МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-47 01 02 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированное программирование»

Тема «Программное средство трехмерный графический редактор на WPF»

Исполнитель

студент 2 курса группы 3 Почебут А.В.

Руководитель

Ст. преподаватель Наркевич А.С.

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Наркевич А.С.

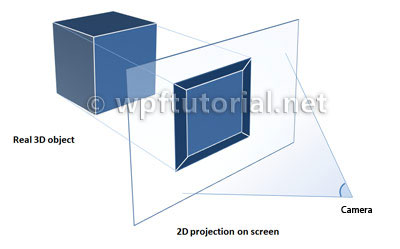
Минск 2018

# **Введение**

Работать с графикой в принципе очень интересно, но работать с трехмерной - особенно интересно. И WPF предлагает нам удобный инструментарий для этого. Конечно, создавать суперсложные трехмерные сцены или игры на WPF - очень непростой процесс, и лучше для этого выбрать DirectX, OpenGl, либо специально заточенные под это фреймворки и движки на подобие Unity, Monogame и т.д. Однако для относительно несложных в том числе бизнес-приложений трехмерные возможности WPF вполне сойдут.

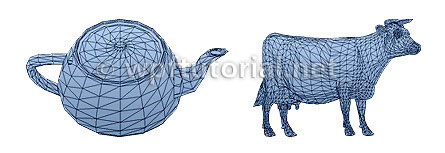
Основная идея создания 3D-графики - иметь трехмерную модель объекта. Поскольку наш экран только двухмерен, мы определяем камеру, которая делает снимок объекта. Изображение представляет собой проекцию объекта на плоскую поверхность. Эта проекция преобразуется в растровое изображение с помощью 3D-рендеринга. Двигатель определяет цвет для каждого пикселя, вычисляя количество света, отражаемого любыми источниками света на поверхности проекции объектами в трехмерном пространстве.

На всех поверхностях объектов есть материал и кисть. Материал определяет, сколько света отражается для определенного угла, а кисть определяет цвет. Кисть может быть либо простым цветом, либо градиентом, либо даже изображением, называемым текстурой.

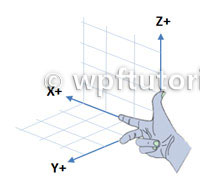


В мире 3D-графики все объекты описываются набором треугольников. Но почему треугольники? Причиной этого является то, что треугольник является самой гранулированной геометрией для описания плоской поверхности. Механизм рендеринга может вычислять цвет каждого треугольника в зависимости от его материала и угла к источнику света в сцене. Если бы мы попытались построить наш мир на прямоугольниках, то это была бы одна из худших идей для рендеринга. Поверхность будет намного сложнее вычислять и визуализировать.

Поверхность 3D-объекта называется сеткой. Полигоны определяются рядом трехмерных точек. Эти точки называются вершинами. Вершины соединены вместе рисунком обмотки для определения треугольников. Каждый треугольник имеет переднюю и заднюю стороны. Отображается только лицевая сторона. Передняя сторона определяется порядком обмотки точек. WPF использует шаблон намотки против часовой стрелки.



WPF использует стандартную систему координат. Это означает, что если вы держите пальцы правой руки, как показано на рисунке, пальцы указывают положительное направление осей дерева.



1. Аналитика пояснение
   1. Постановка задачи

Редактор будет реализовывать следующие функции:

* Возможность создания учетной записи пользователя.
* Возможность создания двумерных фигур в трехмерном пространстве.
* Возможность построения собственных 3D фигур.

Приложение будет являться десктопным и не будет иметь онлайн сервисов, реализация на WPF 4.6, что позволит создать приятный и не требовательный к ресурсам PC интерфейс.

В программном средстве будут применена база данных, которая будет хранить в себе файлы моделей. База данных будет реализована при помощи MS SQL.

Обзор аналогов.

Рассмотрим стандартный редактор Paint 3D, интерфейс которого представлена на рисунке 1.

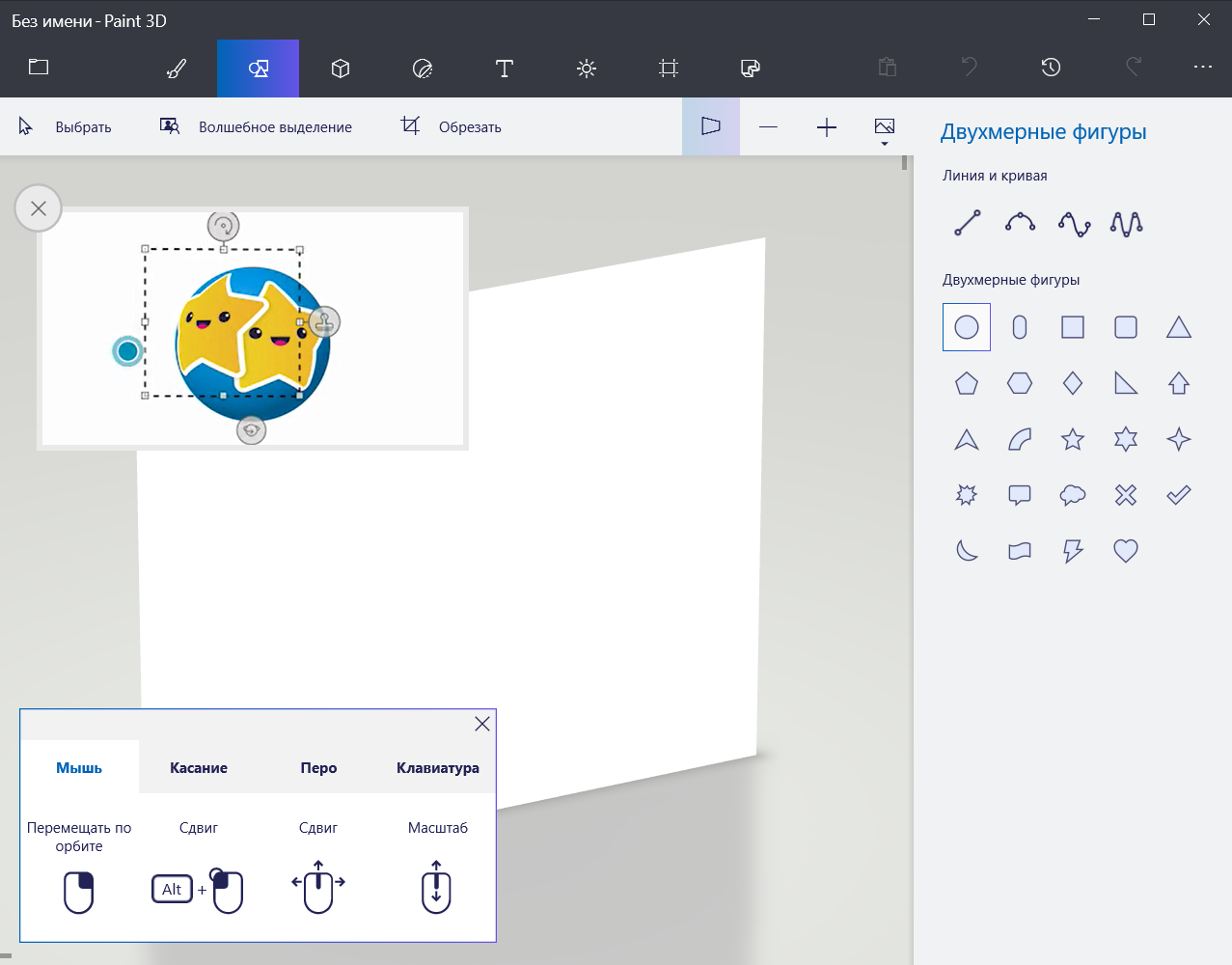


Рисунок 1.

Является представителем облегченного редактора 3D графики. Интерфейс представляет из себя одно окно, в котором отображается модель и доступны некоторые функции. Является очень простым, что крайне важно для таких массовых приложений, но и предоставляет весь необходимый начальный функционал.

Также его популярность обусловлена и тем, что он является стандартным приложением.

Выделим основные плюсы:

* Очень прост в освоении;
* Легкий и приятный интерфейс;

Минусы:

* Интерфейс практически нельзя изменить;
* Имеет только базовые функции.

Blender

Рассмотрим интерфейс Blender на рисунке 2.

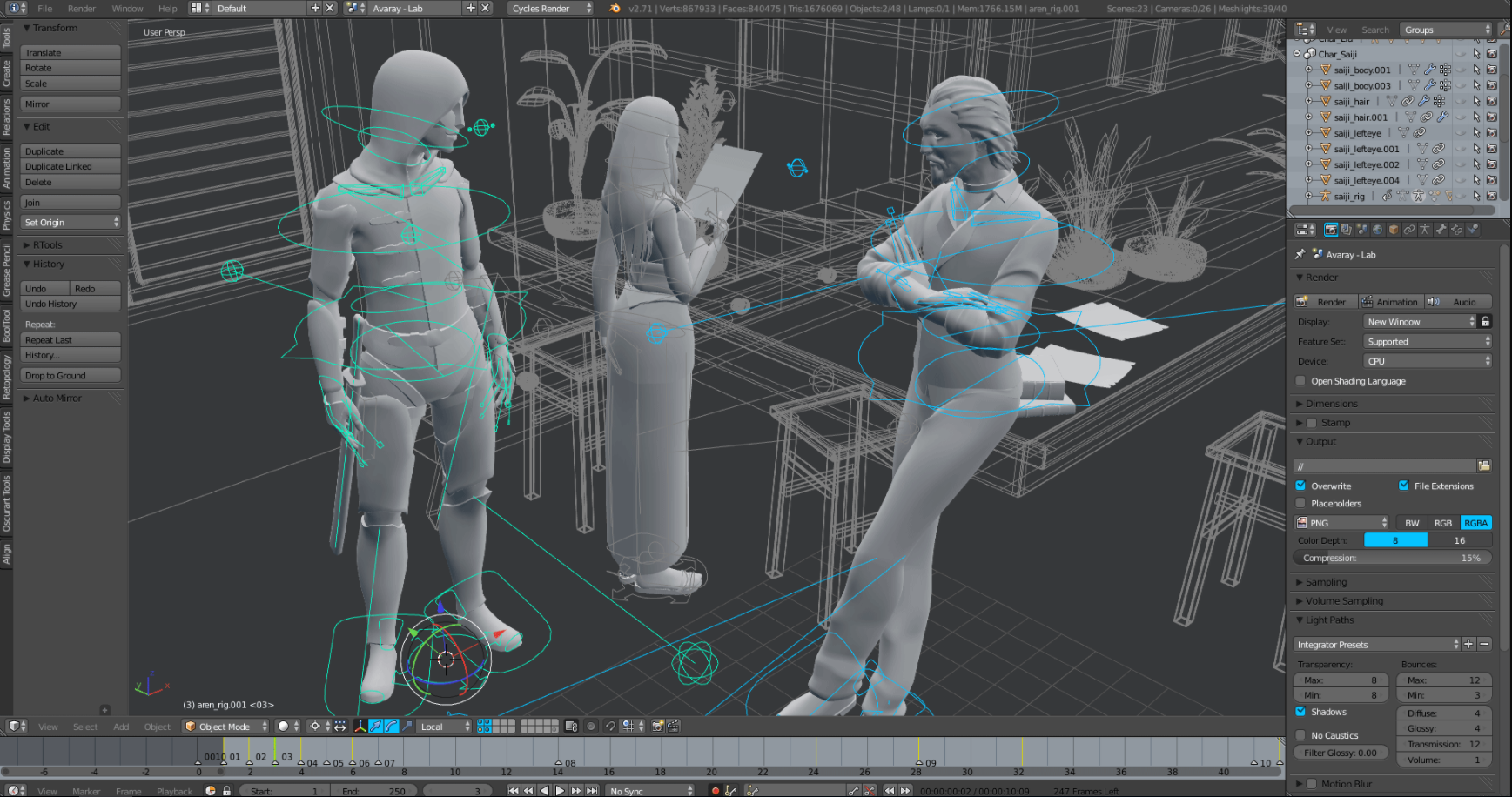


Рисунок 2.

Blender очень популярен благодаря своей бесплатности и представляет собой совершенно другой класс графических редакторов, а именно, умеющий работать с очень тяжелыми сценами. Имеет широкие функциональные возможности, настраиваемый интерфейс.

Плюсы:

* Имеет расширяемый клиент;
* Можно устанавливать пользовательские модификации;
* Очень много инструментов для работы с файлами;

Минусы:

* Интерфейс изначально кажется громоздким;

# **Начало проекта и анализ требований.**

## **Описание предметной области.**

Объектом разработки является WPF приложения «Программное средство трехмерный графический редактор на WPF» - приложение, специализирующееся на предоставлении услуг по конструированию и разработке моделей в трехмерном пространстве.

## **1.2 Цели и задачи проекта.**

Необходимо сделать приложение, в котором любой пользователь мог бы создавать свои сцены, управлять светом и камерой с возможностью их последующего редактирования.

### **1.2.1 Анализ бизнес-целей проекта.**

В результате проведенного исследования были выявлены следующие цели проекта:

* Облегчить создание несложных 3D моделей;
* создать легкое к пониманию 3D приложение.

## **1.3 Целевая аудитория.**

Для приложения «Интернет-аукцион автомобилей» можно выделить такие группы пользователей:

* пользователи, желающие быстро освоить 3D моделирование;
* пользователи, желающие создавать трехмерные модели;
* случайные пользователи.

## **1.4 Список потребностей пользователя.**

Список потребностей пользователей приложения:

* получить навыки в трехмерном дизайне;
* научиться понимать работу 3D приложений;

## **1.5 Список возможностей на проекте.**

Все пользователи смогут в приложении могут:

* ознакомиться со списком созданных фигур;
* отредактировать созданные фигуры;
* удалить фигуры;
* создавать фигуры;
* управлять всеми доступными в WPF источниками света;
* управлять камерой;
* зарегистрироваться;
* авторизоваться.

# **Проектирование**

## **2.1 Выбор используемых технологий**

Данное программное обеспечение разрабатывалось для работы в операционной системе Windows 10, Windows 7 .

Разработка проекта производиться на языке программирования С#, в среде разработки Microsoft Visual Studio 2017.

При разработке будут использованы следующие технологии:

* WPF – создание интерфейса.
* MS SQL Server – хранение данных.
* Xceed.WPF – расширенный инструмент разработки.

WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

MS SQL – [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) СУБД.

Xceed.WPF – на данный момент, инструментария WPF в некоторых задачах может не хватать, из-за чего необходимо использовать дополнительные NuGet пакеты.

## **2.2 Выбор системной архитектуры**

Мною была выбрана патернная архитектура Model**-View-ViewMode (MVVM)** (рис. 3).

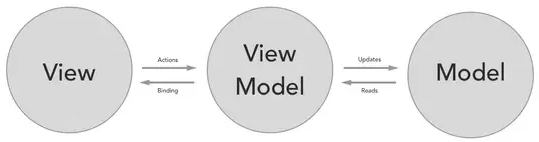


Рисунок 3 – шаблон MVVM.

Шаблон MVVM делится на три части:

* Модель (Model), так же, как в классической MVC, Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.
* Представление (View) — это графический интерфейс, то есть окно, кнопки и т. п. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
* Модель Представления (ViewModel) является, с одной стороны, абстракцией Представления, а с другой, предоставляет обёртку данных из Модели, которые подлежат связыванию. То есть, она содержит Модель, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

## **2.3 Структура проекта**

Структура проекта – “скелет” любой программы. Он состоит из различных элементов. Это могут быть ресурсы, статические классы, формы и прочее.

### **2.3.1 Структура проекта**

|  |  |
| --- | --- |
| Общая структура проекта |  |
| EDM модель базы данных |  |
| Папка «Figures2D» |  |
| Папка «CamerasFolder» |  |
| Папка «CustomClasses» |  |
| Папка «LightingFolder» |  |
| Папка «Resorces» |  |
| Папка«Updaters» |  |

### **2.3.2 Папка «Resourses» (ресурсы приложения)**

В качестве ресурсов приложения были выбраны следующие элементы:

* GridSplitter.xalm (содержит стиль ползунка)

### **2.3.3 Папка «Updaters»**

Файлы, содержащиеся в данной папке предназначены для дополнения/изменения уже существующих фигур.

### **2.3.4 Папка «LightingFolder»**

Файлы, содержащиеся в данной папке предназначены для изменения освещения на сцене.

### **2.3.5 Папка «CustomClasses»**

Файлы, содержащиеся в данной папке являются шаблонами для создания 3D моделей.

### **2.3.6 Папка «CamerasFolder»**

Файлы, содержащиеся в данной папке предназначены для управления камерой.

### **2.3.2 Папка «Figures2D»**

Файлы, содержащиеся в данной папке являются в своем роде View частью архитектурного паттерна.

## **2.4 Разработка графического интерфейса**

Интерфейс данного программного продукта должен отображать меню навигации программы поверху содержащее кнопки: «3D Мастер», «2D Фигуры», «Освещение», «Войти», «Проекция».

|  |  |
| --- | --- |
| Страница «3D Редактор» |  |
| Вкладка «Меню создания пользовательской 3D фигуры» |  |
| Вкладка  «Меню создания четырехугольника» |  |
| Страница «Авторизация» |  |
| Страница «Регистрация» |  |

# **3.Руководство пользователя**

# **3.1 Пользовательские фигуры.**

Самым первым делом необходимо запомнить, как работает трехмерный движок WPF. Для того, чтобы построить простейшую фигуру, необходимо иметь как минимум три точки для объединения их в единый треугольник. Но это еще не самое трудное, сложнее всего удерживать у себя в голове именно правило постройки фигуры. Оно заключается в том, что если вы хотите построить простейший трехугольник одновременно с тем, чтобы он смотрел на вас своей лицевой стороной, то вы должны вообразить некую локальную Z ось, проходящую ваш треугольник насквозь в направлении к вам и относительно этой оси необходимо перечислить точки в обратном порядке хода часовой стрелки.

На первый раз звучит ужасно и непонятно, но если немного попрактиковаться, то все очень быстро начнет получаться. Пример с обозначениями последовательности точек показан на рисунке 4.

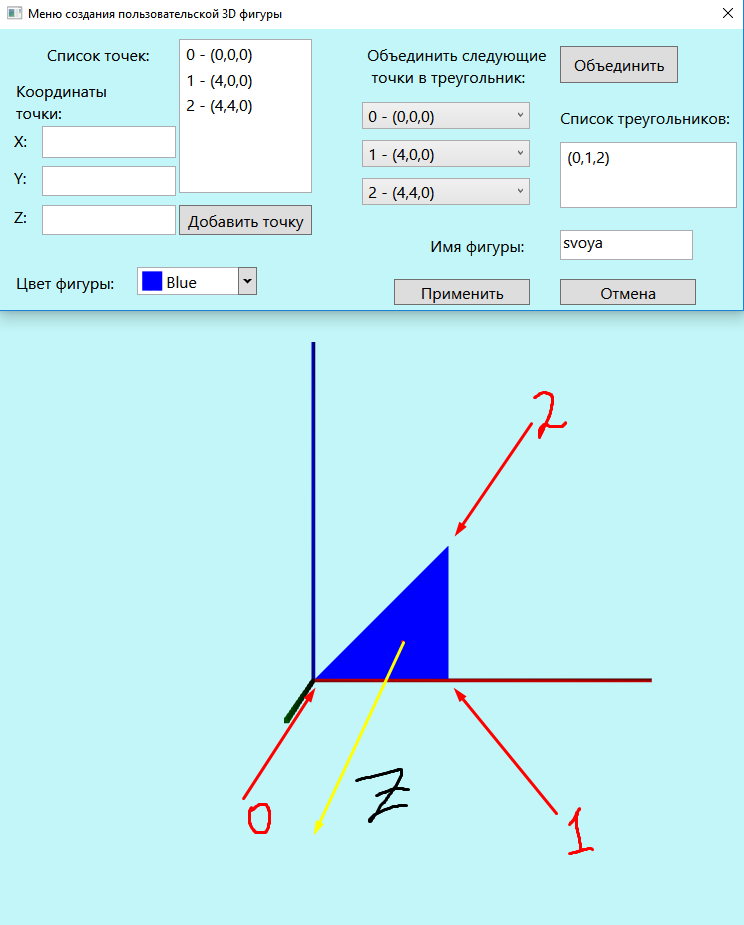


Рисунок 4.

# **3.3 Шаблонные 2D фигуры.**

Обыкновенная оперция создания любого четырехугольника на основании 3 точек (рис. 5)

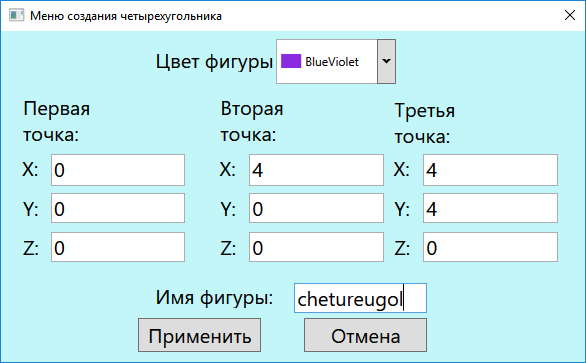


Рисунок 5 – Создание четырехугольника

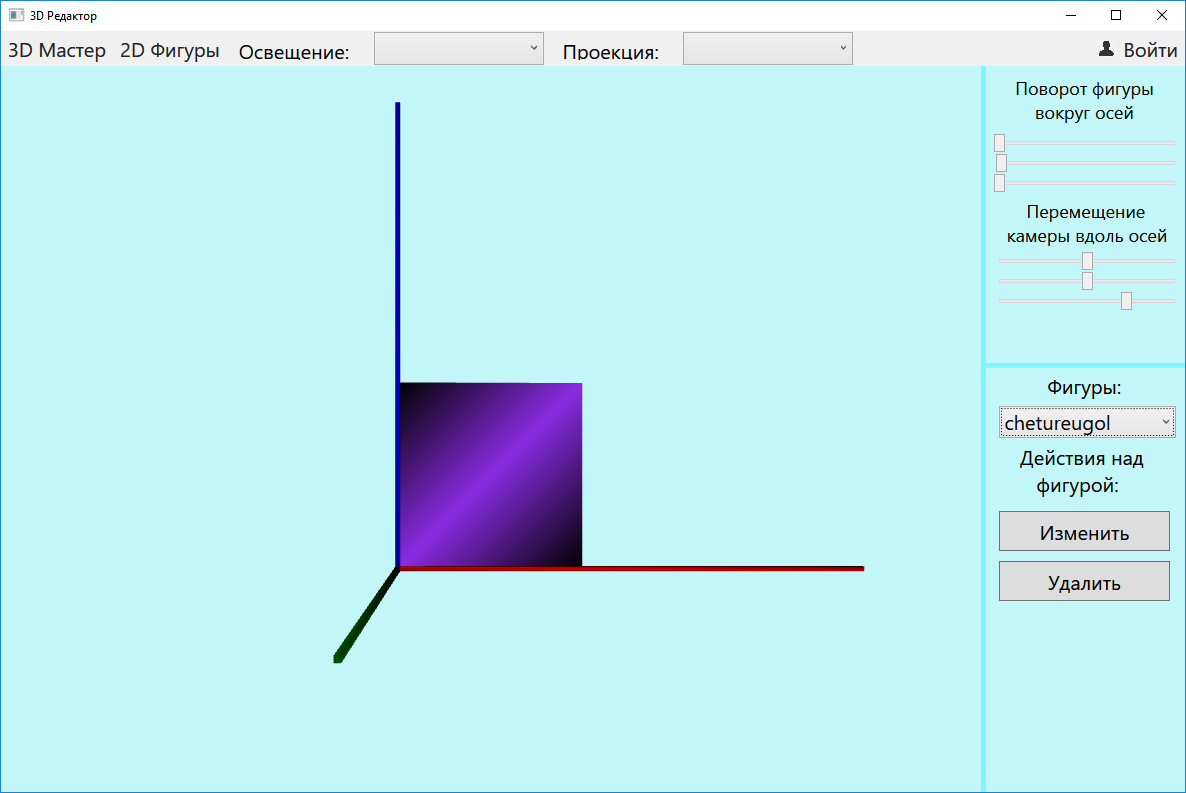


Рисунок 6 – Результат создания четырехугольника

Аналогично строится и треугольник.

Построение круга более увлекательное занятие, чем простых многоульников. Для того, чтобы построить под таким наклоном, который вам необходим, нужно обязательно указать вектор нормали к кругу. Сам угол строится исходя из скалярного перемножения двух векторов. Вектор камеры постоянен и он равен (0, 20, 0). Пожтому, если вы хотите построить круг, который будет изначально смотреть на вас необходимо указать положительное значение в координате Z. Так как это сделано на рисунках 7а, 7б.

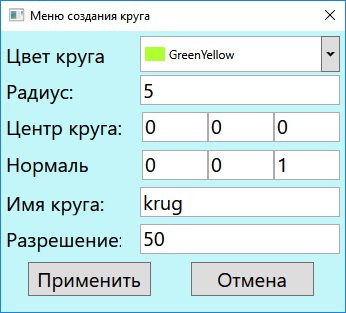


Рисунок 7а- Создание фронтального направленног круга.

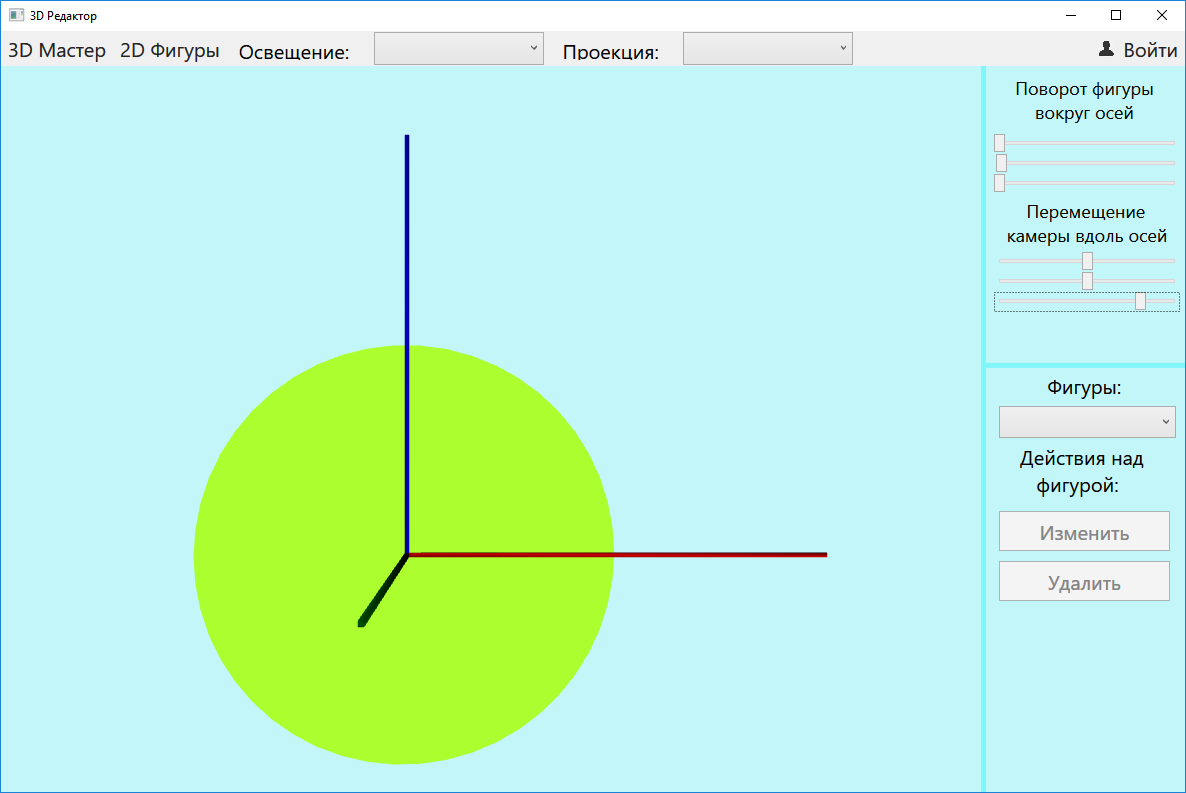


Рисунок 7б – Пример направленного фронтальной стороной к вам круга.

# **3.4 Освещение.**

Всего WPF предоставляет 4 стандартных источника света:

* Направленный
* Рассеяный
* Точечный
* “Камерный”

# Направленный свет представляет собой параллельные лучи света, направленные из одной точки в другую. Начальная точка в направленном освещении всегда распологается в центре координат (0, 0, 0). Соответственно если взять вторую точку (0, 0, -20), то тогда все фигуры, расположенные по оси Z до -20 буду освещены данным источником света (рисунки 8а, 8б).

# 

Рисунок 8а.

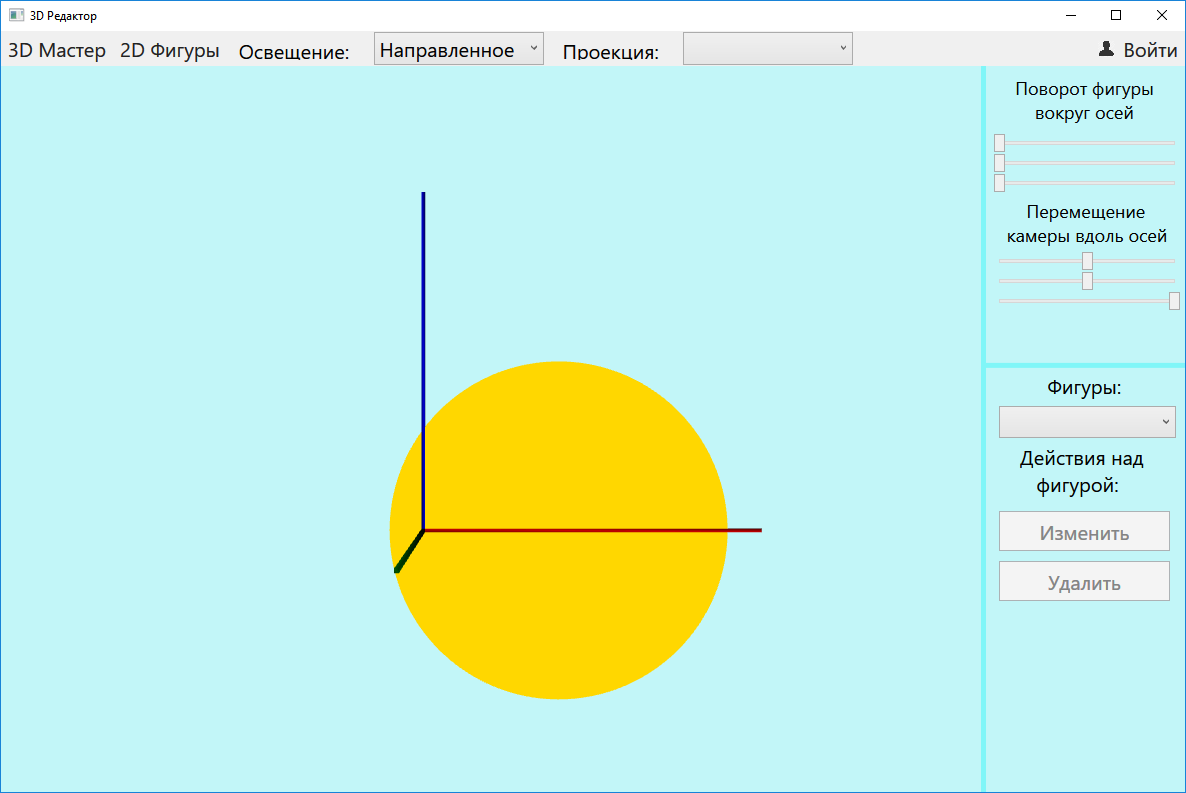


Рисунок 8б.

При рассеяном свете тени не образуются и все фигуры имеют свой чистый свет (рисунки 9а, 9б).

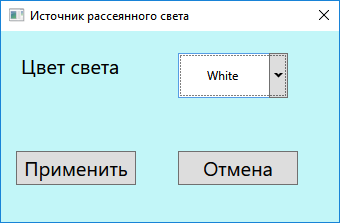


Рисунок 9а.

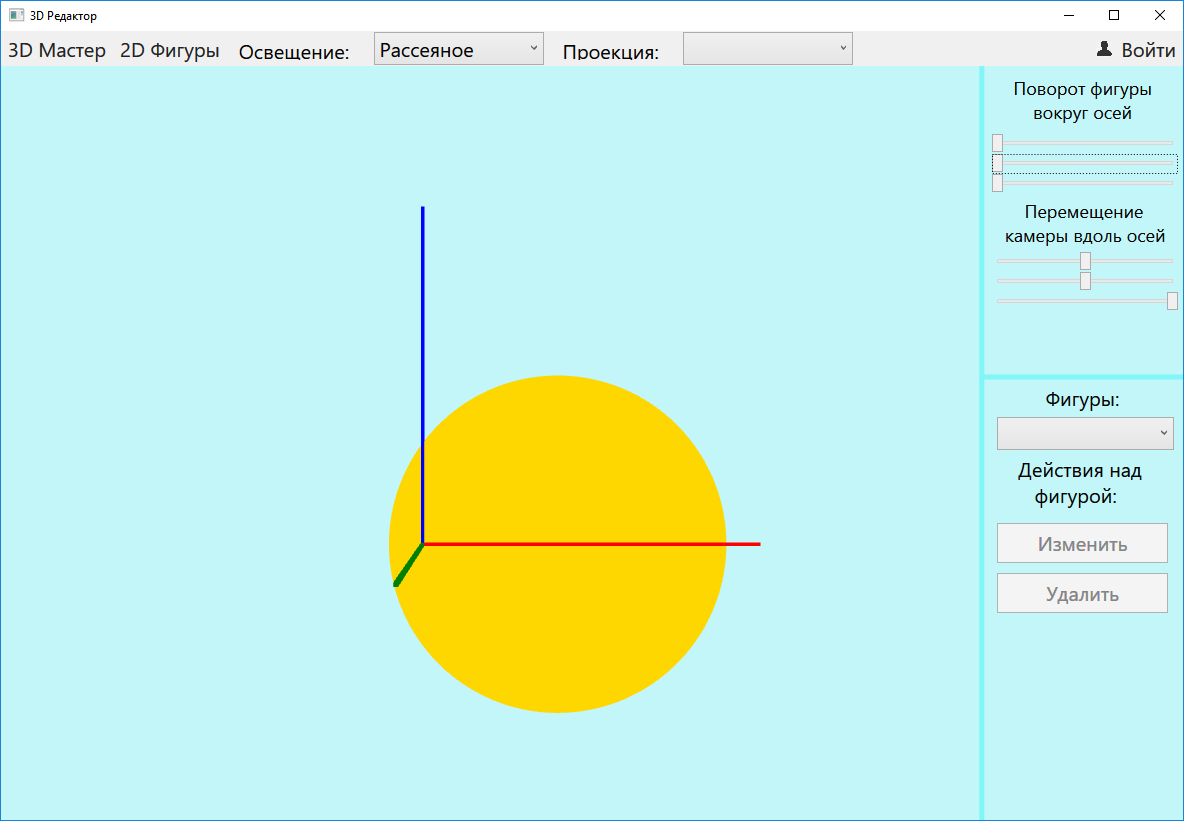


Рисунок 9б.

Точечный источник света создает вокруг себя сферу из постепенно угасающего света, который распостраняется на заданный радиус (рисунки 10а, 10б).

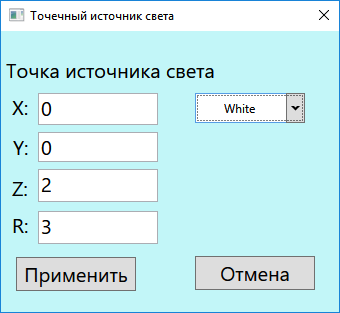


Рисунок 10а.

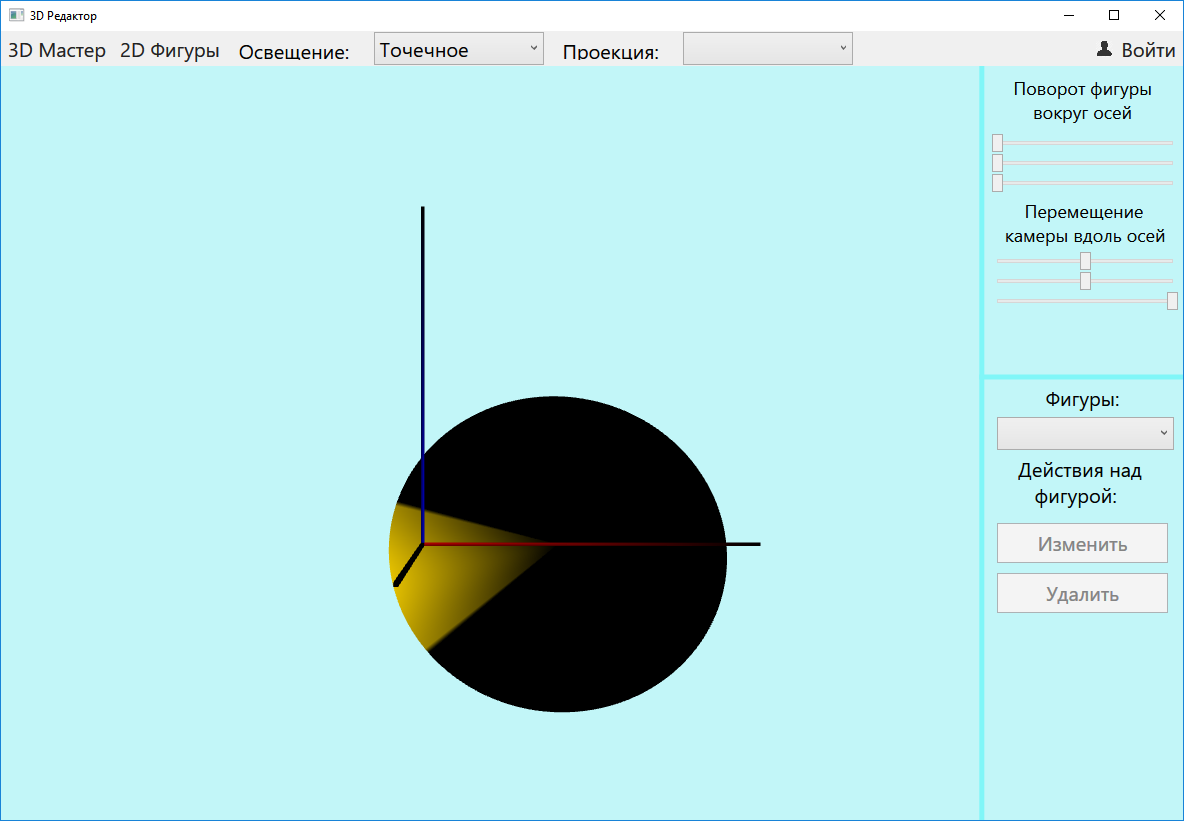
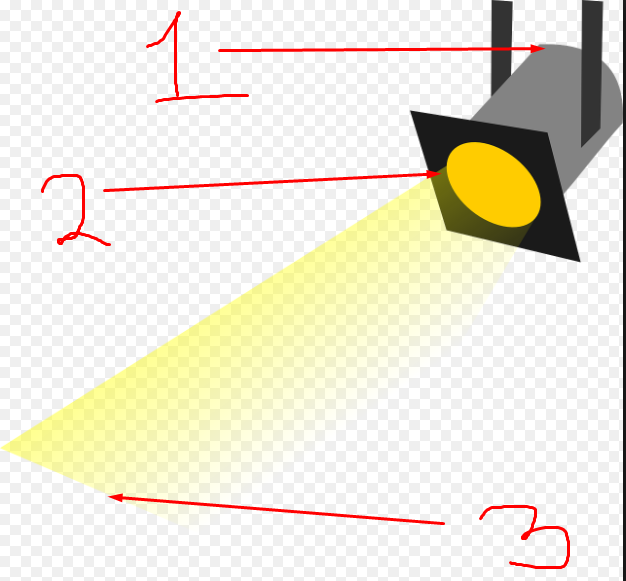


Рисунок 10б.

“Камерное” освещение наиболее понятно можно рассталковать на данной картинке.



В роле 1 выступает основание света и его внешний угол распостранения от самого основания. Под 2 нужно понимать внутренний угол, который будет ограничивать внешний. А под 3 будет находиться основание, до которого свет имеет возможность добраться (рисунки 11а, 11б). При этом всем нужно понимать, что делая угол внутреннего угла больше чем внешнего, никаких результатов вы не достигните.

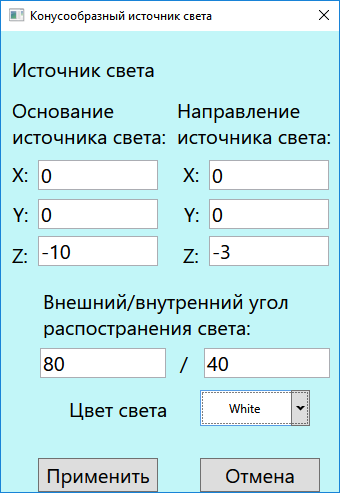


Рисунок 11а.

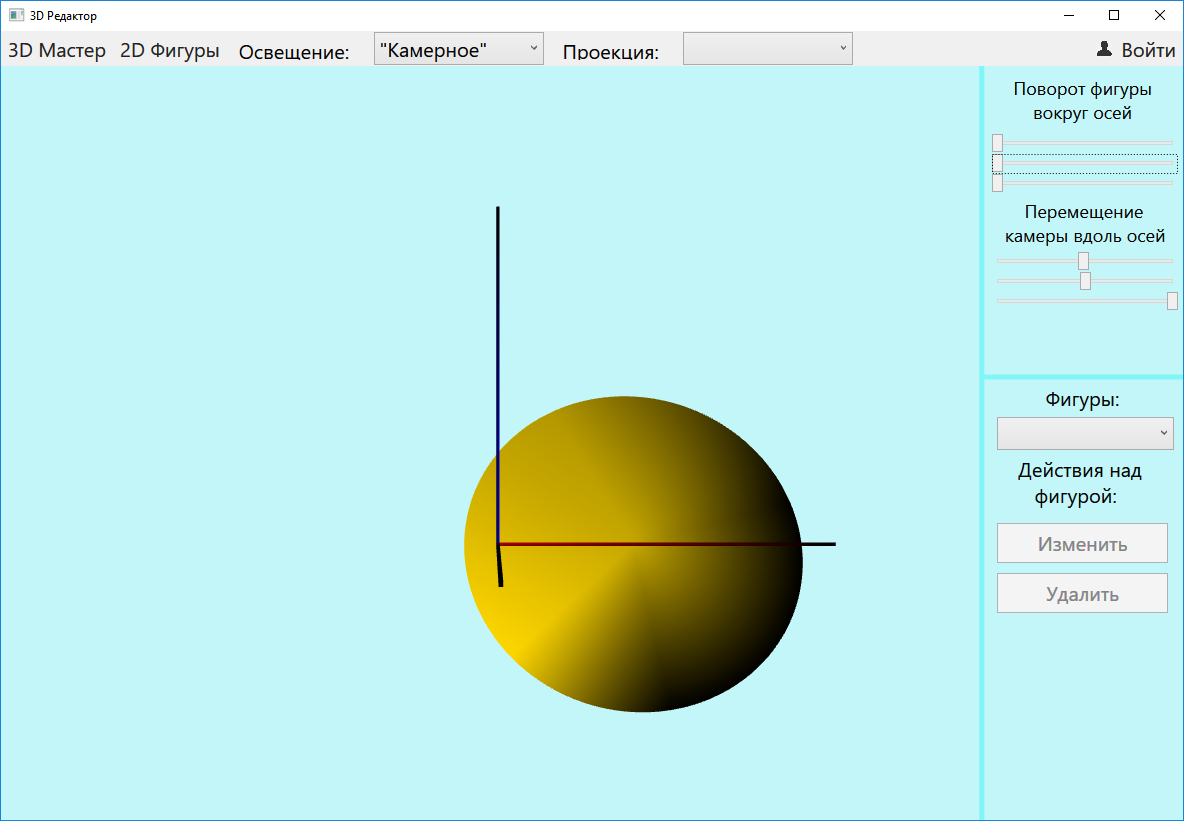


Рисунок 11б.

# **3.5 Проекции камеры.**

В программе реализовано два вида камер:

1. Камера в перспективной проекции.
2. Камера в ортогональной проекции.

В чем же их разница? Разница у них заключается в том, что, смотря через перспективу мы видим предметы, так как видели бы их вживую, а наблюдая через ортогогональную проекцию, дистанция, от камеры до объекта, не играет роли и все предметы выглядели бы соответствующе своим реальным размерам.

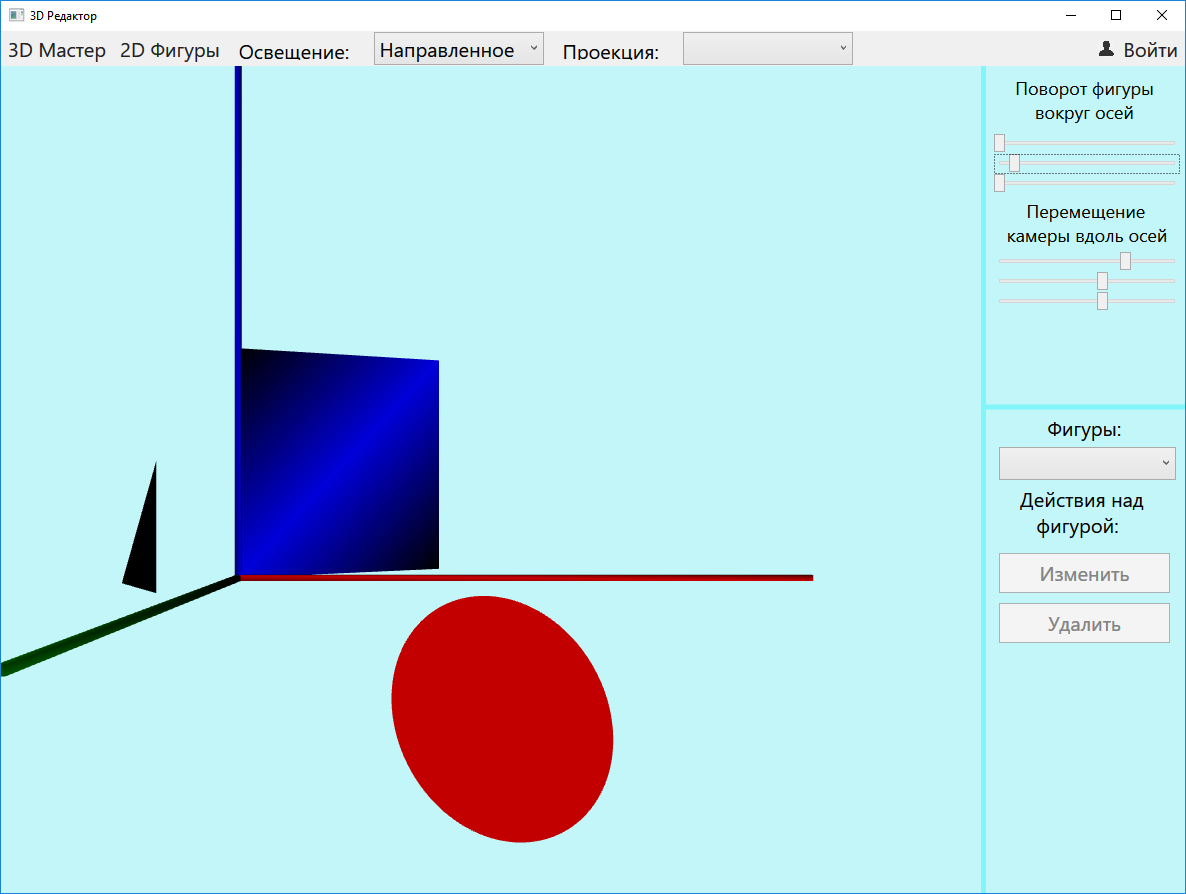


Рисунок 12- Перспектива

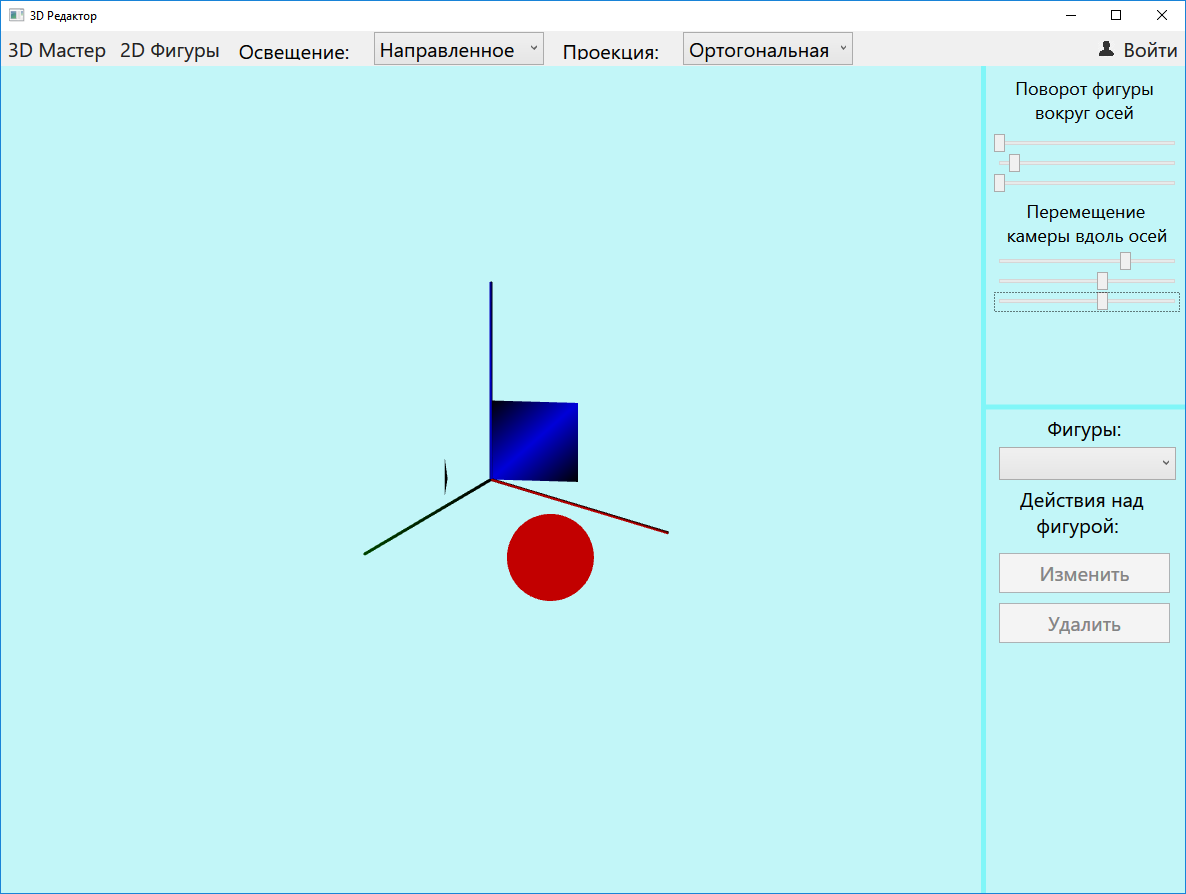


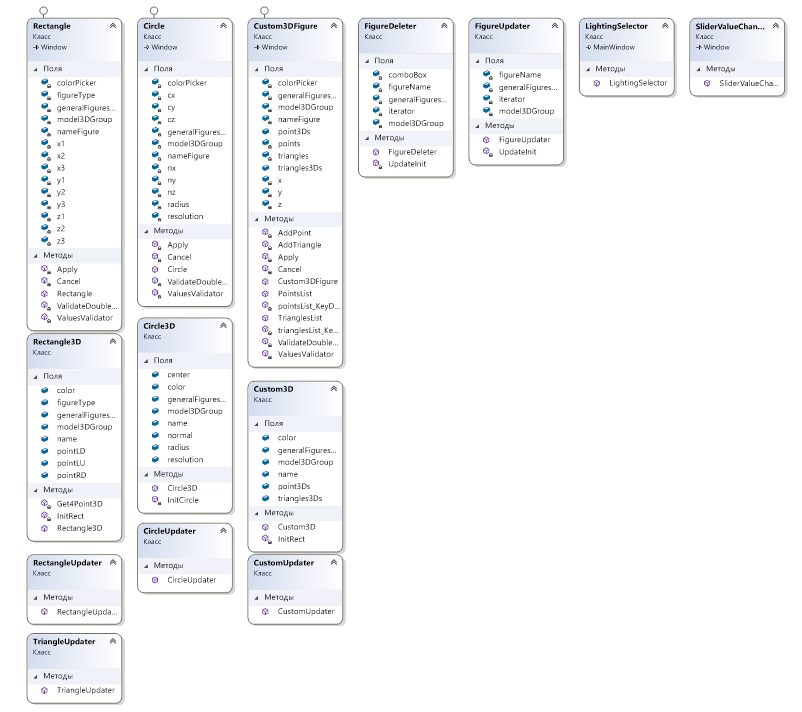
Рисунок 13 - Ортогональность

# **4. Тестирование**

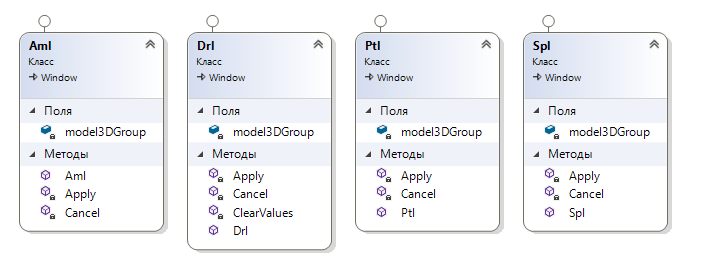
Тестирование данного продукта выявило несколько незначительных недочетов, но, в целом, отзывы о данном приложении были оставлены положительные.

# **5. Скелет проекта**

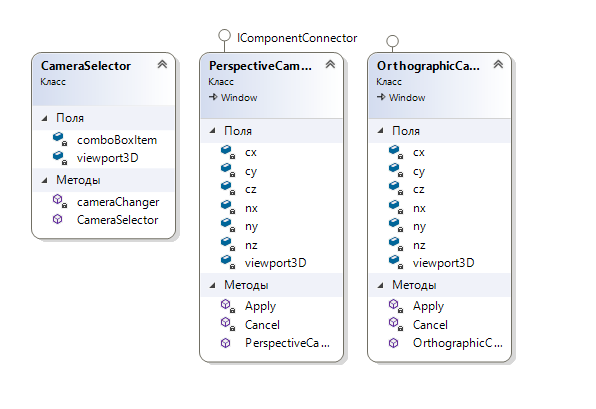
# **5.1 Диаграмма класов, ответственных за фигуры на сцене**



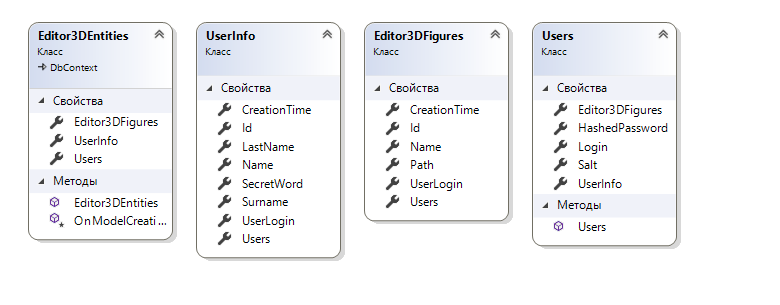
# **5.2 Диаграмма класов, ответственных за освещение на сцене**



# **5.3 Диаграмма класов, ответственных за камеру**



# **5.3 Диаграмма класов, ответственных за БД**



# **Заключение**

Данный проект был разработан для многочисленной аудитории: учителя, студенты, ученики, родители и т.д. Главной задачей было сделать простое и удобное приложение. Для этого был использован API-интерфейс WPF и MS SQL Server. Чтобы сделать программу легко изменяемой и доступной, был подключен паттерн MVVM. Также был проработан ряд исключительных ситуаций с выводом сообщений, в случае некорректной работы пользователя.

Само приложение – первый опыт в данной области. Естественно, существует ряд приложений, имеющих функционал и дизайн в разы лучше. Подводя итог всей курсовой работе, необходимо сделать некоторые выводы.

Использование API-интерфейса WPF значительно упростила разработку приложения. Данный интерфейс был выбран неслучайно: само приложение предназначено для демонстрации работы с графическими моделями. Этот факт предполагает использование привлекательного графического интерфейса, для создания которого на программной платформе .NET идеально подошел WPF.

# **Список использованных источников**

1. <https://msdn.microsoft.com>

2. <https://stackoverflow.com>

3. <https://metanit.com>