OpenGL. Вступ.

Коротко про OpenGL

OpenGL — (англ. **Open G**raphics **L**ibrary — відкрита графічна бібліотека) — специфікація, що визначає незалежний від мови прогамування крос-платформовий програмний інтерфейс для написання застосункув, що використовують дво- та тривимірну графіку. Цей інтерфейс містить понад 250 функцій, які можуть використовуватися для малювання складних тривимірних сцен з простих примітивів.

На базовому рівні, OpenGL — це специфікація, тобто, — документ, який описує набір функцій та їх точну поведінку. На основі цих специфікацій виробники апаратного забезпечення створюють реалізації — бібліотеки функцій, які відповідають заявленій в OpenGL специфікації.

Ефективні реалізації OpenGL існують для ОС Linux, MacOS X, Microsoft Windows та багатьох Unix-подібних ОС, а також для таких ігрових боксів, як Sony PlayStation 3.

Різні програмні реалізації OpenGL існують для платформ, виробники яких не підтримують дану специфікацію. Відкрита (open sourse) бібліотека Mesa — повністю OpenGL сумісний програмний API.

OpenGL обслуговує дві цілі:

- Для того, щоб приховувати складнощі встановлення зв'язку комп'ютера з різними 3D 3D акселераторами, надати програмістові один, загальноприйнятий API
- Для того, щоб приховати можливості базових інструментальних машин, які відрізняються своїм намаганням виконати підтримку повного набору особливостей OpenGL (використання програмної емуляції, якщо необхідно).

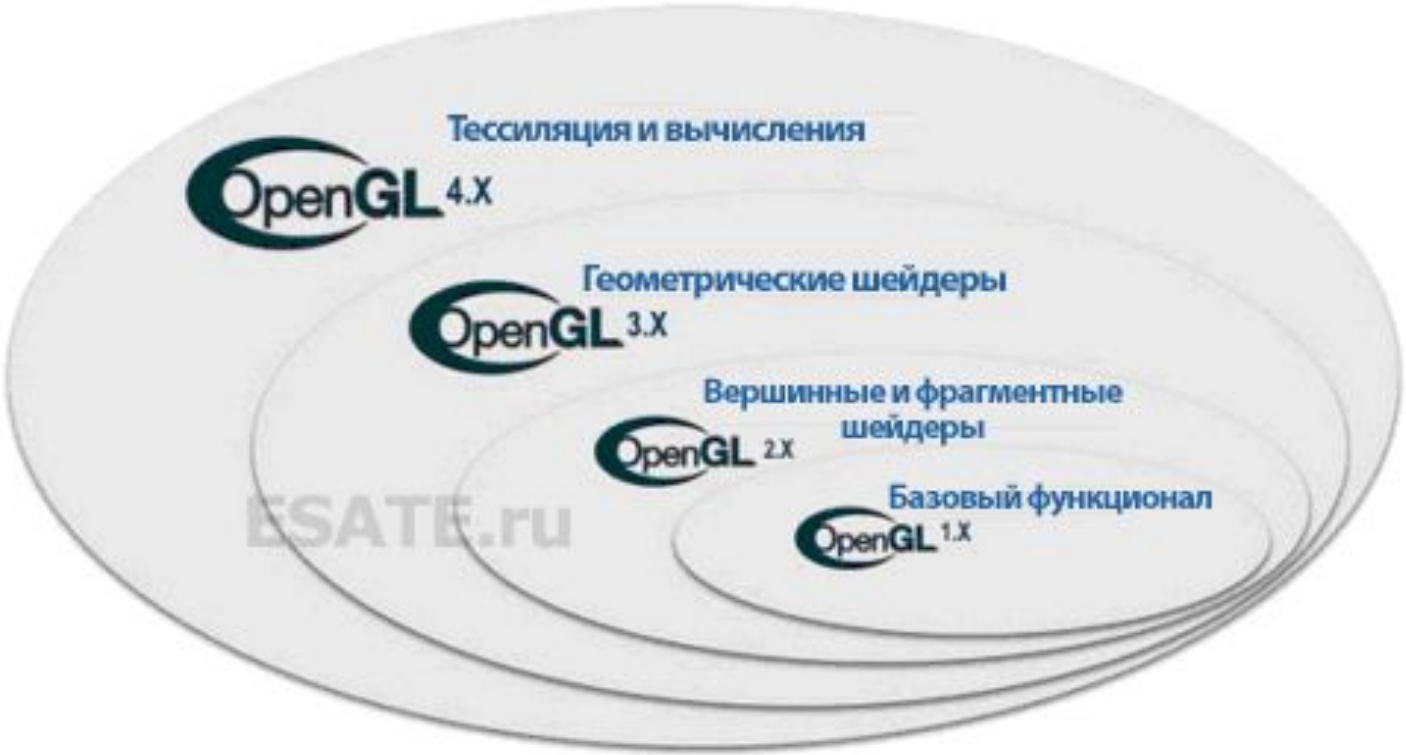
Історія

Стандарт OpenGL Architecture Review Board (OpenGL ARB) був розроблений і затверджений у 1992 році передовими компаніями в області розробки апаратного та програмного забезпечення для роботи з комп'ютерною графікою. Основа - бібліотека IRIS GL, розроблена компанією Silicon Graphics.

Курт Акелей (Kurt Akeley) і Марк Сігал (Mark Segal) створили оригінальну специфікацію OpenGL. Кріс Фразаєр (Chris Frazier) редагував версію 1.1, а Джон Ліч (Jon Leech) версії з 1.2 по 2.0.

Огляд специфікацій

- OpenGL 1.1 - Texture objects
- OpenGL 1.2 - 3D textures, BGRA and packed pixel formats[23]
- OpenGL 1.3 - Multitexturing, multisampling, texture compression
- OpenGL 1.4 - Depth textures
- OpenGL 1.5 - Vertex Buffer Object (VBO), Occlusion Queries[24]
- OpenGL 2.0 - GLSL 1.1, MRT, Non Power of Two textures, Point Sprites,[25] Two-sided stencil[24]
- OpenGL 2.1 - GLSL 1.2, Pixel Buffer Object (PBO), sRGB Textures[24]
- OpenGL 3.0 - GLSL 1.3, Texture Arrays, Conditional rendering, Frame Buffer Object (FBO)[26]
- OpenGL 3.1 - GLSL 1.4, Instancing, Texture Buffer Object, Uniform Buffer Object, Primitive restart[27]
- OpenGL 3.2 - GLSL 1.5, Geometry Shader, Multi-sampled textures[28]
- OpenGL 3.3 - GLSL 3.30 Backports as much function as possible from the OpenGL 4.0 specification
- OpenGL 4.0 - GLSL 4.00 Tessellation on GPU, shaders with 64-bit precision,[29]
- OpenGL 4.1 - GLSL 4.10 Developer-friendly debug outputs, compatibility with OpenGL ES 2.0,[30]
- OpenGL 4.2 - GLSL 4.20 Shaders with atomic counters, draw transform feedback instanced, shader packing, performance improvements
- OpenGL 4.3 - GLSL 4.30 Compute shaders leveraging GPU parallelism, shader storage buffer objects, high-quality ETC2/EAC texture compression, increased memory security, a multi-application robustness extension, compatibility with OpenGL ES 3.0,[31]
- OpenGL 4.4 - GLSL 4.40 Buffer Placement Control, Efficient Asynchronous Queries, Shader Variable Layout, Efficient Multiple Object Binding, Streamlined Porting of Direct3D applications, Bindless Texture Extension, Sparse Texture Extension,[32]
- OpenGL 4.5 - GLSL 4.50 Direct State Access (DSA), Flush Control, Robustness, OpenGL ES 3.1 API and shader compatibility, DX11 emulation features
- OpenGL 4.6 - GLSL 4.60 More efficient geometry processing and shader execution, more information, no error context, polygon offset clamp, SPIR-V, anisotropic filtering


Тессиляция и вычисления
OpenGL 4.X
Геометрические шейдеры
OpenGL 3.X
Вершинные и фрагментные шейдеры
OpenGL 2.X
Базовый функционал
OpenGL 1.X
ESATE.ru

Огляд OpenGL 4.6

Release date: July 31, 2017

- more efficient, GPU-sided, geometry processing
- more efficient shader execution (AZDO)
- more information through statistics, overflow query and counters
- higher performance through no error handling contexts
- clamping of polygon offset function, solves a shadow rendering problem
- SPIR-V shaders
- Anisotropic filtering

Тут: шейдер — спеціальна підпрограма, що виконується на GPU. Шейдери для OpenGL пишуть на C-подібній мові — GLSL.

Посилання: https://www.opengl.org/

Маєте чути про …

- *Direct3D* - основний конкурент OpenGL
- *Metal (API*) - графічний API для iOS, macOS, tvOS, watchOS
- *OpenGL ES* – OpenGL для вкладених систем
- OpenSL ES – API для аудіо, розроблена Khronos Group
- *OpenVG* – API для прискореної 2D графіки, розроблена Khronos Group
- *RenderMan Interface Specification (RISpec)* – Pixar's open API для фотореалістичного автономного рендерінга
- *VOGL* – відладчик для OpenGL
- *Vulkan* – крос-платформений 2D and 3D графічний API, “наступне покоління OpenGL“

Перелік мов і основних бібліотек

Бібліотеки існують для наступних мов: Common Lisp, Java, Delphi, Fortran, Ocaml, Perl, Python, Racket, Ruby, Haskell і т.д.

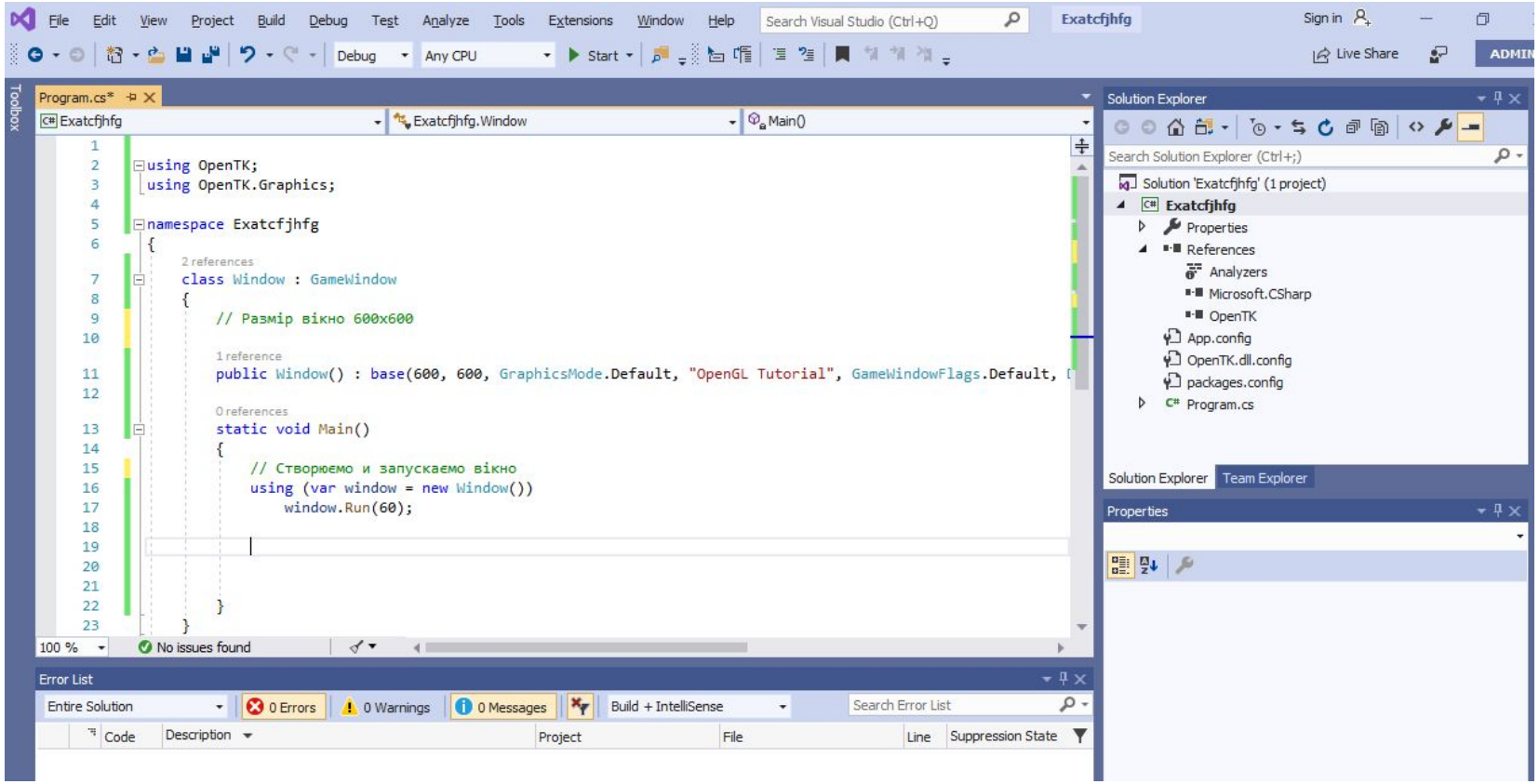
Повний список за посиланням:

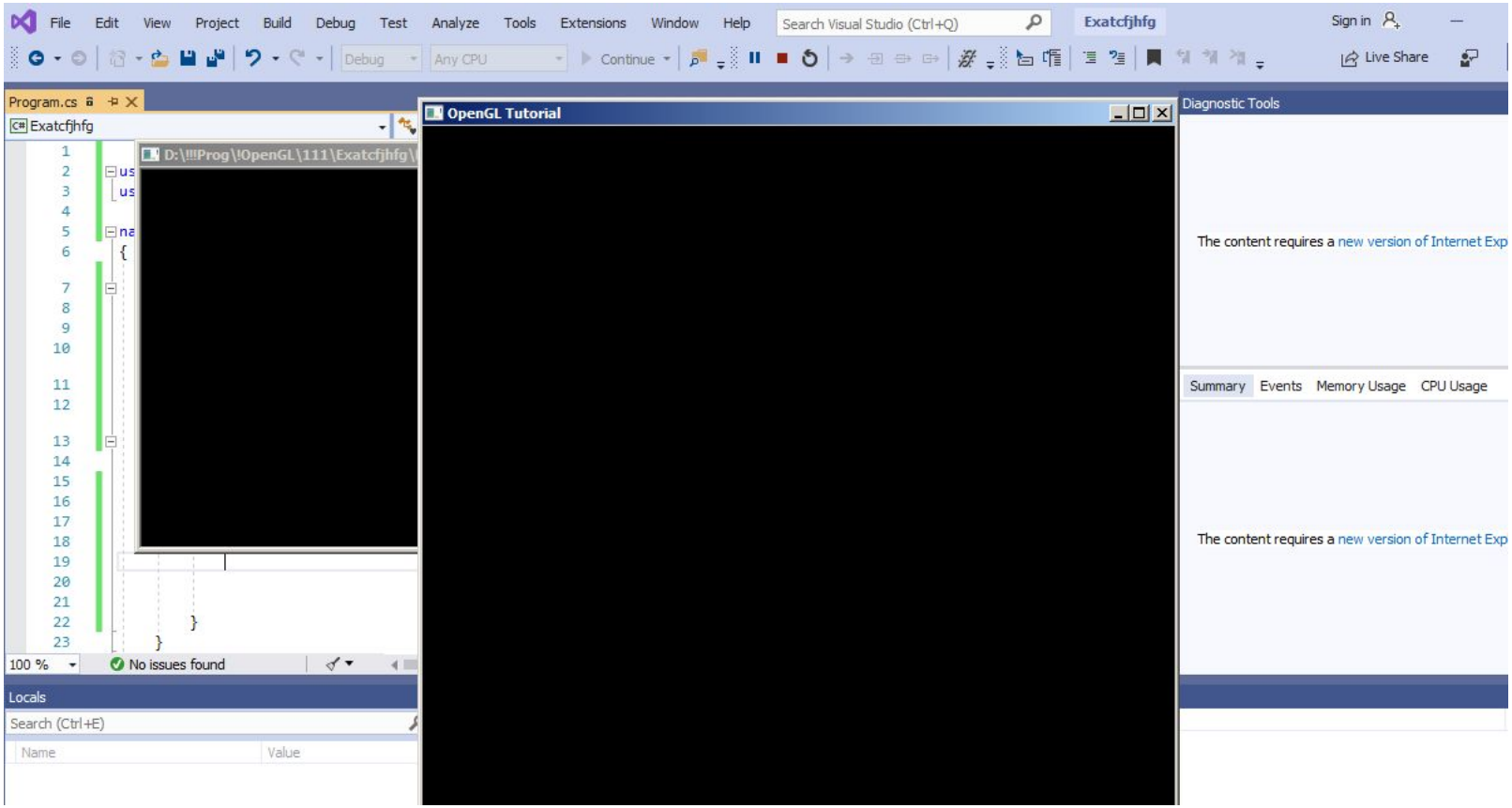
https://www.khronos.org/opengl/wiki/Language_bindings

C# (Windows Forms в Microsoft Visual Studio):

- TaoFramework (останнє оновлення в 2008 р.);
- SharpGL;
- OpenTK;
- OpenGL.Net

Створення вікна




File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
Exatcfjhfg
Sign in
Debug
Any CPU
Continue
Live Share
Program.cs
Exatcfjhfg
D:\!!!Prog\!OpenGL\111\Exatcfjhfg\
OpenGL Tutorial
Diagnostic Tools
The content requires a new version of Internet Exp
Summary Events Memory Usage CPU Usage
The content requires a new version of Internet Exp
100 %
No issues found
Locals
Search (Ctrl+E)
Name
Value

Загальний вигляд команди

rtype glCommand_name {1 2 3 4}{b s i f d ub us ui}[v](type1 arg1, ……type N argN);

де *rtype* – тип, який повертає функція, *gl* – ім'я бібліотеки OpenGl, в якій описана ця функція (для бібліотек Glu, Glut – це імена *glu* та *glut* відповідно), *Command_name* – ім'я команди, *{1 2 3 4}* – кількість аргументів команди, *{b s i f d ub us ui}* – тип аргументу (суфікс): символ *b* означає тип *GLbyte* (аналог *char* у C/C++), символ *f* – тип *GLfloat* (аналог *float*), символ *i* – тип *GLint* (аналог *long*)

Наявність символу *[v]* вказує, що як параметр функції використовується вказівник на масив значень.

Символи у квадратних дужках у деяких назвах не використовуються. Наприклад, команда *glVertex2i( )* описана як базова в бібліотеці OpenGL і використовує як параметри два цілих числа, а команда *glColor3fv(a)* використовує як параметр вказівник на масив *a* із трьох дійсних чисел.

Координатні осі розташовані так, що точка (0, 0) знаходиться в лівому нижньому куті екрана, вісь *x* напрямлена вліво, вісь *y* – вгору, а вісь *z* – з екрана на нас.

Суфікс	Опис	Тип у С	Тип в OpenGL
b	8-бітне ціле	signed char	GLbyte
s	16-бітне ціле	short	GLshort
i	32-бітне ціле	long	GLint, GLsizei
f	32-бітне число з плаваючою крапкою	float	GLfloat, GLclampf
d	64-бітне число з плаваючою крапкою	double	GLdouble, GLclampd
ub	8-бітне беззнакове ціле	unsigned char	GLubyte, GLboolean
us	8-бітне беззнакове ціле	unsigned short	GLushort
ui	8-бітне беззнакове ціле	unsigned long	GLuint, GLenum
		void	GLvoid

Примітиви

GL_POINTS – набір окремих точок;
GL_LINES – пари вершин, які задають окремі відрізки;
GL_LINE_STRIP – незамкнена ламана;
GL_LINE_LOOP – замкнена ламана;
GL_POLYGON – простий опуклий многокутник, внутрішня область якого заповнена поточними кольором;
GL_TRIANGLES – трійки вершин, що визначають вершини окремих трикутників;
GL_TRIANGLE_STRIP – трикутний стріп, де кожна наступна вершина задає трикутник разом із двома попередніми;
GL_TRIANGLE_FAN – віяло трикутників (фен) ;
GL_QUARD_STRIP – чотирикутний стріп, в якому чотирикутник з номером n визначається вершинами з номерами $2n - 1$, $2n$, $2n + 2$, $2n + 1$.

Корисно

……..

glMatrixMode(GL_PROJECTION);

glLoadIdentity();

glOrtho(0.0f, windowWidth, windowHeight, 0.0f, 0.0f, 1.0f);

…….

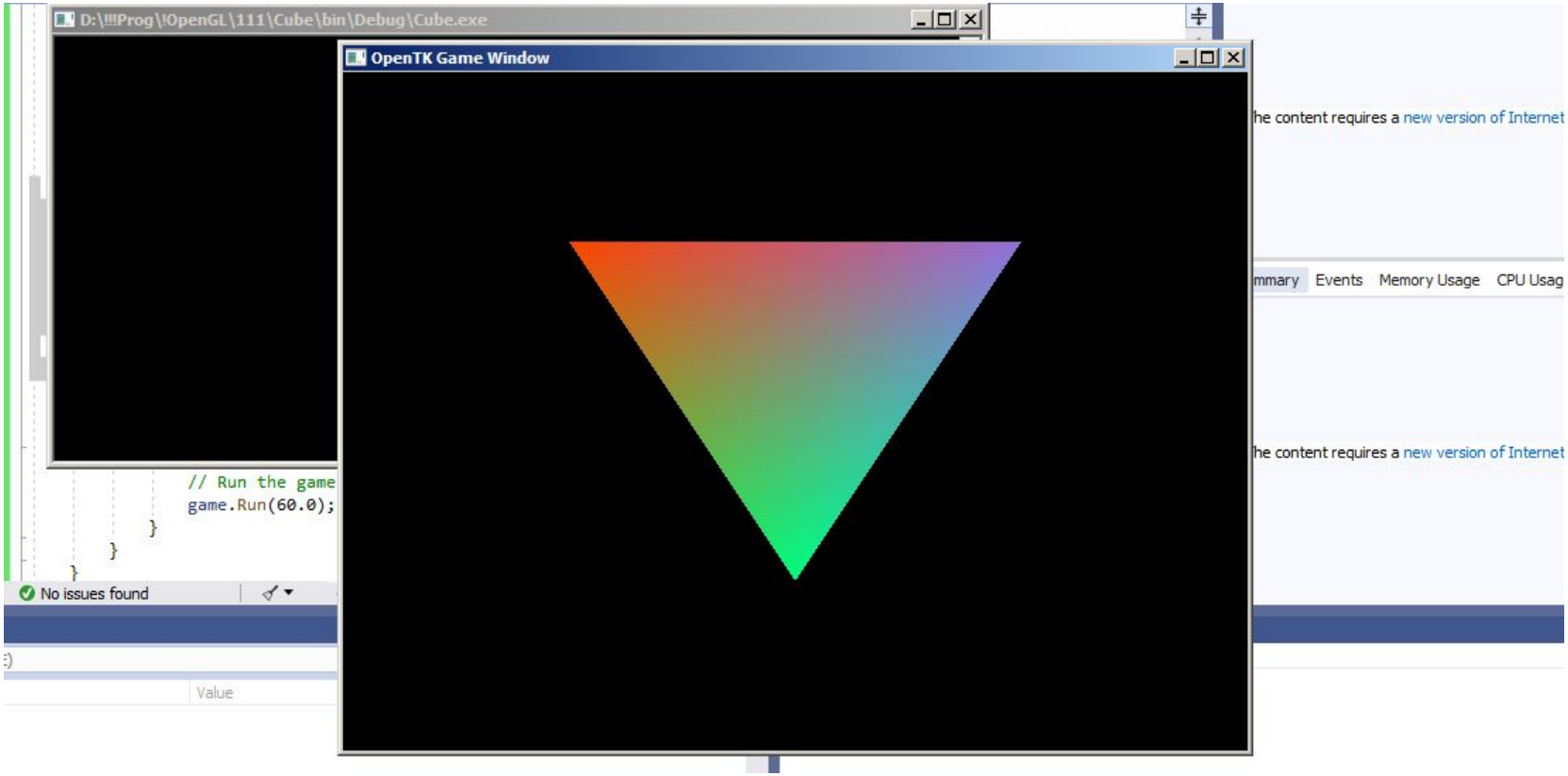
Це перераховує координати OpenGL в еквівалентні значення пікселів (X переходить від 0 до windowWidth і Y переходить від 0 до windowHeight).

ТУТ значення перевернуті: значення (0,0) для Y починається з верхнього лівого кута , а не з НИЖНЬОГО.

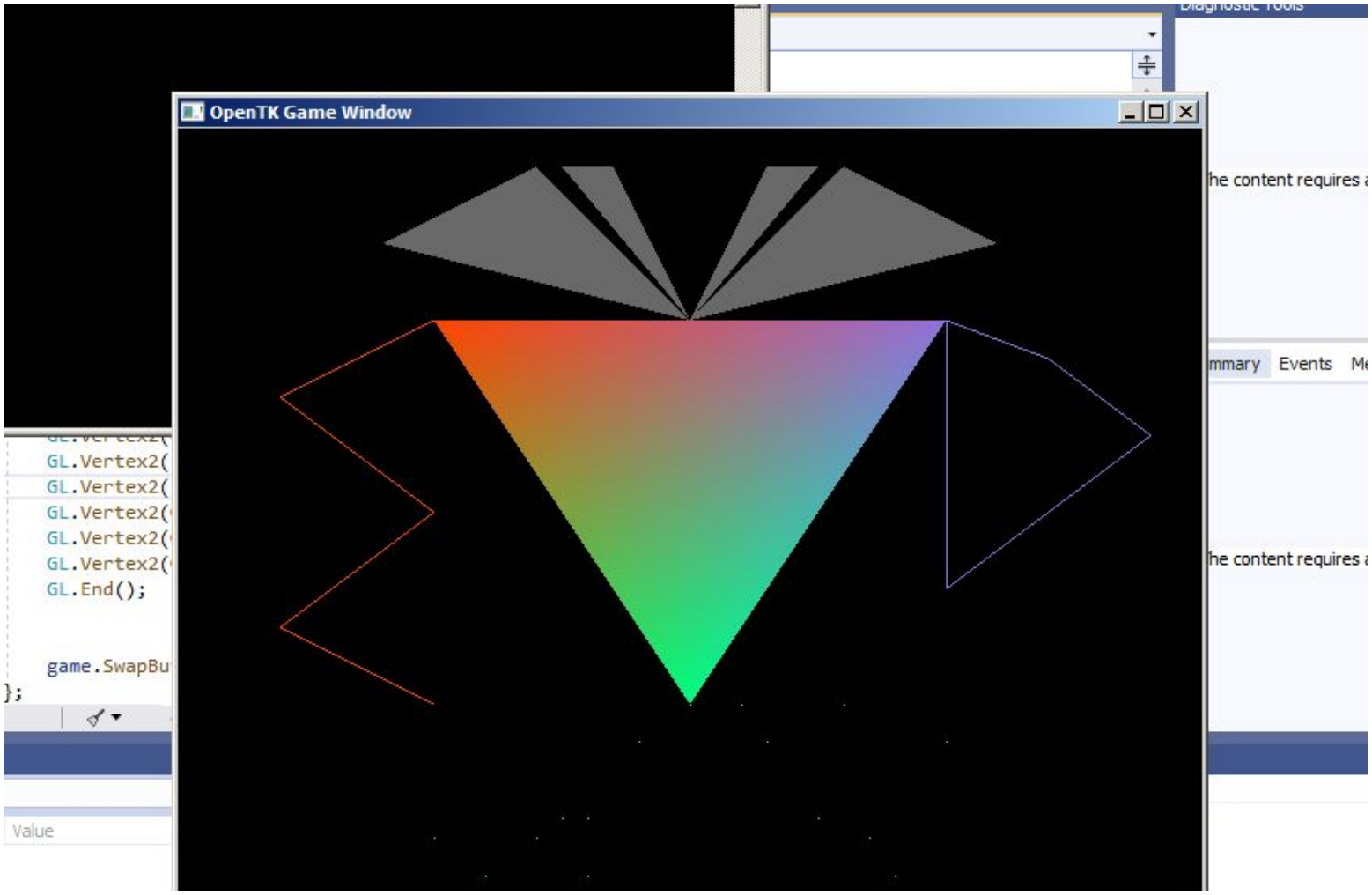
glLoadIdentity() – починає з початку координат

Трикутник

```
……….
GL.Begin(PrimitiveType.Triangles);
 GL.Color3(Color.OrangeRed);
 GL.Vertex2(-0.5f, 0.5f);
 GL.Color3(Color.SpringGreen);
 GL.Vertex2(0.0f, -0.5f);
 GL.Color3(Color.MediumPurple);
  GL.Vertex2(0.5f, 0.5f);
  GL.End();
……..
```


D:\!!!Prog\!OpenGL\111\Cube\bin\Debug\Cube.exe
OpenTK Game Window
// Run the game
game.Run(60.0);
No issues found
he content requires a new version of Internet
mmary
Events
Memory Usage
CPU Usag
Value

Картинка з примітивів



Перетворення

В OpenGL на різних етапах формування зображень об’єктів використовується три типи матриць. Один із них – матриці моделювання, що служать для задання розміщення об’єкта і його орієнтації в просторі, другий – матриці проекцій, що задають спосіб проектування три вимірних об’єктів на площину екрана (OpenGL підтримує 2 види проектування – паралельне і перспективне). Третій тип – матриці текстури, які визначають накладання текстури на об’єкт.

Для того, щоб вибрати поточну матрицю моделювання, проектування чи текстури, використовують процедуру

void glMatrixMode (Glenum mode);.

Параметр *mode* може набувати значення *GL_MODELVIEW*, *GL_PROJECTION* або *GL_TEXTURE*, що дозволяє як поточну матрицю вибрати видову матрицю, матрицю проектування або матрицю перетворення текстури.

$$(x', y', z', 1)^T = M * (x, y, z, 1)^T,$$

де M – матриця перетворення.

Сама матриця M може бути створена за допомогою таких команд:

void glTranslate{f d} (GLtype x, GLtype y, GLtype z);,

яка забезпечує зсув об'єкта на вектор (x, y, z);

void glRotate{f d} (GLtype angle, GLtype x, GLtype y, GLtype z);,

що здійснює поворот об'єкта на кут *angle* в градусах у напрямку проти годинникової стрілки навколо прямої з направляючим вектором *(x, y, z)*;

void glScale{f d} (GLtype x, GLtype y, GLtype z);,

яка виконує масштабування об'єкта (стиснення та розтяг), домножуючи відповідні координати вершин об'єкта на значення своїх параметрів.

Приклад 1

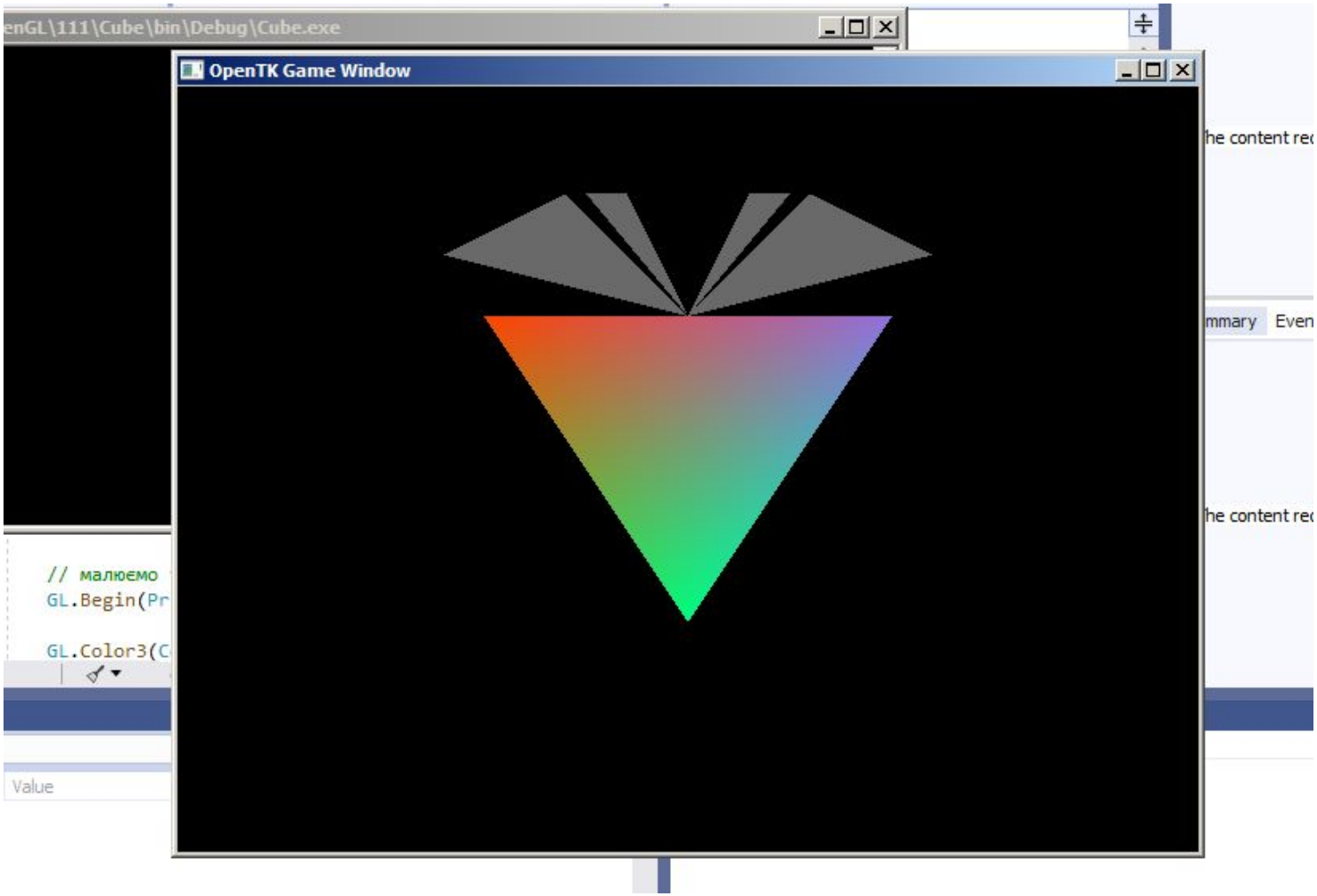
……

….

```
GL.MatrixMode(MatrixMode.Modelview);
            GL.LoadIdentity();
            GL.Translate(0.5,0.1,0.0);
            GL.LoadIdentity();
            GL.Scale(0.8, 0.8, 1.0);
```

……..

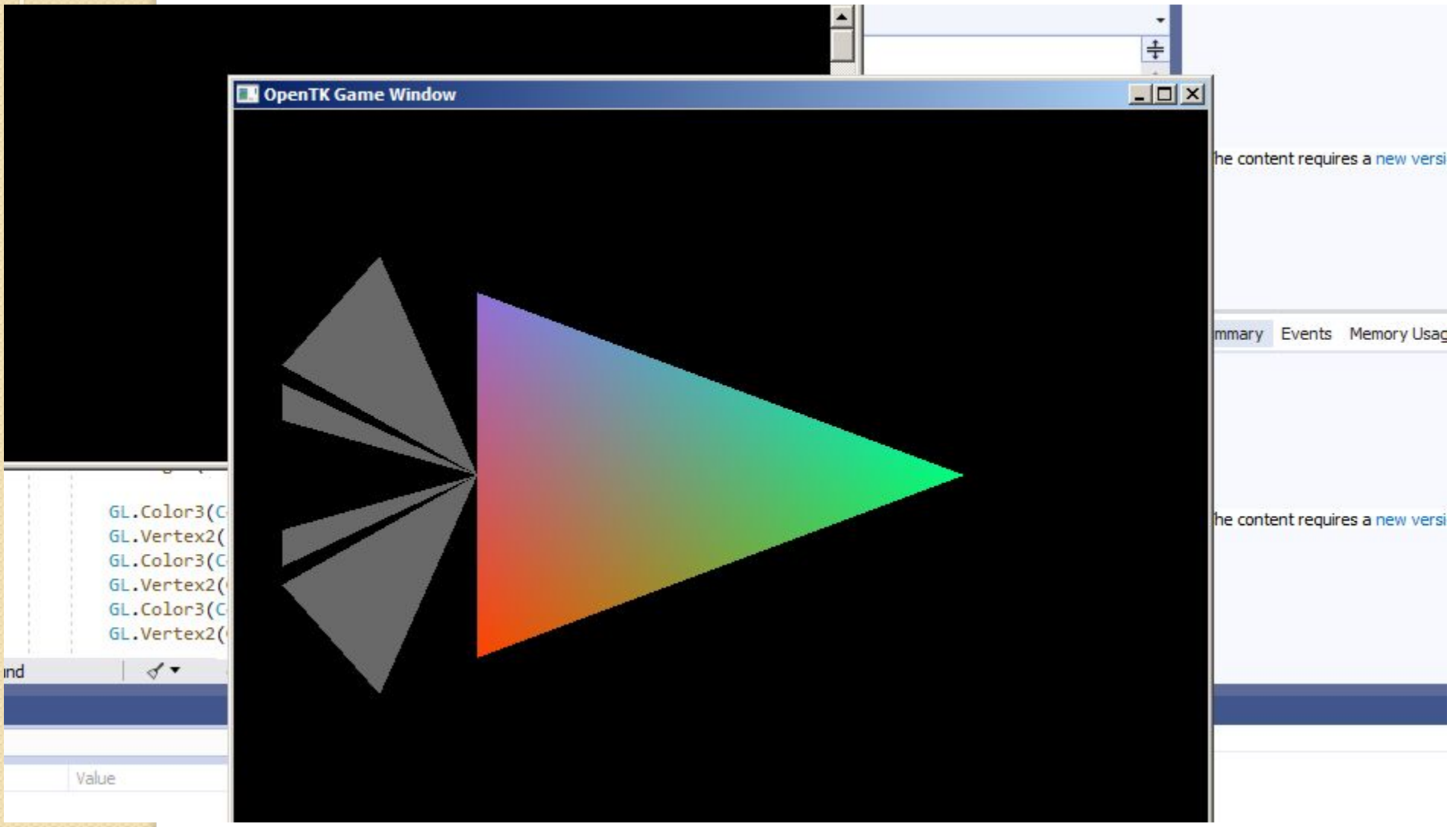
** Реалізовано на VS 2019 (.NET 4.7.2 , OpenTK 3.0.1) для проекту Console App*


enGL\111\Cube\bin\Debug\Cube.exe
OpenTK Game Window
// малюємо
GL.Begin(Pr
GL.Color3(C
Value
he content rec
mmary
Even
he content rec

Приклад 2

```
……….
GL.MatrixMode(MatrixMode.Modelview);
 GL.LoadIdentity();
 GL.Translate(1.0, 1.0, 0.0);
 GL.LoadIdentity();
 GL.Scale(0.8, 0.8, 1.0);
 GL.LoadIdentity();
 GL.Rotate(90.0, 0.0, 0.0, 1.0);
……..
```

** Реалізовано на VS 2019 (.NET 4.7.2 , OpenTK 3.0.1) для проекту Console App*


OpenTK Game Window
GL.Color3(C
GL.Vertex2(
GL.Color3(C
GL.Vertex2(
GL.Color3(C
GL.Vertex2(
Value
he content requires a new versi
mmary
Events
Memory Usag
he content requires a new versi

Приклад 3

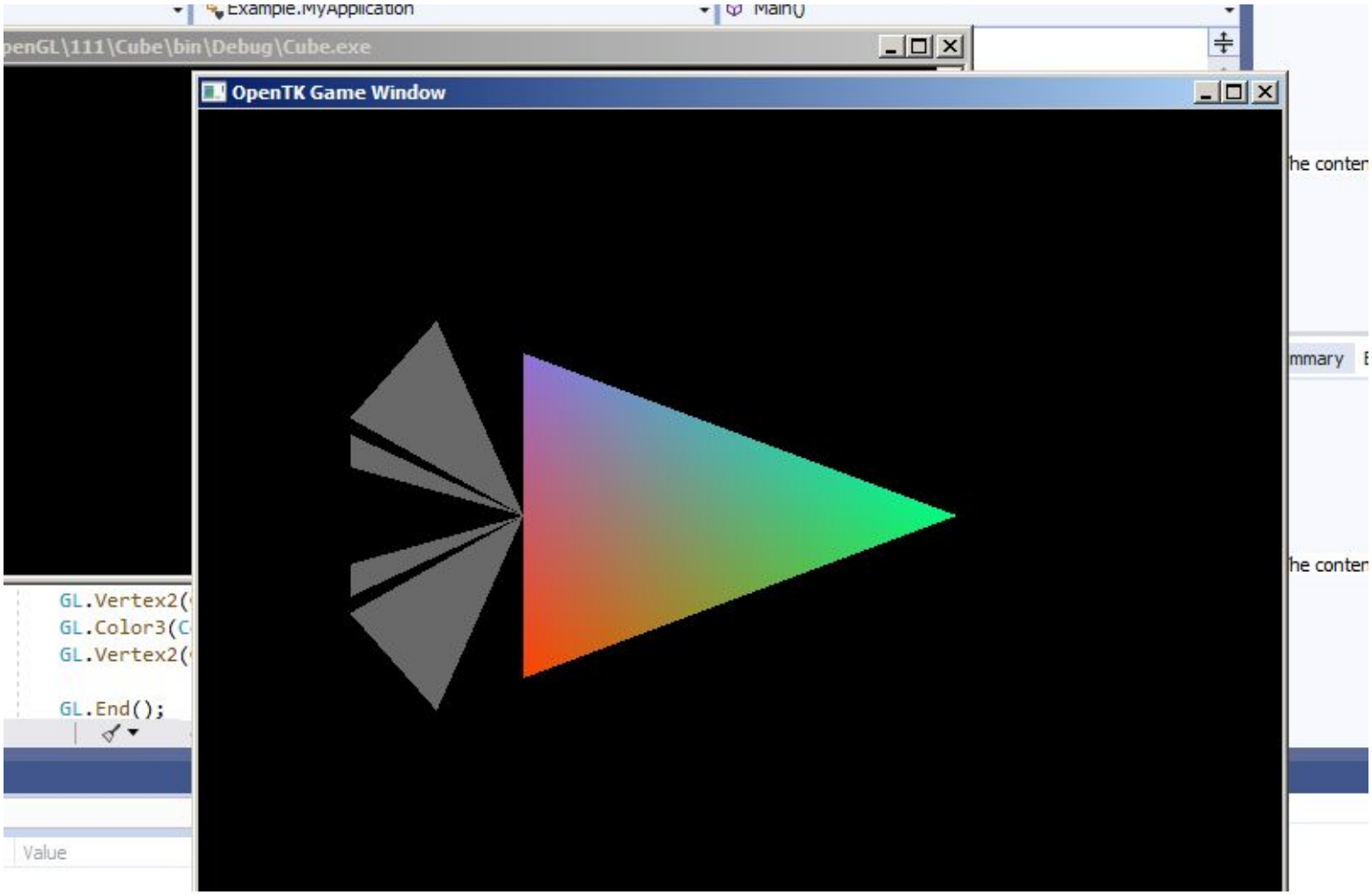
……

……

```
GL.MatrixMode(MatrixMode.Modelview);
        GL.LoadIdentity();
        GL.Translate(1.0, 1.0, 0.0);
        GL.LoadIdentity();
        GL.Scale(0.8, 0.8, 1.0);
        GL.Rotate(90.0,0.0, 0.0, 1.0);
```

……

** Реалізовано на VS 2019 (.NET 4.7.2 , OpenTK 3.0.1) для проекту Console App*


Example.MyApplication
Main()
penGL\111\Cube\bin\Debug\Cube.exe
OpenTK Game Window
GL.Vertex2(
GL.Color3(C
GL.Vertex2(
GL.End();
Value
he conter
mmary
he conter

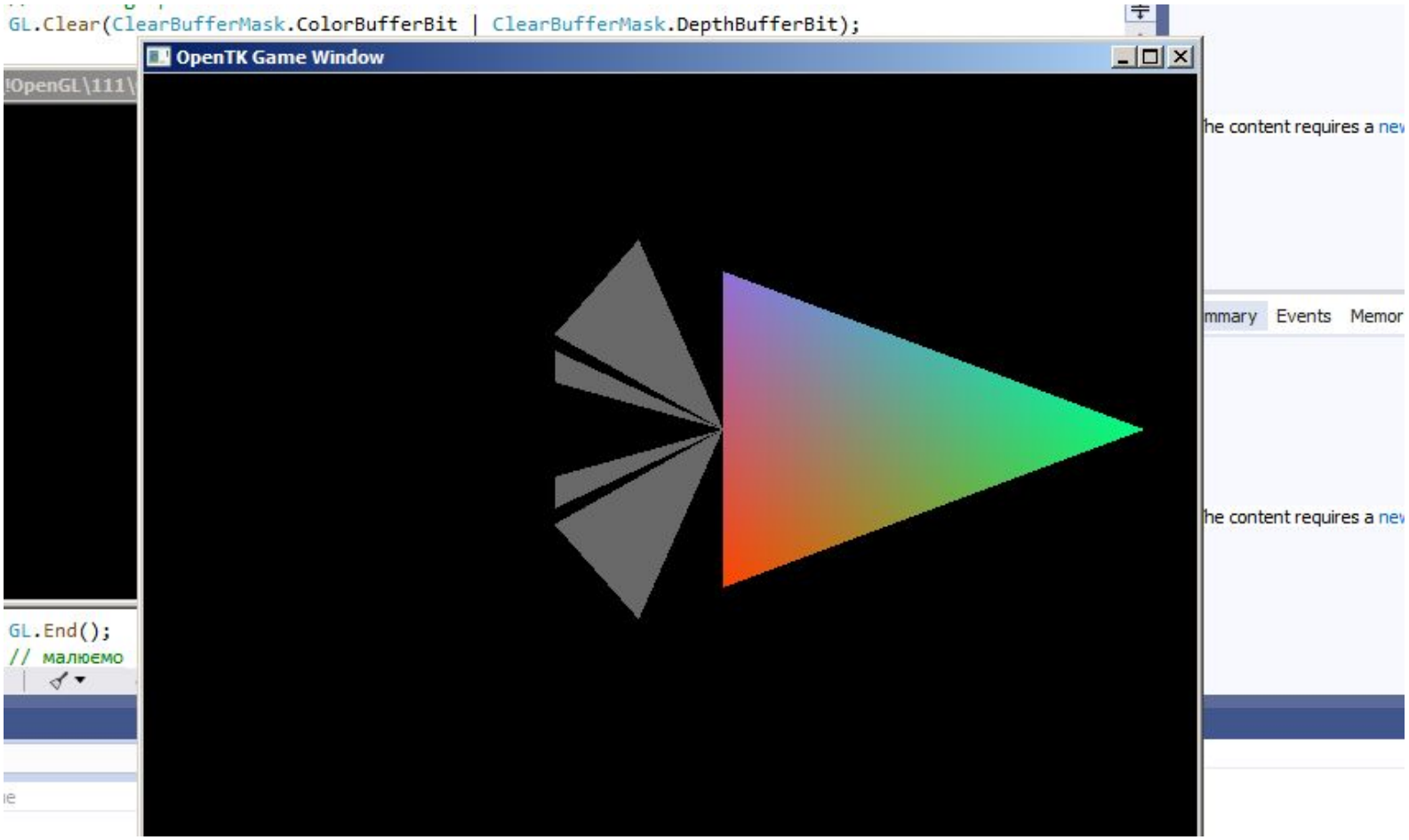
Приклад 4

…….

```
GL.MatrixMode(MatrixMode.Modelview);
        GL.LoadIdentity();
        GL.Translate(0.5,0.1,0.0);
        GL.Scale(0.8, 0.8, 1.0);
        GL.Rotate(90.0,0.0, 0.0, 1.0);
```

……

** Реалізовано на VS 2019 (.NET 4.7.2 , OpenTK 3.0.1) для проекту Console App*


GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);
OpenTK Game Window
!OpenGL\111\
GL.End();
// малюємо
he content requires a new
mmary
Events
Memor
he content requires a new

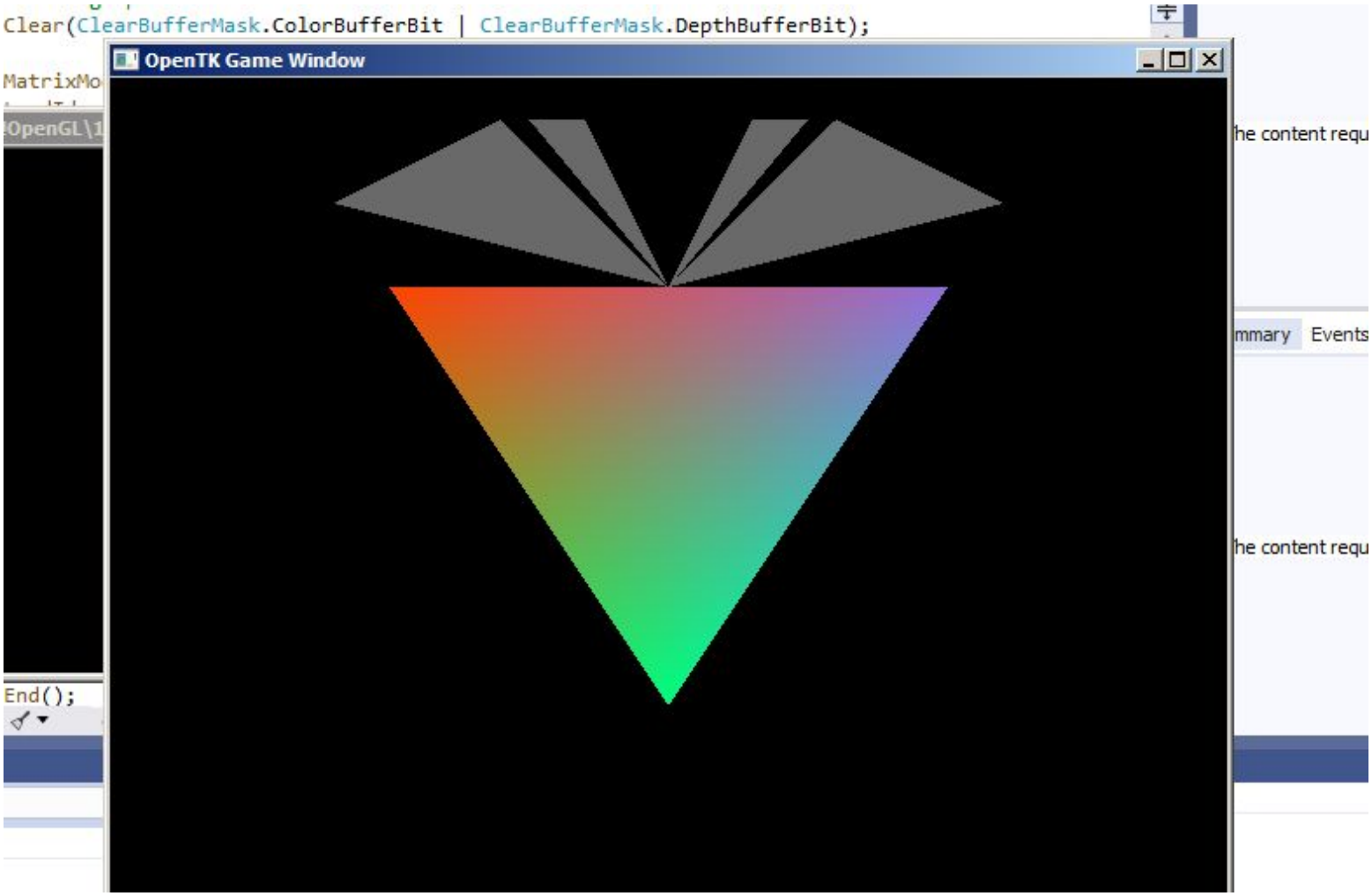
Приклад 5

……

```
GL.MatrixMode(MatrixMode.Modelview);
        GL.LoadIdentity();
        GL.Translate(0.5,0.1,0.0);
        GL.Scale(0.8, 0.8, 1.0);
        GL.Rotate(90.0,0.0, 0.0, 1.0);
        GL.LoadIdentity();
```

……..

** Реалізовано на VS 2019 (.NET 4.7.2 , OpenTK 3.0.1) для проекту Console App*


Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);
MatrixMo
OpenGL\1
End();
OpenTK Game Window
he content requ
mmary
Events
he content requ

Література

1. Маценко В.Г. Комп'ютерна графіка: навчальний посібник.- Чернівці:Рута.- 2009 – 243 с.
2. Eck D. J. Fundamentals of computer graphics with Java, OpenGL and Jogl . – H.and W.Smith colleges, 2010. – pp.131 режим доступу http:// math.hws.edu/graphicsnotes/
3. Fleet D., Hertzmann A. Computer graphics lecture notes. – University of Toronto, 2006. – pp.126.
4. https://habr.com/ru/post/111175/
5. https://www.khronos.org/opengl/wiki/Main_Page
6. http://security-corp.org/programming/29610-vvedenie-v-ispolzovanie-opengl-40-v-net.html