малювання



Рис 2.5 – Вибір товщини пензлика



Рис 2.6 – Вибір кута обернення фігури за один кадр



Рис 2.7 – Вибір переміщення фігури за один кадр



Рис 2.8 – Вибір зміни кольору за один кадр



Рис 2.9 – Вибір зміни масштабу за один кадр



Рис 2.10 – Вибір режиму відображення перетинів фігур



Рис 2.11 – Вибір номера точки, для побудови дотичної



Рис 2.12 – Вибір номера точки, для побудови окружності кривизни



Рис 2.13 – Вибір режиму відображення обмежуючого прямокутника

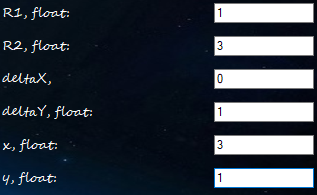


Рис. 2.14 - Вибір геометричних параметрів фігури

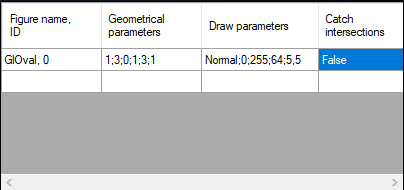


Рис. 2.15 – Перегляд доданих фігур

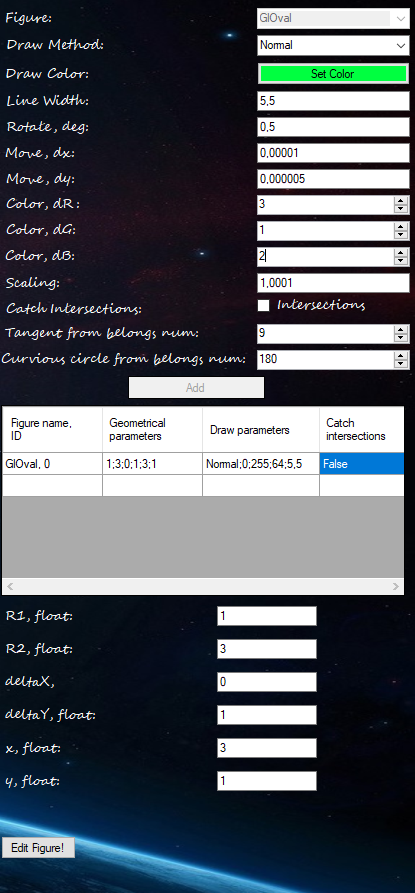


Рис. 2.16 – Редагування фігури після додання

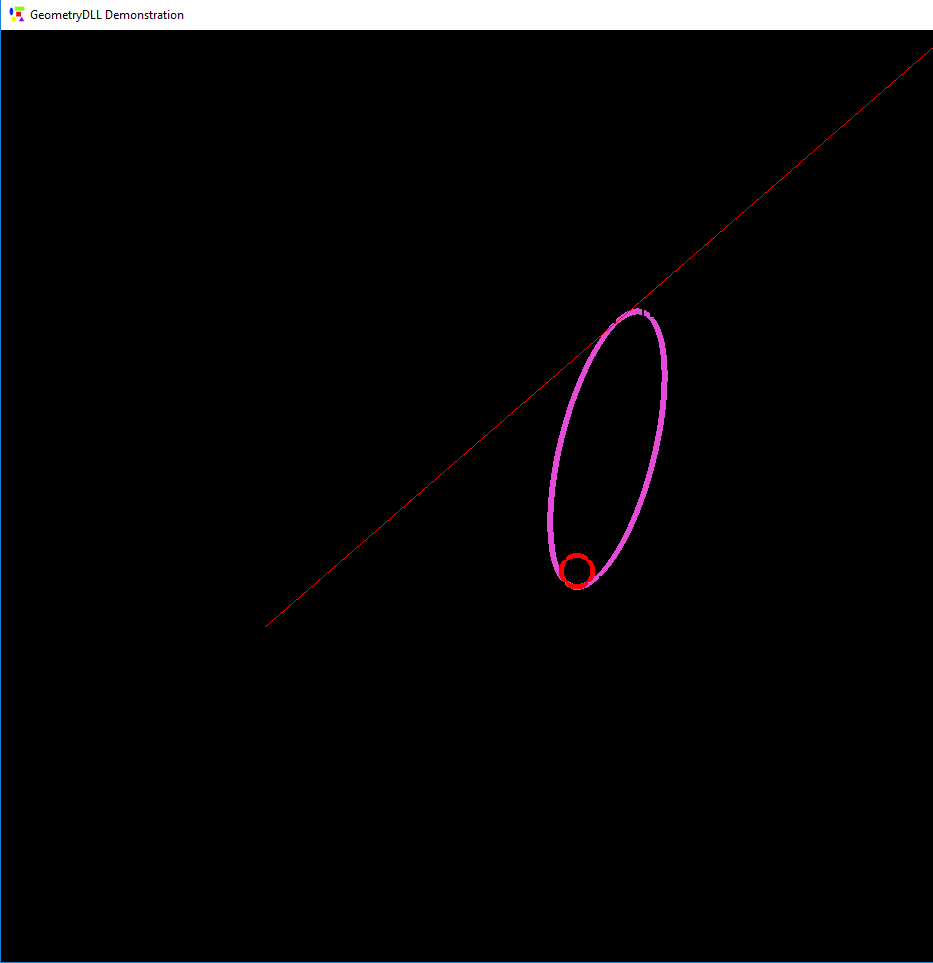


Рис. 2.17 – Результат

ВИСНОВКИ

Під час виконання курсової роботи було розроблено бібліотеку класів для роботи з графічними примітивами на базі OpenGl та Tao Framework. Отриманий програмний продукт значно спрощує роботу з ОpenGl, дозволяє виконувати прості задачі з графічного моделювання за короткі проміжки часу. Також було створено демонстраційну програму, що містить графічний інтерфейс користувича, яка, хоч і не розкриває потенціал бібліотеки в повному обсязі, але надає доступ к основним її функціям. У створеному програмному продукті були реалізовані поставлені задачі у повному обсязі.

Ця курсова робота є прикладом програми, що полегшує та спрощує роботу з графічними данними.

В ході виконання роботи були закріплені навички роботи зі стандартними методами та інструментами програмного забезпечення з використанням технології WinForms платформи .NET, а також бібліотеки OpenGL.

В результаті було створено бібліотеку класів та демонстраційну програму, що являє собою віконний прикладний додаток, написаний об’єктно-орієнтованою мовою C#.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бондарев, В.М. Объектно-ориентированное программирование на С#[Текст] : учеб. пособ. / В.М. Бондарев. – Х. : Компания СМИТ, 2009. – 224 с. – ISBN 978-966-2028-34-8

2. Microsoft Developer Network. Библиотека MSDN. Разработка на .NET [Электронный ресурс] //MSDN – сеть разработчиков Microsoft. – Режим доступа: https://msdn.microsoft.com/ru-ru/ - 17.04.2017 г. – Загл. с экрана.

3. Шилдт Г. C# 4.0.: Полное руководство [Текст]: пер. с англ.- М.: Вильямс, 2011 - 1056 с. - ISBN 978-5- 8459-1684- 6.

4. ProfessorWeb [Электронный ресурс]// Программирование на языке C#, платформа .NET Framework. Режим доступа: https://professorweb.ru 29.04.2017 г. – Загл. с экрана.

5. Metanit [Электронный ресурс] //metanit.com: информ.-справочный портал. Режим доступа: https://metanit.com/ − 22.04.2017 – Загл. с экрана.

ДОДАТОК А

Алгоритм пошуку перетину двох кривих

public override GlPointR2[] getIntersection(GlCurve C){

if (C == null || C.CountOfPoints == 0)

return new GlPointR2[] { };

List<GlPointR2> Intersections =

new List<GlPointR2>();

bool Side = this.isPointInside(C[0]);

for (int i = 1; i < C.CountOfPoints; i++)

if (this.isPointInside(C[i]) != Side){

int next = (i == C.CountOfPoints - 1) ?

0 : i + 1;

int prev = i == 1 ?

C.CountOfPoints - 1 : i - 2;

GlPointR2[] faultInter = this.getIntersection(

new GlLineR2(C[i],

new GlVectorR2(C[next].X - C[prev].X,

C[next].Y -C[prev].Y)));

for (int k = 0; k < faultInter.Length; k++)

if (C.isPointBelongs(faultInter[k])){

Intersections.Add(faultInter[k]);

Side = !Side;

}

}

return Intersections.ToArray();

}