**Министерство образования и науки РФ**

**Федеральное государственное автономное образовательное учреждение**

**Высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ**

Специальность: 09.03.03. – Прикладная информатика

ДИПЛОМНАЯ РАБОТА

**РАЗРАБОТКА КОМПОНЕНТА ПОЛУАВТОМАТИЧЕСКОГО КОНСТРУИРОВАНИЯ ТРЕХМЕРНЫХ СЦЕН ПО АРХИВНЫМ ФОТОГРАФИЯМ.**

**.**

Студент 4 курса

Группа 11-301

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_ (В.Е.Сафин)

**Научный руководитель**

(преподаватель/куратор лаборатории)

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_ (В.В.Кугуракова)

Оглавление

[ВВЕДЕНИЕ 3](#_Toc483812960)

[1. ПОСТАНОВКА ЗАДАЧИ 5](#_Toc483812961)

[1.1 Технология разработки полуавтоматического компонента 5](#_Toc483812962)

[1.2 Техническое задание 6](#_Toc483812963)

[1.2.1 Общие сведения 6](#_Toc483812964)

[1.2.2 Назначение и цели создания 6](#_Toc483812965)

[1.2.3 Требование к функциям компонента 6](#_Toc483812966)

[1.2.4 Требование к программному обеспечению 6](#_Toc483812967)

[1.2.5 Планирование работы по созданию компонента 7](#_Toc483812968)

[2. CAMERA MAPPING, PROJECTION CAMERA 8](#_Toc483812969)

[2.1 Технология 8](#_Toc483812970)

[2.2 Подготовка фотографии 10](#_Toc483812971)

[2.3 Настройка сцены 12](#_Toc483812972)

[2.4 Настройка статичной камеры 13](#_Toc483812973)

[2.4.1 Линии горизонта, перспективные линии 13](#_Toc483812974)

[2.4.2 Фокусное расстояние и углы обзора 14](#_Toc483812975)

[2.5 Моделирование 17](#_Toc483812976)

[2.6 Проецирование 18](#_Toc483812977)

[3. ПРОБЛЕМЫ 20](#_Toc483812978)

[3.1 Недостаток данных 20](#_Toc483812979)

[3.2 Неверно указано фокусное расстояние 20](#_Toc483812980)

[3.3 Не определены опорные точки 20](#_Toc483812981)

[3.4 Дефекты на самих изображениях 21](#_Toc483812982)

[4. РЕАЛИЗАЦИЯ АВТОМАТИЗАЦИИ 22](#_Toc483812983)

[4.1 Разработка функций нахождения фокусного расстояния 22](#_Toc483812984)

[4.2 Разработка интерфейса в Maya (UI) 28](#_Toc483812985)

[ЗАКЛЮЧЕНИЕ 30](#_Toc483812986)

[СПИСОК ЛИТЕРАТУРЫ 31](#_Toc483812987)

[ПРИЛОЖЕНИЕ 32](#_Toc483812988)

# **ВВЕДЕНИЕ**

Современный мир насквозь пропитан информационными технологиями. Развитие растет каждый день, создаются не только инновационные методы решения задач, но и совершенствуются уже существующие.

Люди по своей натуре ленивы и, в связи с этим качеством, стремятся всеми силами облегчить свою работу. Разрабатываются все больше программ, рассчитанных на автоматизацию процессов, в которых человек принимает роль наблюдающего и следит за выполнением работы. Люди все свое время пытаются упростить использование того или иного продукта, сделать его проще, удобнее и быстрее в эксплуатации, путем написания расширений, плагинов или компонентов для них.

Данный вид программирования получил свое названия – “скрипты”. Скрипт - это программа или программный файл сценарий, которые автоматизируют некоторую задачу, которую пользователь делал бы вручную, используя интерфейс программы. [1]

Это тенденция облегчать свой труд, за счет технологий, проникла и в компьютерную графику. Все больше и больше рождается компонентов, дополнений к программам, виджетов для быстрой и удобной работы. Потому что, используя такие компоненты, человек выигрывает во времени выполнения поставленной задачи.

Благодаря компьютерной графике, мы можем искажать реальность, добавлять новое, что человеческий глаз никогда бы не смог увидеть в реальном мире и многое другое. Это занимает много усилий и трудов, не говоря об затраченном времени. Стремление облегчить небольшую части работы в сфере цифровой графики является актуальным характером в наши дни.

Разработка идеи написания скрипта не так проста, как это может выглядеть на первый взгляд. Он должен быть актуальным и действительно решать проблему использующего его. Так или иначе если скрипт не будет должным образом выполнять заданную ей роль, он потеряет свою значимость. Реализация полуавтоматического конструирования трехмерных сцен по архивным фотографиям предназначено для решения проблемы с настройкой камеры для имитации двухмерной фотографии в трехмерном пространстве.

Цели дипломной работы:

1. Изучение материала по проецированию изображения в трехмерное пространство.
2. Процесс нахождения фокусного расстояния на двухмерной фотографии.
3. Формирование интерфейса.
4. Реализация скрипта с включением анимации движущихся элементов.

# **ПОСТАНОВКА ЗАДАЧИ**

## **Технология разработки полуавтоматического компонента**

Была поставлена задача о разработке компонента для автоматизации процесса конструирования трехмерной сцены по двухмерному изображению. В частности, написания расширения в уже существующей программе по работе с трехмерными сценами, для настройки камеры под индивидуальные изображения.

Возможность использовать метод проецирования из камеры имеется у всех трехмерных программ, в которых присутствуют функции создания и настройки виртуальной камеры, и так же присутствует один из методов наложения текстур – “perspective projection”. Рассмотрев все возможные программы по отдельности, было решено использовать продукт – Autodesk Maya, так как в ней удобен интерфейс пользования и понятна внутренняя архитектура. Сама же программа реализованы на C++ с включением поддержки Python, что необходимо для решения поставленных задач.

В выбранном продукте стандартным языком программирования является C++, на котором реализована основная библиотека всех доступных функций - Maya API. Данная оболочка под API имеет свое название MEL (Maya Embedded Language) и с ее помощью возможно обращение к всем функциям Maya. Так же присутствует и поддержка от API: Python, с чем мы и будем работать.

Трехмерный пакет Autodesk Maya на API: Python будет вполне достаточно для решения поставленной задачи.

## **Техническое задание**

### **Общие сведения**

Полуавтоматический компонент для конструирования трехмерных сцен с включением анимации движущихся элементов по архивным фотографиям.

### **Назначение и цели создания**

Компонент предназначен для упрощения нахождения линии горизонта, перспективы и фокусного расстояния у фотографии для повышения точности имитации сцены в трехмерном пространстве.

Компонент создается с целью:

1. Предоставление информации о фотографии;
2. Обеспечении быстрой настройки камеры в виртуальной сцене;
3. Скорость создании трехмерных сцен по заданной фотографии;

### **Требование к функциям компонента**

В компоненте выделены следующие методы:

1. Нахождение фокусного расстояния по средствам вводе плоскостей объектов;
2. Установление перспективных линий для обнаружения точки соприкосновения.

### **Требование к программному обеспечению**

Реализация компонента должна осуществляется в трехмерном пакете – Autodesk Maya, с помощью Script Editor’а на языке Python версии 3.x.

### **Планирование работы по созданию компонента**

1. Сбор информации и материалов
2. Реализация функции установления перспективных линий
3. Реализация компонента установки перспективных линий в Maya
4. Осуществление настройки камеры
5. Разработка интерфейса компонента
6. Тестирование
7. Завершение

# **CAMERA MAPPING, PROJECTION CAMERA**

## **Технология**

Camera Mapping, Projection Camera -  это процесс, который позволяет имитировать фотографию в трехмерную сцену. С ее помощью мы можем получить динамическую сцену (анимацию) из двухмерного изображения, переводя все имеющиеся элементы, изображенные на фотографии, в трехмерное пространство с точным соответствием их положения в своей системе.

Так же используют этот метод и для оптимизации сцен при дальнейшем рендером. Объекты, конструированные по фотографии, имеют представление очень упрощенных форм или простых геометрический фигур. Дело в том, что, используя данный метод, нет необходимости в сильной детализации объектов, так как сама проецируемая фотография выступает в качестве текстуры. Такая текстура передает ту же детализацию, которая и присутствует на ней. И получается, что просчет рендера почти не зависит от количества полигонов, а от качества проецируемого изображения.

Работу такого метода можно представить, как проецирование фильма из кинопроектора на объёмную белую фигуру, а не на плоский экран. [2]

Для чего используется данная технология:

1. Создание реалистичного окружения,
2. Оптимизация рендера,
3. Анимирование статичных кадров, путем анимирования виртуальной камеры или добавления своих элементов с анимацией.

Преимущества у данной технологии:

1. Упрощение процесса текстурирования,
2. Упрощение процесса моделирования

Что нужно знать, чтобы реализовать данную технологию?

Во-первых, это понятие “линия горизонта” – граница неба с земной или водной поверхностью. [3] Не во всех фотографиях она имеет явный вид, но она играет важную роль в создании трехмерной сцены.

Во-вторых, “перспективные линии” - линии, которые указывают в сторону горизонта под разными углами. Эти линии нужны для определения перспективы. Перспектива - техника изображения пространственных объектов на какой-либо поверхности в соответствии с теми кажущимися сокращениями их размеров, изменениями очертаний формы и светотеневых отношений, которые наблюдаются в натуре. [4]

В-третьих, “исчезающая точка” - Точка, в которой угловые линии перспективы объекта визуально продолжить мимо ее края и в конечном итоге сходятся на линии горизонта.

На Рисунке 1 продемонстрировано все необходимые данные для начала работы с данной технологией.

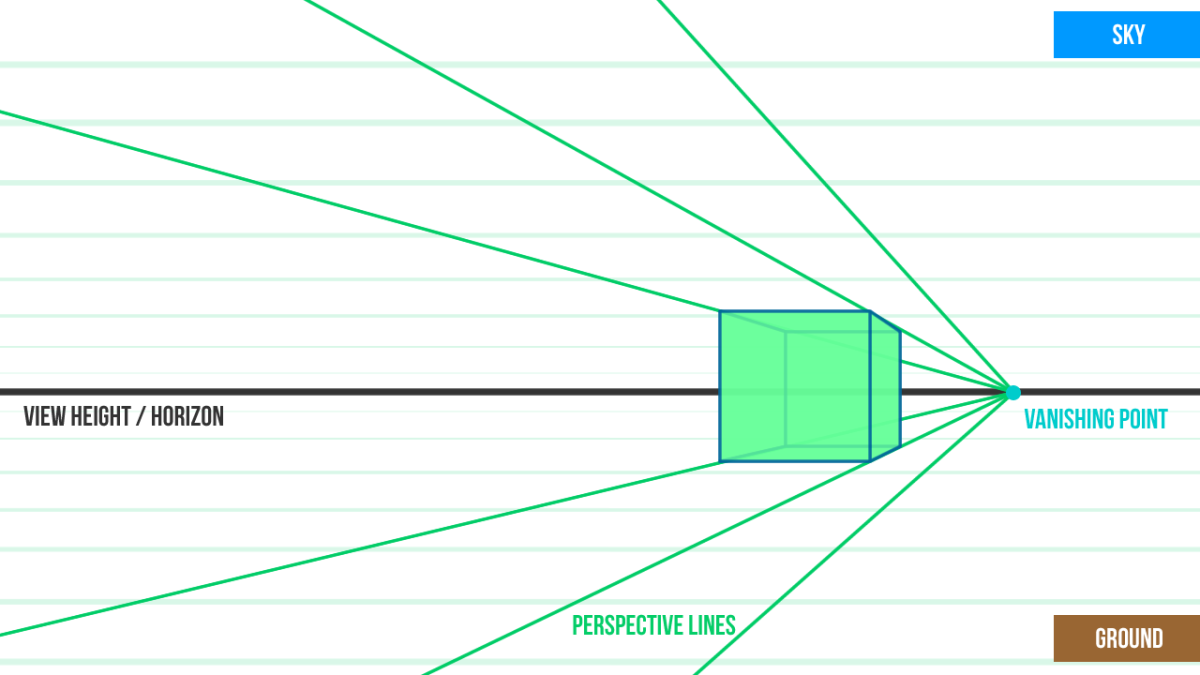


Рисунок 1. Линия горизонта, перспективные линии, исчезающая точка.

Определение данных составляющих помогают верно определить положения объектов на фотографии в трехмерном пространстве. Определив данные атрибуты можно настроить камеру и собрать сцену, проецировав фотографию на объекты.

## **Подготовка фотографии**

Перед началом работы в трехмерном софте, необходимо подготовить изображение (фотографию) к проецированию. Этот этап не столь важен для реализации всего процесса, сколько необходим для достижения корректного и качественного результата.

Для обеспечения точной и аккуратной проекции, на фотографии следует отсоединить выделяющиеся объекты от центрального фона (Рисунок 2, Рисунок 3, Рисунок 4). После таких выделений мы получим отдельные слои с каждым элементом и чисто фон с вырезанными областями, которые нам необходимо закрасить под основной шаблон всего изображения (фотографии). Стоит выделить для себя какие элементы мы будем заменять на собственные, и если они будут мешать целостному восприятию всей картины, то их тоже необходимо удалить с изображения (фотографии). Благодаря таким действиям мы обретаем больше возможностей для работы с трехмерной сценой и получаем в результате качественную анимацию всей картины.

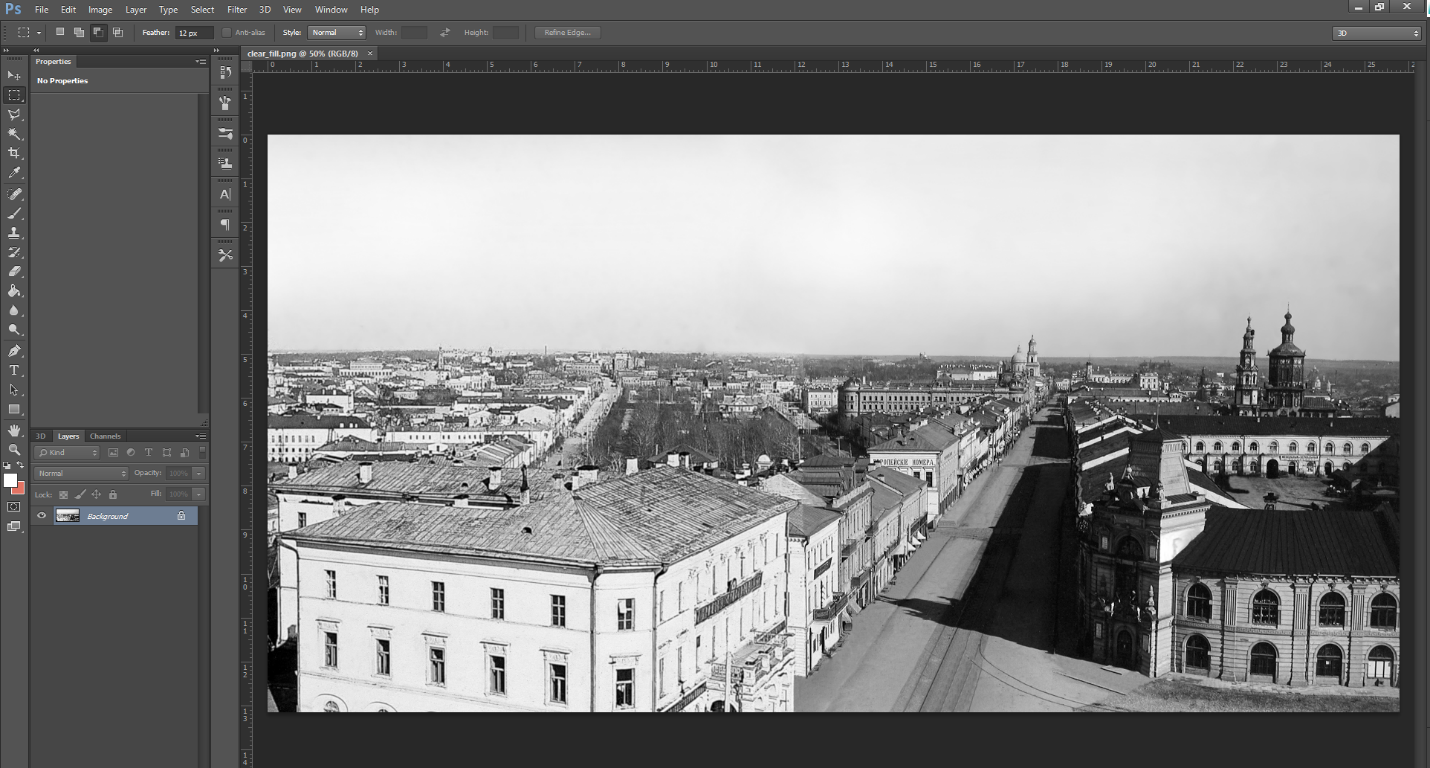


Рисунок 2. Чистая фотография

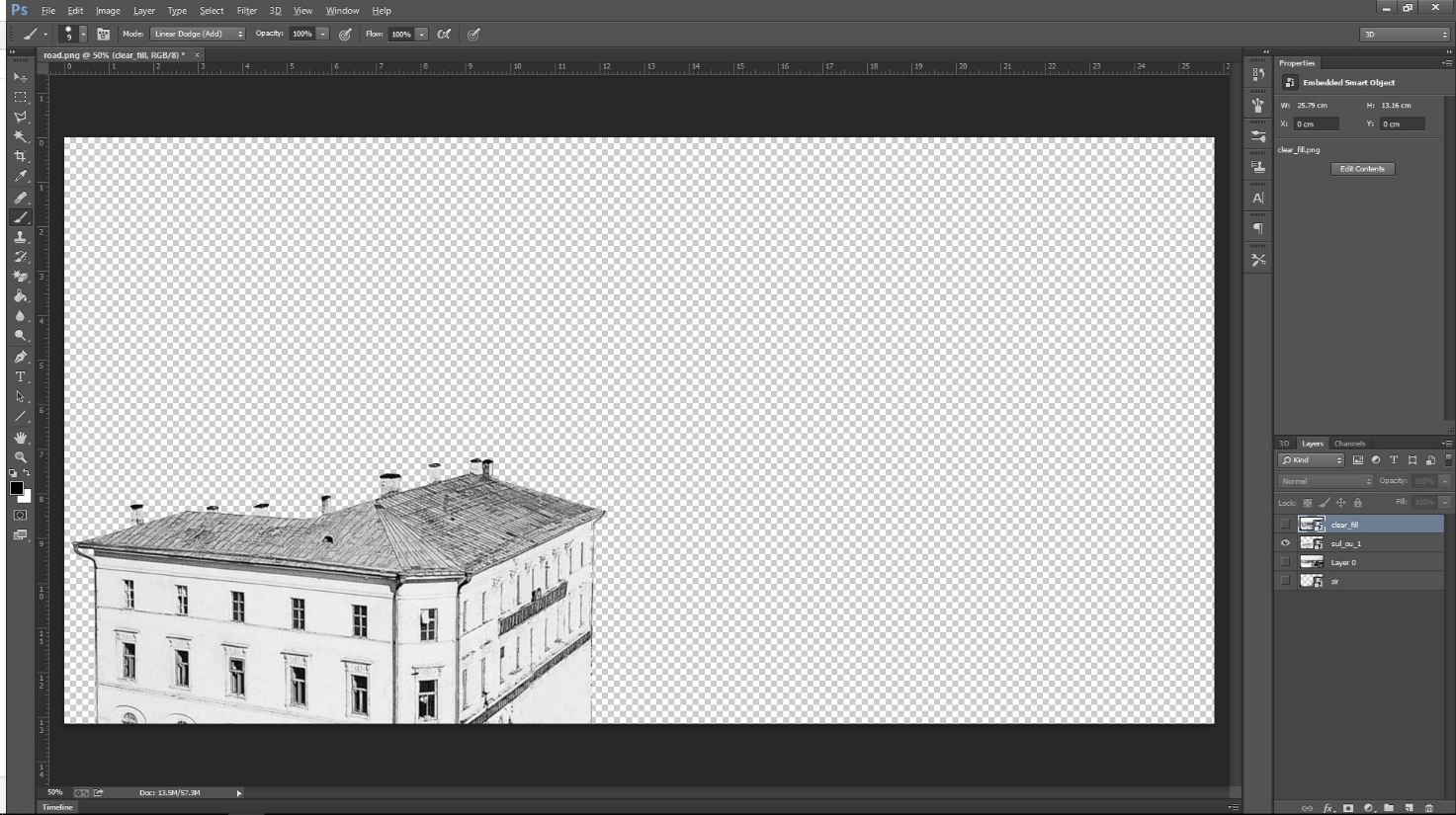


Рисунок 3. Разбиение на слои (Передний дом)

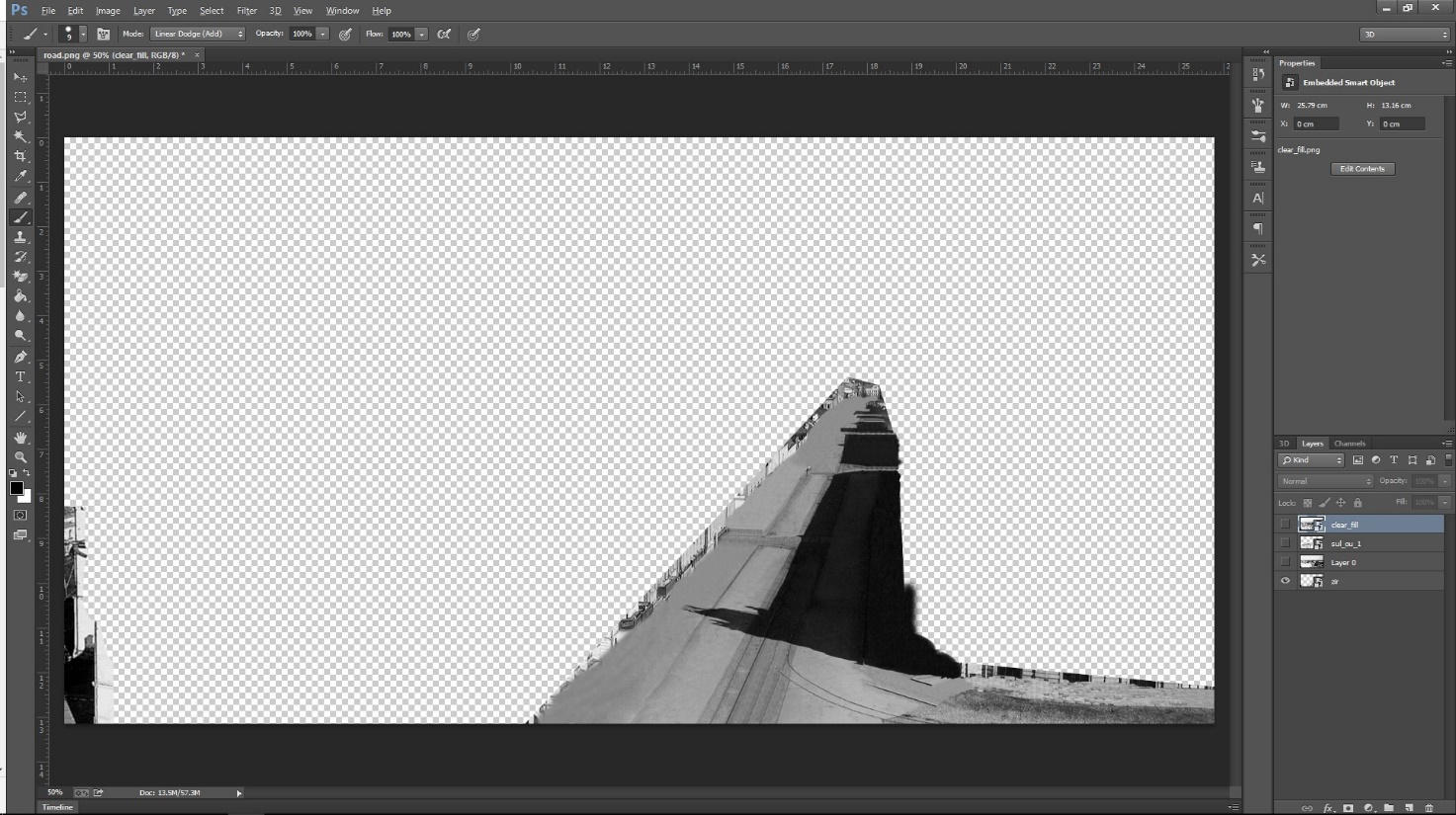


Рисунок 4. Разбиение на слои (Дорога)

## **Настройка сцены**

Как уже говорилось ранее, технология “проецирование из камеры” представляет собой проекцию изображения (фотографии) на сконструированные объекты. Если представить проектор, который проецирует изображение с компьютера на плоский экран, то можно заметить, что камера статична, и при ее перемещении, изменяется и положение проецируемого материала.

Виртуальная камера имеет такую же физику передачи информации, что и проектор. Тем самым мы не сможем проецировать из одной камеры несколько изображений, так же не сможем анимировать ее для финального результата.

Для реализации технологии, создается статичная камера у которой блокируется любая трансформация и замораживается ее положение в пространстве. Это необходимо для невозможности случайно передвинуть или взаимодействовать с ее положением в пространстве. Импортировав плоскость с фотографией, мы получаем виртуальный проектор, который служит базой для конструирования всей сцены. (Рисунок 5).

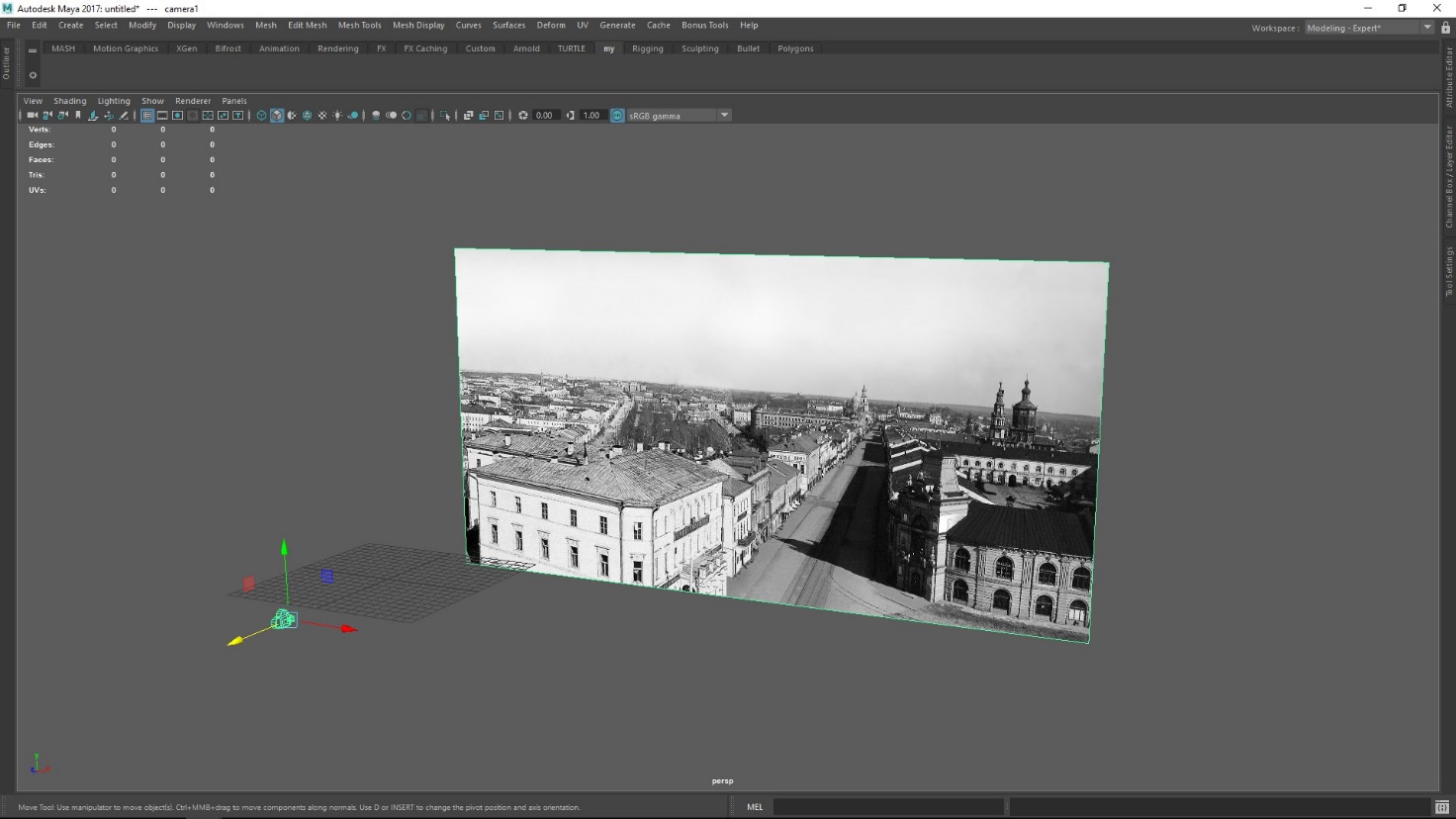


Рисунок 5. Проекция фотографии через камеру.

## **Настройка статичной камеры**

Когда мы настроили сцену, необходимо настроить камеру из который мы проецируем исходное изображение, для точного конструирования сцены. Для установки камеры под изображение, необходимо знать такие параметры: фокусное расстояние, углы обзора по осям, линию горизонта, перспективные линии, исчезающая точка. При наиболее точной установке этих параметров мы получим наиболее точную конструкцию всей сцены.

### **Линии горизонта, перспективные линии**

В Autodesk Maya есть стандартная сетка, которая показывается начальное расположение координат и плоскость горизонта. Ее мы и будем использовать в качестве основы. Суть в том, что, определив грамотно линию горизонта мы получим плоскость земли, и тогда сможем более точно восстановить объекты по фотографии.

Как же определить линию горизонта? Нередко на самих фотографиях присутствуют объекты, которые обтекаются тремя прямыми осями пространства: ось ширины (x), ось длины (y), ось высоты (z). Это может быть, как коробка, квадратный дом, ящик и любая другая деталь по форме похожая на квадрат или прямоугольник. Имея такие объекты на фотографиях возможно сопоставить базовые оси Maya с осями объектов, тем самым мы получим наиболее точное расположение объектов (Рисунок 6).

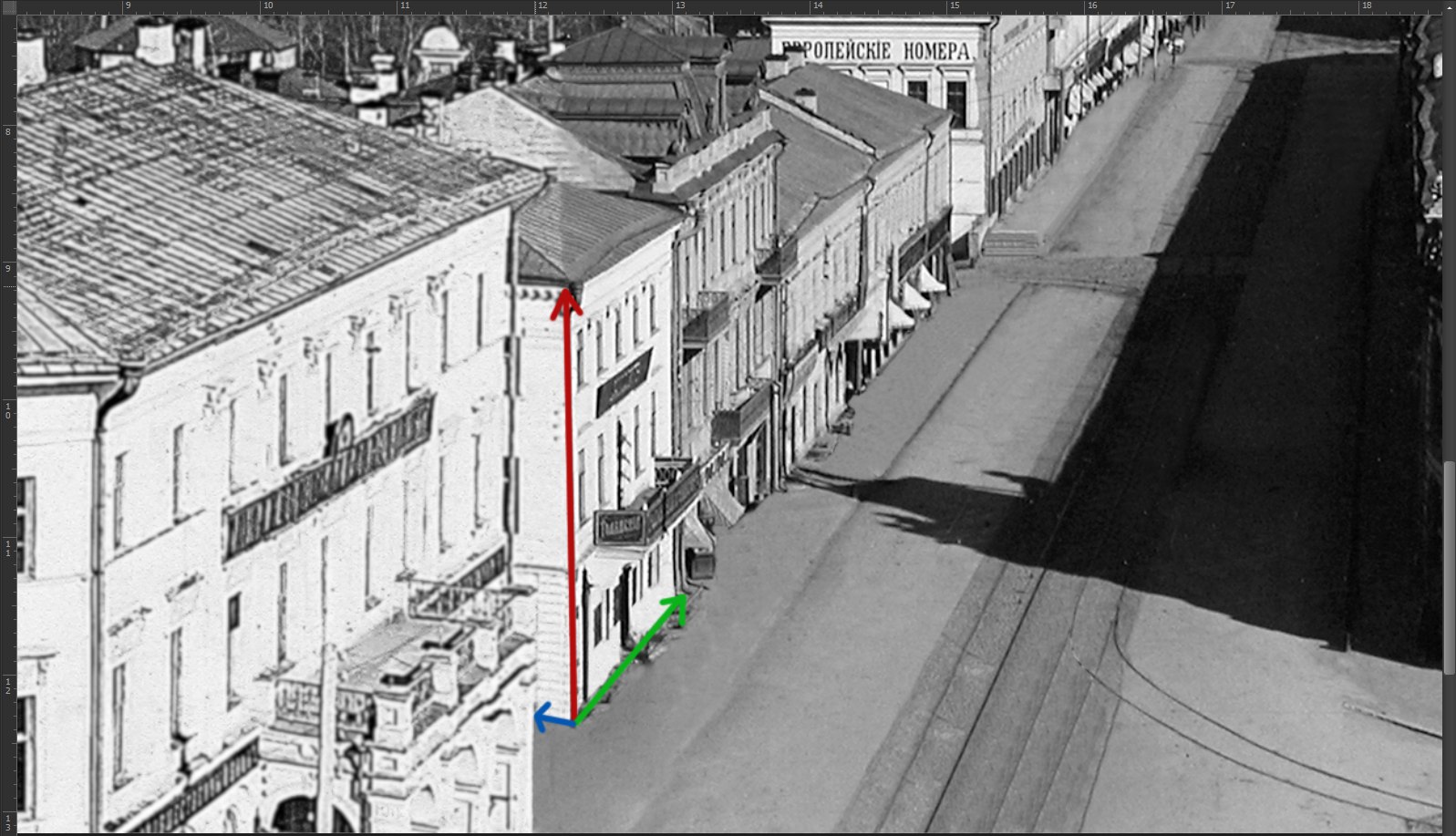


Рисунок 6. Оси объекта.

Определив линию горизонта и плоскость поверхности, необходимо установить фокусное расстояние изображения.

### **Фокусное расстояние и углы обзора**

Фокусное расстояние – физическая характеристика оптической системы [8]. Фокусное расстояние определяет угол изображения. Чем больше фокусное расстояние, тем уже целое изображение, увеличивая все предметы, расположенные на нем, и соответственно наоборот. Фотографию один участок с разным фокусным расстоянием, мы заметим, что где-то объекты на заднем плане далеко позади, а где-то вплотную прилегает к передним объектам.

Фокусное расстояние является одним из главного параметра при настройке камеры, от него зависит верно ли мы расположим объекты на трехмерной сцене или нет. Так же от фокусного расстояния мы сможем узнать углы обзора по вертикали, горизонтали и диагонали, что тоже является ключевым в настройке и соответственно в качестве результата всего процесса. (Рисунок 6)

Определить фокусное расстояние можно либо посмотрев в характеристике самого изображения, либо вручную подстраиваясь под объекты на сцены.

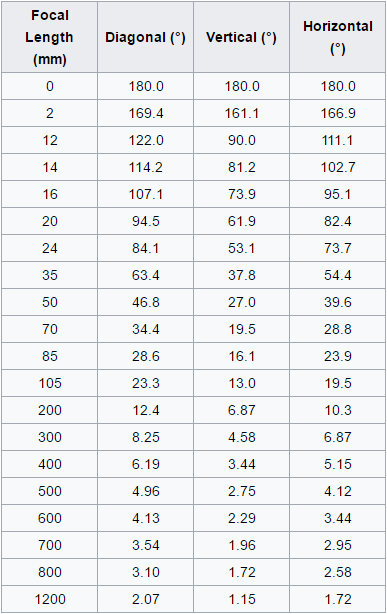


Рисунок 6. Зависимость углов обзора от фокусного расстояния.

Перспективу можно найти несколькими способами:

1. Посмотреть на информацию об изображении. Если фотография была сделана на новые фотоаппараты, то они автоматически задают параметры перспективы и фокусное расстояние у фотографии и нам остается всего лишь вбить эти значения в настройках созданной статичной камеры (Рисунок 8, Рисунок 9). Если же нет этих параметров, а есть только изготовитель камеры, то можно зайти на сайт изготовителей и посмотреть эти параметры.
2. Искать эти параметры вручную, изменять значения в параметрах камеры и смотреть как ведет себя объекты и наиболее точно приблизиться к объектам на фотографии (Рисунок 9).

Используя из любых методов, мы настраиваем камеру на сцене.

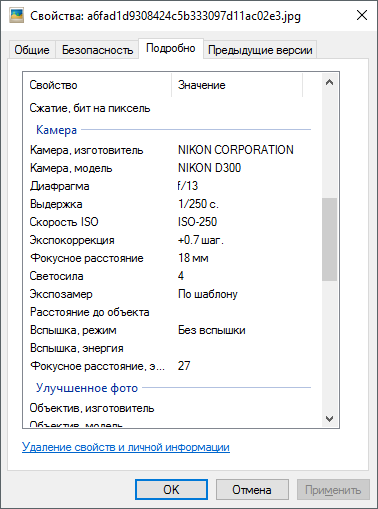


Рисунок 8. Параметры фотографии.

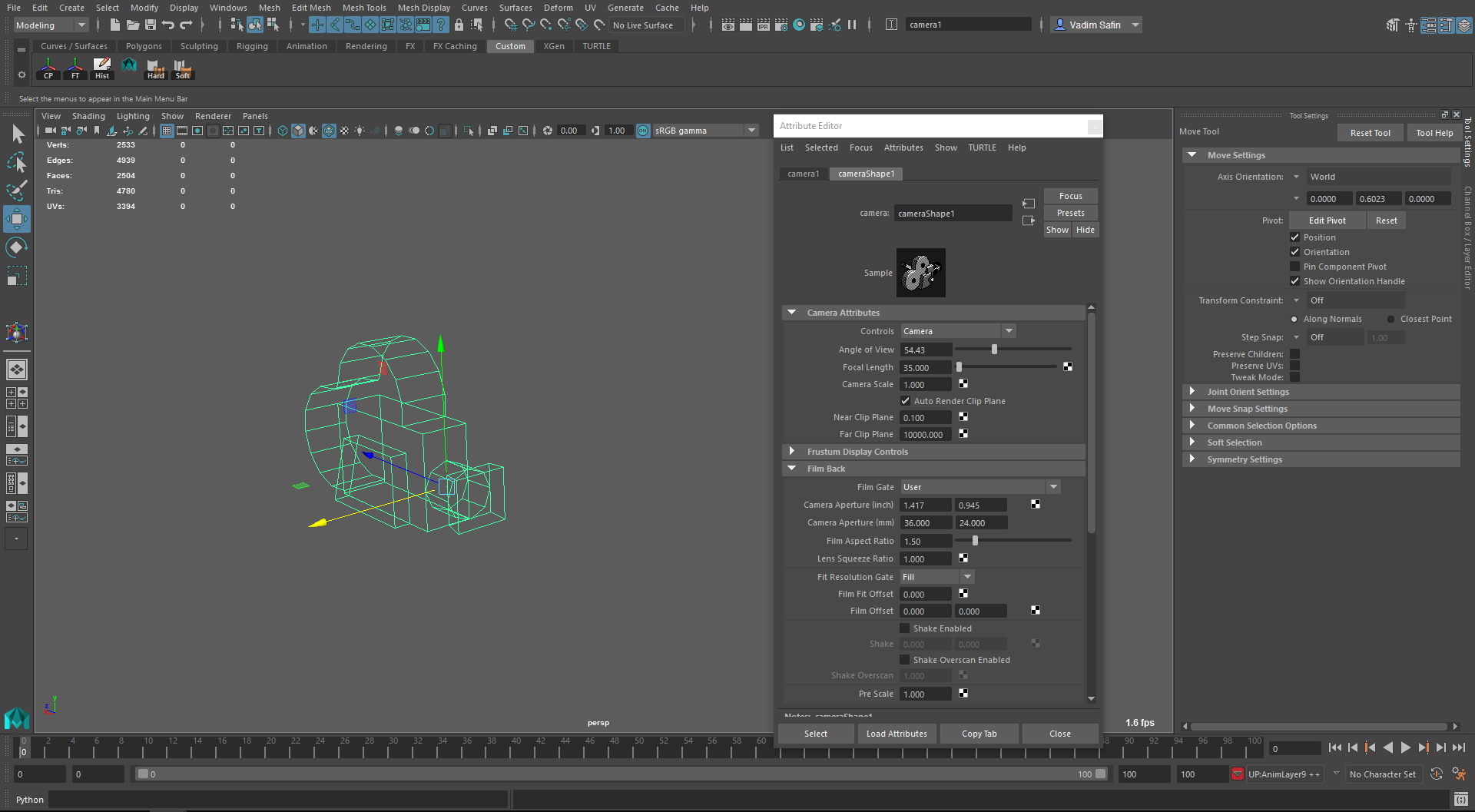


Рисунок 9. Параметры камеры.

## **Моделирование**

Процесс моделирования заключается в создании простых объектов, которые полностью повторяют контур объектов на фотографии (Рисунок 10). Как говорилось раннее, нет необходимости в сильной детализации объектов, так как в качестве текстуры служит проецируемое изображение, и она передает ту детализацию, которая присутствует на изображении.

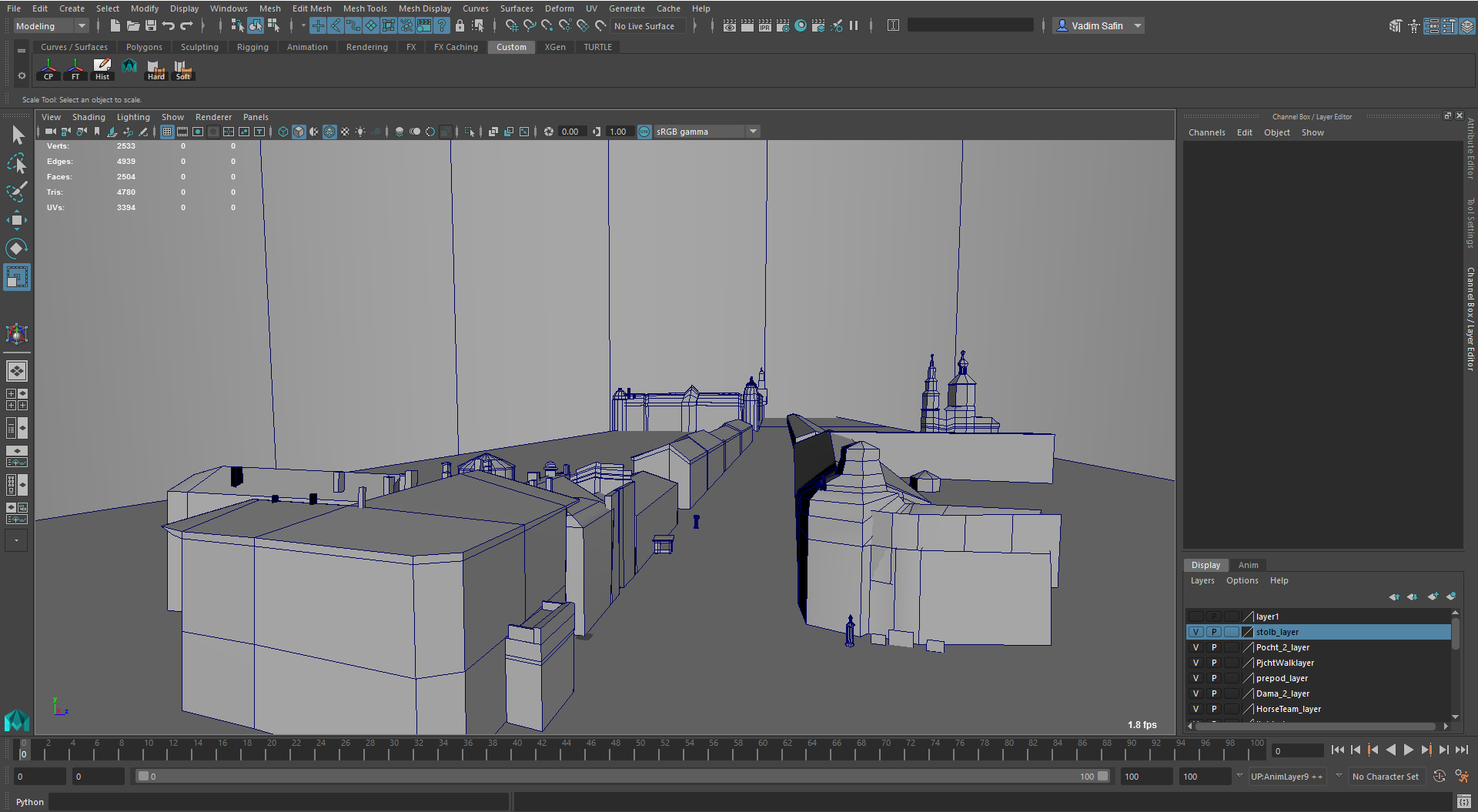


Рисунок 10. Расстановка объектов

## **Проецирование**

Следующим шагом создается динамическая камера для последующей ее анимации. Здесь настраивается только параметры для рендера трехмерной сцены.

После проецируется двухмерное изображение на сцену смотрится, как ведет себя та или иная область при анимации камеры (Рисунок 11).

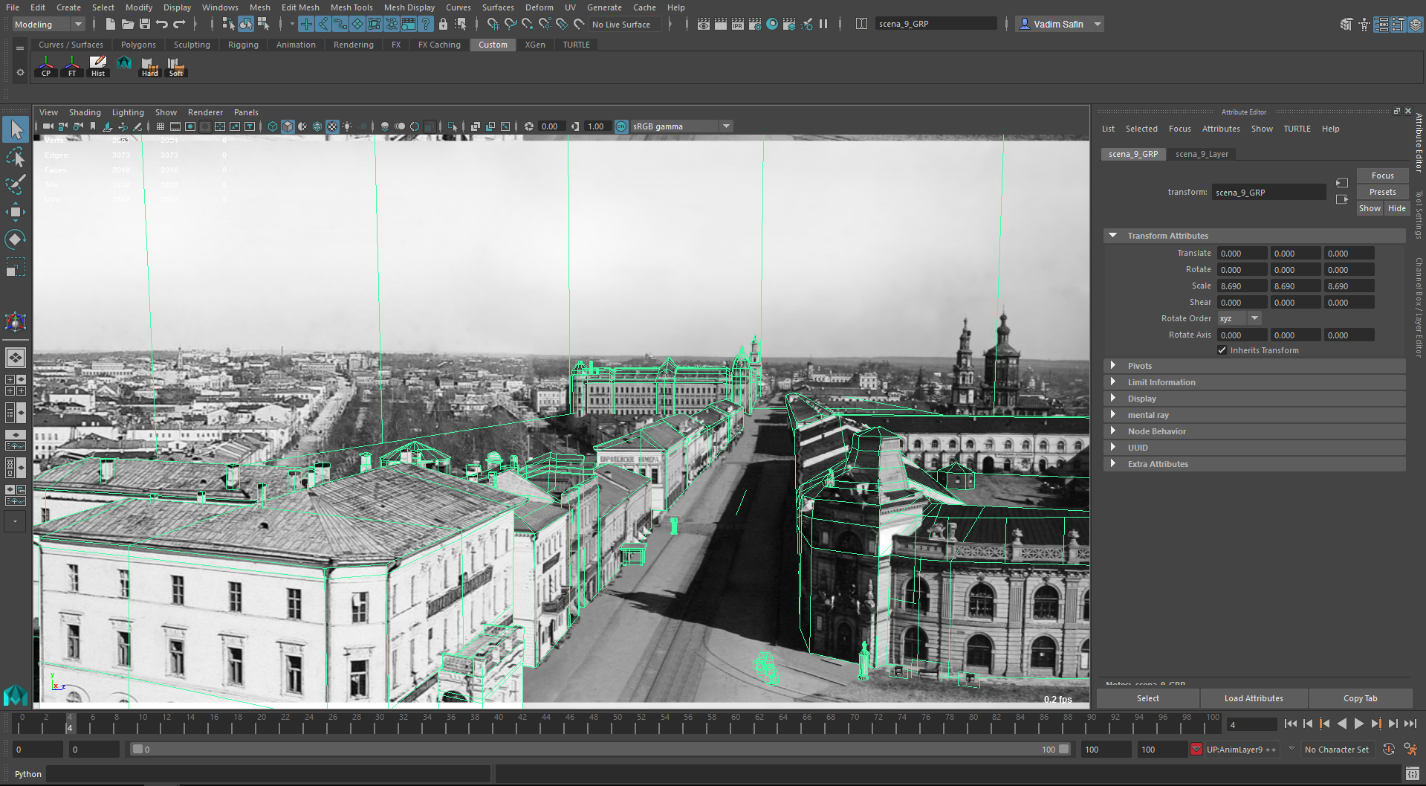


Рисунок 11. Проецирование

Просмотрев искажения и вращение камеры, детализируется объекты где это необходимо или добавляется новые, так же при обнаружении дефектов или артефактов на сцене они исправляются.

Настроив сцену и камеру, проводится финальный рендер сцены и оценивается полученный результат.

# **ПРОБЛЕМЫ**

При работе с данной технологией было выявлено ряд проблем, которые напрямую зависят от настройки камеры.

## **Недостаток данных**

Работая с архивными изображениями, фотографиями, которые были сделаны не в современное время на фотоаппараты старого поколения, возникает проблема в нехватке данных. Сейчас фотоаппараты по дефолту записывают все настройки, при котором была произведена съемка, в атрибуты изображения, и мы можем в любой момент их посмотреть. И зная все необходимые параметры, нам остаётся перенести их в виртуальную камеру и продолжить работу. У архивных фотографий таких атрибутов нет, и бывают искажения нестандартных отклонений, так как использовались индивидуальные линзы, сделанные вручную.

Не установив настройки камере, возникают ряд других проблем, которые сконструируют сцену в нерабочее состояние.

## **Неверно указано фокусное расстояние**

Неверно указав фокусное расстояния и углы обзора, возможно, что расположенные объекты в пространстве будут иметь не те соотношения в размерах, что является в действительности. Так же измениться и их положения относительно друг друга, что приведет к некорректности работы. Данная проблема решается только точной настройкой камеры под заданное изображение.

## **Не определены опорные точки**

При начале конструирования стоит для себя определить ряд моментов, на которые мы будем опираться. Надо указать линию горизонта, где исчезают линии перспектив, плоскость поверхности и элемент с которого мы будем строить сцену. Не определив эти моменты для себя, возможно, что сцена не будет иметь логичности и физики, и объекты будут парить в воздухе и не иметь завершенной конструкции.

## **Дефекты на самих изображениях**

Возможно, что сами изображения подверглись различной деформации:

* Скомбинированные изображения
* Неровные элементы в действительности
* Неровная плоскость поверхности

Решением данной проблемы является настройка камер на каждый элемент, тем сам предотвращая не обнаружения фокусного расстояния всей картины.

# **РЕАЛИЗАЦИЯ АВТОМАТИЗАЦИИ**

# **Разработка функций нахождения фокусного расстояния**

Необходимо создать компонент в трехмерном пакете, который будет автоматически настраивать камеру для последующего моделирования сцены для решения проблем.

Идея компонента заключается в нахождении отклонений плоскостей элементов от нормального состояния. При изменении фокусного расстояния, изменяются и визуальное представление элементов. Чем меньше фокусное расстояние использовали при совершении кадра, тем сильнее подвергается искажению объекты. Эти отклонения нам и понадобиться для нахождения фокусного расстояния.

Для решения данной задачи, был выбран язык программирования – Python версии 3.x. В нем присутствуют библиотеки для работы с изображениями и векторами, а также он поддерживается программой Maya.

Для работы с изображениями использовались библиотеки – PIL (Python Image Library) и для работы с векторами использовалась библиотека – Matplotlib.

Python Image Library – библиотека языка Python, предназначенная для работы с растровой графикой.[6] В частности, будем использовать Pillow, что является собой ответвлением от оригинальной библиотеки PIL. Выбор Pillow был связан с тем, что данный форк включает в себя поддержку Python 3.x, что и необходимо для выполнения поставленных задач.

Данная библиотека позволит манипулировать изображениями, их содержимым и форматами.

MATPLOTLIB – библиотека на языке программирования Python для визуализации данных двумерной и трехмерной графикой.[7] В частности, будем использовать процедурный интерфейс – PyLab, которая предоставляет аналоги команд MATLAB.

Данная библиотека предоставить возможность работать с графиками и осями.

Чтобы начать процесс нахождения фокусного расстояния, нам необходимо загрузить исходную фотографию в нашу систему координат для возможности наложения поверх него векторов. (Рисунок 12)

**def settings**():  
 # open file  
 im = array(Image.open(image))  
 imshow(im)

Рисунок 12. Загрузка изображения.

Загрузив изображение можно указать вектора выбранного объекта для дальнейших вычислений. (Рисунок 13)



Рисунок 13. Обозначение осей перспектив.

Имея такие вектора, мы сможем перенести объект в векторное представление и начать работу с поиском углов отклонений. (Рисунок 14)

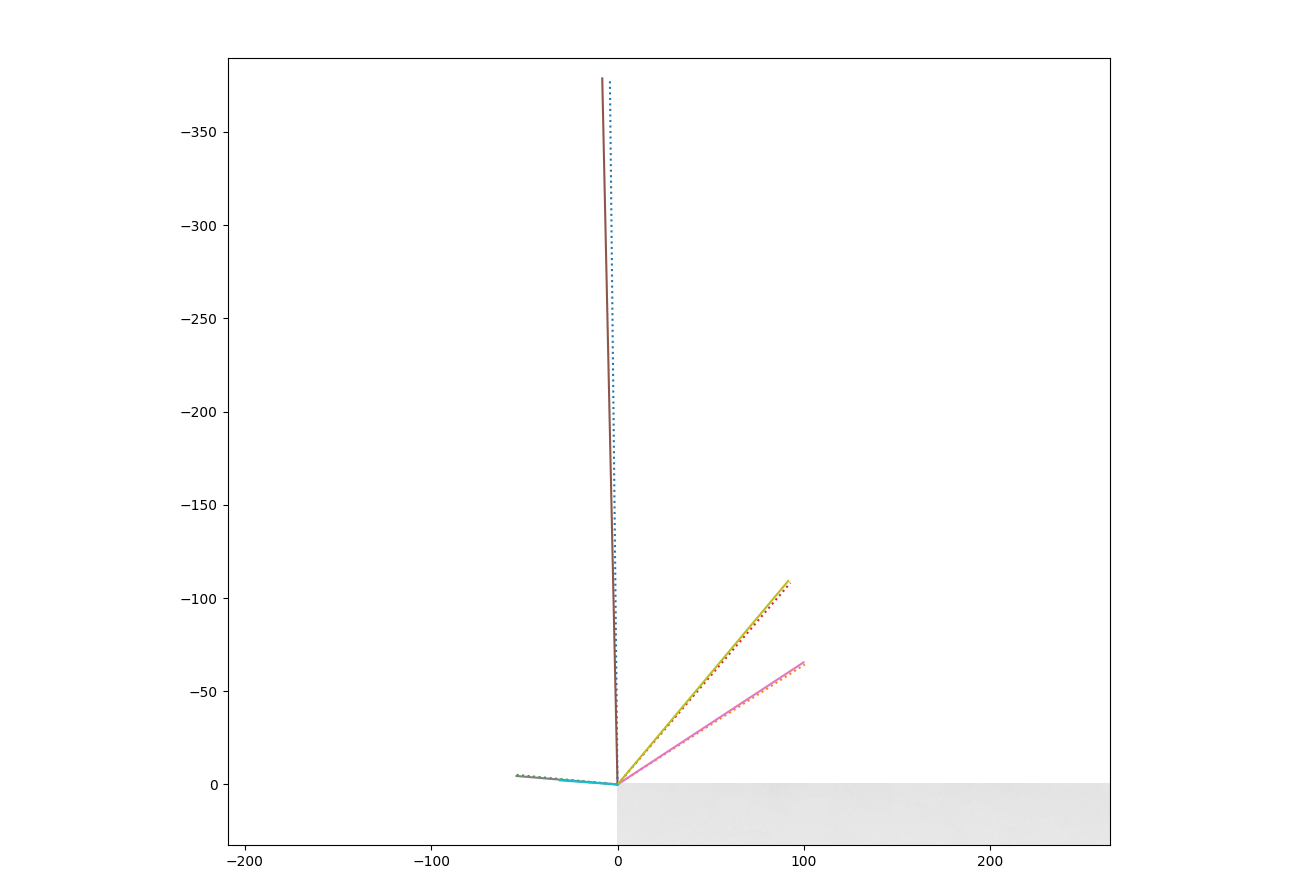


Рисунок 14. Векторное представление.

Указав 5 векторов, мы имеем опорную ось и 2 плоскости которые пересекаются в опорной оси. Опорная ось является статичным элементом, который служит в качестве фундамента искомого угла отклонения.

В качестве опорного вектора, следует указывать ось высоты объекта, желательно стоящего на плоскости поверхности. Так как мы восстанавливаем опорную ось в нормальное положение, мы узнаем угол отклонения всей фотографии относительно нормальных векторов поверхности.

Как уже говорилось выше в Maya есть своя система координат, и мы опираемся на нее. И само проецирования изображения происходит путем наложения текстуры на двухмерную плоскость и есть возможность что нам необходимо будет задать вращение относительно проецируемой оси. Так как фотография возможно имеет отклонение в повороте всей сцены и нам необходимо узнать угол данного отклонения чтобы задать его и настройкам камеры в Maya для точного результата. (Рисунок 15). И зная угол отклонения возможно совершить вращение всей заданной системе векторов. (Рисунок 16)

**def get\_angle\_to\_rotate**(vector\_x, vector\_y):  
 # find scalar of vectors  
 scalar = find\_scalar(vector\_x, vector\_y)  
  
 # find length of vector  
 len\_x = find\_length(vector\_x)  
 len\_y = find\_length(vector\_y)  
  
 # find angle  
 cos\_point = scalar / (len\_x \* len\_y)  
  
 **return** math.degrees(math.acos(cos\_point))

Рисунок 15. Поиск угла отклонения.

**def rotate\_vector**(vector, rotate\_angle):  
 rotate\_angle = math.radians(rotate\_angle + 180)  
  
 **if** vector[1] < 0:  
 x = vector[0] \* math.cos(rotate\_angle) - vector[1] \* math.sin(rotate\_angle)  
 y = vector[1] \* math.cos(rotate\_angle) + vector[0] \* math.sin(rotate\_angle)  
 plot([0, x], [0, y])  
 **else**:  
 x = vector[0] \* math.cos(rotate\_angle) + vector[1] \* math.sin(rotate\_angle)  
 y = vector[1] \* math.cos(rotate\_angle) - vector[0] \* math.sin(rotate\_angle)  
 plot([0, x], [0, y])  
  
 **return** array([x, y])

Рисунок 16. Вращение векторов.

Вернув опорную ось в нормальное состояние так же возвращаем и плоскости смежные ей. В результате мы имеем нормальное представление нашего элемента в векторном пространстве и одну из настроек камеры – угол отклонения всего изображения. (Рисунок 17)

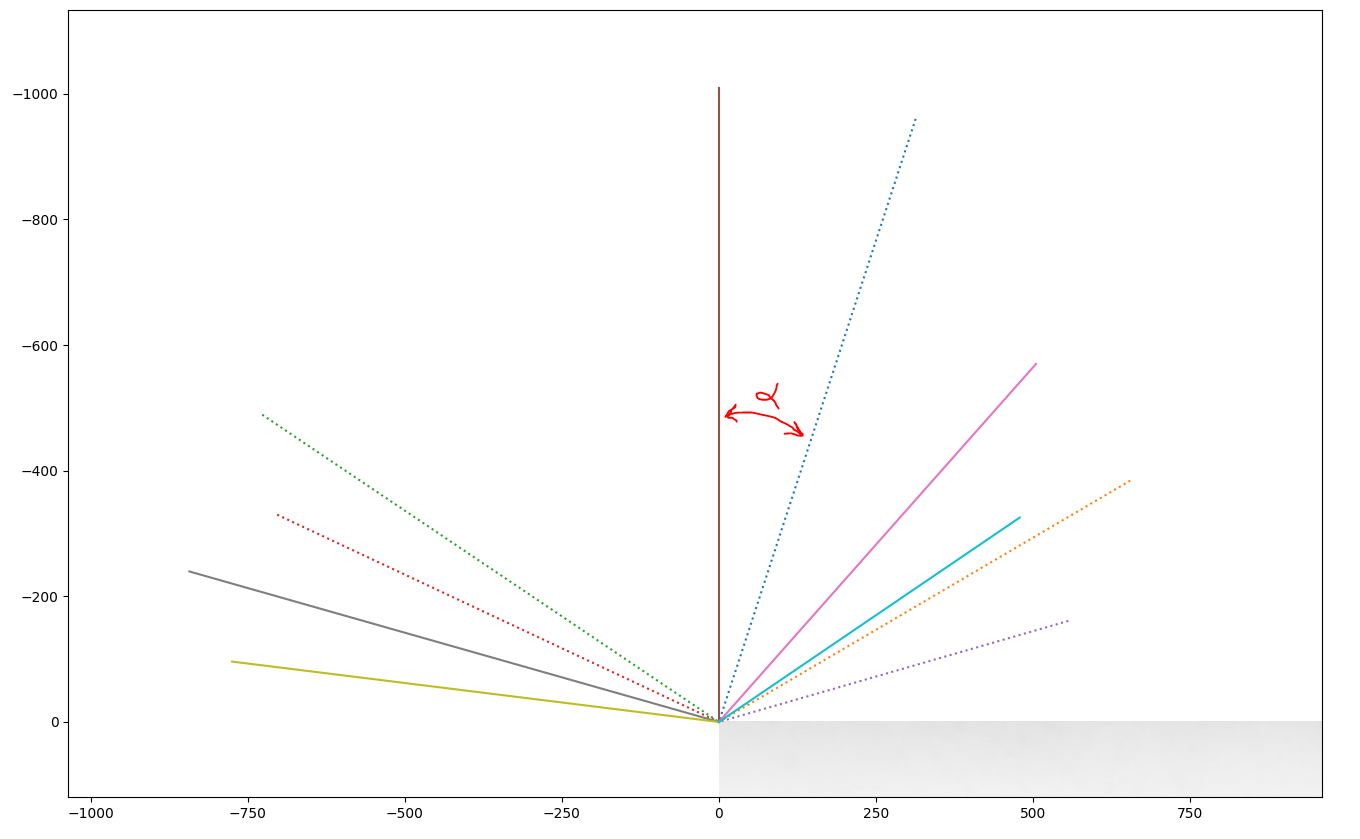


Рисунок 17. Угол отклонения.

Теперь нам необходимо определить угол между двумя плоскостями для нахождения фокусного расстояния. Угол между двумя плоскостями равен углу между их нормальными векторами. Необходимо определить нормальные вектора на плоскостях. Для нахождения нормалей перенесём векторное представление в трехмерное пространство и определим ось Z по формуле:

И получаем пересечение двух плоскостей в пространстве. (Рисунок 18)

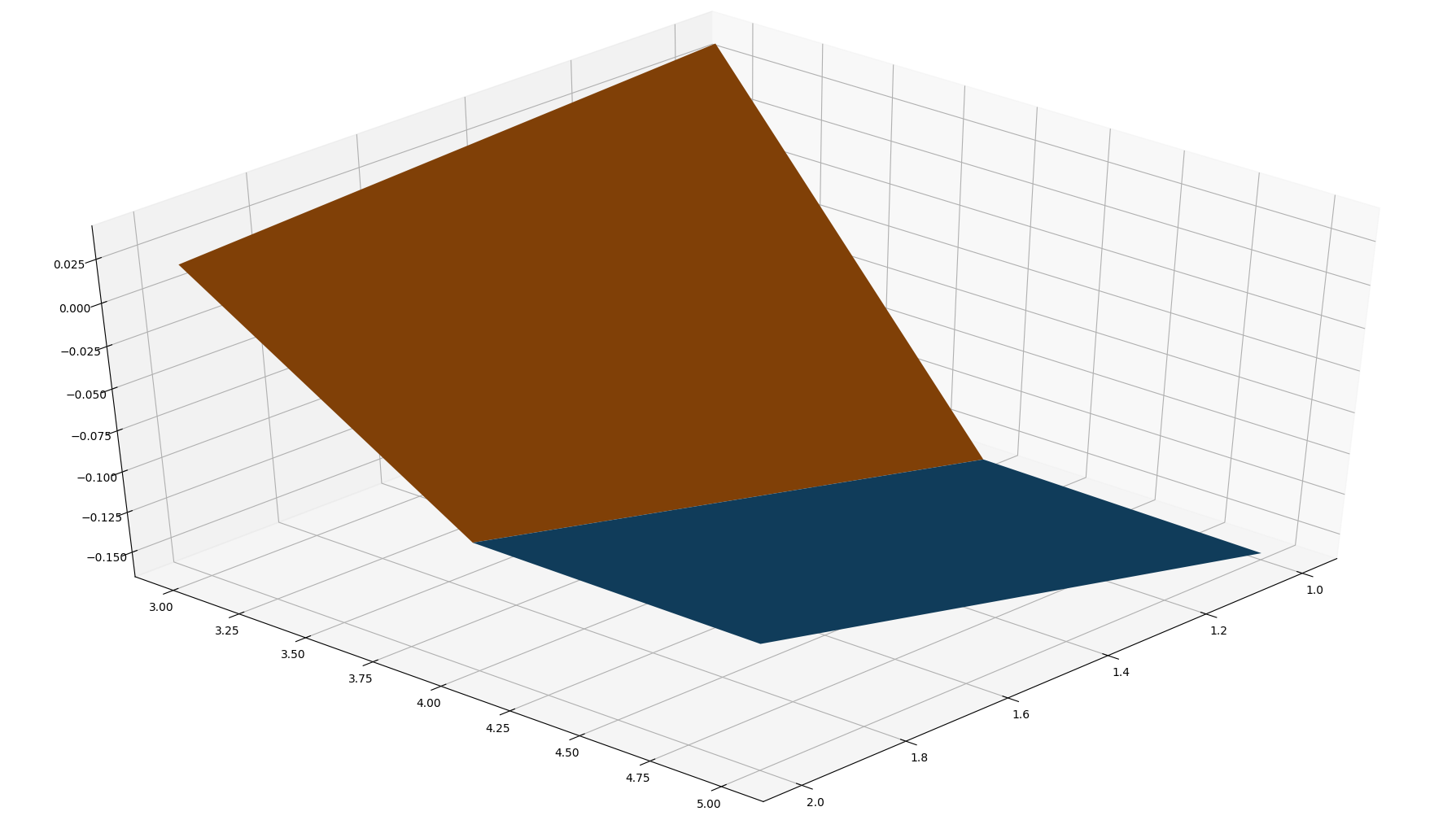


Рисунок 18. Пересечение плоскостей.

И остается определить нормали и высчитать по формуле нахождения угла с помощью нормалей:

Так же возможно зная ось Z определить угол и по формуле ниже:

В результате мы имеем угол отклонения от нормальных поверхностей и можем определить фокусное расстояние и задать ее камере.

# **Разработка интерфейса в Maya (UI)**

Для реализации пользовательского интерфейса использован стандартный инструмент в Maya – Script Editor. (Рисунок 18).

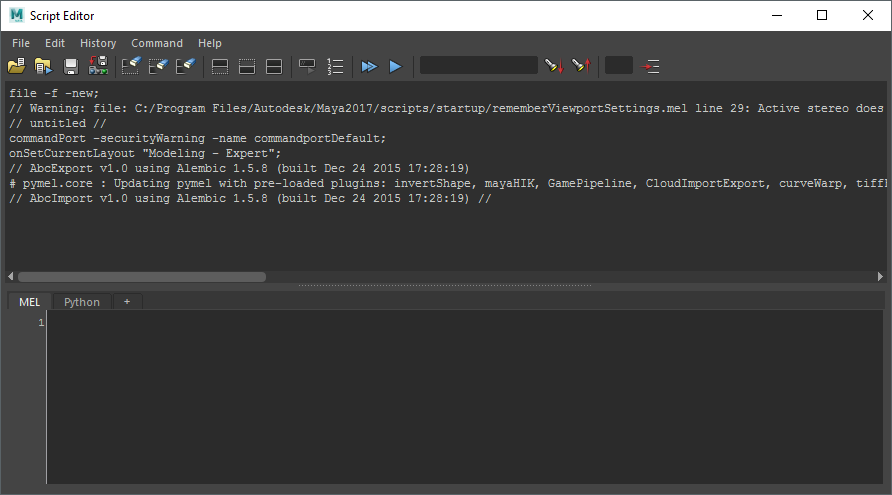


Рисунок 18. Script Editor.

Данный инструмент включает в себя поддержку языка Python и всех его библиотек. Для разработки пользовательского интерфейса были использованы библиотеки: maya.cmds и PySide2 (оболочка PyQt5)

Maya.cmds – это Python оболочка команд MEL, которая может быть использована вместо MEL. [5]

PySide2 – обеспеивает порт для PyQt5 для Maya 2017.

Компонент будет представлять из себя плагин Maya с функционалом по настройке камеры, с элементами ввода и вывода данных.

В качестве функционала у компонента присутствует (Рисунок 19):

1. Окно пользования
2. Создание камеры
3. Наименование камеры
4. Загрузки изображения
5. Кнопки установки осей
6. Редактирование параметров камеры
7. Запуск

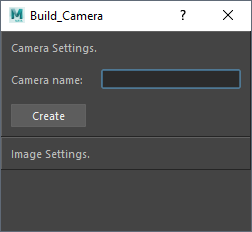


Рисунок 19. Прототип интерфейса компонента

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения дипломной работы, была рассмотрена технология проецирования двухмерных изображение на трехмерные сцены. Была рассмотрена её реализация в программе Autodesk Maya, так же было проведено подготовка исходной фотографии к применению данного метода проецирования на объекты. И было рассмотрено пошаговое исполнение технологии Projection Mapping.

В конце изучения Camera Mapping были выделены интересующие аспекты данной технологии для возможности дальнейших автоматический действий. Так же были выявлены ряд проблем, возникающих при реализации данной технологии. Были предложены подходы к процессу реализации автоматизации для решения проблем. Разработан дальнейший план к решению поставленных задач для достижения основной цели. И был предложен функционал к плагину.

# **СПИСОК ЛИТЕРАТУРЫ**

1. [Электронный ресурс удаленного доступа]: <http://www.x-scripts.com/articles/sis.php> [1]
2. [Электронный ресурс удаленного доступа]: <http://3dyuriki.com/2016/03/24/chto-takoe-camera-projection-mappingspravochnik/> [2]
3. [Электронный ресурс удаленного доступа]: <https://ru.wikipedia.org/wiki/Горизонт> [3]
4. [Электронный ресурс удаленного доступа]: <https://ru.wikipedia.org/wiki/Перспектива> [4]
5. [Электронный ресурс удаленного доступа]: <https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-C0F27A50-3DD6-454C-A4D1-9E3C44B3C990-htm.html> [5]
6. [Электронный ресурс удаленного доступа]: [https://ru.wikipedia.org/wiki/Python\_Imaging\_Library [6](https://ru.wikipedia.org/wiki/Python_Imaging_Library%20%5b6)]
7. [Электронный ресурс удаленного доступа]: [https://ru.wikipedia.org/wiki/Matplotlib [7](https://ru.wikipedia.org/wiki/Matplotlib%20%5b7)]
8. [Электронный ресурс удаленного доступа]: [https://ru.wikipedia.org/wiki/ФокусноеРасстояние [8](https://ru.wikipedia.org/wiki/ФокусноеРасстояние%20%5b8)]

# **ПРИЛОЖЕНИЕ**

Maya UI

import PySide2.QtWidgets as qw

import PySide2.QtCore as qc

import maya.cmds as cmds

class simpleUI(qw.QDialog):

def \_\_init\_\_(self):

qw.QDialog.\_\_init\_\_(self)

self.setWindowTitle('Build\_Camera')

self.setModal(False)

# set size window

self.setMaximumHeight(400)

self.setMinimumHeight(200)

self.setMaximumWidth(400)

self.setMinimumWidth(250)

# set global layout

self.setLayout(qw.QVBoxLayout())

self.layout().setContentsMargins(0,0,0,0)

self.layout().setSpacing(0)

# set place for camera settings

camera\_frame = qw.QFrame()

camera\_frame.setFrameStyle(qw.QFrame.Panel | qw.QFrame.Raised)

picture\_frame = qw.QFrame()

picture\_frame.setFrameStyle(qw.QFrame.Panel | qw.QFrame.Raised)

# add places to global layout

self.layout().addWidget(camera\_frame)

self.layout().addWidget(picture\_frame)

camera\_frame.setSizePolicy(qw.QSizePolicy.Minimum, qw.QSizePolicy.Maximum)

self.layout().setAlignment(qc.Qt.AlignTop)

# add layouts and items to camera settings

camera\_title = qw.QLabel('Camera Settings.')

self.camera\_button = qw.QPushButton('Create')

camera\_frame.setLayout(qw.QVBoxLayout())

camera\_text\_layout = qw.QHBoxLayout()

camera\_frame.layout().setAlignment(qc.Qt.AlignTop)

camera\_frame.layout().addWidget(camera\_title)

camera\_frame.layout().addLayout(camera\_text\_layout)

camera\_frame.layout().addWidget(self.camera\_button)

camera\_frame.layout().setSpacing(15)

self.camera\_button.setSizePolicy(qw.QSizePolicy.Maximum, qw.QSizePolicy.Maximum)

# add items to text\_camera\_frame\_layout

camera\_title\_text = qw.QLabel('Camera name: ')

self.camera\_new\_name = qw.QLineEdit()

camera\_text\_layout.addWidget(camera\_title\_text)

camera\_text\_layout.addWidget(self.camera\_new\_name)

# add layouts and items to image settings

image\_title = qw.QLabel('Image Settings.')

picture\_frame.setLayout(qw.QVBoxLayout())

picture\_frame.layout().addWidget(image\_title)

# actions

self.connect(self.camera\_button, qc.SIGNAL("clicked()"), self.create\_camera)

def create\_camera(self):

new\_name = self.camera\_new\_name.text()

cameraName = cmds.camera(name = new\_name + "#")

# print(cmds.listCameras())

# cmds.rename(cameraName, 'spinning\_ball')

dialog = simpleUI()

dialog.show()

Python set angle of view and focal length

**from** PIL **import** Image  
**from** pylab **import** \*  
  
**def** settings():  
 # open file  
 im = array(Image.open('09.jpg'))  
 imshow(im)  
  
**def** set\_vector(color):  
 # draw vectors  
 vector = ginput(2)  
 **for** point **in** vector:  
 plot(point[0], point[1], 'rx')  
  
 **if** color == 1:  
 plot([vector[0][0], vector[1][0]], [vector[0][1], vector[1][1]], 'b')  
 **if** color == 2:  
 plot([vector[0][0], vector[1][0]], [vector[0][1], vector[1][1]], 'g')  
 **if** color == 3:  
 plot([vector[0][0], vector[1][0]], [vector[0][1], vector[1][1]], 'r')  
  
 **return** vector  
  
**def** normalize\_vector(x):  
 x1 = x[0][0]  
 x2 = x[1][0]  
 y1 = x[0][1]  
 y2 = x[1][1]  
  
 **if** y1 > y2:  
 x2 = x2 - x1  
 y2 = y2 - y1  
 x1 = 0  
 y1 = 0  
 arr = array([x2, y2])  
 **else**:  
 x1 = x1 - x2  
 y1 = y1 - y2  
 x2 = 0  
 y2 = 0  
 arr = array([x1, y1])  
  
 plot([x1, x2], [y1, y2], ':')  
 **return** arr  
  
**def** get\_angle\_to\_rotate(vector\_x, vector\_y):  
 # find scalar of vectors  
 scalar = find\_scalar(vector\_x, vector\_y)  
  
 # find length of vector  
 len\_x = find\_length(vector\_x)  
 len\_y = find\_length(vector\_y)  
  
 # find angle  
 cos\_point = scalar / (len\_x \* len\_y)  
  
 **return** math.degrees(math.acos(cos\_point))  
  
**def** rotate\_vector(vector, rotate\_angle):  
 rotate\_angle = math.radians(rotate\_angle + 180)  
  
 **if** vector[1] < 0:  
 x = vector[0] \* math.cos(rotate\_angle) - vector[1] \* math.sin(rotate\_angle)  
 y = vector[1] \* math.cos(rotate\_angle) + vector[0] \* math.sin(rotate\_angle)  
 plot([0, x], [0, y])  
 **else**:  
 x = vector[0] \* math.cos(rotate\_angle) + vector[1] \* math.sin(rotate\_angle)  
 y = vector[1] \* math.cos(rotate\_angle) - vector[0] \* math.sin(rotate\_angle)  
 plot([0, x], [0, y])  
  
 **return** array([x, y])  
  
**def** find\_scalar(vector\_x, vector\_y):  
 **return** vector\_x[0] \* vector\_y[0] + vector\_x[1] \* vector\_y[1]  
  
**def** find\_length(vector):  
 **return** sqrt(vector[0] \*\* 2 + vector[1] \*\* 2)  
  
**def** test():  
 **return** vector\_x1\_by\_rotate, vector\_x2\_by\_rotate  
  
# set settings  
settings()  
  
# set vector by Z  
normal\_vector\_z = normalize\_vector(set\_vector(1))  
  
# get angle  
angle\_to\_rotate = get\_angle\_to\_rotate(normal\_vector\_z, array([0, 1]))  
print(angle\_to\_rotate)  
  
# set vectors by X  
vector\_x1 = normalize\_vector(set\_vector(2))  
vector\_x2 = normalize\_vector(set\_vector(2))  
  
# set vectors by Y  
vector\_y1 = normalize\_vector(set\_vector(3))  
vector\_y2 = normalize\_vector(set\_vector(3))  
  
# rotate vectors  
vector\_z\_by\_rotate = rotate\_vector(normal\_vector\_z, angle\_to\_rotate)  
vector\_x1\_by\_rotate = rotate\_vector(vector\_x1, angle\_to\_rotate)  
vector\_x2\_by\_rotate = rotate\_vector(vector\_x2, angle\_to\_rotate)  
vector\_y1\_by\_rotate = rotate\_vector(vector\_y1, angle\_to\_rotate)  
vector\_y2\_by\_rotate = rotate\_vector(vector\_y2, angle\_to\_rotate)  
  
show()