Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.04.01 – Информатика и вычислительная техника
Магистерская программа компьютерное моделирование

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____2016 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Компьютерное моделирование технических систем грузового автомобиля КамАЗ-4310

Исполнитель
студент группы 453ом _____ (подпись, дата) Н.С. Зуев

Руководитель
профессор, доктор техн. наук _____ (подпись, дата) И.Е.Еремин

Руководитель
магистерской программы
профессор, доктор техн. наук _____ (подпись, дата) Е.Л.Еремин

Нормоконтроль
доцент _____ (подпись, дата) В.В.Еремина

Рецензент _____ (подпись, дата)

Рецензент _____ (подпись, дата)

Благовещенск 2016

РЕФЕРАТ

Магистерская диссертация содержит 81 с., 30 рисунков, 1 приложение, 29 источников.

МОДЕЛИРОВАНИЕ, МОДЕЛЬ, ПРЕДМЕТНАЯ ОБЛАСТЬ, ГРАФИКА, ВИЗУАЛИЗАЦИЯ, ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА, ПРОГРАММНЫЙ ПРОДУКТ

Объектом исследования данной работы является устройство технических систем грузового автомобиля КамАЗ-4310.

Цель работы – заключается в разработке программного продукта используя мультимедийные технологии, который может являться кроссплатформенным и может быть использован на бортовом компьютере автомобиля, что существенно упростит понимание и принцип работы устройства.

Выполнение работы включает несколько этапов. Первым этапом является исследование предметной области – обзор существующих решений рассмариваемой задачи. На втором этапе выполняется программное и алгоритмическое обеспечение решения задачи. Следующим этапом является описание алгоритмических и кодовых процедур реализации программного продукта и заключающих этапом является описание разработанного программного продукта.

Разработанный программный продукт может способствовать упростит понимание и принцип работы устройства, а также универсальная платформа разработки даёт возможность использовать продукт на разных устройствах, что является большим плюсом, и новизной в настоящее время.

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей магистерской диссертации использованы ссылки на следующие стандарты и нормативные документы:

ГОСТ 2.104-68 ЕСКД Основные надписи

ГОСТ 2.105-95 ЕСКД Общие требования к текстовым документам

ГОСТ 2.111-68 ЕСКД Нормоконтроль

ГОСТ 7.1-2003 Библиографическое описание документа. Общие требования и правила составления

ГОСТ 19.201-78 ЕСПД Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.401-78 ЕСПД Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 ЕСПД Описание программы.

ГОСТ 19.404-79 ЕСПД Пояснительная записка. Требования к содержанию и оформлению

ГОСТ 19.502-78 Описание применения. Требования к содержанию и оформлению.

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению.

ГОСТ 34.601-90 КСАС Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания

ГОСТ 34.602-89 КСАС Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы управления

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

NURBS – Неоднородный рациональный B-сплайн;

ПП – Программный продукт;

САПР – Система автоматизированного проектирования;

ISO – Международная организация по стандартизации;

TGA – Растровый графический формат;

TIFF – Tagged Image File Format;

ADO – (ActiveX Data Objects) объекты данных «ActiveX»;

ОМТО – отдел материально-технического обслуживания;

ПО – программное обеспечение;

ИС – информационная подсистема.

ВВЕДЕНИЕ

В связи со сложившейся ситуацией в сфере специального образования, а именно, необходимости внедрения современных компьютерных технологий и быстрого развития информационных систем, особую актуальность представляет создание трёхмерных макетов грузовых специализированных автомобилей. На данный момент все технические разработки создаются в рамках САПР (система автоматизированного проектирования). Все эти системы разрабатываются в виде приложения к таким программным продуктам как AutoCAD и работают с учетом установленного программного обеспечения, что не всегда рентабельно. С другой стороны, принимая во внимание наглядность разрабатываемой модели, можно создавать необходимый программный продукт, вскрывающий техническое устройство систем автомобиля, необходимых для преподавания тех или иных дисциплин. Существующие модели максимально детализированы. С одной стороны, это удобно, но информация становится избыточной и как следствие программный продукт становится дорогостоящим. Альтернативным решением упрощения данной ситуации является создание высоко наглядных детализированных трёхмерных моделей, которые будут визуализироваться с помощью игрового графического процессора Unity.

Основной целью магистерской диссертации заключается в разработке программного продукта используя мультимедийные технологии, который может являться кроссплатформенным и может быть использован на бортовом компьютере автомобиля, что существенно упростит понимание и принцип работы устройства.

Выбор технологии Unity3D позволяет разработать двух - или трехмерное приложение, работающее под любыми операционными системами: Windows, OS X, Android, Apple iOS, Linux.

Unity3d является современным кроссплатформенным движком для создания приложений, рассчитанный на то, что весь процесс разработки будет происходить в поставляемой в комплекте интегрированной среде разработки. Среда разработки содержит редактор сцен, объектов и скриптов. В Unity3D есть отложенное осве-

щение, встроенный редактор шейдеров, стандартный набор постпроцессинговых эффектов. Основной концепцией Unity3d является использование в сцене легко управляемых объектов, которые, в свою очередь, состоят из множества компонентов. Создание отдельных объектов и последующее расширение их функциональности с помощью добавления различных компонентов позволяет бесконечно совершенствовать и усложнять проект.

Практическая значимость заключается в том, что конечный программный продукт, разработанный в графическом визуализаторе Unity будет являться кросс-платформенным и может быть использован на бортовом компьютере автомобиля, что существенно упростит понимание и принцип работы устройства.

Апробация результатов диссертационного исследования отражаются в следующих научных публикациях:

1) Зуев. Н.С., Куксенко П.Е. Трёхмерное компьютерное моделирование технического устройства автомобиля // Молодёжь XXI века: шаг в будущее: материалы XVII региональной научно-практической конференции (24 мая 2016 года) : в 4 т. – Благовещенск: Изд-во БГПУ, 2016. – Т. 3. – 258 с.

2) Зуев, Н.С. Трёхмерное компьютерное моделирование технического устройства автомобиля / П.Е. Куксенко // Вестник магистратуры (15 июня 2016 года): № 6(57). – Йошкар-Ола: Изд-во ООО «Коллоквиум», 2016. – Т. II. – 134-135 с.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор существующих методов решения рассматриваемой задачи

В данном разделе будет рассмотрена основная теоретическая часть в области решения рассматриваемой задачи в нашем случае – это компьютерная графика.

Трёхмерная графика (3D (от англ. 3 Dimensions – «3 измерения») Graphics, Три измерения изображения) – раздел компьютерной графики, совокупности приёмов и инструментов (как программных, так и аппаратных), предназначенных для изображения объёмных объектов.

Трёхмерная графика обычно имеет дело с виртуальным, воображаемым трёхмерным пространством, которое отображается на плоской, двухмерной поверхности дисплея или листа бумаги. В настоящее время известно несколько способов отображения трёхмерной информации в объемном виде, хотя большинство из них представляет объёмные характеристики весьма условно, поскольку работают со стереоизображением. Из этой области можно отметить стерео очки, виртуальные шлемы, 3D-дисплеи, способные демонстрировать трёхмерное изображение. Несколько производителей продемонстрировали готовые к серийному производству трёхмерные дисплеи. Однако и 3D-дисплеи по-прежнему не позволяют создавать полноценной физической, осязаемой копии математической модели, создаваемой методами трёхмерной графики. Развивающиеся с 1990-х годов технологии быстрого прототипирования ликвидируют этот пробел. Следует заметить, что в технологиях быстрого прототипирования используется представление математической модели объекта в виде твердого тела (воксельная модель).

Трёхмерное изображение на плоскости отличается от двумерного тем, что включает построение геометрической проекции трёхмерной модели сцены на плоскость (например, экран компьютера) с помощью специализированных программ (однако, с созданием и внедрением 3D-дисплеев и 3D-принтеров, трёхмерная графика не обязательно включает в себя проецирование на плоскость). При этом модель может, как соответствовать объектам из реального мира (автомобили, здания, ураган, астероид), так и быть полностью абстрактной (проекция четырёх-

мерного фрактала).

Существуют следующие разновидности 3D-графики:

– полигональная;

– аналитическая;

– фрактальная;

– сплайновая.

Полигональная графика является наиболее распространенной. Это объясняется, прежде всего, высокой скоростью ее обработки. Любой объект полигональной графики задается набором полигонов. Полигон – это плоский многоугольник. Простейшим вариантом являются треугольные полигоны, ибо, как известно, через любые три точки в пространстве можно провести плоскость. Каждый полигон задается набором точек. Точка задается тремя координатами – X, Y, Z. Таким образом, можно задать 3-мерный объект как массив или структуру. Полигональная сетка – это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трехмерной компьютерной графике и объёмном моделировании. Гранями обычно являются треугольники, четырехугольники или другие простые выпуклые многоугольники (полигоны), так как это упрощает рендеринг, но сетки могут также состоять и из наиболее общих вогнутых многоугольников, или многоугольников с дырками. На рисунке 1 изображен пример полигональной сетки, изображающей автомобиль.

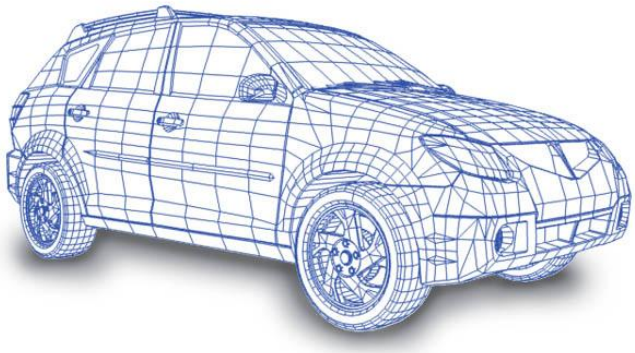

Рисунок 1 – Пример полигональной сетки, изображающей автомобиль

Аналитическая графика состоит, по сути, в том, что объекты задаются аналитически, т. е. формулами. Комбинируя различные формулы друг с другом, можно получить объекты сложной формы. Но вся сложность состоит в нахождении формулы требуемого объекта.

Другой способ создания аналитических объектов – это создание тел вращения. Так, вращая круг вокруг некоторой оси, можно получить тор, а вращая одновременно сильно вытянутый эллипс вокруг собственной и внешней осей, можно получить достаточно красивый рифленый тор.

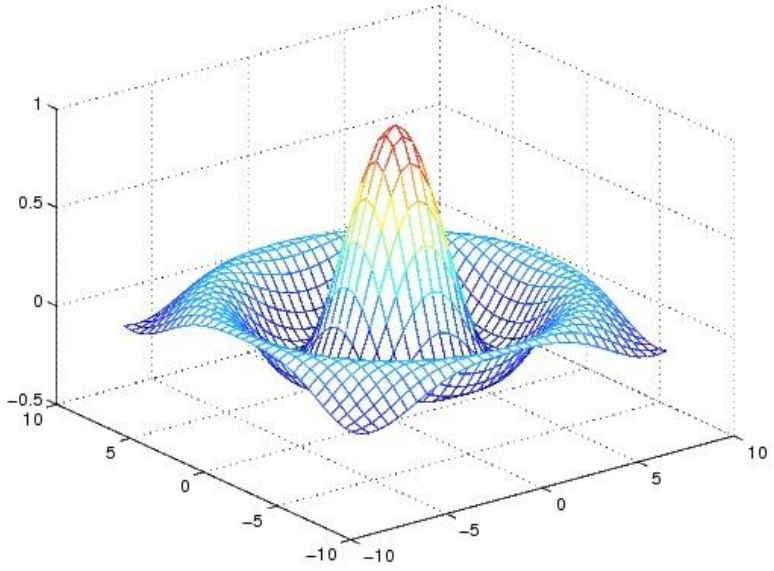

Рисунок 2 – Пример использования аналитической графики

Фрактальная графика основана на понятии фрактала – само подобия. Объект называют само подобным, когда увеличенные части объекта походят на сам объект и друг на друга. К само подобному классу относится местность. Так зазубренный край сломанного камня похож на горный хребет на горизонте. Фрактальная графика, как и векторная, основана на математических вычислениях. Базовым элементом фрактальной графики является математическая формула, в связи с этим никаких объектов в памяти компьютера не хранится и изображение строится исключительно по уравнениям.

Таким образом, строят как простейшие регулярные структуры, так и сложные иллюстрации, имитирующие природные ландшафты и трехмерные объекты.

Алгоритмы фракталов могут создавать невероятные трехмерные изображения.

Рисунок – 3 Пример использования фрактальной графики

Сплайновая графика основана на понятии сплайна. Термин «сплайн» от английского spline. Так принято называть гибкая полоска стали, при помощи которой чертежники проводят через заданные точки плавные кривые. В былые времена подобный способ плавных обводов различных тел (корпус корабля, кузов автомобиля) был широко распространен в практике машиностроения. В результате форма тела задавалась при помощи набора очень точно изготовленных сечений-плазов.

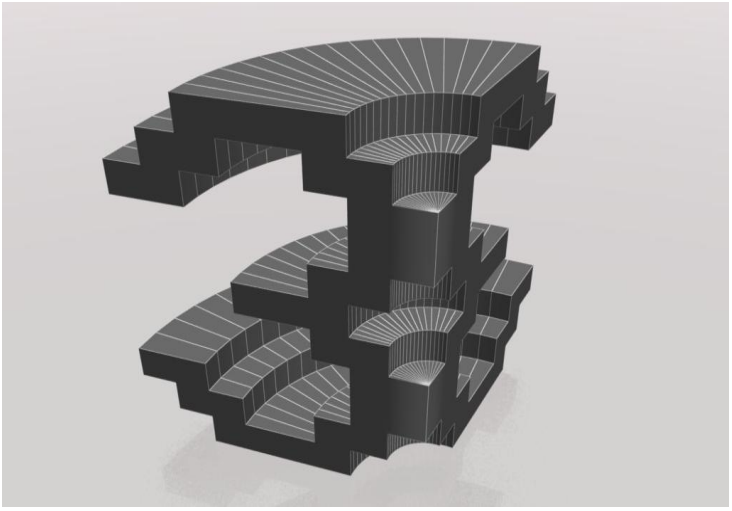

Рисунок 4 – Пример использования сплайновой графики

Появление компьютеров позволило перейти от этого, плазово-шаблонного,

метода к более эффективному способу задания поверхности обтекаемого тела. В корне этого подхода к описанию поверхностей лежит использование сравнительно несложных формул, позволяющих воспроизводить облик изделия с крайне важной точностью.

При моделировании сплайнами чаще всего применяется метод бикубических рациональных B-сплайнов на неравномерной сетке (NURBS). Вид поверхности при этом определяется расположенной в пространстве сеткой опорных точек. Каждой точке присваивается коэффициент, величина которого определяет степень ее влияния на часть поверхности, проходящей вблизи точки. От взаимного расположения точек и величины коэффициентов зависит форма и «гладкость» поверхности.

Деформация объекта обеспечивается перемещением контрольных точек. Другой метод называют сеткой деформации. Вокруг объекта или его части размещается трехмерная сетка, перемещение любой точки которой вызывает упругую деформацию, как самой сетки, так и окруженного объекта.

После формирования «скелета» объекта крайне важно покрыть его поверхность материалами. Все многообразие свойств в компьютерном моделировании сводится к визуализации поверхности, т. е. к расчету коэффициента прозрачности поверхности и угла преломления лучей света на границе материала и окружающего пространства. Для построения поверхностей материалов используют пять базовых физических моделей:

1) Bouknight – поверхность с диффузным отражением без бликов (к примеру матовый пластик);

2) Phong – поверхность со структурированными микро поверхностями (к примеру, металлические);

3) Blinn – поверхность со специальным распределением микронеровностей с учетом взаимных перекрытий (к примеру, глянец);

4) Whitted – модель, позволяющая дополнительно учитывать поляризацию света;

5) Hall – модель, позволяющая корректировать направления отражения и па-

раметры преломления света.

Закраска поверхностей осуществляется методами Гуро (gouraud) или Фонга (Phong). В первом случае цвет примитива рассчитывается в его вершинах, а затем линейно интерполируется по поверхности. Во втором случае строится нормаль к объекту в целом, ее вектор интерполируется по поверхности составляющих примитивов и освещение рассчитывается для каждой точки.

Свет, уходящий с поверхности в конкретной точке в сторону наблюдателя, представляет собой сумму компонентов, умноженных на коэффициент, связанный с материалом и цветом поверхности в данной точке. К таковым компонентам относятся:

1) Свет, пришедший с обратной стороны поверхности, т. е. преломленный свет (Refracted);

2) Свет, равномерно рассеиваемый поверхностью (Diffuse);

3) Зеркально отраженный свет (Reflected);

4) Блики, т. е. отраженный свет источников (Specular);

5) Собственное свечение поверхности (Self Illumination).

Свойства поверхности описываются в создаваемых массивах текстур (двух или трехмерных). Таким образом, в массиве содержатся данные о степени прозрачности материала, коэффициенте преломления, коэффициентах смещения компонентов (их список указан выше), цвете в каждой точке, цвете блика, его ширине и резкости, цвете рассеянного (фонового) освещения, локальных отклонениях векторов от нормали (т. е. учитывается шероховатость поверхности).

Следующим этапом является наложение («проектирование») текстур на определенные участки каркаса объекта. При этом крайне важно учитывать их взаимное влияние на границах примитивов. Проектирование материалов на объект – задача трудно формализуемая, она сродни художественному процессу и требует от исполнителя хотя бы минимальных творческих способностей.

Из всех параметров пространства, в котором действует создаваемый объект, с точки зрения визуализации самым важным является определение источника света. В трехмерной графике принято использовать виртуальные эквиваленты физи-

ческих источников:

1) Растворенный свет (Ambitnt Light), являющийся аналогом равномерного светового фона. Он не имеет геометрических параметров и характеризуется только цветом и интенсивностью.

2) Удаленный не точечный источник называют удаленным светом (Distant Light). Ему присваивают конкретные параметры (координаты). Аналог в природе – Солнце.

3) Точечный источник света (Point Light Source) равномерно испускает свет во всех направлениях и также имеет координаты. Аналог в технике – электрическая лампочка.

4) Направленный источник света (Direct Light Source) кроме местоположения характеризуется направлением светового потока, углами раствора полного конуса света и его наиболее яркого пятна. Аналог в технике – прожектор.

Процесс расчета реалистичных изображений называют ренедерингом (визуализацией). Большинство современных программ рендеринга основаны на методе обратной трассировки лучей. Его суть состоит в следующем, из точки наблюдения сцены посылается в пространство виртуальный луч, по траектории которого должно прийти изображение в точку наблюдения.

Для определения параметров приходящего луча все объекты сцены проверяются на пересечение с траекторией наблюдения. В случае если пресечения не происходит, то считается, что луч попал в фон сцены и приходящая информация определяется параметрами фона. В случае если траектория пересекается с объектом, то в точке соприкосновения рассчитывается свет, уходящий в точку наблюдения в соответствии с параметрами материала.

После завершения конструирования и визуализации объекта приступают к его «оживлению», т. е. заданию параметров движения. Компьютерная анимация базируется на ключевых кадрах. В первом кадре объект выставляется в исходное положение. Через определенный промежуток (к примеру, в восьмом кадре) задается новое положение объекта и так далее до конечного положения. Промежуточные положения вычисляет программа по специальному алгоритму. При этом про-

исходит не просто линейная аппроксимация, а плавное изменение положения опорных точек объекта в соответствии с заданными условиями. Эти условия определяются иерархией объектов (т. е. законами их взаимодействия между собой), разрешенными плоскостями движения, предельными углами поворотов, величинами ускорений и скоростей.

Такой подход называют методом инверсной кинематики движения. Он хорошо работает при моделировании механических устройств. В случае с имитацией живых объектов используют так называемые скелетные модели. Т. е. создается некий каркас, подвижный в точках, характерных для моделируемого объекта. Движения точек просчитываются предыдущим методом.

3D-моделирование – это процесс создания трёхмерной модели объекта. Задача 3D-моделирование – разработать визуальный объемный образ желаемого объекта. С помощью трёхмерной графики можно и создать точную копию конкретного предмета, и разработать новое, даже нереальное представление до сего момента, не существовавшего объекта.

Для получения трёхмерного изображения на плоскости требуются следующие шаги:

1) Моделирование – создание трёхмерной математической модели сцены и объектов в ней;

2) Текстурирование – назначение поверхностям моделей растровых или процедурных текстур (подразумевает также настройку свойств материалов – прозрачность, отражения, шероховатость и пр.);

3) Освещение – установка и настройка источников света;

4) Анимация (в некоторых случаях) – придание движения объектам;

5) Динамическая симуляция (в некоторых случаях) – автоматический расчёт взаимодействия частиц, твёрдых/мягких тел и пр. с моделируемыми силами гравитации, ветра, выталкивания и др., а также друг с другом;

6) Рендеринг (визуализация) – построение проекции в соответствии с выбранной физической моделью;

7) Композитинг (компоновка) – доработка изображения;

8) Вывод полученного изображения на устройство вывода – дисплей или специальный принтер.

Моделирование сцены (виртуального пространства моделирования) включает в себя несколько категорий объектов:

1) Геометрия (построенная с помощью различных техник (напр., создание полигональной сетки) модель, например, здание);

2) Материалы (информация о визуальных свойствах модели, например, цвет стен и отражающая/преломляющая способность окон);

3) Источники света (настройки направления, мощности, спектра освещения);

4) Виртуальные камеры (выбор точки и угла построения проекции);

5) Силы и воздействия (настройки динамических искажений объектов, применяется в основном в анимации);

6) Дополнительные эффекты (объекты, имитирующие атмосферные явления: свет в тумане, облака, пламя и пр.)

Задача трёхмерного моделирования – описать эти объекты и разместить их в сцене с помощью геометрических преобразований в соответствии с требованиями к будущему изображению.

Назначение материалов: для сенсора реальной фотокамеры материалы объектов реального мира отличаются по признаку того, как они отражают, пропускают и рассеивают свет; виртуальным материалам задается соответствие свойств реальных материалов – прозрачность, отражения, рассеивания света, шероховатость, рельеф и пр.

Наиболее популярными пакетами сугубо для моделирования являются:

– Pixologic Zbrush;

– Autodesk Mudbox, Autodesk 3D max;

– Robert McNeel & Assoc. Rhinoceros 3D;

– Google SketchUp;

– Blender.

Для создания трёхмерной модели человека или существа может быть использована как прообраз (в большинстве случаев) скульптура.

Текстурирование это – проецирование растровых или процедурных текстур на поверхности трёхмерного объекта в соответствии с картой UV-координат, где каждой вершине объекта ставится в соответствие определенная координата на двухмерном пространстве текстуры.

Как правило, многофункциональные редакторы UV-координат входят в состав универсальных пакетов трёхмерной графики. Существуют также автономные и подключаемые редакторы от независимых разработчиков, например, Unfold3D Magic, Deep UV, Unwrella и др.

Освещение – заключается в создании, направлении и настройке виртуальных источников света. При этом в виртуальном мире источники света могут иметь негативную интенсивность, отбирая свет из зоны своего «отрицательного освещения». Как правило, пакеты 3D-графики предоставляют следующие типы источников освещения:

1) Omni light (Point light) – всенаправленный;

2) Spot light – конический (прожектор), источник расходящихся лучей;

3) Directional light – источник параллельных лучей;

4) Area light (Plane light) – световой портал, излучающий свет из плоскости;

5) Photometric – источники света, моделируемые по параметрам яркости свечения в физически измеримых единицах, с заданной температурой накала.

Существуют также другие типы источников света, отличающиеся по своему функциональному назначению в разных программах трёхмерной графики и визуализации. Некоторые пакеты предоставляют возможности создавать источники объемного свечения (Sphere light) или объемного освещения (Volume light), в пределах строго заданного объёма. Некоторые предоставляют возможность использовать геометрические объекты произвольной формы.

Анимация – это придание движения трёхмерной модели, либо имитация движения среди трёхмерных объектов. Универсальные пакеты трёхмерной графики обладают весьма богатыми возможностями по созданию анимации. Существуют также узкоспециализированные программы, созданные сугубо для анимации и обладающие очень ограниченным набором инструментов моделирования:

– Autodesk Motion Builder;

– PMG Messiah Studio.

Перед тем как непосредственно перейти к самому моделированию необходимо собрать и проанализировать как можно больше информации о грузовых автомобилях и строении их систем.

Моделирование структуры автомобиля можно разделить на три этапа:

1) Непосредственно создание интерактивной модели автомобиля;

2) Создание UV развертки;

3) Создание текстуры из UV разверти.

В следующем разделе будет подробно описано, как создавалась трехмерная модель грузового автомобиля.

1.2 Постановка задачи

Проанализировать известные программные средства, мы пришли к выводу, что на данное время практически не существуют кроссплатформенных программных средств которые могли бы использоваться на бортовом компьютере автомобиля, либо на каком, либо ином мобильном устройстве по устройству технических систем грузовых автомобилей. Основной задачей является разработать интерактивный программный продукт, используя не обычные САПР технологии, так как все эти системы, разрабатываются в виде приложения к таким программным продуктам как AutoCAD и работают с учетом установленного программного обеспечения, что не всегда рентабельно, а также эти системы являются избыточными и дорогостоящими. Альтернативным решением является создание высоко наглядных детализированных трёхмерных моделей, используя мультимедийный, графический, игровой визуализатор Unity 3D, обоснование выбора данного визуализатора рассмотрим в следующей главе.

Для решения данный задачи в качестве объекта моделирования является грузовой автомобиль КамАЗ-4310.

Автомобиль КамАЗ-4310 по своему типу относится к автомобилям многоцелевого назначения. Это полноприводный автомобиль, предназначенный для перевозки личного состава и различных грузов, буксирования прицельных систем, а

также монтажа вооружения и военной техники (ВВТ) всех видов ВС и родов войск в согласованном объёме, в том числе и на его шасси.

В настоящее время автомобили КамАЗ-4310 являются одними из основных образцов военной автомобильной техники, имеющимся на вооружении в ВС РФ.

На автомобиле КамАЗ установлен двигатель КамАЗ-740.

Основными частями автомобиля являются:

– двигатель;

– шасси;

– кузов;

– электронное и специальное оборудование.

Далее рассмотрим каждую часть автомобиля более подробно.

Двигатель является источником механической энергии, приводящей автомобиль в движение. На современных автомобиля применяются поршневые двигатели – двигатели внутреннего сгорания. В них теплота, выделяющиеся при сгорании топлива в цилиндрах, преобразуется в механическую работу.

Шасси состоит из трансмиссии, ходовой части и систем управления, в которые входят агрегаты и механизмы, служащие для передачи усилия от двигателя к ведущим колесам, а также для управления автомобилем и его передвижением.

Кузов – это часть автомобиля, включающая кабину, грузовую платформу, оперение и предназначенная для размещения и защиты от окружающей среды (дождя, снега, пыли) водителя и пассажиров, а также двигателя и перевозимого груза.

Электрооборудование составляют источники электрической энергии и потребители электрической энергии. Источниками электрической энергии на автомобиле является аккумуляторные батареи, генераторы тока, регуляторы напряжения с фильтрами подавления радиопомех.

К потребителям электрической энергии относятся:

– система электрического пуска;

– приборы освещения, световой и звуковой сигнализации;

– контрольно-измерительные приборы.

Благодаря электрооборудованию обеспечивается поворот коленчатого вала двигателя при его пуске, освещение проезжей части дороги и салона автомобиля, световая и звуковая сигнализация и питание электроизмерительных приборов.

К специальному оборудованию относятся лебедка, система регулирования давления воздуха в шинах, подъемник запасного колеса и др.

Рисунок 5 – Грузовой автомобиль КамАЗ-4310

В ходе работы будет разработан программный продукт, который должен будет включать следующие функции:

– вывод трёхмерного изображения элементов общего технического устройства автомобиля;

– вывод справочных материалов общего технического устройства грузового автомобиля;

–изменение детализации выбранного элемента общего технического устройства грузового автомобиля;

– изменение угла обзора и расстояния до объекта трёхмерного изображения общего технического устройства автомобиля.

Входными данными разрабатываемого программного продукта являются

выбранные элементы технических систем грузового автомобиля в меню программного продукта. Входные данные должны быть представлены в соответствующих полях меню программного продукта.

В соответствии с входными данными программное изделие должно выводить трёхмерное изображение выбранного элемента общих технических систем грузового автомобиля, позволяющее изучить общие технические системы автомобиля. Логическим завершением преобразования входных данных будет являться вывод высоко наглядного детализированного трёхмерного изображения общих технических систем грузового автомобиля.

Результатами моделирования является возможность изучение общих технических систем грузового автомобиля, позволяющий значительно упростить понимание и принцип работы устройства.

Разработка должна включать следующие стадии: теоретические и экспериментальные исследования, разработка приложения для имитационного моделирования, тестирование разработанного приложения.

Программный продукт будет являться кроссплатформенным, т.е. работать на различных операционных систем, в том числе и мобильных. В качестве основного носителя информации для разрабатываемого программного изделия выступает жесткий диск или флэш карта, на котором хранится данное приложение, который позволит достаточно быстро запускать программное изделие, осуществлять изучение нужной информации. Для переноса программного продукта на другие персональные компьютеры или мобильные устройства в качестве носителей могут выступать гибкие и компакт диски или флэш карты.

2 ПРОГРАММНОЕ И АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

2.1 Обзор существующих программных продуктов

Развитие компьютерных технологий открывает новые возможности в различных сферах человеческой деятельности. Автомобилисты тоже смогли найти применение компьютерным технологиям в своей деятельности. Одним из примеров этого стало компьютерное моделирование автомобилей.

Для чего может понадобиться компьютерное изображение автомобиля? В первых моделированием автомобилей при помощи специальных программ занимаются разработчики новых моделей машин. Инженеры и дизайнеры сначала создают трехмерную модель будущего автомобиля на экране компьютера, производят расчет технических характеристик, строят чертежи. Также существуют программы, которые позволяют моделировать отдельные запчасти автомобилей и проводить разноплановое тестирование с целью определить будущие характеристики – прочность, устойчивость, жесткость, плавность хода, а также изучения общего технического устройства автомобиля.

На сегодняшний день, к сожалению, практически не существует интерактивных информационных программных продуктов по устройству технических систем грузового автомобиля КамАЗ-4310 которые бы являлись кроссплатформенными, преимущества кроссплатформенной разработки очевидны. Если программа написана под несколько платформ – у неё будет больше пользователей. Затраты на разработку будут гораздо меньше, чем если бы под каждую платформу программа писалась бы с нуля. Разработчик не привязан к единственной платформе.

Разработка кроссплатформенного приложения заставляет задуматься о хорошем дизайне программы, избегать подключения к «нестандартностям» и недокументированным функциям платформы, что приводит к меньшему числу багов. Есть и рекурсивный аргумент: если программа написана «в кроссплатформенной манере» – добавление поддержки ещё одной платформы происходит гораздо легче, что даёт возможность установить такую систему на бортовой компьютер авто-

мобиля. В настоящее время существуют подобные программные продукты, относящиеся к устройству технических устройств автомобилей. Рассмотрим наиболее популярные программные продукты, которые связаны с техническим устройством автомобиля:

«Устройство легкового автомобиля» – это программный продукт, изображенный на рисунке 6, представляет собой видео руководство помогает понять и изучить все особенности об устройстве автомобиля, принцип работы всех его систем, методах правильной эксплуатации, факторах возникновения поломок, и методах их профилактики и устранения.

Достоинства:

– руководство предназначено как для новичков, которые приобрели машину недавно, так и для автовладельцев с большим опытом;

– видеокурс помогает самостоятельно определить и вовремя устранить неисправности автомобиля.

Недостатки:

– в данном руководстве используется двухмерное изображение устройства автомобиля;

– программный продукт является не кроссплатформенным.

Далее рассмотрим ещё один программный продукт «Обучающая программа автомеханика по ремонту автомобилей»

Система предназначена для получения учащимся знаний, умений и практических навыков, что обеспечивается максимально подробным моделированием действий обучаемого в реальных условиях, вплоть до перемещения объектов. Меню программы представлено на рисунке 6. Работа сопровождается аудио, включает в себя изучение виртуальных моделей, оборудования, инструментов, а также анимированные примеры динамических процессов, проходящих в изучаемых узлах и агрегатах, пример представлен на рисунке 7.

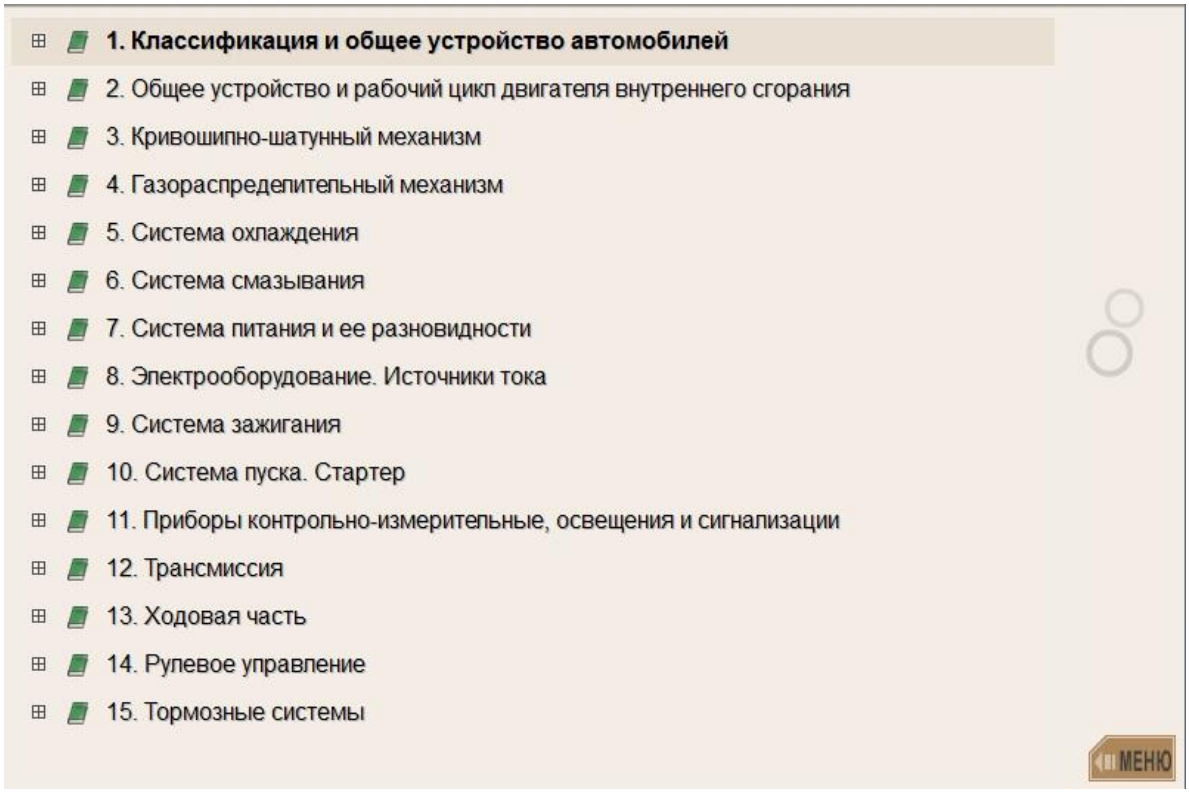

Рисунок 6 – Меню «Обучающая программа по ремонту автомобилей»

Рисунок 7 – Общее устройство и рабочий цикл двигателя

Достоинства:

– руководство предназначено как для новичков, которые приобрели машину недавно, так и для автовладельцев с большим опытом;

– видеокурс помогает самостоятельно определить и вовремя устранить неисправности автомобиля;

– большое количество информации, информация представлена в виде логической последовательности мультимедийных компонентов: 15 фрагментов трехмерной анимации, 60 минут высококачественных видеофрагментов, 250 фрагментов флэш анимации, 500 иллюстраций, 700 тестовых заданий, 450 страниц текста

– звуковая демонстрация.

Недостатки:

– в данном руководстве используется двухмерное изображение устройства автомобиля;

– программный продукт является не кроссплатформенным;

– программный продукт требует первичной инсталляции;

– сложное меню для понимания обычного пользователя.

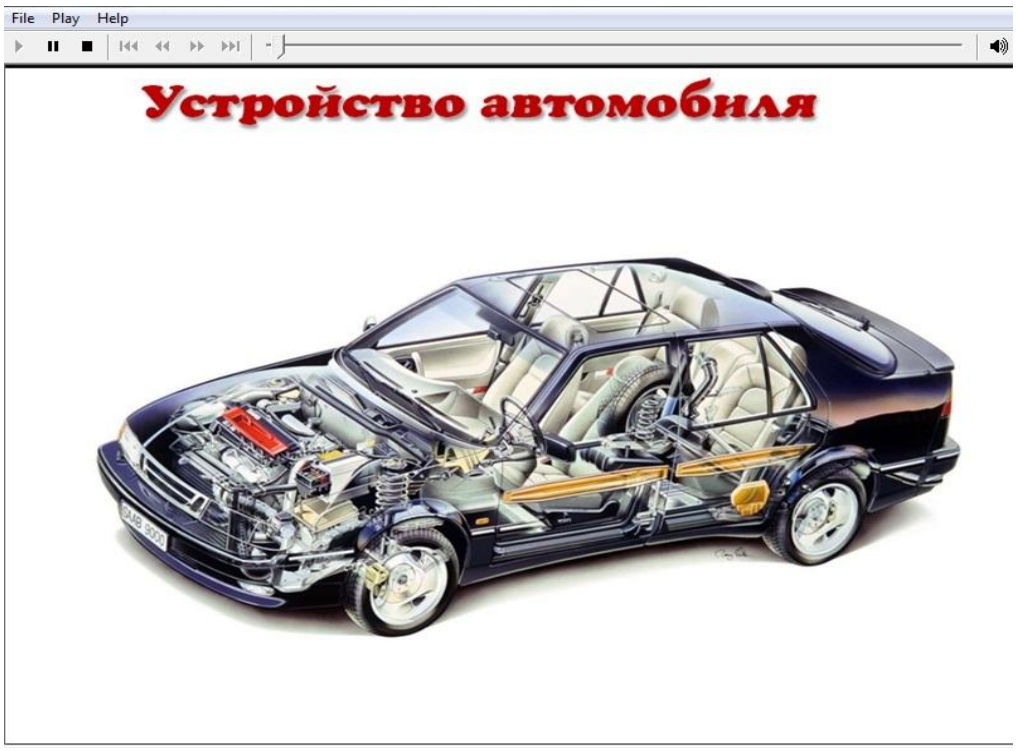

Рисунок 8 – Устройство легкового автомобиля

«Устройство автомобиля» – приложение рассчитано для неопытных пользователей. Приложение разработано для начинающих водителей, тех, кто только что получил права и не приобрёл навыки по устройству автомобиля, а также содержит советы для тех, кто собирается купить автомобиль или уже приобрёл его. Приложение изображено на рисунке 9.

Приложение содержит общую информацию по устройству автомобиля, его составных частей как они работают и как их обслуживать:

– двигателя;

– трансмиссии;

– подвески;

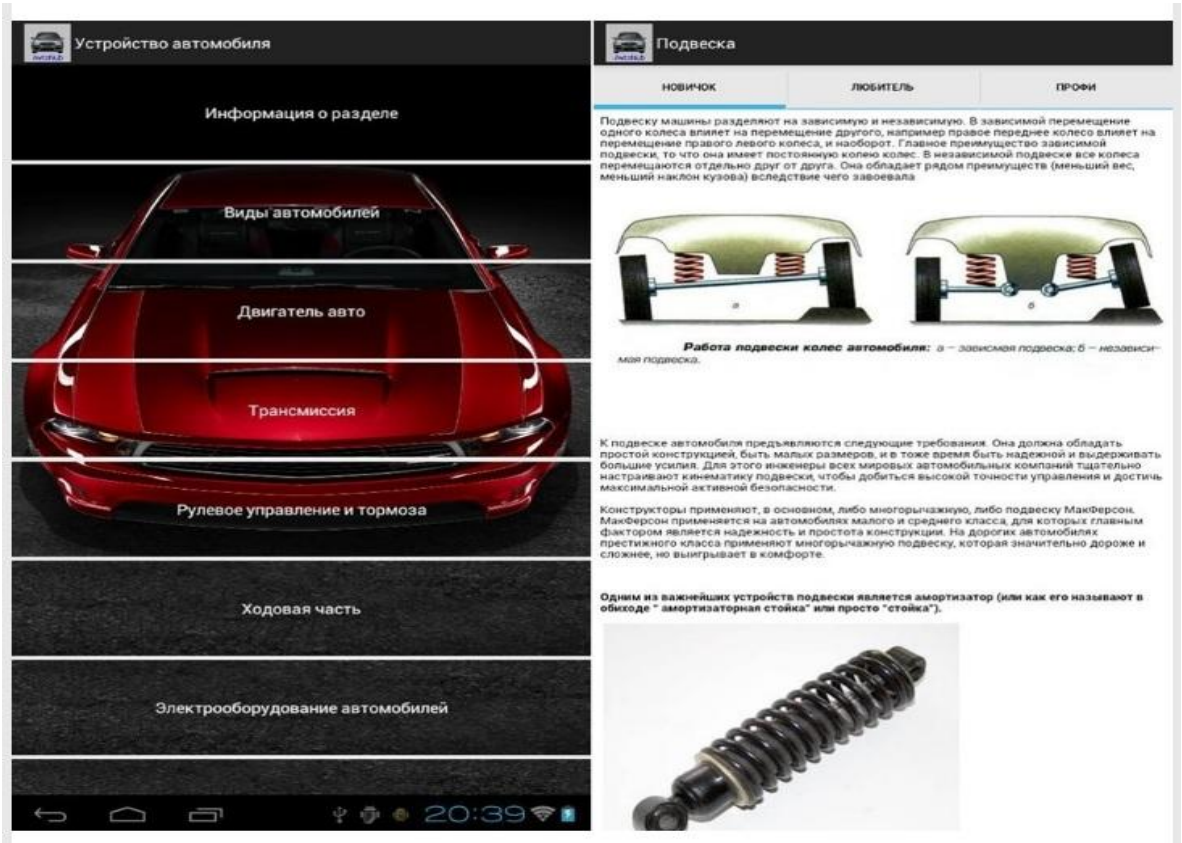

Рисунок 9 – Основное меню приложения «Устройство автомобиля»

– тормозной системы;

– электрооборудование.

Также содержится информация по основам тюнинга автомобиля:

– чип тюнинг;

– тюнинг салона;

– акустики;

– двигателя, трансмиссии.

Достоинства:

– данное приложение является мобильным;

– простое меню, для понимания любого пользователя;

– достаточный объём информации;

– современный дизайн.

Недостатки:

– в данном руководстве используется двухмерное изображение устройства автомобиля;

– программный продукт является не кроссплатформенным;

– программный продукт требует первичной инсталляции.

2.2 Программное обеспечение

Программное обеспечение – это программные средства информационных технологий. Они подразумевают создание, использование компьютерных программ различного назначения и позволяют техническим средствам выполнять операции с машиночитаемой информацией.

Для создания 3D – модели грузового автомобиля нами был выбран пакет для создания трёхмерной компьютерной графики Autodesk 3Ds MAX, включающий в себя средства моделирования, анимации и рендеринга. Дальнейшая разработка информационной системы осуществлялась при помощи мультиплатформенного графического процессора Unity3D с применением высокоуровневого языка программирования C#.

2.2.1 Обоснование выбора языка программирования

Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования (включая нестандартные, визуальные и эзотерические языки). Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть десятком и более разных языков программирования.

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т.п. Язык программирования отличается от естественных языков тем, что предназначен для взаимодействия человека с ЭВМ, в то время как естественные языки используются для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений. Рассмотрим самые распространённые языки программирования.

Java – это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. Дата официального выпуска – 23 мая 1995 года. Изначально язык назывался Oak («Дуб») разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и серверного программного обеспечения. Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с дымящимся кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофемашину как пример бытового устройства, для программирования которого изначально язык создавался. Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) – программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор. Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью

контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Бейсик (от BASIC, сокращение от англ. Beginner's All-purpose Symbolic Instruction Code – универсальный код символических инструкций для начинающих) – семейство высокоуровневых языков программирования.

Был разработан в 1964 году профессорами Дартмутского колледжа Томасом Курцем и Джоном Кемени.

Язык создавался как инструмент, с помощью которого студенты-непрограммисты могли самостоятельно создавать компьютерные программы для решения своих задач. Получил широкое распространение в виде различных диалектов, прежде всего как язык для домашних компьютеров. К настоящему моменту претерпел существенные изменения, значительно отойдя от характерной для первых версий простоты, граничащей с примитивизмом и превратившись в достаточно ординарный язык высокого уровня с типичным набором возможностей. Используется как самостоятельный язык для разработки прикладных программ, главным образом, работающих под управлением ОС Windows различных версий. Также широко распространён в качестве встроенного языка прикладных программных систем различного назначения и в качестве языка для программируемых калькуляторов.

Бейсик был придуман в 1964 году преподавателями Дартмутского Колледжа Джоном Кемени и Томасом Курцем, и под их руководством был реализован командой студентов колледжа. Со временем, когда стали появляться другие диалекты, изначальную реализацию стали называть Dartmouth BASIC.

Бейсик был спроектирован так, чтобы студенты могли без затруднений писать программы, используя терминалы с разделением времени. Он предназначался для более «простых» пользователей, не столько заинтересованных в скорости исполнения программ, сколько просто в возможности использовать компьютер для решения своих задач, не имея специальной подготовки.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Python на самом Python, что уменьшает затраты на изменения языка и постановку экспериментов над новыми возможностями.

Python – активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

C# (Си-шарп) – объектно-ориентированный язык программирования. Создан в 2000 году Андерсом Хейлсбергом, Скоттом Вилтамутом и Питером Гольде, работавшими тогда в «MicrosoftResearch». Основным принципом C# является высказывание: «всякая сущность есть объект». Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

C# – это полнофункциональный объектно-ориентированный язык, который поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию обработки исключений, безопасности типов.

Язык C# разрабатывался «с нуля» и вобрал в себя много полезных свойств таких языков, как C++, Java, VisualBasic, а также Pascal, Delphy и др. При этом необходимость обратной совместимости с предыдущими версиями отсутствовала, что позволило языку C# избежать многих отрицательных сторон своих предшественников.

Как и Java, C# разрабатывался для сети Интернет и примерно 75 % его синтаксических возможностей аналогичны языку программирования Java, его также называют «очищенной версией» Java. 10 % подобны языку программирования C++, а 5 % – заимствованы из языка программирования VisualBasic. Объем новых концептуальных идей в языке C# около 10 %.

Выделение и объединение лучших идей современных языков программирования делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

На основе собранных эмпирических данных, был выбран язык программирования C#.

2.2.2 Обоснование выбора визуализатора

Так как разрабатываемая информационная система, подразумевает использование интерактивной 3D модели, было принято решение выбрать визуализатор с поддержкой Direct3D: интерфейс вывода трёхмерных примитивов. Рассмотрим несколько визуализаторов использующих технологию Direct3D.

Unreal Engine – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Интерфейс графического визуализатора представлен на рисунке 10. Написанный на языке C++, движок позволяет создавать игры для большинства операционных систем и платформ: Microsoft Windows, Linux, Mac OS и Mac OS X; консолей Xbox, Xbox 360, PlayStation 2, PlayStation 3, PSP, PS Vita, Wii,

Dreamcast, GameCube и др., а также на различных портативных устройствах, например, устройствах Apple (iPad, iPhone), управляемых системой iOS и прочих. (Впервые работа с iOS была представлена в 2009 году, в 2010 году продемонстрирована работа движка на устройстве с системой webOS).

Для упрощения портирования движок использует модульную систему зависимых компонентов, поддерживает различные системы рендеринга (Direct3D, OpenGL, Pixomatic, в ранних версиях: Glide, S3, PowerVR), воспроизведения звука (EAX, OpenAL, DirectSound3D; ранее: A3D), средства голосового воспроизведения текста, распознавание речи), модули для работы с сетью и поддержки различных устройств ввода.

Source (рус. «Источник») – игровой движок, разработанный компанией Valve Corporation и впервые задействованный в играх самого производителя – Counter-Strike: Source и Half-Life 2.

Source является игровым движком, поэтому содержит в себе связку различных компонентов, в числе которых: графический движок, физический движок и т.п.

Одной из особенностей движка является его система анимации персонажей, в частности, лицевая анимация, которая содержит множество средств для создания выразительной мимики и точной синхронизации речи актеров с анимацией; также движок отличает продвинутый игровой искусственный интеллект, который может эффективно управлять противниками или союзниками игрока, графический движок был одним из первых, где применялись сложные шейдерные эффекты; в играх на движке активно использовалась шейдерная вода, отражающая окружающий мир.

Физический движок создан на основе Havok. Он позволяет рассчитывать многие физические объекты, такие как твёрдые тела, упругие тела, верёвки, поверхности и т.п. Существует возможность создания реалистичных транспортных средств, от машины до катера на воздушной подушке и вертолёта. Для просчёта поведения транспортного средства на дороге или в воздухе используется много параметров, например сцепление колёс с дорогой, масса машины. Для придания

реалистичного движения телу, используется физика «тряпичной куклы»; созданная заранее анимация может смешиваться с физикой реального времени.

С развитием Source, в него были добавлены: HDR-рендеринг, динамическое освещение и затенение с возможностями само затенения объектов, мягкими тенями от объектов (присутствует возможность использования традиционных карт освещения), многоядерный рендеринг для многоядерных процессоров, развитая система частиц.

Одним из средств разработки является Source SDK – набор утилит для создания модификаций на движке Source, бесплатно доступный через Steam игрокам. В набор входят: Valve Hammer Editor (редактор карт), Faceposer (утилита для создания лицевой анимации моделей), Model Viewer (программа просмотра моделей формата .MDL).

Помимо трёх основных утилит набор включает утилиту для распаковки базовых файлов при создании нового мода, а также файлы исходного кода библиотек части игр Valve, что позволяет вручную создавать игры с изменёнными характеристиками без декомпиляции движка. Однако для компиляции новых файлов необходимо знание языка C++ и компилятора (например, Visual Studio).

Рисунок 10 – Интерфейс графического визуализатора Unreal Engine

Unity 3D – это бесплатный, набирающий популярность мощный движок для создания и разработки игр и различных приложений.

Unity3D имеет простой в обращении интерфейс, позволяющий с легкостью проводить настройку по желанию пользователя; меню представляет собой совокупность рабочих окон, которые могут быть дополнены либо заменены другими. Все это позволяет производить отладку приложения прямо в самом редакторе.

Графической процессор обладает возможностью поддержки DirectX11 и HDR, физические расчеты выполняет физический процессор PhysX от NVIDIA. Также Unity3D поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll («тряпичная кукла»). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

Проект в Unity3D делится на сцены (уровни) – отдельные файлы, содержащие набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые объекты – объекты, которые не имеют модели «пустышки». Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity3D допускается наличие двух и более объектов с одинаковым названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform – он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент MeshRenderer, делающий модель объекта видимой.

С его помощью вы можете создавать игры для различных платформ Windows, iOS, Blackberry, OSX, Wii, Android, Playstation 3, Xbox и Flash. Каждое созданное приложение в программе Unity 3D способно поддерживать OpenGL и DirectX.

Рассмотрим краткий обзор игрового движка Unity 3D, интерфейс графического визуализатора представлен на рисунке 11, обозреватель ресурсов крайне

удобен тем, что все необходимые ресурсы можно создавать прямо там, без необходимости открывать проводник операционной системы. Все созданные ресурсы сортируются по имени. При двойном клике, на каком либо файле открывается программа, для просмотра этого файла ассоциированная с ним в вашей системе, или же файл открывается в редакторе, если это сцена. После добавления игрового объекта (3D модели, префаба, эффекта частиц, камеры) в сцену она появляется в списке игровых объектов.

Список игровых объектов или же иерархия – это некая структура из объектов. Тут можно назначить один объект дочерним у другого путем перетаскивания этого объекта в другой, а так же можно выбрать объект для редактирования его в инспекторе.

При выборе игрового объекта в обозревателе, списке объектов, или в сцене, в инспекторе появляются его характеристики, а так же список переменных в назначенном на него скрипте. Все применённые к игровому объекту компоненты так же отображаются в инспекторе. Главная особенность инспектора заключается в том, что все переменные кроме числовых и строковых назначаются из обозревателя ресурсов, или из списка игровых объектов путем перетаскивания нужного типа ресурса, или игрового объекта на строку с переменной в инспекторе. Еще в инспекторе можно изменять позицию и поворот игрового объекта в сцене, назначать теги и слои, активировать и деактивировать объект.

Игру можно тестировать в редакторе без необходимости ее компиляции. Для этого нужно всего лишь нажать на соответствующую кнопку в верхней панели. А так же игру можно поставить на паузу, чтобы внести какие либо поправки. В левой части верхней панели находятся кнопки изменения перемещения по сцене, поворота и масштабирования игровых объектов. В правой же части можно выбрать слой для отображения его в редакторе и режим расположение окон.

В редакторе сцен строятся игровые уровни. Уровни состоят из игровых объектов, поведение которых программируется на любом из трёх языков программирования – javascript, C# и Boo. Любой объект, будь то 3D модель, эффект частиц, префаб, или камера считается игровым объектом и отображается в списке игровых

объектов. Чтобы назначить скрипт игровому объекту, который находится в сцене, достаточно перетащить его на этот объект, что очень удобно. Положение объектов в сцене легко меняется. Для этого нужно просто перетаскивать их по трём координатным осям в то место, куда вы хотите. Материалы объектам назначаются из обозревателя ресурсов, которые так же перетаскиваются на объект. В верхней части окна сцены можно изменить режим отображения содержимого сцены. Список режимов: textured, wireframe, tex – wire, render paths, и light map resolution. Перемещение по сцене осуществляется клавишами W, A, S, D с зажатой правой кнопкой мыши.

В консоль поступают все сообщения из создаваемой вами игры. Все ошибки, предупреждения и уведомления выводятся в консоли. При ошибке указывается ее причина, и строка где она допущена.

В меню File можно создать, открыть и сохранить сцену или проект, а так же скомпилировать игру. Меню Edit более насыщено, чем все остальные. Основные его пункты – это настройки управления, графики, физики, звука и еще много чего. Меню Game Object позволяет создавать различного типа игровые объекты и управлять ими. В меню Component можно добавлять игровому объекту компоненты, например скрипты, или источники звука. В меню Terrain можно создать и управлять ландшафтом. В меню Window можно управлять всеми окнами редактора. В меню Help находится справочник по скриптам, мануал и различного рода помощь.

Компиляция игры выполняется довольно просто, из пункта меню File-Build settings. В настройках компиляции вы указываете, под какую платформу нужно строить проект, какие сцены будут компилироваться и общие настройки проекта. В общих настройках проекта можно сменить иконку готовой игры, её название, автора проекта и еще много чего.

Графика в Unity3D на достаточно высоком, современном уровне. Довольно качественные тени и шейдеры. Благодаря полностью настраиваемой графике, игры, созданные на Unity3D, запускаются даже на старых компьютерах. Однако разработчикам нужно будет попотеть над оптимизацией, так как большое количество

высоко полигональных моделей и большой ландшафт, заселённый растительностью, сильно сказывается на производительности.

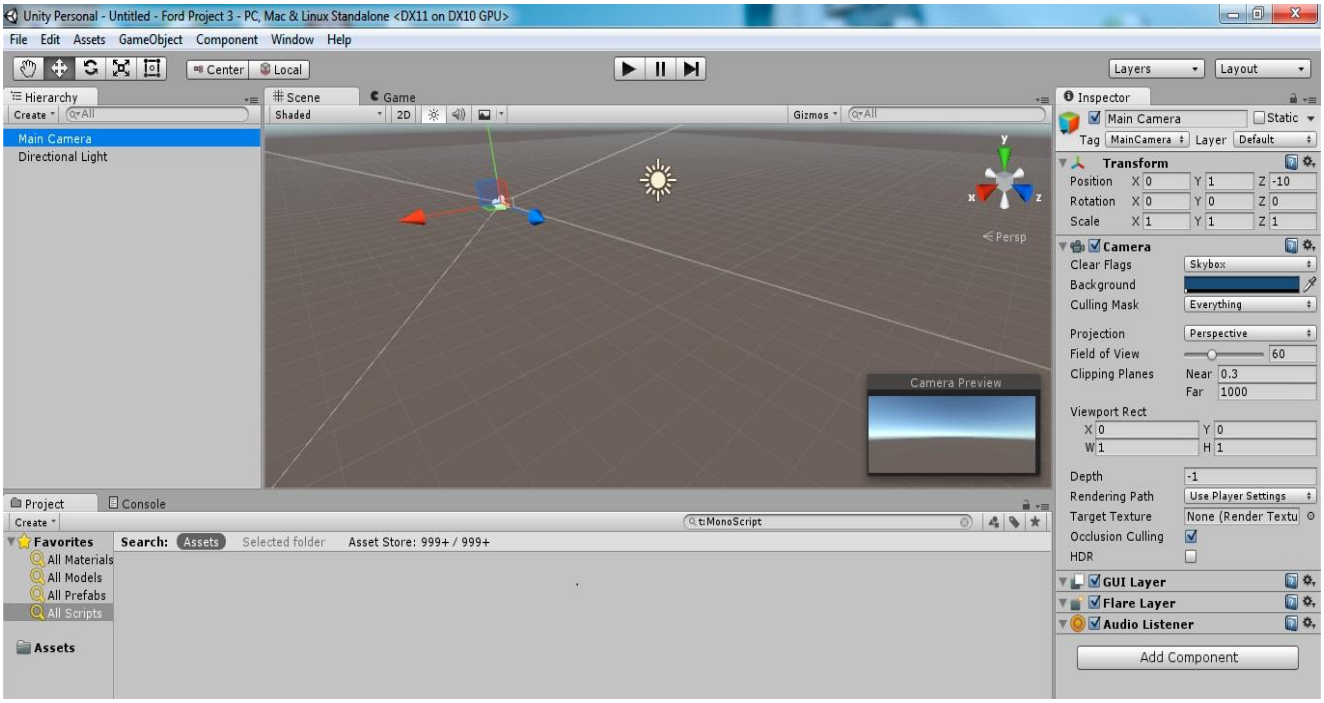

Рисунок 11 – Интерфейс графического визуализатора

Возможность переноса программного продукта на множество современных платформ является одним из важнейших достоинств данного программного продукта.

На основе проведенного обзора 3D движков, в качестве визуализатора был выбран Unity3D, из-за бесплатного его распространения и возможности производства программного обеспечения на любые платформы.

2.2.3 Обоснование выбора программы для 3D моделирования

Рассмотрим основные разновидности программ для 3D моделирования.

3Ds MAX – популярный программный пакет, предназначенный для редактирования трёхмерной графики и ее визуализации. Интерфейс программы изображен на рисунке 12. Утилита великолепно подходит для создания простых и сложнейших структурированных трехмерных объектов – животных, людей, зданий. Программа также позволяет выполнять глубокое моделирование природной среды, включая освещение, воду, деревья, ветер. 3Ds MAX – настоящий лидер среди инструментов, которые используются в дизайне интерьеров и архитектуре.

В приложение интегрирован мощный модуль анимации, предоставляющий огромные возможности касательно управления параметрами анимированного изображения. Создаваемый при этом видеоряд обладает высокой реалистичностью. Графическая среда большинства компьютерных игр создана средствами 3Ds MAX. Кроме этого, данный пакет широко используется в теле- и киноиндустрии.

Потребность в применении утилиты появляется тогда, когда необходимо получить изображение одной и той же сцены или предмета в разных проекциях. Стоит отметить, что прорисовка какой-либо сцены в 2D-редакторе займет меньше времени. Однако создав проект в 3Ds MAX, пользователь получает возможность генерировать сцену в неограниченном числе проекций.

Немаловажен также тот факт, что процесс наложения теней и света в 3Ds MAX осуществляется автоматически (главное правильно настроить источники света), в то время как в 2D-редакторах данная операция возлагается на пользователей. Работа в программе осуществляется в четыре этапа:

– моделирование – создание каркаса, структуры объектов, подлежащих визуализации, их математических моделей;

– текстурирование – формирование текстуры, основных визуальных характеристик для объектов;

– постановка света – отдельная и трудоемкая задача, для решения которой 3Ds MAX предлагает широкий ассортимент разных типов источника света и немалые возможности по их настройке;

– рендеринг – получение конечного результата – растрового изображения.

Математическая модель, созданная на предыдущих этапах, трансформируется в изображение, когда идет речь об анимации – в набор изображений.

Поддерживаются следующие форматы файлов: 3Ds MAX, lwo, jpg, png.

Визуализация трехмерной сцены в 3Ds MAX может осуществляться разными модулями рендеринга, предназначенными непосредственно для 3D-редакторов. Большой популярностью пользуется VRay – внешний визуализатор, характеризирующийся более реалистичными изображениями и огромным числом настроек, сравнительно со встроенным в 3Ds MAX визуализатором Scanline.

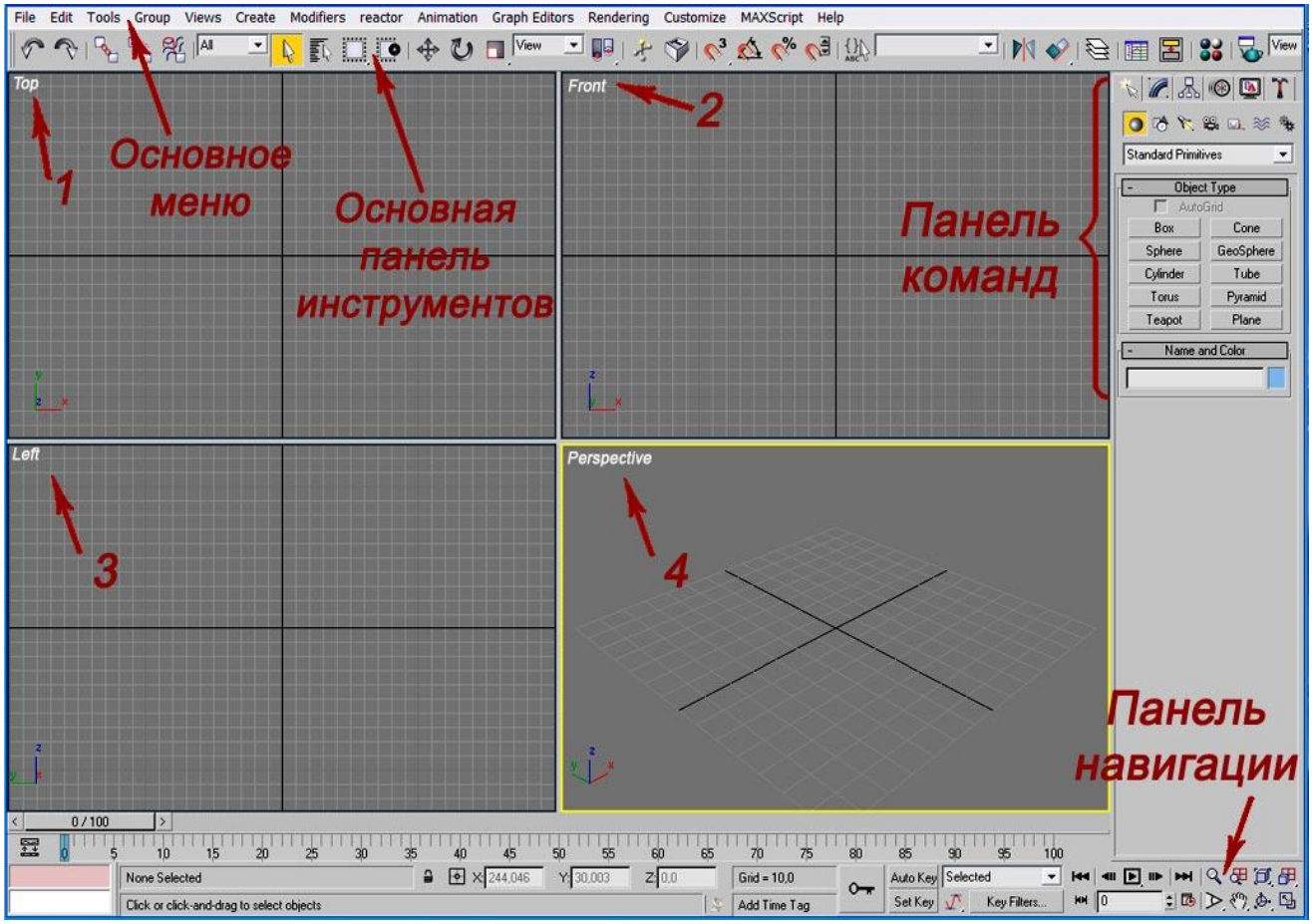

Рисунок 12 – Интерфейс программы 3Ds MAX

Blender – это полноценный бесплатный редактор. Интерфейс программы Blender изображен на рисунке 13. Необычный внешний вид приложения свидетельствует о том, что проект создавался разработчиками «с нуля», они не привязывались к внешнему виду иных подобных утилит.

Технология, реализованная в Blender, позволяет видоизменять интерфейс программы до неузнаваемости. Изюминка состоит в том, что во время создания 3-мерной сцены, окно утилиты можно разделить на части, каждая из них будет представлять собой независимое окно с определенным видом 3D сцены, настройками объекта, линейкой временной шкалы.

Количество таких частей ограничивается лишь разрешением экрана. Независимо от их числа, они никогда не будут между собой пересекаться и накладываться друг на друга.

Еще одна положительная сторона Blender – великолепная поддержка «горя-

чих» клавиш. С их помощью можно выполнять любые операции. Разумеется, сочетаний много, поэтому тяжело запомнить их все сразу, но их знание существенно упрощает работу в программе.

Создание трёхмерных моделей выполняется посредством полигональных и NURBS-поверхностей. Приложение также располагает инструментами сплайнового моделирования, а для формирования 3D-объектов используются еще кривые Безье и B-сплайны.

Инструментарий утилиты настолько универсален, что посредством Blender можно воссоздавать составные органические формы. Для этого применяются метаболы и технология «трёхмерной лепки» при помощи виртуальных кистей. Изменения в геометрию вносятся в режиме симметрии, что крайне необходимо в процессе моделирования персонажей.

Программа пригодна для создания обычной анимации и работы над оснасткой персонажей. Приложение позволяет формировать скелет и привязывать кости к внешней оболочке. Работа редактора основывается на прямой и инверсной кинематике.

Помимо описанных выше инструментов, Blender обладает следующими возможностями:

– поддержка различных методик моделирования (скульптурной, геометрической, метасфер и шрифтов);

– встроенные механизмы рендеринга и функции редактирования видео;

– интегрированный модуль для создания своей логической части игр;

– файловая система, сохраняющая в одном файле несколько отрендеренных сцен;

– небольшой инсталляционный файл – около 50 Мбайт.

Blender – отличное приложение для 3D-моделирования и анимации. Без сомнений, у данной утилиты есть будущее, она стабильно совершенствуется и пополняется новыми функциями.

Помимо собственного формата BLEND, программа работает со следующими форматами файлов:

– 2D:

TGA, JPG, PNG, OpenEXR, DPX, Cineon, Radiance HDR, Iris, SGI Movie, IFF, AVI, GIF, TIFF, PSD, MOV, KML, KMZ.

– 3D:

3D Studio3ds, AC3D, COLLADA, FBX Export, DXF, Wavefront OBJ, DEC Object File Format, DirectX, Lightwave, MD2, Motion Capture, Nendo, OpenFlight, PLY, Pro Engineer, Radiosity, Raw Triangle, Softimage, STL, TrueSpace, VideoScape, VRML, VRML97, X3D Extensible 3D.

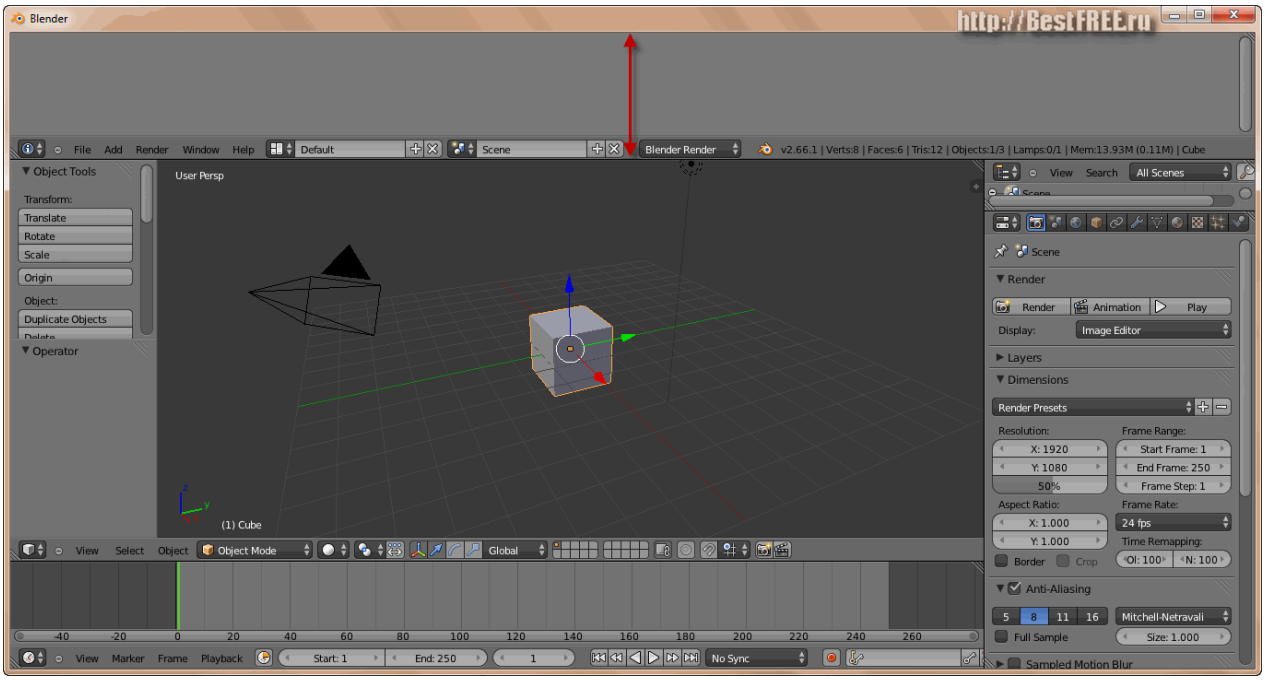

Рисунок 13 – Интерфейс программы Blender

ZBrush – это приложение, которое предназначено для 3D-моделирования. Она отлично подойдет для создания моделей любой сложности. В основном программу используют для моделирования цифровых скульптур путем организации частиц в целую структуру. Интерфейс программы Zbrush представлен на рисунке 14.

Скачав это приложение, пользователь получает отличное средство для высококачественного трехмерного моделирования. Программа востребована среди начинающих пользователей, поскольку обладает настраиваемым интерфейсом и структурированным набором функций.

Профессиональные программисты полюбили эту утилиту за возможность лепки 3D-объектов. Это похоже на создание глиняной скульптуры. Некоторые оскароносные фильмы создавались, используя это приложение.

Кроме того, можно отметить еще одну особенность программы – псевдотрехмерность. Она предоставляет возможность рисовать объемные модели на поверхности. Суть этого метода заключается в том, что в процессе работы можно задавать виртуальную точку высоты над поверхностью, вследствие чего штрихи получаются объемными.

Запустив приложение, на экране появляется множество красок и кистей. С первого взгляда может показаться, что это обычный графический редактор. Все функции приложения расположены внутри палитр, при этом каждая из них предназначена для выполнения определенных действий, связанных между собой.

Для выполнения операций, необходимо нажимать на округлые кнопки. Также есть возможность воспользоваться переключателями и бегунками, что очень удобно. Первые предназначены для включения определенных функций, а вторые позволяют устанавливать необходимое значение.

В процессе работы стоит обратить внимание на Note Bar. Он представляет собой полоску, которая размещена над палитрой и отображает действия пользователя, название раздела, подсказки и другое. Для начинающих пользователей предусмотрена функция автоматических подсказок. Чтобы получить справку по какому-либо пункту, необходимо зажать кнопку CTRL и переместить курсор на нужный раздел.

Приложение обладает неограниченными возможностями, которые лимитируются только производительностью компьютера. Чтобы использовать программу на профессиональном уровне, необходимо просмотреть видео уроки или специальные курсы по обучению.

Программа позволяет создавать модели объектов, используя множество уникальных функций. Кроме того, приложение располагает широким набором палитр и кистей. Также ZBrush обеспечивает высококачественную прорисовку деталей. Помимо этого, утилита обладает псевдо объёмным режимом построения моделей.

К тому же есть возможность использовать подключаемые модули.

Программа снабжена оригинальной функцией лепки, которая похожа на создание фигур из глины. Кроме того, интерфейс можно полностью переделать по своему усмотрению. Помимо этого, программа не требует большого количества ресурсов.

Из недостатков можно отметить применение одной технологии создания моделей. Кроме того, программа официально не переведена на русский язык. Помимо этого, начинающие пользователи потратят много времени на ознакомление с программой.

В общем, ZBrush отлично справляется с возложенными на нее задачами. Она не нуждается в большом количестве системных ресурсов и способна работать на компьютерах, на которых не смог запуститься Photoshop. Это приложение способно работать даже на 32-разрядных системах, чего не могут остальные подобные приложения.

Бесплатно пользоваться приложение можно на протяжении 45-ти дней, после чего необходимо приобрести дорогостоящую лицензию.

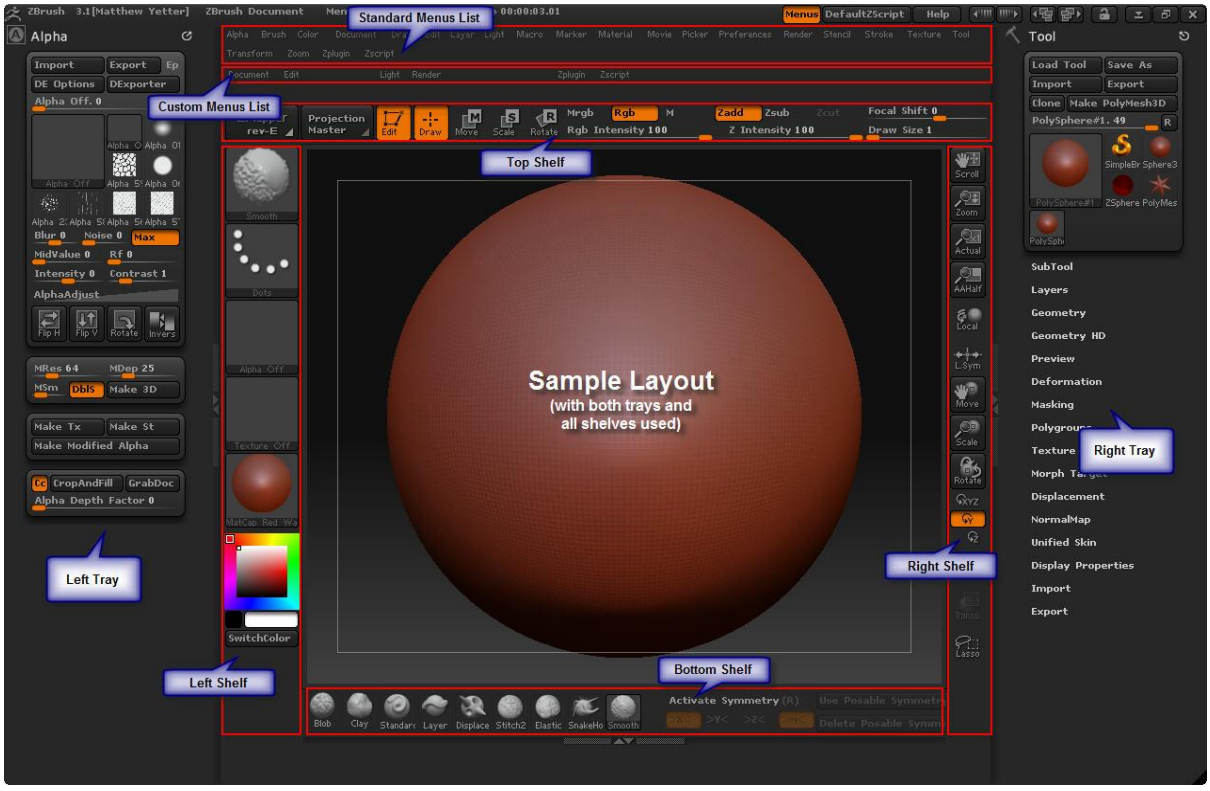

Рисунок 14 – Интерфейс программы ZBrush

Средой для 3D моделирования был выбран пакет для создания трёхмерной компьютерной графики 3Ds MAX, включающий в себя средства моделирования, анимации, рендеринга, постобработки и монтажа видео со звуком.

2.3 Техническое обеспечение

Так как разработанный программный продукт, включает в себя одну трехмерную модель с множеством под объектов, то и требования предъявляемые, к параметрам технического средства будут соответствовать надлежащему уровню для полноценной визуализации 3D сцены.

Для успешного выполнения программы необходимо чтобы используемая конфигурация компьютера удовлетворяла минимальному составу аппаратных средств:

1) двух ядерный процессор с частотой не ниже 1200 МГц;

2) объем оперативной памяти не меньше 2 Гбайт;

3) объем свободной внешней памяти 50 Мбайт;

4) разрешение монитора 640x480;

5) объём видеопамяти не меньше 512 Мбайт.

Также для успешного выполнения программы необходимо чтобы используемая конфигурация компьютера удовлетворяла минимальному составу программных средств: операционная система Windows XP\7\8 и новее, Microsoft NET Framework 4.

Так как программа может быть портированна и на смартфоны, то и для них существуют минимальные системные требования. Для работы с обучающей системой на смартфоне достаточно 50 Мбайт свободного места и наличие поддержки операционной системы Android 2.4.

2.4 Эргономическое обеспечение

Созданный программный продукт обладает всеми необходимыми характеристиками, которые важны полноценной работы пользователя, такие как:

– эффективность: с помощью данного программного продукта пользователь может достичь целей и решить задачи, которые перед ним стоят, с минимальными трудозатратами;

– продуктивность, результативность;

– субъективная удовлетворенность трудом пользователя: отсутствие раздражения, недовольства, негативных эмоций, фрустрации.

Для использования системы пользователь должен обладать базовыми знаниями работы с системой Windows.

Программный продукт обладает возможностью переноса программного продукта на множество других платформ, а также на браузерные расширения.

Интерфейс системы разработан таким образом, что при взаимодействии с ним пользователь ориентируется на интуитивном уровне, это возможность позволяет расширить аудиторию потребителей.

Автоматизированные рабочие станции должны быть расположены в соответствии всем принятым нормам, чтобы вредоносные воздействия со стороны техники наименьшим образом сказывалось на здоровье пользователя.

3 ОПИСАНИЕ АЛГОРИТМИЧЕСКИХ И КОДОВЫХ ПРОЦЕДУР РЕАЛИЗАЦИИ ПП

3.1 Создание 3D модели автомобиля и систем

Построение трехмерной модели легкового автомобиля осуществлялось в программе Autodesk 3Ds MAX Design. На рисунке 15 представлена интерактивная модель грузового автомобиля, на примере некоторых деталей которой мы рассмотрим основные приемы 3D моделирования, которые были использованы в ходе работы.

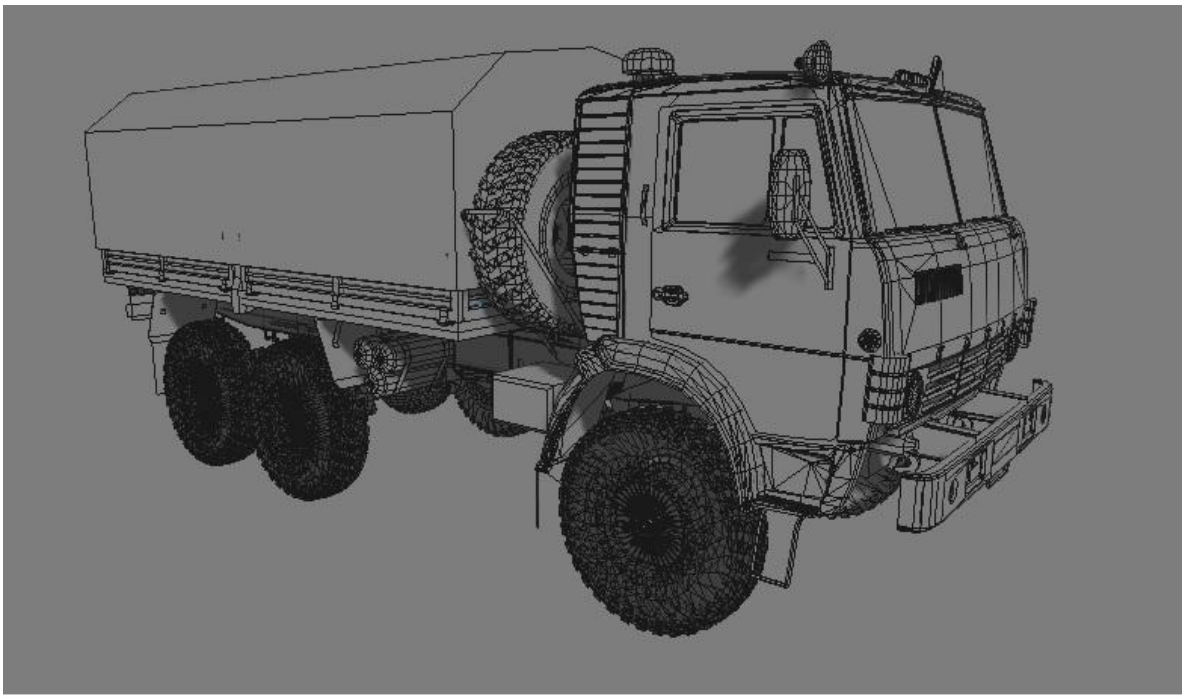

Рисунок 15 – Модель грузового автомобиля

Мы создавали модель, используя данный набор инструментов:

1) Экструдирование;

2) Модификатор Subdivisionsurface (Subsurf);

3) Модификатор Bevel;

4) Модификатор Boolean.

В наборе любой среды 3D-моделирования имеется ограниченный набор объектов-шаблонов. Например, в Autodesk 3Ds MAX Design есть куб, сфера, цилиндр, конус и даже чайник. Да можно сказать, вообще ничего нет, кроме ограниченной

кучки примитивов. Сложные объекты создаются различными способами, одним из которых является изменение mesh-объектов. В свою очередь, для изменения mesh-объектов предусмотрено множество инструментов, одним из которых является инструмент Extrude.

Инструмент Extrude (в переводе с англ. – выдавливать, выпячивать и т.п.) позволяет изменять mesh-объекты в режиме редактирования за счет создания копий вершин, рёбер и граней и их последующего перемещения, а также изменения размеров (если это ребра или грани).

Subdivisionsurface, он же Subsurf – без сомнения, самый часто используемый модификатор, он позволяет строить сложные сглаженные поверхности с помощью относительно простой геометрии.

Модификатор Bevel (скос, фаска) добавляет возможность скосить ребра меш объекта. Также модификатор позволяет определить место и конфигурацию скоса (фаски) ребра меш-объекта.

Модификатор Boolean служит для объединения двух объектов в один. Он может вычитать их друг из друга, складывать и учитывать только разницу любых меш-объектов. Модификатор Boolean полезный и практичный инструмент Autodesk 3ds max design.

В процессе могут участвовать практически любые объекты. Главное чтобы они были меш-объектами. Для того чтобы конвертировать например текст или кривую «бизье» достаточно нажать Alt+C и выбрать соответствующий пункт.

Рассмотрим, как применяются эти инструменты на примере создания изогнутой трубы:

Для начала создадим окружность, нажатием кнопки circle в разделе addprimitive. На рисунке 16 представлена окружность.

Перейдя в режим редактирования нажатием клавиши «Tab», выделим окружность и воспользуемся горячей клавишей «E» для включения инструмента «extrude», а затем, используя горячую клавишу «S» вызовем функцию «scale», для изменения масштаба полученных граней и нажмем ЛКМ для прекращения редактирования масштаба полученных граней. На рисунке 16 представлена окружность.

Выделим часть полигонов правой кнопкой мыши и удалим клавишей «Delete». Результат этого действия представлен на рисунке 17.

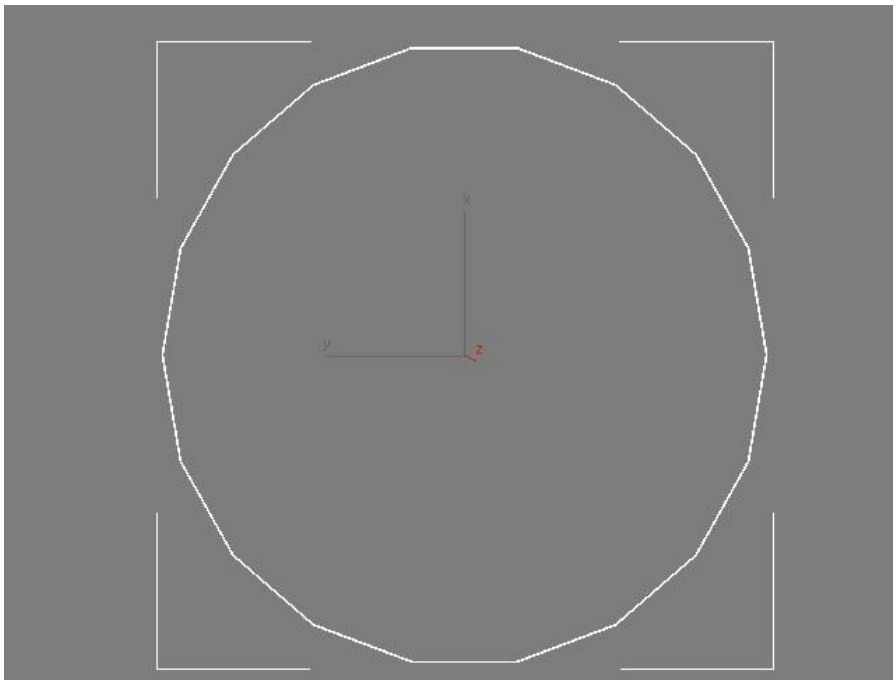

Рисунок 16 – Окружность

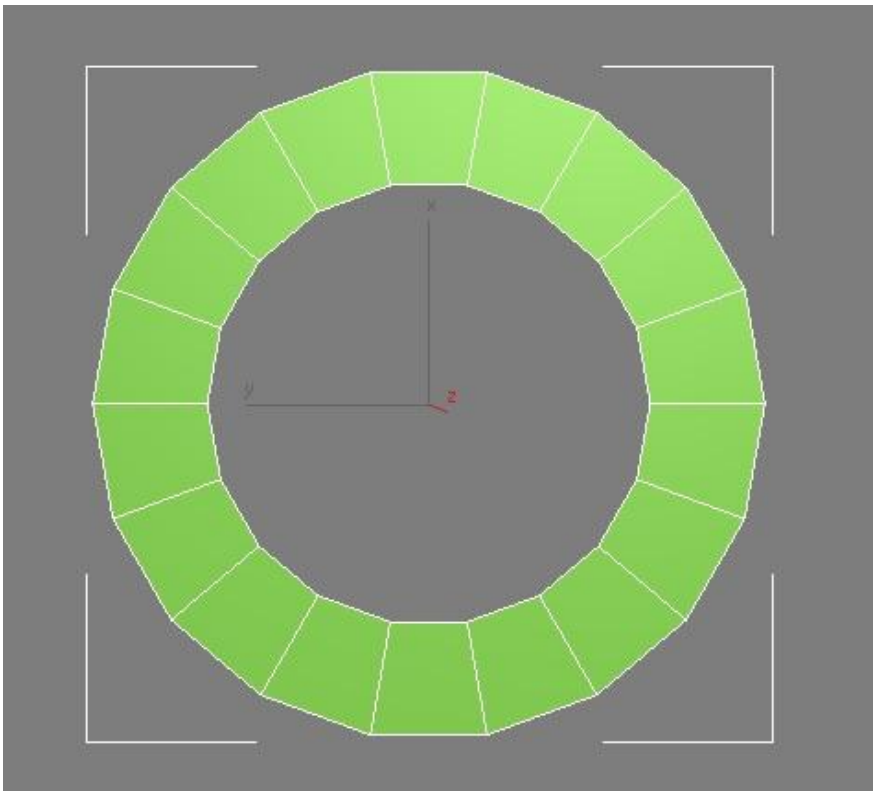

Рисунок 16 – Окружность с расширенными гранями

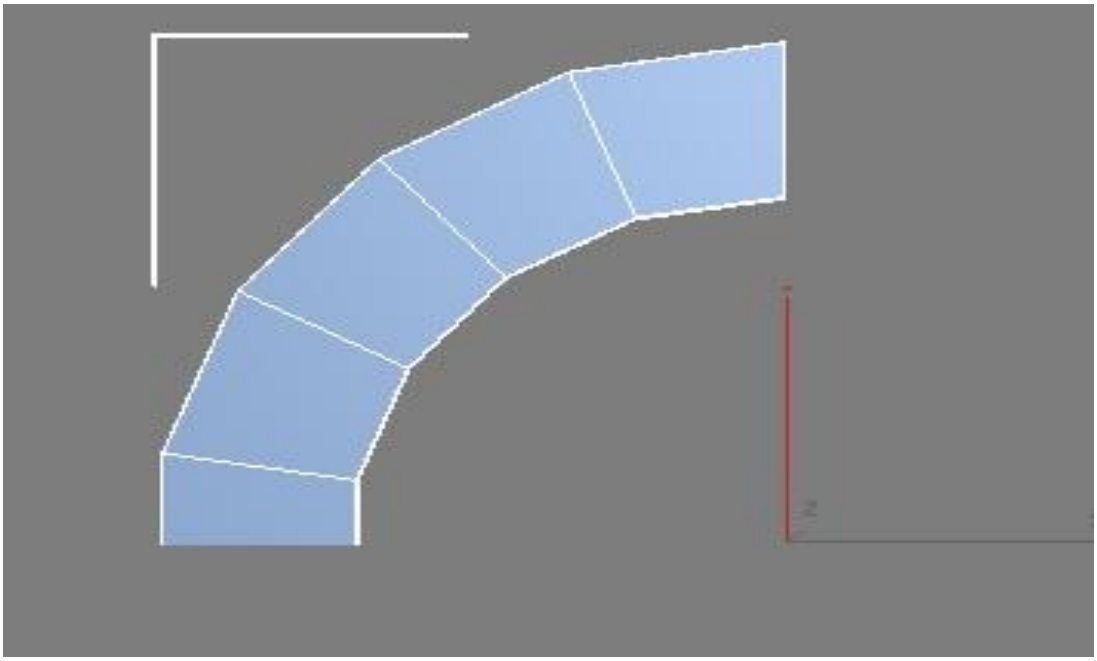

Рисунок 17 – Оставшиеся полигоны

Выделим оставшиеся полигоны и вновь воспользуемся инструментом «extrude» и создадим некое подобие «квадратной» трубы, представленную на рисунке 18.

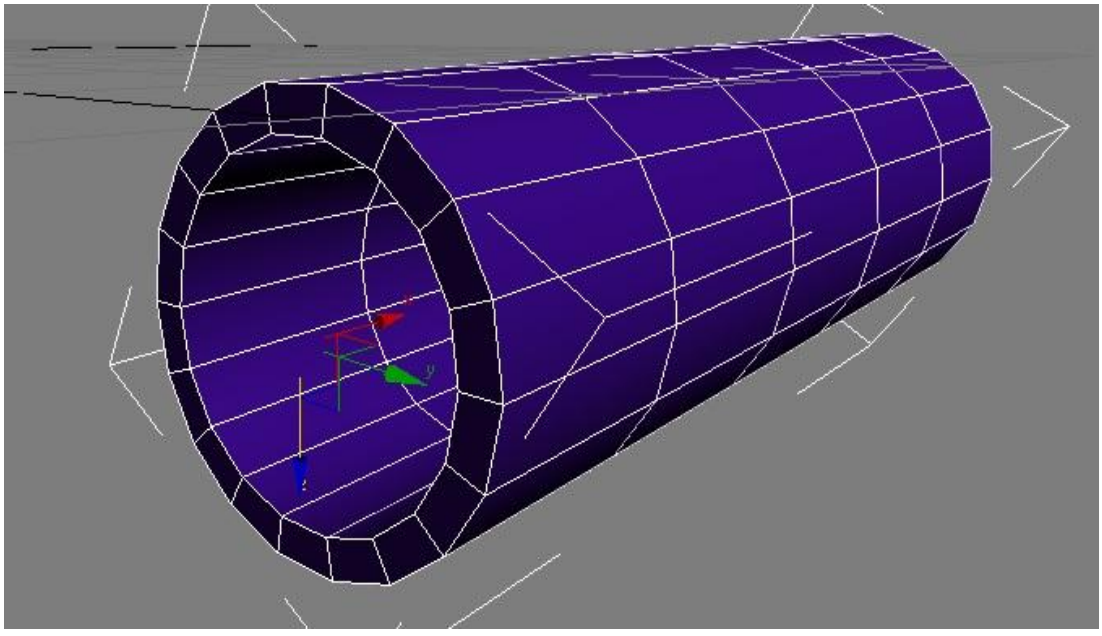

Рисунок 18 – Некое подобие квадратной трубы

Используя модификатор «Subdivisionsurface» сделаем трубу округлой.

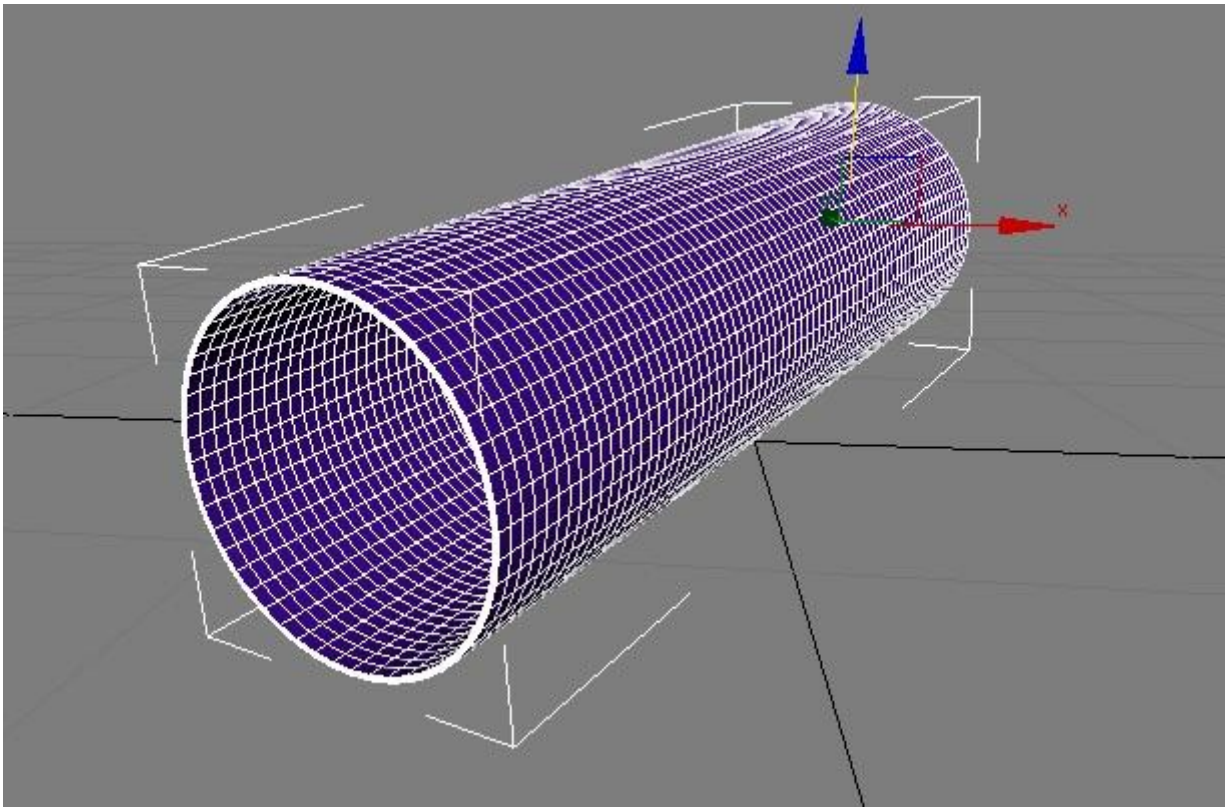

Рисунок 19 – Труба

С использованием этого комплекса методов и средств создавалась 3D модель грузового автомобиля.

3.2 Создание UV развертки

UV-преобразование или развёртка в трёхмерной графике (англ. UV map) – соответствие между координатами на поверхности трёхмерного объекта (X, Y, Z) и координатами на текстуре (U, V). Значения U и V обычно изменяются от 0 до 1. Развёртка может строиться как вручную, так и автоматически – например, в Autodesk 3Ds MAX Design есть несколько алгоритмов автоматического развертывания модели.

Современное трёхмерное аппаратное обеспечение считает, что UV-преобразование в пределах одного треугольника является аффинным – поэтому достаточно задать U и V для каждой вершины каждого из треугольников. Впрочем, как именно стыковать треугольники друг с другом, выбирает 3D-моделер, и умение строить удачную развёртку – один из показателей его класса. Существует несколько противоречащих друг другу показателей качества развёртки:

1) Максимально полное использование площади текстуры. Впрочем, в зависимости от разрыва между «минимальными» и «максимальными» системными требованиями, по краям развёртки текстуре нужен определённый «припуск» на генерацию текстур меньшего размера.

2) Отсутствие областей с недостаточной или избыточной детализацией текстуры.

3) Отсутствие областей с излишними геометрическими искажениями.

4) Сходство со стандартными ракурсами, с которых обычно рисуется или фотографируется объект – упрощает работу художника по текстурам.

5) Для частично симметричных объектов: удачное сочетание симметричных и асимметричных участков развёртки. Симметрия повышает детализацию текстуры и упрощает работу художника по текстурам; асимметричные детали «оживляют» объект.

Так как 3D модель грузового автомобиля задумывалась и исполнялась низко полигональной, то и UV развертка будет создана упрощенная, для экономии ресурсов рабочей станции. Исходя из этого, большая часть рабочего пространства 3D моделей покрашена однотонно, лишь технически сложные элементы имеют высокую детализацию. Рассмотрим порядок создания UV развертки на примере 3D модели колеса автомобиля. Выделим по контуру полигоны колеса, которые будут высоко детализированы и воспользуемся функцией «Markseam»(от англ. создать шов). Нажмем на горячую клавишу «U» для открытия панели UV mapping и выберем Unwrap для создания UV развертки. Перейдем в режим UV editor и нажмем на нижней панели export layout для сохранения развертки в файл. На рисунке 20 представлена UV развертка и 3D модель с нанесенными швами. В следующем разделе будет представлено, как из UV развертки создается текстура.

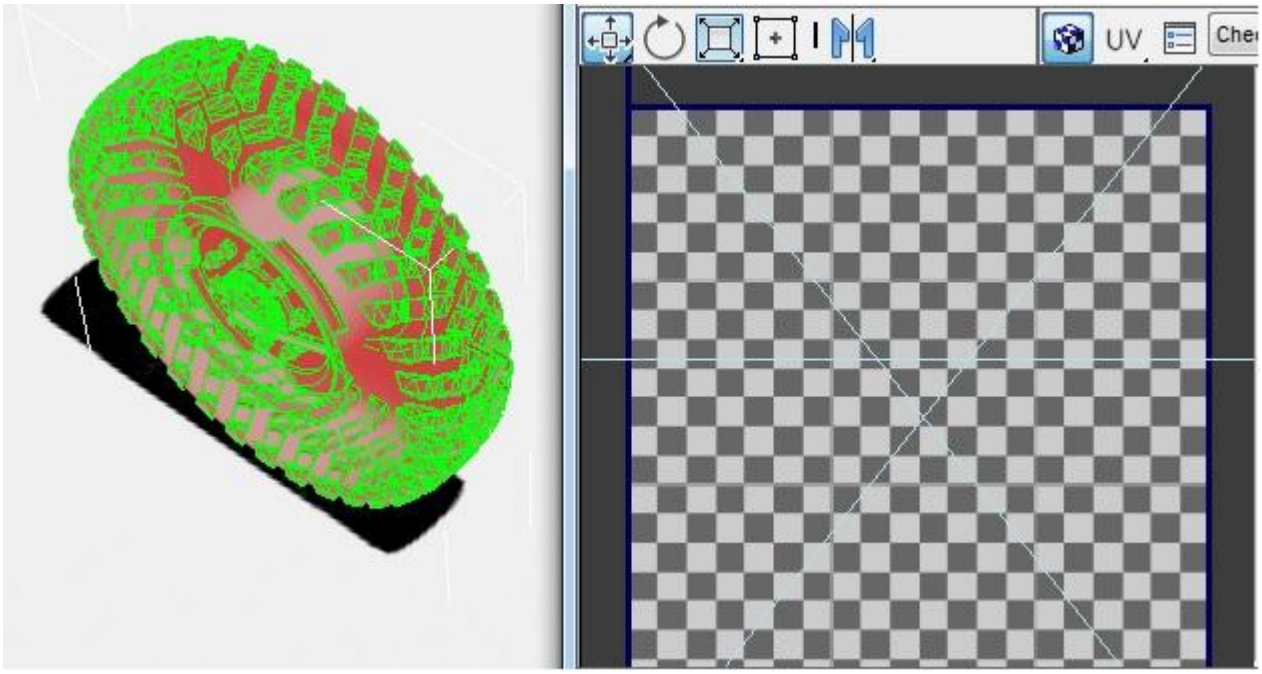

Рисунок 20 – UV развертка и 3D модель с нанесенными швами

3.3 Создание мульти текстуры из UV развертки

Мульти текстура создавалась при помощи программы Adobe Photoshop CS4 это – многофункциональный графический редактор, разработанный и распространяемый фирмой Adobe Systems. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты. Продукт является лидером рынка в области коммерческих средств редактирования растровых изображений и наиболее известным продуктом фирмы Adobe. Часто эту программу называют просто Photoshop. В настоящее время Photoshop доступен на платформах OS X, Windows, в мобильных системах iOS, Windows Phone и Android. Также существует версия Photoshop Express для Windows Phone 8 и 8.1.Сейчас в США идёт бета-тестирование стриминговой версии продукта для Chrome OS. Ранние версии редактора были портированы под SGI IRIX, но официальная поддержка была прекращена, начиная с третьей версии продукта. Для версий 8.0 и CS6 возможен запуск под Linux с помощью альтернативы Windows API – Wine.

Для начала создания мульти текстуры, загрузим сохраненную UV развертку в Photoshop. Затем создадим новый слой и на нем закрасим те места, где находятся грани с развертки, теми цветами, которыми хотим покрасить грани 3D модели. На

рисунке 21 представлена UV развёртка и мульти текстура.

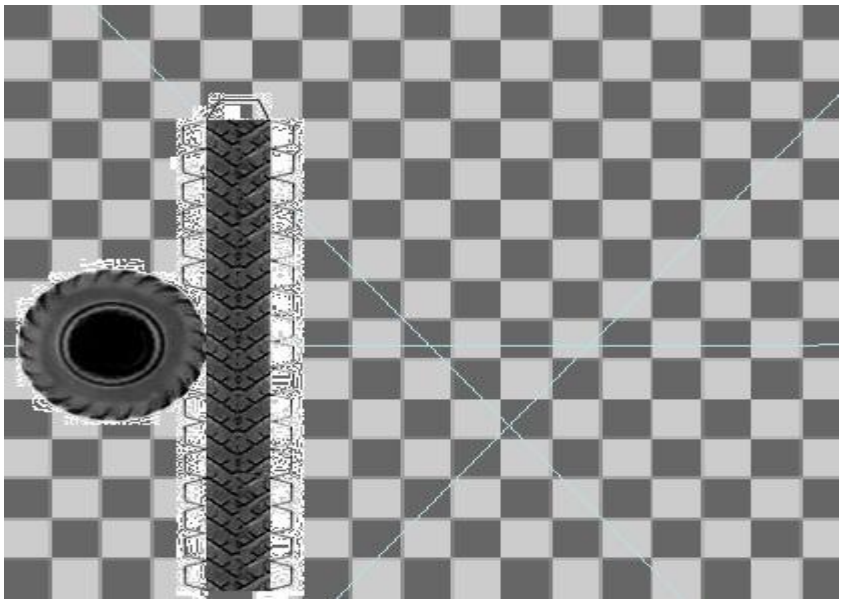

Рисунок 21 – UV развёртка и мульти текстура

3.4 Создание пользовательского интерфейса

На данном этапе будет рассмотрено создание пользовательского интерфейса. Пользовательский интерфейс создавался как интуитивно понятный любому пользователю знакомому с операционной системой Microsoft Windows. Для это экспортируем созданную нами в Autodesk 3ds max модель грузового автомобиля в графический визуализатор Unity 3D. Благодаря встроенной в Unity3D технологии пользовательских интерфейсов. Рассмотрим основы системы пользовательских интерфейсов. Мы начнем со знакомства с Canvas, внутри которого располагаются все элементы пользовательского интерфейса, Rect Tool и Rect Transform, которые используются для верстки элементов. Затем мы рассмотрим визуальные элементы, такие как Text и Image, а также элементы управления: Button и Slider.

Для начала создадим новый проект. Создадим участок для размещения грузового автомобиля (Terrain), зададим ему положение и размер. Добавим к созданной сцене источник освещения (Directional Light). При создании рельефа земли, предадим нашему участку форму плоскости.

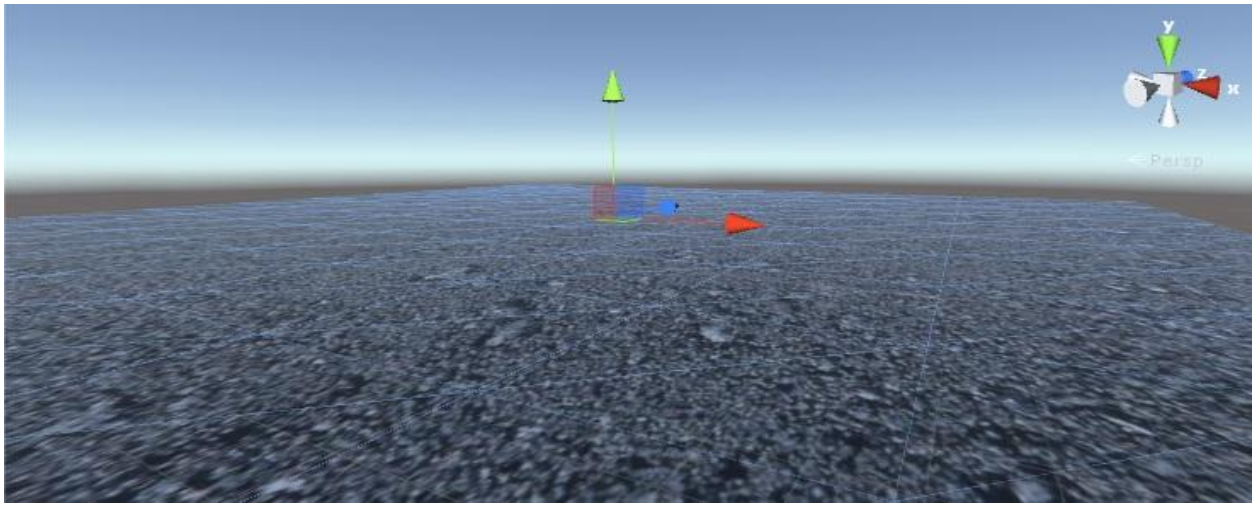

Рисунок – 22 Участок для размещения грузового автомобиля

При помощи компонента «Paint Texture» зададим одномерную текстуру цвета «асфальт» всему участку представленную на рисунке 22. Добавим в наш проект небо, данные компоненты являются стандартными и входят в состав условно – бесплатной версии Unity3D.

В проект были добавлена и размещена созданная в приложении «3D Studio Max» модель грузового автомобиля КамАЗ-4310.

В данном проекте были использованы учебные наработки, сделанные по дисциплине «Основы трехмерного моделирования в Unity3D».

Теперь добавим модель грузового автомобиля КамАЗ-4310 для возможности взаимодействия пользователя с игровым пространством.

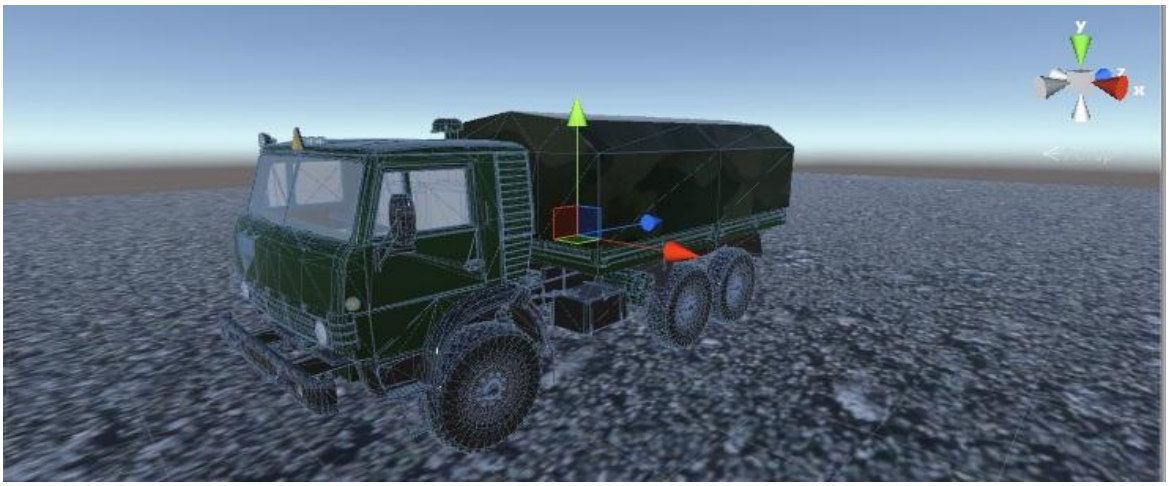

Рисунок 23 – Размещение модели грузового автомобиля в Unity3D

Для создания меню, с помощью которого возможен выбор необходимого элемента грузового автомобиля на переднем фоне данной сцены используем компонент «Canvas» на который добавим «кнопки» «Button», а также «Panel». Меню для выбора элементов грузового автомобиля представлено на рисунке 24.

Рисунок 24 – Меню для выбора элементов грузового автомобиля в Unity3D

Дадим пользователю возможность «управлять» грузовым автомобилем. Для этого установим на грузовой автомобиль необходимые скрипты для вращения «Camera Rotate» и возможности управления расстоянием до объекта «Zoom».

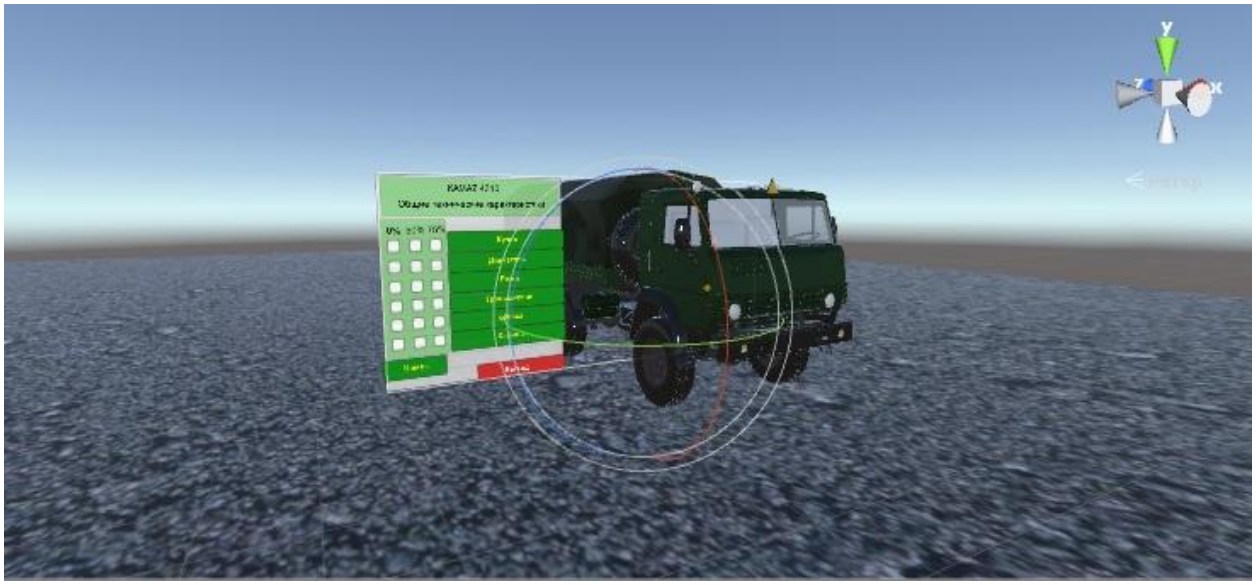

Рисунок 25 – Настройка вращением и управление расстоянием до объекта

Для реализации выбора нужного нам элемента грузового автомобиля используем настройки кнопок «Button» с которой с помощью компонента «Set active» привязываем нужный нам элемент. Демонстрация выбора элемента грузового автомобиля представлена на рисунке 26.

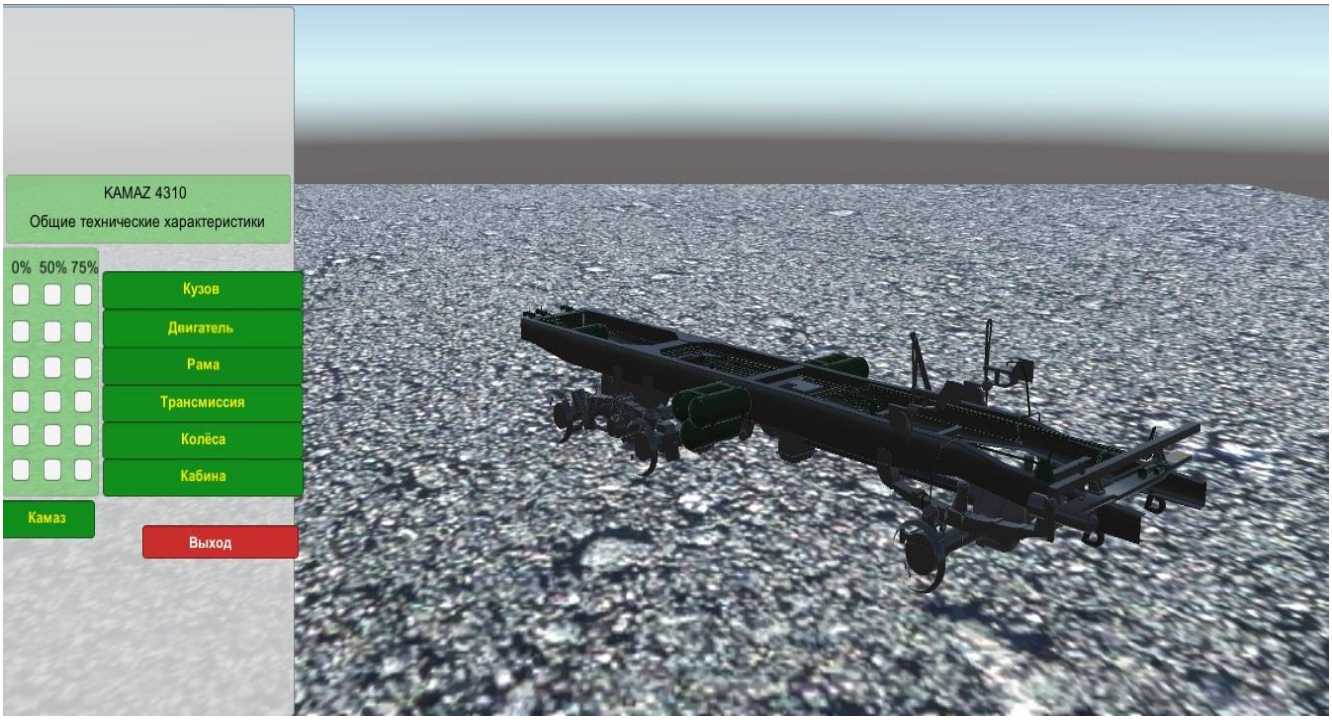

Рисунок 26 – Демонстрация выбора элемента грузового автомобиля

В результате исследования была продемонстрирована реализация проекта по созданию интерактивного приложения для простоты понимания устройства технических систем с возможностью просмотра отдельных элементов технического устройства грузового автомобиля КамАЗ-4310 которое может использоваться на бортовом компьютере автомобиля.

4 ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ПРОДУКТА

4.1 Актуальность программы

На сегодняшний день практически не существует программных продуктов, которые могли бы интерактивно показать устройства технических систем грузового автомобиля, в связи в этом разработан продукт который является кроссплатформенным, т.е. программа может быть установлена на бортовой компьютер автомобиля, наглядно показывает устройство грузового автомобиля и его технических систем.

Кроссплатформенный подход к разработке имеет следующие положительные моменты:

1) Требуется меньше ресурсов для реализации приложения сразу под несколько платформ. В этом собственно и суть кроссплатформенного подхода – один и тот же код работает и на iOS, и на Android. Программистов, занимающихся проектом, нужно ровно в два раза меньше. Дизайнер делает только один набор графики. Все это снижает количество рабочих часов и как следствие бюджет проекта.

2) Меньшее время на разработку. За счет отсутствия уникальных элементов интерфейса и более простых технологий время на создание простых продуктов, как правило, меньше.

3) Упрощенный цикл обновления продукта. Если в проект нужно что-то добавить или исправить какую-то ошибку, это делается сразу для всех платформ, на которых распространяется проект.

4) Возможность использования мобильной версии сайта. Большинство кроссплатформенных решений используют семейство JavaScript языков, поэтому если у вас уже есть мобильная версия сайта, значительная часть кода и материалов может быть использована в приложении без изменений.

5) Использование единой логики приложения. Заложенная логика в работу приложения будет работать гарантированно одинаково для всех платформ. Довольно часто это может являться и минусом из-за разной архитектуры операцион-

ных систем (яркий пример – кнопка. Назад в навигации между экранами: в Android предусмотрена аппаратная кнопка Back для этих целей, в iOS – движение пальцем от левой части экрана или же наличие кнопки в левой части навигационной панели; если кнопку не делать вовсе, пользователи iOS не смогут вернуться назад, если сделать, но не на том месте и выглядящую нестандартно – пользователям iOS будет непривычно и неудобно, если сделать как в iOS, будет непривычно пользователям Android), однако написанная и отлаженная один раз логика содержит потенциально меньшее количество ошибок и расхождений в своей работе, вам не придется проделывать двойную и тройную работу по поиску проблем на каждой платформе.

4.2 Характеристика программы

Программный продукт для компьютерного макетирование общего технического устройства КамАЗ-4310 разрабатывается на языке совместимом с языком системы Unity. Машинный язык совместим с ЭВМ на базе процессоров серии x86, x64. Сообщения и экранные формы должны иметь стандартный вид операционной системы Windows.

Для корректной работы приложения необходимо наличие операционной системы Windows XP/7/Vista/8/10, Linux, Android

Объём разработанного программного продукта составляет 50 Мбайт.

Аппаратное и программное обеспечение должно в первую очередь соответствовать минимальным требованиям, предъявляемым выбранной версией Unity. Минимальные требования:

- Intel совместимый процессор не ниже Pentium;
- операционная система Windows XP/7/Vista, Linux;
- минимально 128 Мбайт ОЗУ, рекомендуется 256 Мбайт;
- свободное пространство на жестком диске, в зависимости от комплекта установки, от 400 Мбайт;
- указатель «мышь».

Минимальные требования для смартфонов:

- минимально 128 Мбайт ОЗУ, рекомендуется 256 Мбайт;

− свободное пространство на съемном накопителе, в зависимости от комплекта установки, от 50 Мбайт;

− наличие поддержки операционной системы Android 2.4.

3.2 Руководство пользователя

Запустив программу, вниманию пользователя предстает следующее окно, изображенное на рисунке 27.

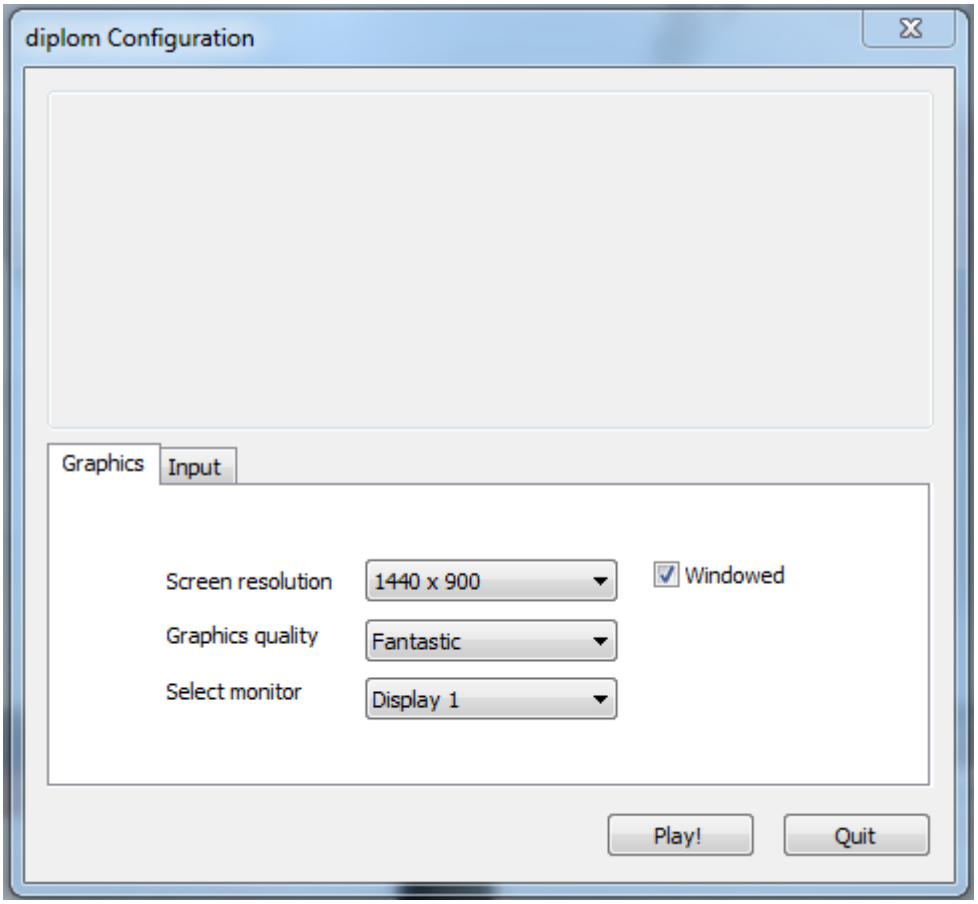

Рисунок 27 – Окно конфигурации приложения

Как мы видим, данное окно предоставляет нам возможность выбора необходимого разрешения, качества и установки запуска в на весь экран или в окне нашего предложения. Так мы можем запускать приложение на разных рабочих станциях в зависимости от мощности и требований пользователя. Для первого запуска рекомендуется запускать в режиме Fastest. Выставив нужные параметры, нажимаем кнопку Play после чего откроется непосредственно информационная система с загруженной 3D моделью грузового автомобиля, представлен-

ная на рисунке 28.

Теперь разберем принцип работы с программой. Управление приложением осуществляется с использованием лишь двух кнопок: левой кнопки мыши, для перемещения положения камеры в пространстве и колесо мыши для отдаления и приближения камеры. Для вращения камеры, достаточно удерживая левую кнопку мыши, передвигать её в каком либо направлении. Для масштабирования камеры достаточно вращать колесиком мыши.

Рисунок 28 – Рабочее окно программы

На рисунке 12 переключатели установлены на 100 % видимость всех деталей 3D модели. Если переустановить переключатель, допустим для кузова автомобиля на 50 % , то он станет полупрозрачным, а если установить на 0 % то полностью прозрачным. На рисунке 29 представлена 3D модель рамы автомобиля с прозрачностью 50 %.

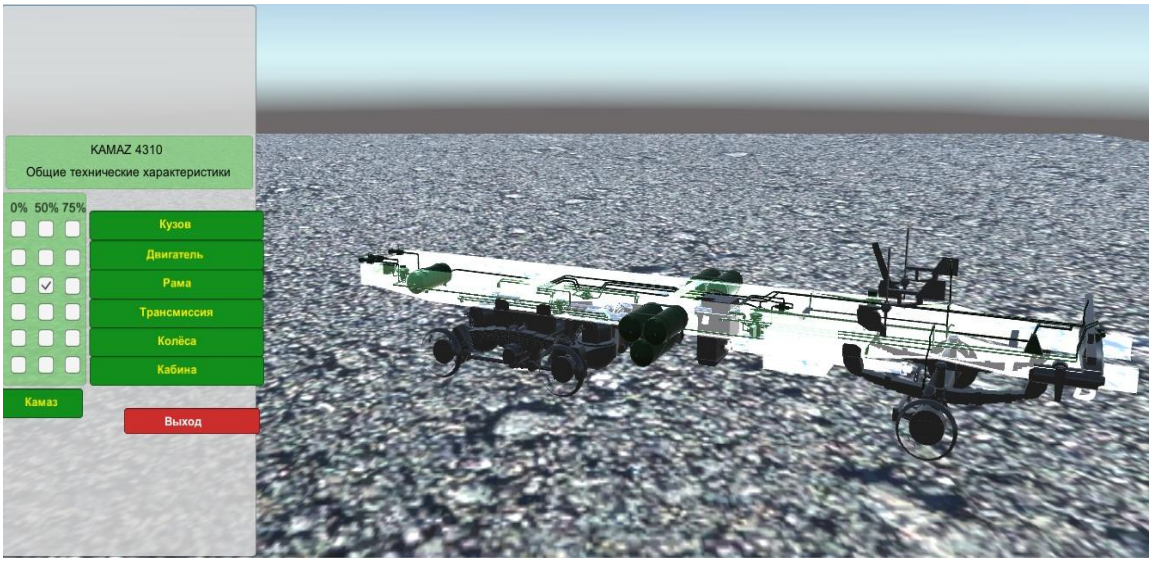

Рисунок 29 – 3D модель рамы грузового автомобиля, с несколькими прозрачными деталями

Далее если выберем поле «Рама» рама автомобиля вернется в исходное состояние, показано на рисунке 30.

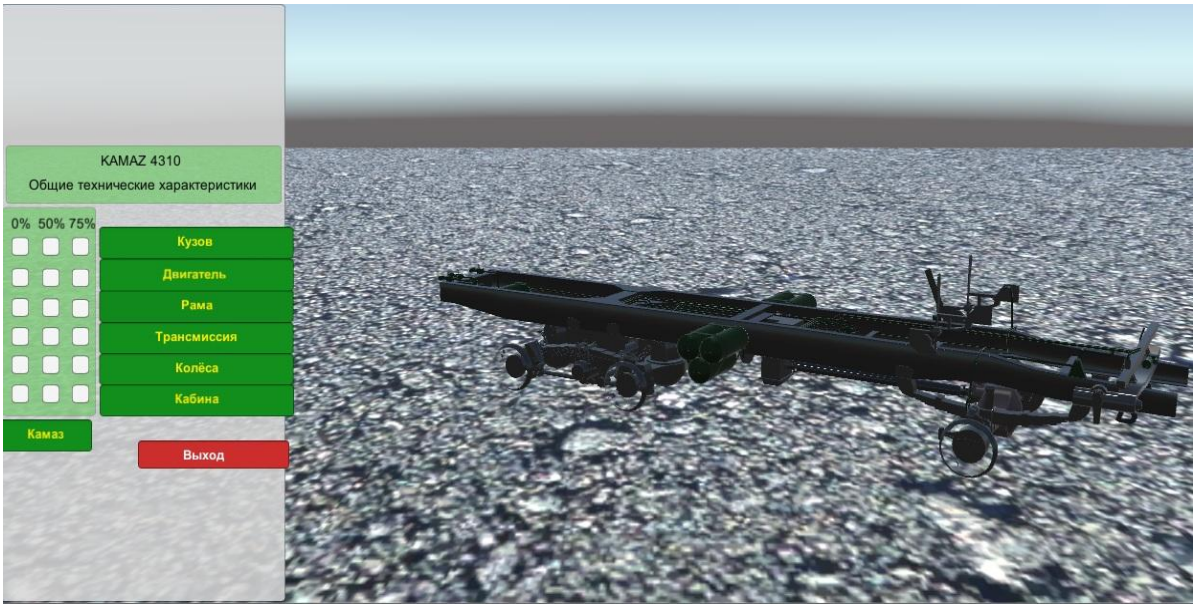

Рисунок 30 – 3D модель рамы грузового автомобиля

Правильно используя все инструменты предложенные программой можно разглядеть вблизи и под любым углом детали автомобиля, а также их крепления. Это позволяет достичь максимальной наглядности и простоты в познавательном процессе.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы, была изучена предметная область проводимого исследования, проанализированы существующие программные средства рассматриваемой задачи. Так же в ходе работы было представлено обоснование создания информационной системы по устройству технических систем грузового автомобиля. Было представлено техническое и эргономическое обеспечение. Создана наглядная, простая, распространяющаяся без каких либо ограничений программа, с возможностью быстрого и подробного ознакомления с устройством грузового автомобиля, а так же с возможностью установки на мобильные устройства, так как это информационная система является кроссплатформенной, что позволит использовать данную систему на бортовом компьютере грузового автомобиля, что является новинкой в настоящее время.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Баронов, В. В. Основы 3D моделирования / Г.Н. Калянов, Ю. Н. Попов, А.И. Рыбников, И. Н. Титовский – М.: Академия, 2009. – 334 с.

2 Атре, Ш. Современные методы 3D моделирования. - М.: «Моделирование и проектирование», 2003 – 336 с.

3 Благодатских, В.А. Разработка и использование программного обеспечения ЭВМ / М.А. Енгибарян, Е.В. Ковалевская – М.: Проектирование и программирование, 1997 – 210 с.

4 Вендров, А.М. CASE – технологии. Современные методы и средства проектирования информационных систем. – М.: Проектирование и программирование, 1998. – 176 с.

5 Гайкович, В. С. Основы работы с графическими редакторами / А. А. Першин – М.: «Единая Европа», 1994 – 150 с.

6 Титоренко, Г.А. Автоматизированные информационные технологии. Учебник. – М.: ЮНИТИ, 2001 – 200 с.

7 Зеленко, Л.С. Виртуальная дистанционная обучающая среда: особенности реализации и применения в учебном процессе / Д.А. Загуменнов. ученые записки ИСГЗ, 2013. – 305 с.

8 Зеленко, Л.С. Основы построения виртуальной информационно-образовательной среды / Д.А. Загуменнов, А.О. Зинченко // Вестник Самарского Государственного Аэрокосмического Университета им. академика С.П. Королева (Национального Исследовательского Университета), 2012. – 56 с.

9 Галкин, Н.С. Создание трехмерной модели местности в Unity3D / Е.А. Ромин // Инновационные технологии: теория, инструменты, практика, 2014. – 316 с.

10 Ходос, О.С. Обучение трехмерному моделированию в UNITY3D / Р.И. Баженов // Современные научные исследования и инновации, 2014. – 38 с.

11 Баженов, Р.И. Выявление общих критериев совершенствования предметной компетентности студентов «информационных специальностей» в рамках кур-

са по выбору «Трехмерное моделирование в среде Unity» / О.С. Ходос // Современные информационные технологии и ИТ-образование, 2011. – 259 с.

12 Ходос, О.С. Формирование предметной компетентности у студентов в контексте среды виртуальной реальности // Вестник бурятского государственного университета, 2014. – 86 с.

13 Ходос, О.С. Совершенствование предметной подготовки будущих учителей информатики в рамках дисциплины «компьютерное моделирование» / Р.И. Баженов // Актуальные вопросы методики преподавания математики и информатики. Биробиджан: Минобрнауки России, 2011. – 136 с.

14 Якобсон, М.Е. Использование мультиплатформенный среды Unity для визуализации учебных материалов / Г.М. Гринберг // Актуальные проблемы авиации и космонавтики, 2010. – 438 с.

15 Bergera, M. CFD Post-processing in Unity3D / V.Cristie // Procedia Computer Science, 2015. – 292 с.

16 Becker-Asanoa, C. Multi-agent System based on Unity 4 for Virtual Perception and Wayfinding / F. Ruzzolia, C. Holscherb, B. A. Nebel // Transportation Research Procedia, 2014. – 455 с.

17 Wang, J. Coordinated hybrid virtual environments: Seamless interaction contexts for effective virtual reality / R. Lindeman // Computers & Graphics, 2015. – 200 с.

18 Richards D., Taylor M. A Comparison of learning gains when using a 2D simulation tool versus a 3D virtual world: An experiment to find the right representation involving the Marginal Value Theorem // Computers & Education, 2015. – 175 с.

19 Ходос О.С. Основы трехмерного моделирования в Unity3D. Методические указания к выполнению лабораторных работ по направлениям / Баженов. Р.И. – Биробиджан: Изд-во ПГУ им Шолом-Алейхема, 2013. – 101 с.

20 Простой искусственный интеллект // Блоги Unity3D по-русски! URL: http://blogs.Unity3D.ru/2010/09/simple-ai (дата обращения: 26.05.2015).

21 Верстак, В.И. 3dsMax 9. Секреты мастерства / В. И. Верстак, О. И. Нессонов. – СПб.: Питер, 2009. – 320 с.

22 Кудрявцев, Е.М. Компас-3D.Основы работы в системе / Е. М. Кудрявцев.

– М.: ДМК Пресс, 2008. – 528 с.

23 Черепашков, А.А. Компьютерные технологии, моделирование и автоматизированные системы в машиностроении / А. А. Черепашков, Н. В. Носов. – Волгоград : Ин-Фолио, 2009. – 640 с.

24 Чепак, Л. В. Методические указания к выполнению курсовой работы по дисциплине: «Базы данных» / А. Г. Масловская Практикум. – Благовещенск: Амурский гос. ун-т, 2010 – 100 с.

25 Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite / С. В. Маклаков. – М.: Диалог-МИФИ, 2009. – 432 с.

26 Романченко, А.А. Дизельные автомобили «КамАЗ» / А. А. Романченко, Н. Н. Чиненов, В. Т. Иванов. − М.: Транспорт, 1984. − 208 с.

27 Кашев, Л.Е. Армейские автомобили КамАЗ-4310 / Л. Е. Кашев. − М.: Russian Motor Books, 2000. − 20 с.

28 Ведерников, А.А. КамАЗ-4310 / 4310-31 и их модификации. Руководство по эксплуатации / А. А. Ведерников. − Миасс: ОАО «Автомобильный завод КамАЗ», 2003. − 219 с.

29 Кащеев, Л.Б. КамАЗ-4310/ 5323/ 6301 − военные грузовики (Военные машины №65) / Л. Б. Кащеев. − Киров : Кировское общество Любителей военной техники и моделизма, 2004. − 64 с.

ПРИЛОЖЕНИЕ А

Основная часть программного кода

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
public class texture : MonoBehaviour {
bool a = true;
public GameObject rolimage;
public void Texture()
{
if (a == false) {
rolimage.SetActive (false);
a = true;
} else {
rolimage.SetActive (true);
a = false;
}
}
}
SubProgram "gles3 " {
"!!GLES3
#ifdef VERTEX
#version 300 es
precision highp float;
precision highp int;
uniform    vec4 _Time;
uniform    vec4 _SinTime;
uniform    vec4 _CosTime;
uniform    vec4 unity_DeltaTime;
```

```
uniform    vec3 _WorldSpaceCameraPos;
uniform    vec4 _ProjectionParams;
uniform    vec4 _ScreenParams;
uniform    vec4 _ZBufferParams;
uniform    mat4 unity_CameraProjection;
uniform    mat4 unity_CameraInvProjection;
uniform    vec4 _WorldSpaceLightPos0;
uniform    vec4 _LightPositionRange;
uniform    vec4 unity_4LightPosX0;
uniform    vec4 unity_4LightPosY0;
uniform    vec4 unity_4LightPosZ0;
uniform    mediump vec4 unity_4LightAtten0;
uniform    mediump vec4 unity_LightColor[8];
uniform    vec4 unity_LightPosition[8];
uniform    mediump vec4 unity_LightAtten[8];
uniform    vec4 unity_SpotDirection[8];
uniform    mediump vec4 unity_SHAr;
uniform    mediump vec4 unity_SHAg;
uniform    mediump vec4 unity_SHAb;
uniform    mediump vec4 unity_SHBr;
uniform    mediump vec4 unity_SHBg;
uniform    mediump vec4 unity_SHBb;
uniform    mediump vec4 unity_SHC;
uniform    mediump vec3 unity_LightColor0;
uniform    mediump vec3 unity_LightColor1;
uniform    mediump vec3 unity_LightColor2;
uniform    mediump vec3 unity_LightColor3;
uniform    vec4 unity_ShadowSplitSpheres[4];
```

```
uniform vec4 unity_ShadowSplitSqRadii;
uniform vec4 unity_LightShadowBias;
uniform vec4 _LightSplitsNear;
uniform vec4 _LightSplitsFar;
uniform mat4 unity_World2Shadow[4];
uniform mediump vec4 _LightShadowData;
uniform vec4 unity_ShadowFadeCenterAndType;
uniform mat4 glstate_matrix_mvp;
uniform mat4 glstate_matrix_modelview0;
uniform mat4 glstate_matrix_invtrans_modelview0;
uniform mat4 _Object2World;
uniform mat4 _World2Object;
uniform vec4 unity_LODFade;
uniform vec4 unity_WorldTransformParams;
uniform mat4 glstate_matrix_transpose_modelview0;
uniform mat4 glstate_matrix_projection;
uniform lowp vec4 glstate_lightmodel_ambient;
uniform mat4 unity_MatrixV;
uniform mat4 unity_MatrixVP;
uniform lowp vec4 unity_AmbientSky;
uniform lowp vec4 unity_AmbientEquator;
uniform lowp vec4 unity_AmbientGround;
uniform lowp vec4 unity_FogColor;
uniform vec4 unity_FogParams;
uniform vec4 unity_LightmapST;
uniform vec4 unity_DynamicLightmapST;
uniform vec4 unity_SpecCube0_BoxMax;
uniform vec4 unity_SpecCube0_BoxMin;
```

```
uniform      vec4 unity_SpecCube0_ProbePosition;
uniform      mediump vec4 unity_SpecCube0_HDR;
uniform      vec4 unity_SpecCube1_BoxMax;
uniform      vec4 unity_SpecCube1_BoxMin;
uniform      vec4 unity_SpecCube1_ProbePosition;
uniform      mediump vec4 unity_SpecCube1_HDR;
uniform      vec3 unity_LightPosition0;
in highp vec4 in_POSITION0;
in highp vec3 in_NORMAL0;
out highp vec3 vs_TEXCOORD0;
out highp vec4 vs_TEXCOORD1;
out highp vec2 vs_TEXCOORD2;
highp vec4 t0;
void main()
{
t0 = in_POSITION0.yyyy * glstate_matrix_mvp[1];
t0 = glstate_matrix_mvp[0] * in_POSITION0.xxxx + t0;
t0 = glstate_matrix_mvp[2] * in_POSITION0.zzzz + t0;
gl_Position = glstate_matrix_mvp[3] * in_POSITION0.wwww + t0;
vs_TEXCOORD0.xyz = in_NORMAL0.xyz;
vs_TEXCOORD1.xyz = in_POSITION0.xyz;
vs_TEXCOORD1.w = 1.0;
t0.xyz = in_POSITION0.yyy * unity_World2Shadow[0][1].xyw;
t0.xyz = unity_World2Shadow[0][0].xyw * in_POSITION0.xxx + t0.xyz;
t0.xyz = unity_World2Shadow[0][2].xyw * in_POSITION0.zzz + t0.xyz;
t0.xyz = t0.xyz + unity_World2Shadow[0][3].xyw;
t0.xy = t0.xy / t0.zz;
vs_TEXCOORD2.xy = t0.xy * vec2(0.5, 0.5) + vec2(0.5, 0.5);
```

```
return;
}
#endif
#ifdef FRAGMENT
#version 300 es
precision highp float;
precision highp int;
in highp vec3 vs_TEXCOORD0;
in highp vec4 vs_TEXCOORD1;
in highp vec2 vs_TEXCOORD2;
layout(location = 0) out lowp vec4 SV_Target0;
highp vec4 t0;
highp float t1;
lowp vec4 t10_1;
lowp vec4 t10_2;
highp vec4 t3;
mediump vec3 t16_4;
highp vec3 t5;
bool tb5;
highp float t10;
bool tb10;
highp float t15;
void main()
{
t0.x = unity_World2Shadow[1][0].x;
t0.y = unity_World2Shadow[1][1].x;
t0.z = unity_World2Shadow[1][2].x;
t0.w = unity_World2Shadow[1][3].x;
```

```
t0.x = dot(t0, vs_TEXCOORD1);
tb5 = 0.5<t0.x;
t0.x = (-t0.x) * 0.25 + 1.0;
t0.x = clamp(t0.x, 0.0, 1.0);
t10 = (-t0.x) + 1.0;
t10_1 = texture(unity_SplashScreenShadowTex1, vs_TEXCOORD2.xy);
t10 = t10 * t10_1.x;
t10_2 = texture(unity_SplashScreenShadowTex0, vs_TEXCOORD2.xy);
t0.x = t10_2.x * t0.x + t10;
t0.x = tb5 ? t0.x : float(0.0);
t3.x = unity_World2Shadow[1][0].y;
t3.y = unity_World2Shadow[1][1].y;
t3.z = unity_World2Shadow[1][2].y;
t3.w = unity_World2Shadow[1][3].y;
t5.x = dot(t3, vs_TEXCOORD1);
tb10 = 0.5<t5.x;
t5.x = (-t5.x) * 0.25 + 1.0;
t5.x = clamp(t5.x, 0.0, 1.0);
t15 = (-t5.x) + 1.0;
t15 = t15 * t10_1.y;
t5.x = t10_2.y * t5.x + t15;
t5.x = tb10 ? t5.x : float(0.0);
t0.x = max(t5.x, t0.x);
t3.x = unity_World2Shadow[1][0].z;
t3.y = unity_World2Shadow[1][1].z;
t3.z = unity_World2Shadow[1][2].z;
t3.w = unity_World2Shadow[1][3].z;
t5.x = dot(t3, vs_TEXCOORD1);
```

```
t10 = (-t5.x) * 0.25 + 1.0;
t10 = clamp(t10, 0.0, 1.0);
tb5 = 0.5<t5.x;
t15 = (-t10) + 1.0;
t15 = t15 * t10_1.z;
t10 = t10_2.z * t10 + t15;
t5.x = tb5 ? t10 : float(0.0);
t3.x = unity_World2Shadow[1][0].w;
t3.y = unity_World2Shadow[1][1].w;
t3.z = unity_World2Shadow[1][2].w;
t3.w = unity_World2Shadow[1][3].w;
t10 = dot(t3, vs_TEXCOORD1);
t15 = (-t10) * 0.25 + 1.0;
t15 = clamp(t15, 0.0, 1.0);
tb10 = 0.5<t10;
t1 = (-t15) + 1.0;
t1 = t1 * t10_1.w;
t15 = t10_2.w * t15 + t1;
t10 = tb10 ? t15 : float(0.0);
t5.x = max(t10, t5.x);
t0.x = max(t5.x, t0.x);
t0.x = (-t0.x) + 1.0;
t5.xyz = (-vs_TEXCOORD1.xyz) + unity_LightPosition0.xyzx.xyz;
t1 = dot(t5.xyz, t5.xyz);
t1 = inversesqrt(t1);
t5.xyz = t5.xyz * vec3(t1);
t5.x = dot(vs_TEXCOORD0.xyz, t5.xyz);
t5.x = clamp(t5.x, 0.0, 1.0);
```

```
t10 = t5.x * t5.x;
t5.x = t10 * t5.x;
t0.x = t0.x * t5.x;
t16_4.xyz = unity_LightColor0.xyzx.xyz + (-unity_LightColor1.xyzx.xyz);
t16_4.xyz = t0.xxx * t16_4.xyz + unity_LightColor1.xyzx.xyz;
SV_Target0.xyz = t16_4.xyz;
SV_Target0.w = 1.0;
return;
}
in  vec4 in_POSITION0;
in  vec3 in_NORMAL0;
out vec3 vs_TEXCOORD0;
out vec4 vs_TEXCOORD1;
out vec2 vs_TEXCOORD2;
vec4 t0;
void main()
{
t0 = in_POSITION0.yyyy * glstate_matrix_mvp[1];
t0 = glstate_matrix_mvp[0] * in_POSITION0.xxxx + t0;
t0 = glstate_matrix_mvp[2] * in_POSITION0.zzzz + t0;
gl_Position = glstate_matrix_mvp[3] * in_POSITION0.wwww + t0;
vs_TEXCOORD0.xyz = in_NORMAL0.xyz;
vs_TEXCOORD1.xyz = in_POSITION0.xyz;
vs_TEXCOORD1.w = 1.0;
t0.xyz = in_POSITION0.yyy * unity_World2Shadow[0][1].xyw;
t0.xyz = unity_World2Shadow[0][0].xyw * in_POSITION0.xxx + t0.xyz;
t0.xyz = unity_World2Shadow[0][2].xyw * in_POSITION0.zzz + t0.xyz;
t0.xyz = t0.xyz + unity_World2Shadow[0][3].xyw;
```

Основные скрипты

```
var speed = 10.0f;
var mainCamera : Camera;
var isSmoothCamera : boolean=false;
function Update()
{
var movement = Vector3.zero;
if (Input.GetKey("i")){
movement.z++;}
if (Input.GetKey("k")){
movement.z--;
if (Input.GetKey("u")){
movement.y++;}
if (Input.GetKey("j")){
movement.y--;}
if (Input.GetKey("j"))
movement.x--;
if (Input.GetKey("l"))
movement.x++;
if(Input.GetAxis("Mouse ScrollWheel") < 0){
movement.z=movement.z-5;
}
else if(Input.GetAxis("Mouse ScrollWheel") > 0){
movement.z=movement.z+5;
}
transform.Translate(movement * speed * Time.deltaTime, Space.Self);
}
```

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
public class texture : MonoBehaviour {
bool a = true;
public GameObject rolimage;
public void Texture()
{
if (a == false) {
rolimage.SetActive (false);
a = true;
} else {
rolimage.SetActive (true);
a = false;
}
}
}

using UnityEngine;
using System.Collections;
using UnityEngine;
using System.Collections;
public class CameraRotateAround : MonoBehaviour
{
public Transform target;
public Vector3 offset;
public float sensitivity = 3f;
public float limit = 80f;
```

```
private float Y;
private float X;
void Start()
{
limit = Mathf.Abs(limit);
if (limit > 90) limit = 90;
transform.position = target.position + offset;
}
void Update()
{
X = transform.localEulerAngles.y + Input.GetAxis("Mouse X") * sensitivity;
Y += Input.GetAxis("Mouse Y") * sensitivity;
Y = Mathf.Clamp(Y, -limit, limit);
transform.localEulerAngles = new Vector3(0, X, 0);
Vector3 position = transform.localRotation * offset + target.position;
transform.position = position;
}
}

using UnityEngine;
using System.Collections;
public class NewBehaviourScript : MonoBehaviour {
public float speed = 10f;
Update is called once per frame
void Update () {
transform.Rotate(Vector3.up, speed * Time.deltaTime);
}
public void ExitGame()
```

```
{
Application.Quit();
}
public void Hod()
{
}
}

var target : Transform; //What to rotate around
var distance = 5.0; //How far away to orbit
var xSpeed = 125.0; //X sensitivity
var ySpeed = 50.0; //Y sensitivity
private var x = 0.0; //Angle of the y rotation?
private var y = 0.0; //Angle of the x rotation?
script AddComponentMenu("Scripts/Mouse Orbit")//Add to menu
function Start() {
Initialize the angles
var angles = transform.eulerAngles;
x = angles.y;
y = 0;
}
function LateUpdate() { //Every frame, do this as late as you can
if (target) {//There's a target
Change the angles by the mouse movement
x += Input.GetAxis("Mouse X") * xSpeed * 0.02;
y -= Input.GetAxis("Mouse Y") * ySpeed * 0.02;
Rotate the camera to those angles
var rotation = Quaternion.Euler(0, x, 0);
```

```
transform.rotation = rotation;
Move the camera to look at the target
var position = rotation * Vector3(0.0, 0.0, -distance) + target.position;
transform.position = position;
}
}

using UnityEngine;
using System.Collections;
public class RotationPlane : MonoBehaviour {
private bool mouseDown;
private bool rot;
private GameObject selectedObject;
private Vector3 vec;
private float startPosition;
private float nowPosition;
private float tempAngle;
private float nowAngle;

void Start ()
{
}
Update is called once per frame
void Update ()
{
if(mouseDown)
{
nowPosition= Input.mousePosition.x;
```

```
Debug.Log("2-"+tempAngle);
tempAngle=-(nowPosition-startPosition)/5.0f;
Debug.Log("3-"+tempAngle);
if(tempAngle!=0)
{
nowAngle=tempAngle;
if(selectedObject)
{
vec =new Vector3 (selectedObject.transform.eulerAngles.x);
this.transform.eulerAngles=new Vector3(this.transform.eulerAngles.x) ;
selectedObject.transform.eulerAngles=vec;
}
}
void OnMouseUp ()
{
mouseDown=false;
Debug.Log("4-"+tempAngle);
}
void OnMouseDown()
{
nowAngle=this.transform.rotation.y;        //Debug.Log("1-"+nowAngle);
startPosition= Input.mousePosition.x;
mouseDown=true;
selectedObject= GameObject.FindGameObjectWithTag("SelectedObject");
}
}
```

```
using UnityEngine;
using System.Collections;
[AddComponentMenu("Infinite Camera-Control/Mouse Orbit with zoom")]
public class MouseOrbitInfiniteRotateZoom : MonoBehaviour {
public Transform target;
public float xSpeed = 12.0f;
public float ySpeed = 12.0f;
public float scrollSpeed = 10.0f;
public float zoomMin = 1.0f;
public float zoomMax = 20.0f;
public float distance ;
public Vector3 position;
public bool isActivated;
float x = 0.0f;
float y = 0.0f;
Use this for initialization
void Start () {
Vector3 angles = transform.eulerAngles;
x = angles.y;
y = angles.x;
}
void LateUpdate () {
only update if the mousebutton is held down
if (Input.GetMouseButtonDown(1)){
isActivated = true;
}
if mouse button is let UP then stop rotating camera
if (Input.GetMouseButtonUp(1))
```

```
isActivated = false;
}
if (target && isActivated) {
x += Input.GetAxis("Mouse X") * xSpeed;
y -= Input.GetAxis("Mouse Y") * ySpeed;
transform.RotateAround(target.position,transform.up, x);
transform.RotateAround(target.position,transform.right, y);
x=0;
y=0;
} else {
if (Input.GetAxis("Mouse ScrollWheel") != 0)
{
distance = Vector3.Distance (transform.position , target.position);
distance = ZoomLimit
position = -(transform.forward*distance) + target.position;
transform.position = position;
}
}
}
public static float ZoomLimit(float dist, float min, float max)
{
if (dist < min)
dist= min;
if (dist > max)
dist= max;
return dist;
}
```