Министерство образования и науки Украины  
Одесский Национальный Университет им. И.И. Мечникова

Институт Математики, Экономики и Механики

**Создание иерархии классов на тему «Геометрические фигуры: многогранники»**

выполнил: Кукишев А.А., студент I курса факультета Компьютерной Инженерии

руководитель:

канд. ф.

Одесса – 2016

Содержание

**1 Постановка задачи**

Необходимо создать классы: «Многогранник» (абстрактный) и его подклассы: «Пирамида» (характеризуется многоугольником в основании и высотой), «Призма» (характеризуется многоугольником в основании и высотой), «Додекаэдр» (характеризуется стороной). Создать абстрактный класс «Многоугольник» с наследниками: «Треугольник», «Прямоугольник», «Правильный многоугольник» (характеризуется количеством сторон и стороной). Объявить методы площадь, периметр, количество вершин для многоугольника и реализовать их в наследниках. Объявить методы площадь поверхности, объем, количество вершин, количество ребер и количество граней для многогранника и реализовать их в конкретном многограннике.

В проекте предусмотреть возможность создания новых многогранников произвольных типов и выдачи полных характеристик многогранников (в табличной форме и подробно для каждого многогранника). Для этого определить методы выдачи характеристик многогранника кратко (в одну строку) и подробно.

**2 Используемые средства**

2.1 Qt — кроссплатформенная библиотека разработки GUI на С++.

Библиотека Qt задумывалась и начиналась как кроссплатформенный тулкит (toolkit) для быстрой разработки графических интерфейсов (GUI) приложений на языке C++, с целью упростить жизнь программистов, пишущих на C++ кроссплатформенные, переносимые GUI-приложения, которые должны работать и в среде Windows, Symbian, и в среде Unix/Linux под X11, и на компьютерах Macintosh. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

В настоящее время Qt значительно переросла рамки тулкита для разработки графических интерфейсов приложений. Она предоставляет использующему её программисту целостный фреймворк (framework), позволяющий при написании большей части приложения использовать только «родные» классы Qt и практически полностью отказаться от написания системно-зависимого кода. В библиотеке предусмотрены классы и для работы со строками, и для работы с файлами, сетью, базами данных, XML, и для обеспечения многопоточности в приложении, и многое другое.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q\_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Qt комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы в режиме WYSIWYG. В поставке Qt есть «Qt Linguist» — графическая утилита, позволяющая упростить локализацию и перевод программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кроссплатформенную справку для разрабатываемого на основе Qt ПО. Начиная с версии 4.5.0 в комплект Qt включена среда разработки «Qt Creator», которая включает в себя редактор кода, справку, графические средства «Qt Designer» и возможность отладки приложений. «Qt Creator» может использовать GCC или Microsoft VC++ в качестве компилятора и GDB в качестве отладчика. Для Windows версий библиотека комплектуется компилятором, заголовочными и объектными файлами MinGW.

2.2 OpenGL Mathematics

GLM (OpenGL Mathematics — математика для OpenGL) — библиотека для OpenGL, предоставляющая программисту на C++ структуры и функции, позволяющие использовать данные для OpenGL.

Одна из особенностей GLM состоит в том, что его реализация основана на спецификации GLSL (OpenGL Shading Language).

**3 Программная реализация**

Реализован абстрактный класс CPolygon, определяющий абстрактный многоугольник.

Класс CPolygon имеет следующие поля:

std::vector<glm::vec2> mAngles;

Последовательный контейнер типа vec2, который представляет собой массив точек(углов) двухмерного пространства.

std::string mName;

Имя фигуры.

Основные публичные методы класса CPolygon:

virtual QJsonObject toJSON() const = 0;

Абстрактный метод, преобразовывающий объекты потомков в json-представление, которое используется для записи в файл формата json.

virtual const string type() const;

Виртуальный метод, возвращающий тип фигуры.

virtual glm::vec3 toVec3(Uint index) const;

Виртуальный метод, преобразовывающий двухмерную точку в трехмерную. index – индекс точки из mAngles;

virtual vec2& operator[] (Uint index);

Перегруженный оператор квадратный скобок для получения доступа к точкам фигуры из mAngles.

virtual const vec2& operator[](Uint index) const;

Перегруженный оператор квадратный скобок для получения доступа к точкам константной фигуры mAngles.

virtual double area() const = 0;

Абстрактный метод площади фигуры.

virtual double perimeter () const = 0;

Абстрактный метод периметра фигуры.

Реализован класс CRectangle, , который является наследником CPolygon, определяющий прямоугольник.

Основные методы класса:

double getWidth() const;

double getLength() const;

vec2 getDownLeftAngle() const;

Методы возвращения длины ширины, высоты и координаты левого нижнего угла.

CRectangle& setWidth(double width);

CRectangle& setLength( double length);

CRectangle& setDownLeftAngle(vec2 point);

Методы для установки значений ширине, высоте и нижнему левому углу прямоугольника.

static std::shared\_ptr<CRectangle> load(const QJsonObject& object);

Статический метод, возвращающий умный указатель, который создается по переданному параметру типа QJsonObject.

CRectangle имеет макрос DeclareFigure.

Реализован класс CTriangle, который является наследником CPolygon, определяющий треугольник. Класс имеет только переопределенные методы своего родителя.

CTriangle имеет макрос DeclareFigure.

static std::shared\_ptr<CTriangle> load(const QJsonObject& object);

Статический метод, возвращающий умный указатель, который создается по переданному параметру типа QJsonObject.

Реализован абстрактный класс CFigure, определяющий трехмерную фигуру.

Основные поля класса:

std::string mNameFigure;

Имя фигуры.

glm::mat4 mMat;

Матрица, которую использует метод getCalculatedVertex(Uint index).

std::shared\_ptr<CPolygon> mBasis;

Умный указатель на класс CPolygon, который используется в качестве основы трехмерной фигуры.

Основные публичные методы:

virtual double surfaceArea() const = 0;

virtual double volume() const = 0;

Методы, которые возвращают поверхностную площадь и объём фигуры.

virtual void setMatrix(const glm::mat4& mat);

virtual glm::mat4 getMatrix() const;

Геттер и сеттер для mMat.

virtual glm::vec3 operator[](Uint index) const;

Перегруженный оператор квадратный скобок, который возвращает точку по указанному индексу.

virtual std::shared\_ptr<CFigure> operator()(Uint index) const;

Перегруженный оператор круглых скобок, который возвращает указатель на класс CFigure. Переопределен в наследнике CComplexFigure.

virtual QJsonObject toJSON() const = 0;

Абстрактный метод, возвращающий преобразовывает объекты наследников класса CFigure в QJsonObject.

virtual const std::string type() const;

Виртуальный метод, возвращающий тип фигуры.

virtual std::string getName() const;

virtual void setName(const std::string& name);

Геттер и сеттер для mNameFigure.

virtual Uint countVertex() const= 0;

virtual Uint countEdge() const = 0;

virtual Uint countFacets() const = 0;

Абстрактные методы, которые возвращают количество вершин, ребер и граней фигуры.

virtual Uint countFigures() const ;

Метод, возвращающий количество фигур. Переопределен в наследнике CComplexFigure.

std::shared\_ptr<CPolygon> getBasis() const;

Геттер для mBasis.