**BC/NW 2018 № 1 (32):10.1**

**ОБЗОР ПОДХОДОВ К СОЗДАНИЮ КОНЦЕПТ-ИЗОБРАЖЕНИЙ СРЕДСТВАМИ IT-ТЕХНОЛОГИЙ**

Сафронов И.А**.**

**Введение**

Компьютерные игры долгие годы были главным двигателем прогресса как центральных процессоров, так и графических подсистем. Но в последнее время темпы прироста вычислительных мощностей и внедрения новых возможностей заметно замедлились. Наверное, дело в том, что большинство игроков вполне удовлетворят текущее качество и реалистичность компьютерной графики. Кроме того, возможности сегодняшних технологий, реализуемых в компьютерных комплектующих массовой доступности, позволяют решать большинство задач обычного пользователя.

Для достижения количественного и качественного отображения, а также преодоления предела технических ограничений, приходится прибегать к хитростям и уловкам разнообразных методов и средств использования 3D графики, которые при аналогичном результате значительно уменьшают потребление вычислительных мощностей, что позволяет увеличить качество отображения конечного игрового продукта.

Областью исследования являются видеоигры, исследуемым аспектом их формирование являются модели игрового окружения.

Создание модели игрового окружения – является одним из этапов производства видеоигр, от которого зависят сроки и качество всего проекта.

Разнообразие используемых методов и разрозненность подходов в области создания игровых моделей могут приводить к ошибкам в топологии объектов и при импорте в игровые движки, которые понижают качество проекта и повышают потребление вычислительных мощностей. Нахождение решений проблем, связанных с моделированием 3д-объектов и последующей их эксплуатацией, особо значимо для проектов в области 3д-моделирования в сферах кино, телевидения, культуры, образования, промышленного дизайна и других, где возможно использование 3д-объектов. Это обосновывает актуальность данного диссертационного исследования.

Тема проработана в базовой концепции понятия 3д-моделирования. Общепризнанную методологию, работающую как стандарт, еще необходимо прорабатывать. Люди в разных сферах работы с 3д-графикой сами определяют методы и алгоритмы использования имеющихся средств. Каждый пакет 3-хмерной графики используют зачастую совершенно разные подходы в меру разработанности программ.

Целью данного исследования является формирование методики создания и оптимизации модели игрового окружения с критерием сокращения используемых вычислительных мощностей.

Для достижения поставленной цели исследования необходимо решить следующие задачи:

* провести анализ игровых движков и выбрать одних из них для тестирования модели игрового окружения;
* провести анализ 3D-редакторов и выбрать один из них для создания модели игрового окружения;
* провести анализ этапов создания модели и сформировать свод правил по данным этапам;
* провести анализ подготовки модели игрового окружения к импорту в игровой движок и сформировать технические рекомендации по данному процессу;
* следуя выведенному своду правил создать модель игрового окружения;
* исследовать созданную модель игрового окружения, провести коррекцию выявленных недостатков;
* протестировать импорт модели с использованием технических рекомендация и без них;

Объектом исследования является процесс создания 3д-моделей игрового окружения по методике.

Предметом исследования является методика создания модели игрового окружения.

Отработанная методика позволит выбрать в начале работы приемлемый алгоритм создания модели и снизить потребление вычислительных мощностей на обработку 3д-сцены.

В своей работе я использовал такие методы:

* Теоретический – сравнение и обобщение методов создания 3д-моделей, моделирование усредненного алгоритма по стандартным параметрам.
* Эмпирический – эксперимент по созданию моделей по алгоритму, тестирование моделей.
* Статистический – количественное сравнение исходных данных и полученных.

1. **ТРЕХМЕРНАЯ ГРАФИКА**
   1. **Трехмерный объект**

Трехмерный объект состоит из вершин, граней, полигонов, материалов.

1. Вершина (ед. Vertex, мн. Vertices) - это абстрактная геометрическая точка с координатами X, Y и Z. Вершиной она называется потому, что является крайней точкой либо замкнутого полигона (плоского многоугольника), либо объёмной фигуры.

2. Грань (Edge) - отрезок прямой, соединяющий две вершины. Опять же, в трёхмерной графике это не самостоятельное нечто, а лишь ограничитель для полигонов.

3. Полигон (poly, polygon, face) - основная функциональная составляющая: плоская многоугольная фигура (обычно трёхмерные редакторы и другие приложения предпочитают оперировать только треугольниками и четырёхугольниками), из множества которых состоит поверхности трёхмерной фигуры.

Любую даже самую сложную фигуру или поверхность можно представить в виде множества простых фигур: такая идеальная фигура, как шар (точнее, сфера), в компьютерной графике в любом случае представляется в виде множества треугольников и четырёхугольников. Чем их будет больше, тем выше степень приближения, то есть, тем более гладкой, тем более сферической будет поверхность, изображено на рисунке 1. Но опять-таки, это вопрос степени приближения.

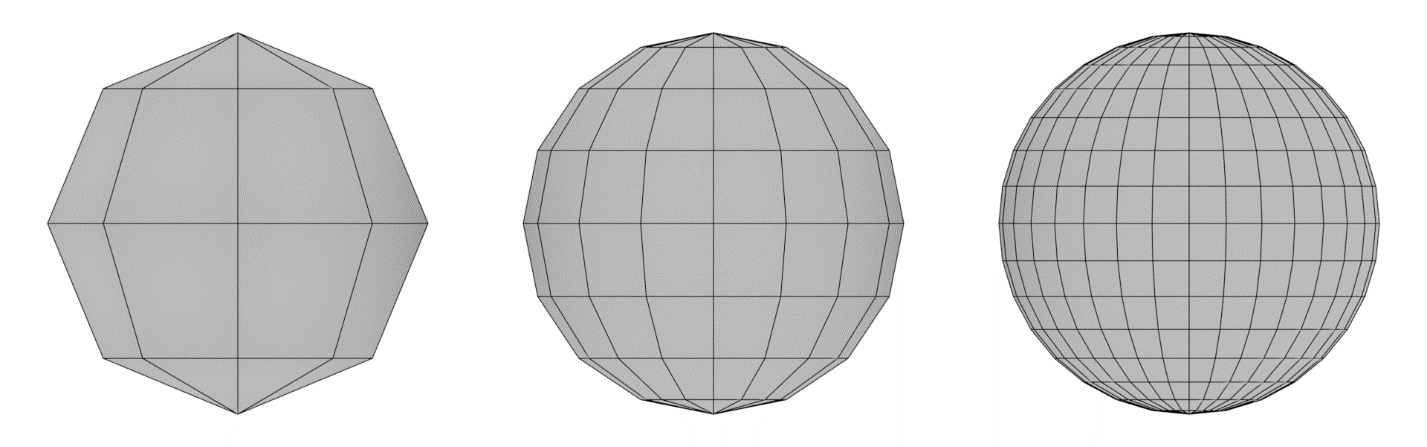


Рисунок 1. Степень приближения геометрии сферической формы

Материал трехмерного объекта (текстура) – растровое изображение, нанесенное на полигональную геометрию трехмерного объекта для придания цвета. Использование текстур позволяет воспроизвести малые объекты поверхности, создание которых полигонами оказалось бы чрезмерно ресурсоёмкими. Качество текстурированной поверхности определяется текселями – количеством пикселей на минимальную единицу текстуры.

* 1. **Общие сведения геометрии трехмерного объекта**

Геометрические объекты описывают предмет и явления, обладающие геометрическими свойствами. В общем случае реально существующий объект, не может в точности соответствовать своему описанию. Для этого бы потребовалось бесконечное число троек координат (x, y, z) – по одной для каждой точки поверхности объекта.

В настоящее время при моделировании объектов используют несколько основных типов геометрических моделей.

Для описания каркасной модели используются геометрические объекты первого порядка – линии или ребра. Каркасные модели применяют для задания объектов, представляющих собой полиэдры, замкнутые многогранники произвольной формы, ограниченные плоскими гранями. Каркасная модель содержит список координат вершин полиэдра с указанием связей между ними.

Каркасное представление объекта часто используется не при моделировании, а при отображении моделей как метод визуализации.

Преимуществами каркасной модели являются низки требования к вычислительным ресурсам, недостатком – невозможность построения высоко реалистичных изображений, так как совокупность отрезков не является адекватным описанием объекта – отрезки сами по себе не определяют поверхностей. Рисунок 2.

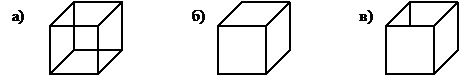


Рисунок 2. Одна и та же каркасная модель (а) может описывать и куб (б), и открытую коробку (в)

Развитием каркасной модели является кусочно-аналитическая граневая модель, которая задается перечислением всех отдельных граней. Объект задается множеством ограничивающих его граней и нормалью, направленной из объекта. Каждая грань задается циклом ограничивающих ее ребер, каждое ребро – двойкой ограничивающих его точек (вершин), каждая точка – тройкой координат в трехмерном пространстве. Т.е. граневая модель представляет трехмерный объект в виде замкнутой поверхности.

Совокупность граней, предоставленных плоскими многоугольниками и ограниченных прямолинейными ребрами, образует полигональную сетку. Грани могут иметь любую форму, но в подавляющем большинстве случаев используются выпуклые многоугольники с минимальным количеством вершин (треугольники и четырехугольники), т.к. их обсчет выполняется проще.

Основным недостатком полигональной сетки является приблизительность представления формы объекта при описании искривленных поверхностей. Для улучшения кусочно-линейной аппроксимации таких объектов увеличивают число граней, что приводит к дополнительным затратам памяти и увеличению объема вычислений.

В рамках граневой модели грани могут представлять собой и искривленные поверхности, ограниченные криволинейными ребрами. Наиболее часто в качестве граней используются параметрические бикубические куски, ограниченные параметрическими кубическими кривыми.

При использовании бикубических кусков для представления объекта с заданной точностью требуется значительно меньшее число граней, чем при аппроксимации полигональной сеткой. Однако, вычисления при работе с бикубическими поверхностями значительно сложнее, чем при работе с плоскими гранями.

В отличие от граневой модели, объемно-параметрическая модель рассматривает объект как сплошное тело. Объект описывается как совокупность некоторых базовых объемных элементов формы (объемных примитивов). Каждый примитив в модели задается двумя группами параметров:

* размерные параметры – определяют геометрические размеры примитива;
* параметры положения – устанавливают положение и ориентацию примитива относительно мировой системы координат.

В качестве примитивов используются простые геометрические тела: цилиндр, конус, усеченный конус, параллелепипед, шар, тор.

В качестве параметров положения обычно используют координаты центральной точки примитива и координаты единичного вектора, направленного вдоль высоты примитива.

Кроме этих параметров задаются операции над примитивами, в качестве которых используются три основные операции теории множеств – объединение, пересечение и вычитание. Объединением двух примитивов является объект, включающий все точки исходных примитивов. Пересечением двух примитивов является объект, все точки которого принадлежат одновременно и первому, и второму примитиву. Результатом вычитания двух примитивов является объект, состоящий из тех точек первого примитива, которые не принадлежат второму примитиву.

Недостатком объемно-параметрической модели является отсутствие явных границ отсеков граней в случае взаимопроникновения примитивов.

В рамках кинематической модели объект может быть задан совокупностью объемных элементов, каждый из которых представляет собой объем, «вырезаемый» в пространстве при движении по определенной траектории замкнутого плоского контура. Траектория движения контура может быть, как прямой, так и искривленной.

Вид элемента определяется формой контура и траекторией его движения. Например, цилиндр в рамках кинематической модели может быть описан как движение круга вдоль отрезка, представляющего собой высоту цилиндра.

Для моделирования элементов сложной формы можно использовать изменение размеров контура или его положения относительно траектории во время движения.

Достоинством модели является практическое отсутствие ограничений на сложность формируемого объекта. К недостаткам относится сложность задания элементов.

**Полигональные сетки**

Полигональные сетки представляют собой совокупность вершин, ребер и многоугольников.

Сетку можно представить несколькими различными способами.

При явном задании многоугольников каждый многоугольник представляется в виде списка координат его вершин:

P = ((x1, y1, z1), (x2, y2, z2), …(xn, yn, zn)).

Вершины многоугольника запоминаются в том порядке, в котором они встречаются при обходе вокруг многоугольника. Все последовательные вершины многоугольника, а также первая и последняя вершины, соединяются ребрами.

Этот способ записи является эффективным для каждого отдельного многоугольника. Для полигональной сетки этот метод приводит к большим потерям памяти из-за дублирования информации о координатах общих вершин многоугольников, а также не дает явного описания общих ребер и вершин. Для поиска всех многоугольников, имеющих общую вершину, требуется сравнение троек координат одного многоугольника с тройками координат всех остальных многоугольников. Если же из-за ошибок округления одна и та же вершина в разных многоугольниках имеет разные координаты, то результат поиска может быть нулевой.

При визуализации полигональной сетки общие ребра рисуются дважды – по одному разу для каждого многоугольника.

При задании многоугольников с помощью указателей в список вершин каждый узел полигональной сетки запоминается лишь один раз в списке вершин. Каждый многоугольник определяется списком указателей в список вершин.

Этот метод требует меньшего объема памяти, но общие ребра многоугольников по-прежнему рисуются дважды.

При явном задании ребер многоугольников полигональная сетка задается в виде иерархической модели. Координаты каждого узла полигональной сетки запоминаются в списке вершин.

Создается список ребер, где каждое ребро указывает на две вершины в списке вершин. Каждый многоугольник задается как совокупность указателей на элементы списка ребер.

Полигональная сетка изображается вычерчиванием всех ребер (а не всех многоугольников).

Заданную таким способом полигональную сетку наиболее просто проверять на непротиворечивость представления, т.е. на выполнение некоторых условий: что все многоугольники замкнуты, что все ребра используются по крайней мере один раз, но не более некоторого максимального числа раз, что на каждую вершину есть ссылка по крайней мере от двух ребер и т.п. Проверка непротиворечивости представления наиболее важна для полигональных сеток, созданных в интерактивном режиме (например, при оцифровке объектов), когда ошибки практически неизбежны.

* 1. **Общие сведения о текстурировании в трехмерной графике**

Любые объекты, окружающие нас в реальной жизни, имеют свой характерный рисунок, по которому мы можем безошибочно узнать объект.

Подобная идентификация происходит на подсознательном уровне. Когда мы видим проходящий через предмет свет, мы понимаем, что он сделан из стекла, а отражение на поверхности объекта дает нам право предположить, что он отполирован.

Созданные в трехмерном редакторе объекты выглядят, как каменные скульптуры с однотонным цветом, и совсем не похожи на настоящие. Чтобы «раскрасить» все элементы сцены, а также наделить их такими физическими свойствами материалов, как прозрачность, шершавость, способностью отражать и преломлять свет и т. д., необходимо для каждого объекта сцены установить характеристики материала, или текстурироватъ сцену.

Это очень непростая задача, особенно для неподготовленного пользователя. В реальной жизни мы воспринимаем объекты такими, какие они есть, не задумываясь о коэффициентах отражения и преломления, размере блика и других физических параметрах объекта. В трехмерной графике все эти свойства материала необходимо устанавливать вручную.

Для статического изображения продемонстрировать форму гораздо сложнее, поэтому для статического изображения особую роль играют факторы, раскрывающие суть объекта. Например, изображение белого, просвечивающегося перышка создает впечатление легкости, иллюзии того, что оно может улететь при малейшем дуновении ветерка. Если то же самое перо будет темным и не будет пропускать свет, то при взгляде на картинку такие мысли не возникнут. Очевидно, что черный цвет кажется тяжелым, и если в сцене будут изображены два предмета (белый и черный), то зрителю черный будет казаться тяжелее. Материалы, которые имитируются в трехмерной графике, могут быть самыми разнообразными: металл, дерево, пластик, стекло, камень и многое другое. При этом каждый материал определяется большим количеством свойств (рельеф поверхности, зеркальность, рисунок, размер блика и т. д.).

Для описания характеристик материала используются числовые значения параметров (процент прозрачности, размер блика и др.).

Одну из основных ролей в описании характеристик материала играют процедурные карты (карты текстур) — двухмерные изображения, генерируемые программой или загруженные из графического файла. Процедурная карта позволяет определенным образом задать изменение параметра материала. Например, использование в качестве карты прозрачности стандартной процедурной карты Checker (Шахматная текстура) делает материал прозрачным и клетчатым.

Визуализируя любой материал, нужно помнить, что качество материала на полученном изображении очень сильно зависит от множества факторов, среди которых: параметры освещения (яркость, угол падения света, цвет источника света и т. д.), алгоритм визуализации (тип используемого визуализатора и его настройки) и разрешение растровой текстуры.   
Большое значение также имеет метод проецирования текстуры на объект. Из-за неудачно наложенной текстуры на трехмерном объекте может возникнуть шов или некрасиво повторяющийся рисунок.

Кроме того, обычно реальные объекты не бывают идеально чистыми. Если смоделируем кухонный стол, то несмотря на то, что кухонная клеенка имеет повторяющийся рисунок, ее поверхность не должна казаться однородной — клеенка может быть потерта на углах стола, иметь порезы от ножа и т. д.

Чтобы трехмерные объекты не выглядели неестественно чистыми, можно использовать сделанные вручную (например, в программе Adobe Photoshop) карты «загрязненности» и смешивать их с имеющимися процедурными картами, получая реалистичный, «изношенный» материал.

* 1. **Общее описание процесса создания трехмерного объекта**

Процесс создания трехмерного объекта делится на несколько обязательных и последовательных этапов. Они одинаковы, независимо от того, каким способов делаются. Трехмерный объект часто называют сценой. Сначала создают декорации для виртуальной сцены, затем создают актеров — трехмерных персонажей (моделирование), заставляют их двигаться и разговаривать (анимация). После чего устанавливают освещение (освещение) и выбирают точку съемки (съемка). 3D-художникам также приходится раскрашивать все предметы виртуального мира, делая их похожими на настоящие (текстурирование).

**Моделирование.** Виртуальный мир наполняется разными объектами, которые создает пользователь в 3D-редакторе: трехмерными персонажами, зданиями, горами, лесами и другими вещам виртуального мира. Процесс создания трехмерных объектов называется моделированием, а сами объекты — моделями. В отличие от рисованного изображения, трехмерную модель можно вращать и перемещать в трех измерениях, глядя на нее со всех сторон. В зависимости от того, какой именно объект нужно создать, на его моделирование может уйти от нескольких секунд до нескольких месяцев. Например, шкаф можно сделать за пару минут, а создание человеческой головы может занять не один день. Существует множество разных способов моделирования.

**Текстурирование.** На этапе моделирования объекты, созданные в трехмерном пространстве, отличаются друг от друга только формой. Для того чтобы трехмерный объект напоминала реальный предмет, недостаточно точно повторить его форму, нужно еще и раскрасить его соответствующим образом. Процесс раскраски трехмерных объектов называется текстурированием — от термина «текстура», которым обозначается рисунок, накладываемый на трехмерный объект. В общем смысле слово «текстура» обозначает особенности строения твердого вещества, обусловленные характером расположения его частей.

В трехмерной графике текстурой может служить как фотография реального объекта, наложенная на трехмерный объект, так и двумерный рисунок, созданный в программе типа Adobe Photoshop. Существуют и другие типы текстур, о которых будет рассказано позже.

**Анимация.** В отличие от моделирования, анимация не является обязательным этапом создания трехмерных проектов. Например, в этом нет необходимости при создании проекта интерьера квартиры. Но при создании рекламного ролика или эффекта наподобие взрыва без анимации не обойтись. Анимация подразумевает изменение во времени положения объектов в трехмерном пространстве. В процессе создания анимации 3D-художнику необходимо разместить уже созданные им объекты в каждом кадре

**Съемка.** Конечный продукт трехмерной анимации — видео, которое может «сниматься» с разных точек. Выбор точки и метода съемки в трехмерной анимации не менее важен, чем в настоящем кино или на телевидении. Для съемки анимации, созданной в 3D-редакторе, применяются виртуальные камеры. Их можно перемещать, переключаться между ними в процессе анимации. Виртуальные камеры обладают теми же свойствами, что и настоящие: они могут фиксировать картинку под разным углом зрения, имитировать эффект глубины резкости и т. д. При работе над анимационным проектом необходимо уметь правильно подобрать расположение камер, настроить их, чтобы точка съемки была наиболее удачной, продумать, как камеры будут перемещаться в процессе съемки.

**Освещение.** Для имитации освещения в трехмерной графике используют специальные объекты, которые так и называются — источники света. Чтобы освещение трехмерных объектов было похоже на настоящее, приходится учитывать множество факторов: где расположен источник света, с какой силой он светит, от каких предметов свет отражается и т. д.

**Визуализация.** После моделирования объектов, текстурирования, анимации, освещения, установки и настройки камер можно приступить к визуализации. В результате получается фото или видео, которое записывается в файл. Визуализацию также называют рендерингом, или просчетом.

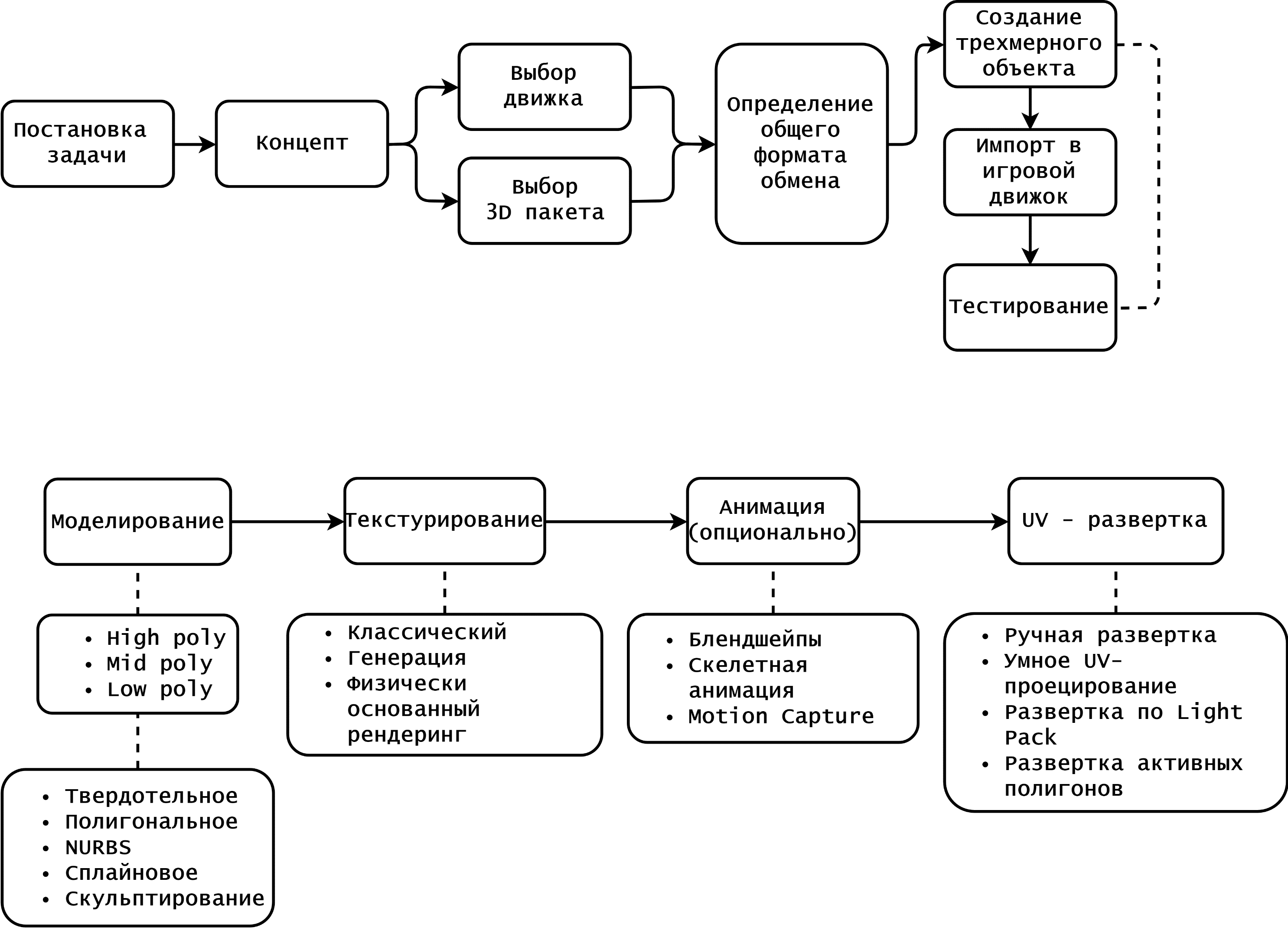
**Импорт в игровой движок.** Перенос трехмерного объекта и его составных частей в игровой движок, для последующей настройки в игровой сцене.

Этапы создания трехмерного объекта для игры и создания видео/статичной картинки изображены на рисунке 3 и 4.

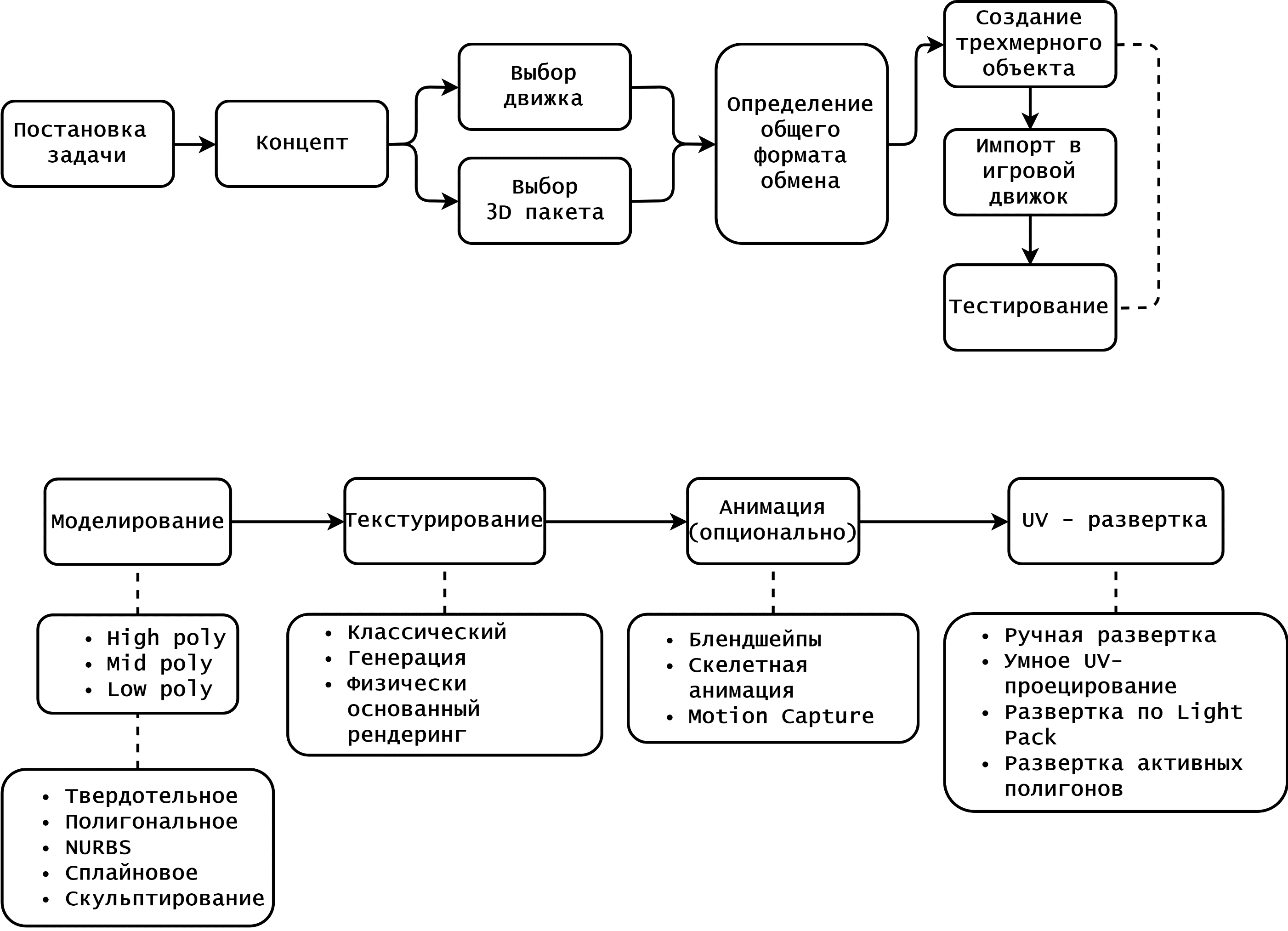
Рисунок 3. Этапы создания трехмерного объекта для игры.

Рисунок 4.Этапы создания трехмерного объекта для визуализации.

Общая схема полного цикла создания модели трехмерного окружения игрового мира



Подробная схема создания трехмерного объекта



Исследование методов и средств трехмерной графики

* 1. **Сравнительный анализ инструментов трехмерной графики**

Почти все игровые движки, имеют встроенный редактор ландшафта, позволяющий моделировать и текстурировать игровые локации, но не имеет инструментов по созданию сложных полигональных трехмерных объектов.

Для этих целей служат специализированные программные пакеты трехмерного моделирования, некоторые из которых включают в себя не только функции моделирования, но и весь инструментарий, необходимый для создания полноценного трехмерного объекта.

Для создания компьютерной игровой графики используется множество различных приложений. Условно их можно разделить на следующие группы:

* программы для цифрового скульптинга (Pixologic ZBrush, Autodesk Mudbox)
* узкоспециализированные приложения, под конкретные задачи (анимация жидкостей – RealFlow, создание текстур – Mari, и пр.)
* универсальные 3D редакторы (Cinema 4D, 3Ds MAX, Maya, Houdini, Blender и т.д.)

Комбинирование различных приложений зависит от конкретной задачи, стоящей перед художником. Рассмотрим универсальные 3D редакторов.

Универсальные 3D редакторы, как правило уже содержат все необходимые инструменты для моделирования, анимации и визуализации.

При выборе универсального 3D-редактора необходимо учесть следующие факторы:

* функционал программы
* удобство пользования
* доступность, цена

**3Ds Max [1]** – «пионер» среди 3D редакторов, очень популярный инструмент, занимает ведущие позиции в сфере дизайна и архитектурной визуализации. Часто используется в игровой индустрии.

Возможности:

* Основные виды моделирования (полигональное, сплайновое, NURBS)
* Мощная система частиц
* Модуль волосы/шерсть
* Расширенные шейдеры Shader FX
* поддержка новых и усовершенствованных механизмов Iray и mental ray.
* анимация толпы,
* импорт из Revit и SketchUp,
* интеграция композитинга.

Плюсы: огромный функционал, множество плагинов и обучающей информации.

Минусы: не так прост в освоении, платная основа.

**Maya [2]** – промышленный стандарт 3D графики в кино и телевидении.  Maya популярна среди крупных студий и масштабных проектов в рекламе, кино, игровой индустрии. Пакет идеален для создания анимации.

Возможности:

* полный набор инструментов для NURBS- и полигонального моделирования;
* мощные средства общей и анатомической анимации;
* развитая система частиц;
* технология Maya Fur (создание меха, волос, травы);
* технология Maya Fluid Effects (моделирование жидкостей, атмосферы);
* динамика твердых и мягких тел;
* широкий набор средств создания динамических спецэффектов;
* UV-текстуры, нормали и цветовое кодирование;
* многопроцессорный гибкий рендеринг.

Плюсы: огромный функционал и возможности.

Минусы: длительное и сложное обучение, высокие требования к системе, высокая цена.

**Cinema 4D [3]** – один из самых лучших и удобных 3D пакетов на сегодняшний день. Огромный функционал: от моделирования, анимации, эффектов до «лепки» и модуля BodyPaint 3D.  У Cinema 4D более понятный и удобный интерфейс нежели у 3Ds Max и Maya. Широко используется в моушен-дизайне, киноиндустрии и рекламе.

Возможности:

* полигональное и NURBS-моделирование;
* BodyPaint 3D (модуль для создания разверток UV и текстурных карт);
* генерация и анимация объектов;
* персонажная анимация;
* динамика мягких и твердых тел;
* модуль для создания реалистичных волос;
* система частиц Thinking Particles;
* неплохой встроенный визуализатор.

Плюсы: легкость в освоении, интуитивный интерфейс, отличный функционал, множество обучающих материалов, тесная связь с Adobe After Effects, Houdini и т.д.

Минусы: неотлаженная система перехода между версиями, платная основа.

**Modo [4]**– полноценный продукт для моделирования, рисования, анимации и визуализации. Включает также инструменты скульптинга и текстурного окрашивания. Благодаря удобству пользования и высокой производительности, у Modo репутация одного из самых быстрых инструментов моделирования. Modo популярен в сфере рекламы, разработки игр, спецэффектов и архитектурной визуализации.

Возможности:

* полигональное и моделирование SDS;
* современные инструменты анимации;
* динамика твердых и мягких тел;
* система рисования;
* материал Fur (мех) для создания волос, травы и меха;
* инструменты лепки;
* быстрая и качественная визуализация.

Плюсы: мощный и понятный инструментарий, высокая производительность.

Минусы: мало информации, платная основа.

**Houdini [5]** – мощный профессиональный пакет для работы с 3D графикой, в его основе процедурная, нодовая система. Houdini идеально подходит для создания сложной динамики, симуляции: частиц, жидкости, дыма, огня, имитации природных явлений и т.д. А также это отличный инструмент для создания впечатляющих визуальных эффектов. Основная область применения Houdini – киноиндустрия.

Возможности:

* полигональное и NURBS-моделирование,
* анимация (ключевая, процедурная),
* персонажная анимация,
* система частиц,
* динамика твердых и мягких тел, тканей, шерсти/волос, газов и жидкостей,
* работа с объемным звуком,
* мощный рендер движок Mantra,
* встроенный инструмент композитинга.

Плюсы: высококлассные спецэффекты и анимация.

Минусы: мало информации, высокая цена.

**Blender [6]** - единственный в списке бесплатный 3D пакет, который практически не уступает по функционалу платным приложениям. Blender включает в себя средства для 3D моделирования, анимации, а также набор опций для создания игр, визуальных эффектов и скульптинга. Отличная альтернатива «монстрам» 3D анимации. Благодаря поддержке Blender Foundation, программа очень быстро и стабильно развивается.

Возможности:

* полигональное моделирование, сплайны, NURBS-кривые и поверхности;
* режим лепки;
* система частиц;
* динамика твердых и мягких тел: жидкость, шерсть/волосы и т.д.;
* скелетная анимация;
* встроенные механизмы рендеринга и интеграция со сторонними визуализаторами;
* редактор видео;
* функции создания игр и приложений (Game Blender).

Плюсы: доступность, открытый код, кроссплатформенность, небольшой размер (около 50 мегабайт), широкий функционал, расширения, возможность создания игр.

Минусы: отсутствие документации в базовой поставке.

**Цифровой скульптинг** – функционал трехмерного моделирования, представляющий собой имитацию процесса «лепки» трехмерного объекта, то есть деформирования её полигональной сетки специальными инструментами – кистями. Можно провести аналогию с лепкой фигур руками из пластилина или глины. Хотя универсальные редакторы и предоставляют возможности цифрового скульптинга, для удобной и правильной работы лучше воспользоваться специализированными программами, такими как ZBrush, Mudbox.

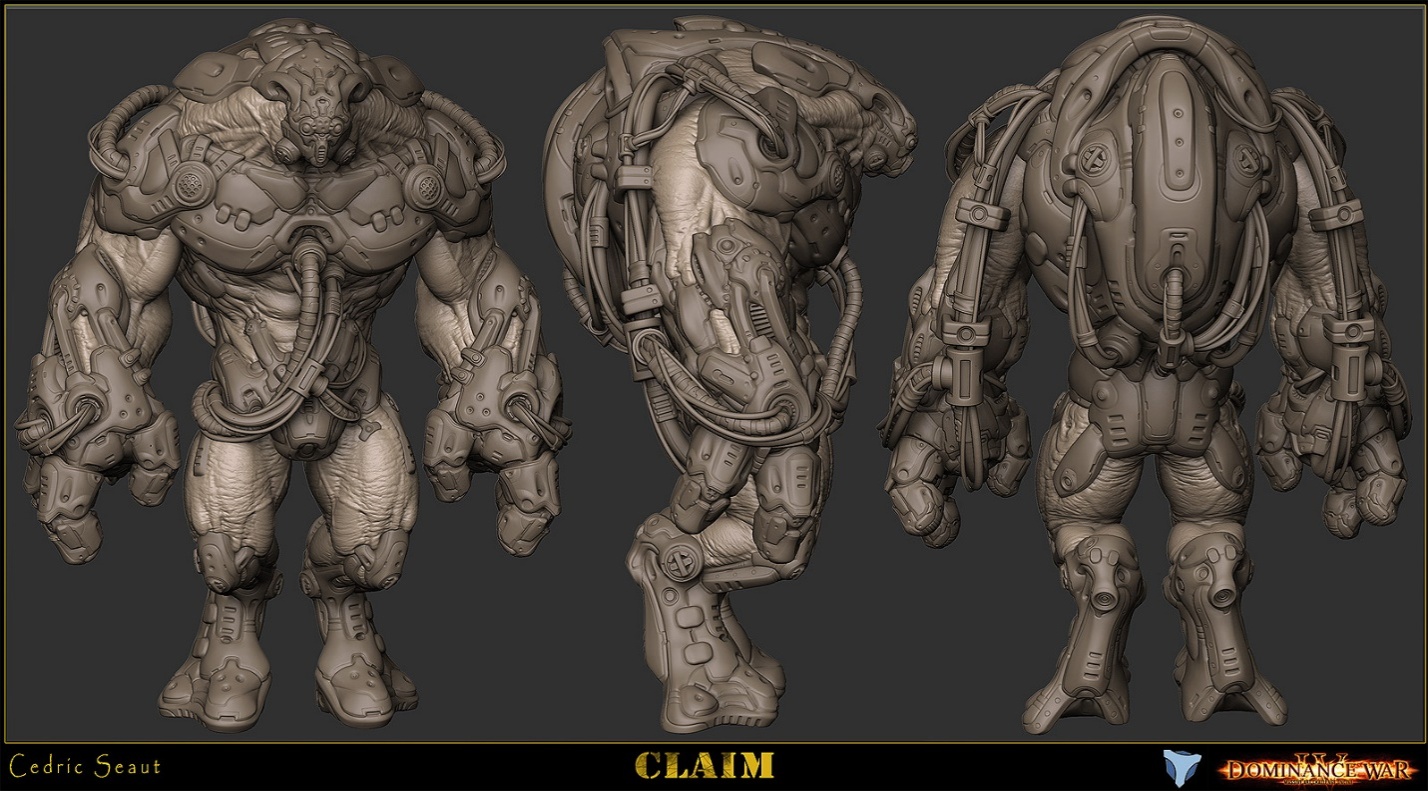


Рисунок 5. Результат цифрового скульптинга

Всего существует две программы для цифрового скульптинга, которые, по моему мнению, полностью удовлетворяют потребности художников трехмерной графики:

ZBrush [7] и Mudbox [8].

Для сравнения программных продуктов необходимо иметь базовый трехмерный объект, его можно создать и в универсальном трехмерном редакторе, но рассмотрим в рамках этих двух программ.

В ZBrush существует масса возможностей для создания трехмерного объекта. Персонажей и различных существ можно создавать с помощью ZSpheres. ShadowBox идеально подходит для работы с твердыми объектами, позволяя создавать сложные объекты из примитивов.

У Mudbox более ограниченные возможности по созданию трехмерного объекта, при этом зачастую приходится использовать уже готовый объект, заранее сделанный в универсальном трехмерном редакторе.

Скалптинг.В ZBrush больше встроенных кистей для лепки. Есть кисти, с помощью которых можно вытягивать общую форму трехмерного объекта, лепить поверх существующего, заострять грани. Кроме того, ZBrush предлагает широкие возможности по настройке кистей и их отпечатков.

Стандартные кисти Mudbox не сильно отличаются от кистей ZBrush. С помощью кистей так же можно перемещать геометрию и лепить поверх других объектов, но есть один момент, который сильно выделяет Mudbox от ZBrush. Это использование векторных карт смещения в качестве штампов, представляют собой сохранение любого слепленного объекта в виде кисти для дальнейшего взаимодействия.

Ретопология.Создание новой корректной топологии по некорректной сетке лепки, является важным моментом с скалптинге. Как в ZBrush, так и в Mudbox есть свои решения для ретопологии.

Есть автоматизированные способы, с помощью которых можно быстро создать новый объект по направляющим исходя из поликаунта. ZBrush использует ZRemesher, а Mudbox отвечает ему опцией Retopologize. Но инструменты ретопологии ZBrush существуют дольше, а поэтому они более продуманные.

Для текстурирования лучше использовать узкоспециализированный программные продукты. Как и в случае со скульптингом, можно воспользоваться и универсальными редакторами, но специализированные программы позволяют сделать это быстрее, удобнее и качественней.

Из существующих программ можно выделить:

* The Foundry Mari [9]
* 3D Coat [10]
* Substance Painter [11]

The Foundry Mari - одна из самых известных разработок для текстурирования трехмерных объектов, используется для обработки экстремальных проектов в первую очередь. Объекты создаваемые в данном приложении часто можно наблюдать в разных фильмах, например Район номер девять, Милые кости, даже Аватар пользовался наработками. Платная основа.

3D-Coat так же предоставляет набор инструментов для текстурирования объектов.

Поддержка до 1600x16000 текстур, 32 битный диспэйсмент смещения. Возможность использования несколько текстурных разверток на объект. Можно проецировать текстуры на объект, работать совместно с Photoshop или Gimp. Поддержка всех карт материалов. Платная основа.

Substance Painter аналогичен 3D Coat, предоставляет обширное количество лицензий как учебных так и для ознакомительных целей, дешевле.

Как и говорилось ранее все необходимые модули можно использовать и в универсальном редакторе, но при разбиении процесса работы на разные инструменты позволяют добиться лучшего результата.

Минимальные требования для запуска любого программного продукта из рассмотренных:

* ОС: Windows 2000, XP, Vista, 7, Mac OS 10.4.5 (Intel) или новее
* Процессор: процессор Core i3 с тактовой частотой 2,4 ГГц или лучше
* Оперативная память: 2048 МБ или лучше
* Жесткий диск: 15 ГБ свободного места
* Видеокарта: Nvidia GTX560 1024 mb видеопамяти или лучше
* DirectX®: 9.0c
  1. **Сравнительный анализ игровых движков**

Игровой движок (англ. game engine) — это центральный программный компонент компьютерных и видеоигр, и других интерактивных приложений с графикой, обрабатываемой в реальном времени. Он обеспечивает основные технологии, упрощает разработку и часто даёт игре возможность запускаться на нескольких платформах, таких как игровые консоли и настольные операционные системы, например, GNU/Linux, Mac OS X и Microsoft Windows.

Словосочетание «игровой движок» подразумевает целый комплекс прикладных программ, включающий движок рендеринга («визуализатор») для 2D или 3D графики, физический движок, или обнаружение столкновений (и реакции на столкновение), звук, скриптинг, анимацию, искусственный интеллект, сетевой код, streaming, управление памятью, threading и граф сцены. Строго говоря, все части кода, написанные программистами при разработке игры, являются компонентами движка. Игровой процесс (геймплей) определяется функциями, реализованными в этих программах.

Если на заре компьютерной эры каждая игра создавалась разработчиками «с нуля», то всё изменилось с момента появления игры Doom (более 15 лет назад). В настоящее время практически все коммерческие игры создаются с использованием готового инструментария: в процессе разработки можно сэкономить за счет повторного использования одного игрового движка для создания множества различных игр.

Иногда «мододелы» делают какие-то видоизменения с помощью встроенного в игру языка программирования через скриптовые сценарии. Можно подкрутить некоторые параметры функций движка, чем с удовольствием пользуются читеры (увеличивая, например, количество денег у игрока). Однако серьезные изменения геймплея невозможны без вмешательства в исходный код игры. И тут начинаются проблемы. Во-первых, не каждая игровая компания код обнародует. Во-вторых, далеко не каждый может разобраться самостоятельно в огромном количестве функций движка. А другого пути часто нет, поскольку отсутствует справочная информация и техническая поддержка (спросить не у кого). В-третьих, как правило, публикуют исходный код уже технически устаревших движков (например, Quake Engine).

От правильного выбора инструмента для разработки игр зависит судьба проекта, студии или разработчика. Данный выбор зависит от многих факторов. Рисунок

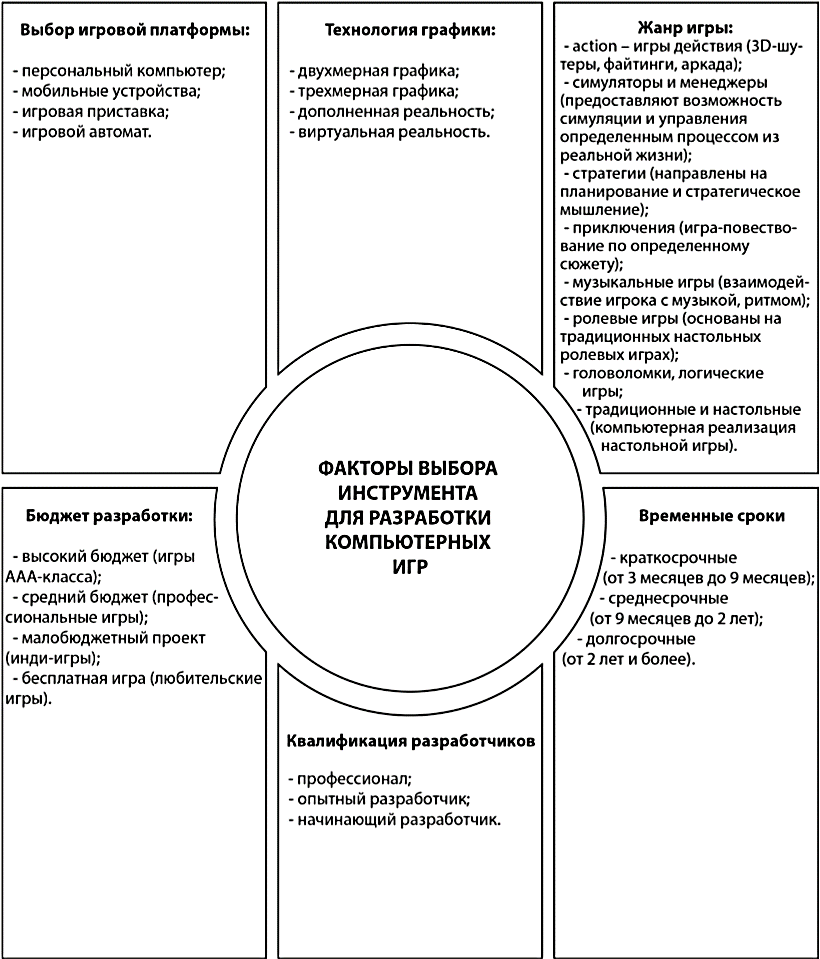


Рисунок 6. Факторы выбора игрового движка

В крупных компаниях для выбора средств разработки будущей игры зачастую привлекаются эксперты, либо проводятся собственные исследования, начинающему же разработчику приходится решать данную проблему самостоятельно.

По результатам работы планируется создание игровой модели окружения. Необходимо выбрать игровой движок, в котором будет проходить тестирование модели. На основе этих данных проведем сравнительный анализ популярных игровых движков:

1. Unity [12]
2. UnrealEngine [13]
3. CryEngine [14]

Ниже в Таблице 1 и 2 приведены основные характеристики и сравнительный анализ описанных выше игровых движков:

Таблица 1. Основные характеристики игровых движков

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | Unity | Unreal Engine | CryEngine |
| Создание игр | 2D, 3D | 2D, 3D | 3D |
| Физический движок | PhysX | PhysX | Soft-body |
| Поддержка искусственного интеллекта | Есть | Есть | Есть |
| Мультиплеер | Есть | Есть | Есть |
| Возможность создания открытого мира | Есть | Есть | Есть |
| Сценарный язык | C#, Java Script | C++, редактор Blueprint | Lua Script |
| Магазин ассетов | Unity Asset Store | Unreal Engine Marketplace | CryEngine Marketplace |
| Наличие бесплатной лицензии | Есть | Есть | Есть |
| Уровень усвоения | Низкий | Средний | Высокий |
| Поддержка платформ | Все | Все основные | Основные |
| Примеры игр на движке | Rust, Cities skylines, Hearthstone | Unreal Tournament, Fortnite, Lineage 2: Revolution | Crysis, ArcheAge, FarCry |

Таблица 2. Сравнительный анализ игровых движков

|  |  |  |  |
| --- | --- | --- | --- |
|  | Unity | Unreal Egine | CryEngine |
| Достоинства | Мульти  платформенность | Большое количество инструментов, постоянное добавление новых | Продвинутые инструменты визуализации и графики |
| Интеграция с сервисами вовлечения и удержания игроков | Полный доступ к исходному коду | Полный доступ к исходному коду |
| Высокий уровень технической поддержки | Высокий уровень технической поддержки |  |
| Широкое сообщество и обилие обучающего материала | Широкое сообщество и обилие обучающего материала |  |
| Поддерживает большинство форматов импорта | Кинематографические эффекты постобработки |  |
| Простой настраиваемый интерфейс, автоматическое обновления связи при изменении файлов внутри проекта | Встроенный редактор визуального скриптинга Blueprint |  |
| Широкий выбор активов в Unity Asset Store (более 10 тыс. бесплатных и платных) |  |  |
| Недостатки | Набор готовых инструментов ограничен (необходимо разрабатывать самому либо покупать) | Для разработки игры необходим очень мощный компьютер | Для разработки игры необходим очень мощный компьютер |
|  |  | Высокий порог вхождения |

**Unity** - игровой движок для разработки двухмерных и трехмерных приложений и игр под различные платформы, такие как: Windows, OS X, Windows Phone, Android, Apple iOS, Linux, Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One. Также можно попытаться сделать игру под Web Player, но его уже редко используют, так что из списка вычеркиваем. Xbox 360, как и PlayStation 3, тоже можно вполне вычеркнуть, так как эти консоли уже не потянут мощные проекты, а на инди игрушки на Xbox 360 и PS3 спрос невелик. Unity очень тесно сотрудничают с Oculus, Apple и рядом других компаний, поэтому быть Unity Developer очень хорошо.

Привыкание к интерфейсу Unity происходит за неделю. Полностью осваивается движок примерно за полгода, год. Единственное, что мешает в Unity, так это очень странное приближение к объекту.  Скроллинг мышью позволяет то сильно быстро отдаляться от объекта, то слишком медленно приближаться, что не очень хорошо. Это, конечно, не проблема, если клацать 2 раза по нужному объекту в Hierarchy, но если нужна точность фокусировки, то в Unity пока с этим проблемы. Вообще Unity разбит на несколько окошек: Hierarchy, где можно наблюдать названия объектов на сцене, группировать их, Scene, Inspector ( настройки выделенного объекта ), Project, где видны все материалы проекта, ToolBar, т.е. меню с инструментами.  
Расчеты физики производит NVIDIA PhysX.

Unity можно скачать абсолютно бесплатно. Эта версия не будет демо-версией или на определенное время, с ее помощью можно создать свой собственный проект. Только выставить платным в маркет нельзя, придется ставить рекламу в приложение или игру. Если доходы будут больше 100.000$, то необходимо будет купить PRO версию за 3000$ или 75$/месяц. С одной стороны, это грамотный шаг, чтобы разработчики получали нормальный доход, а с другой стороны – устаревшая модель распространения. Epic Games, в отличие от Unity, заявили, что если вам что-то нравится, set it free, т.е. одним из мощнейших движков можно теперь пользоваться бесплатно! Единственное - делиться 5% от продажи копии игры. Это разумнее, чем столько не нужных действий с Unity, Epic Games не останавливает удивлять, если вы нуждаетесь в средствах, то Epic Games предоставит 5000$ для выпуска проекта в массы.  
Unity 5 использует C# и JavaScript. С этими языками особых проблем в Unity нет. Раньше Unity 4 использовал Boo, но никто им не пользовался, поэтому команда разработчиков убрала этот язык из 5-ки.  
На данный момент Unity имеет самое большое коммьюнити. В российском сегменте unity является самым популярным. По данным из интернета, Unity используют более 50% разработчиков. И всё больше разработчиков покидают старые игровые движки и переходят делать игры в Unity.  Но если изучить статистику подробнее, в Unity более 60% - это начинающие программисты, которые не выпустили ещё ни одной игры. 20% людей повелись на рекламу в сообществах, 10% людей изучили интерфейс Unity и «учат» делать «игры» на YouTube, а вот остальные 10% - это те самые Unity Developers, которые по-настоящему делают игры.

Unity имеет очень мощный магазин Unity Asset Store (только из-за этого магазина можно спокойно выбирать Unity) Здесь есть абсолютно все: от текстур и 3D моделей до готовых проектов. Такой огромный ассортимент ни у одного конкурента не найти. Магазин Asset Store выдвигает Unity в лидеры. Теперь о сборках:

Unity очень быстро собирает сцены. Если нужно будет быстро проверить тот или иной момент в игре, а в окне Scene это не всегда удобно, то смело собираем проект, и через 1-2 минуты у нас готовая папка с исполняющим файлом внутри. Что же касается оптимизации, то в Unity не лучшая оптимизация под PC и Mac. ААА-проекты, вроде Mortal Kombat X, Tom Clansy: The Division лучше не пробовать создавать на Unity. Все-таки данный движок создан для инди-проектов, а не для жестоких стрелялок. Тот же самый Rust, The Forest, если бы они были сделаны на Unreal Engine, то они были бы более детализированней с той же производительностью. Создавать фильмы или кат-сцены лучше в другом редакторе, Unity потребует очень мощный ПК.  
В Unity3d есть ряд проблем. Во-первых, в Unity 5 странный мультиплеер. Например, в The Forest до сих пор не доделанный мультиплеер, а связано это с тем, что настраивается он очень не просто. Есть и хороший пример - это HeartStone. Во-вторых, бесплатная версия не позволяет выложить проект под все платформы, нужно будет брать Full Version за 75$/месяц или 3000$. В-третьих, ограниченный набор инструментов. Придется создавать некоторые дополнения к движку. Или докупать в Asset Store.

Минимальные требования для запуска программного продукта Unity:

* ОС: Windows 2000, XP, Vista, 7, Mac OS 10.4.5 (Intel) или новее
* Процессор: процессор Dual core с тактовой частотой 1,8 ГГц или лучше
* Оперативная память: 1024 МБ или лучше
* Жесткий диск: 10 ГБ свободного места
* Видеокарта: Nvidia GTX560 1024 mb видеопамяти или лучше
* DirectX®: 9.0c

**CryEngine 5.5** - игровой движок, разработанный компанией Crytek. CryEngine, в отличие от других игровых движков, нацелен на создание игр для PC и консолей. Crytek сделали очень мощный движок, который выдает лучшую графику из всех движков. Здесь есть и динамический свет, и затенение и затуманивание в реальном времени, и управление уровнем детализации ландшафта. Если нужна внешняя составляющая игры, то CryEngine сделает это лучше всех. Лица и персонажи в CryEngine получаются по-настоящему захватывающими. А вот несколько примеров игр, сделанных на CryEngine: Crysis, Far Cry, Warface, Star Citizen, Ryse: Son of Rome, Homefront: The Revolution, Evolve, Проект Армата, Kingdom Come: Deliverance.

Минимальные требования для запуска программного продукта CryEngine 5.5:

* ОС: Windows 2000, XP, Vista, 7, Mac OS 10.4.5 (Intel) или новее
* Процессор: процессор Core i5 с тактовой частотой 2,4 ГГц или лучше
* Оперативная память: 4096 МБ или лучше
* Жесткий диск: 15 ГБ свободного места
* Видеокарта: Nvidia GTX560 1024 mb видеопамяти или лучше
* DirectX®: 9.0c

**Unreal Engine 4** - игровой движок, созданный Epic Games. Unreal Engine 4 - самый популярный игровой движок для создания фильмов и ААА-проектов. Данный движок обладает высокими графическими возможностями. С Unreal Engine 4 можно разрабатывать игры под PC, Mac, консоли, IOS, Android, Windows Phone. В отличие от Unity, UE4 имеет мощный инструмент для дизайна игровых уровней прямо в движке, невероятно удобную систему Blueprint, не имеющую аналогов, красивый дизайн самого движка, интуитивность в использовании и много других улучшений. Из всех игровых движков, Unreal Engine 4 является самым инновационным. Он сочетает в себе высокую производительность при сборке, лучшую графику, простой язык программирования и удобность в использовании. Есть очень мощное коммьюнити, помогающее решить все проблемы. Epic Games раздает UE4 абсолютно бесплатно, весь исходный код открыт, а все, чем мы должны отблагодарить Epic Games, так это 5% с продажи одной копии. Если собрались делать фильм, то платить и вовсе не нужно. Команда Epic тесно сотрудничает с разработчиками VR технологий, а значит, что уже сейчас можно создавать игры под PlayStation VR, Oculus и т.д.

На сайте Unreal Engine есть много документации. Также есть раздел для тех, кто перешел с Unity3d. В российском сегменте есть множество форумов и ВК-групп, в которых с радостью готовы помочь. На YouTube естьбольшое разнообразие различных видео, где молодые геймдеверы расскажут об интерфейсе, азах скриптинга, правильной сборке и т.д. Магазин в Unreal Engine, честно говоря, отстает от конкурентов. Во-первых, здесь слишком мало контента, во-вторых, цены здесь баснословные, кусающиеся. Как и говорил ранее, Unity Asset Store пока лидирует, и Epic Games есть к чему стремиться.

На Unreal Engine 4 было сделано огромное количество ААА-проектов, таких как: Mortal Kombat X, Dead Island 2, Smite, Paragon, Infinity Blade.

Минимальные требования для запуска программного продукта Unreal Engine 4:

* ОС: Windows 2000, XP, Vista, 7, Mac OS 10.4.5 (Intel) или новее
* Процессор: процессор Dual core с тактовой частотой 1,8 ГГц или лучше
* Оперативная память: 2048 МБ или лучше
* Жесткий диск: 15 ГБ свободного места
* Видеокарта: Nvidia GTX560 1024 mb видеопамяти или лучше
* DirectX®: 9.0c
  1. **Исследование существующих методик создания трехмерных объектов**
     1. **Моделирование геометрии трехмерного объекта**

Полигональное моделирование дает возможность производить различные манипуляции с сеткой 3d объекта на уровне подобъектов: вершин, ребер, граней. Сам полигон состоит из граней, но в системах, которые поддерживают многосторонние грани, полигоны и грани будут равнозначны.

Это самый первый и основной вид моделирования, так как при помощи его можно создать объект любой сложности путем соединения групп полигонов.

Полигональное моделирование подразделяется на три типа: низкополигональное, среднеполигональное и высокополигональное.

1. Низкополигональное моделирование (Low-Poly) предназначено для создания объектов с небольшим числом полигонов обычно для экономии ресурсов, когда не требуется высокая детализация, а также для создания низкополигональных иллюстраций, которые набирают большую популярность в последнее время;
2. Среднеполигональное моделирование (Mid-Poly) ориентировано обычно только на необходимый результат при рендеринге, то есть при моделировании нужной геометрии, например, с применением булевых операций; над полигональной сеткой никакие работы по её оптимизации не производят, или они минимальны;
3. Высокополигональное моделирование (High-Poly) представляет собой создание объекта с большим числом полигонов, обычно, приближенной его копии.

Стандартная схема High-Poly моделирования происходит с постепенным наращиванием уровня детализации 3d объекта:

* первый уровень является базовым, и представляет собой общую форму объекта;
* на втором уровне происходит уточнение базовой формы, обычно, путём добавления фасок;
* третий уровень завершающий, то есть на нем производится четкая детализация объекта, обычно, путем применения плагинов сглаживания.

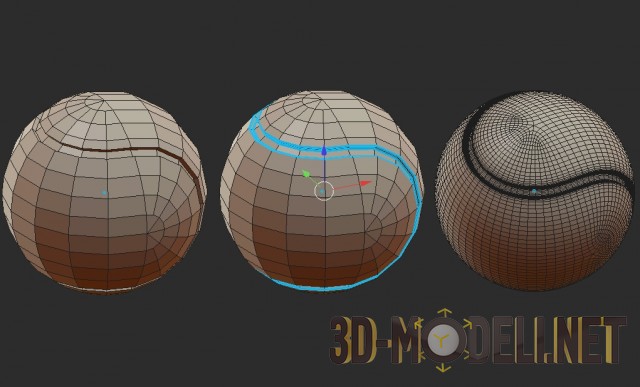


Рисунок 7. На рисунке предоставлены все вышеперечисленные уровни при High-Poly моделировании на примере теннисного мяча.

Сплайновое моделирование

Сплайновое моделирование представляет собой создание 3d объектов при помощи кривых линий (сплайнов). Сплайнами могут выступать линии различной формы: окружности, прямоугольники, дуги и т.д. Объекты при этом получаются плавной формы, в связи с чем, данный метод получил широкое применение в создании органический моделей, таких как растения, люди, животные и т.д.

Преимущество данного метода в гибкости изменения формы сплайна.

Данный вид моделирования часто сравнивают с полигональным, как векторную графику с растровой. Преимущество векторной графики в том, что при увеличении объекта, его качество не изменяется, в отличие от растрового, где становятся видны пиксели. Так же и при увеличении объекта, созданного сплайнами, его качество останется неизменным, а при полигональном моделировании будут уже видны полигоны.

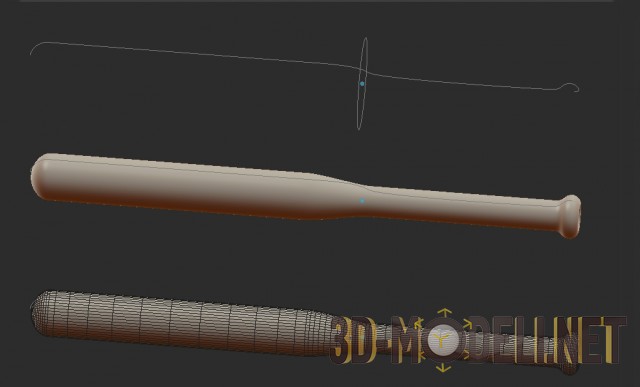


Рисунок 8. Пример построения бейсбольной биты при помощи сплайнов

NURBS моделирование

NURBS расшифровывается как «Non-Uniform Rational B-Spline», и представляет собой технологию создания 3d объектов при помощи специальных кривых, которые называются B-сплайнами. Некоторые специалисты выделяют данный вид моделирования в отдельный, а некоторые – в подвид сплайнового моделирования.

Принцип моделирования состоит в следующем: при помощи B-сплайнов, расположенных по вертикали и горизонтали, строится нужная форма объекта, а затем все это соединяется при помощи полигонов.

Существуют две разновидности этого моделирования:

1. при помощи P-кривых (Point), форму которых можно изменять при помощи вершин, которые расположены непосредственно на самой линии;
2. при помощи CV-кривых (Control Vertex), форму которых можно изменять при помощи вершин, которые расположены за пределами линии.

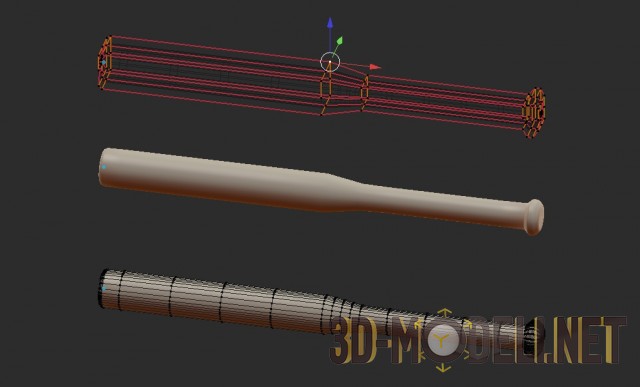


Рисунок 9. Пример биты, созданной при помощь P-кривых методом NURBS

Скульптинг

3d-скульптинг он же «цифровая скульптура» представляет собой имитацию процесса «лепки» 3d модели из пластилина. Только в программах 3d моделирования пальцы заменены на инструмент «кисть», а «пластилином» является полигональная сетка.

Но это - сотни тысяч, миллионы полигонов



Рисунок 10. Пример трехмерного объекта, выполненного с помощью Digital Sculpting

Твердотельное моделирование

Если при полигональном моделировании куб разрезать пополам, то там внутри будет пустота. При твердотельном моделировании, если разрезать куб, то там не будет пустоты, как если бы разрезали реальный твердый предмет.

При построении модели работают сразу со всей оболочкой, а не с отдельными поверхностями. Сначала создается простая форма оболочки, например, сферы, а затем к ней применяют различные операции: резка, объединение с другими телами, булевые операции и др.

Твердотельное моделирование идеально подходит для создания твердых 3d моделей несложной формы: шестеренок, двигателей, и т.д., но не применим к созданию мягких: мятой одежды, животных и т.д.



Рисунок 11. Пример твердотельного моделирования

При создании сложных моделей вышеописанные методы моделирования часто используются совместно, так как это ускоряет процесс моделирования. Так, например, при создании персонажа для игр используется 3d-скульптинг, с помощью которого прорисовываются необходимые мелкие детали, а затем на её основе создается Low-Poly модель полигональным моделированием.

**Построение UV-развертки трехмерного объекта**

После моделирования объекта необходимо вдохнуть в него жизнь путем применения к нему некоторых материалов, но прежде, чем это сделать, нужно пройти процесс, который называется UV развертку. Это то, где мы сопоставляем координаты текстуры на объект. Это крайне важно, даже для самых простых предметов, но действительно необходимо на более сложных объектах, таких, как предметы мебели. Рассмотрим пару основных способов этого процесса. Процесс создания UV-развертки разберем в программе Blender.

Шаг 1: Модификатор UVW Map.

Самый элементарный способ назначить развёртку к текстурам простого объекта — это применить модификатор UVW Map. Таким образом, добавим модификатор UVW Mapping в наш набор модификаторов. Выберем его в наборе на панели модификаторов появятся редактируемые параметры.

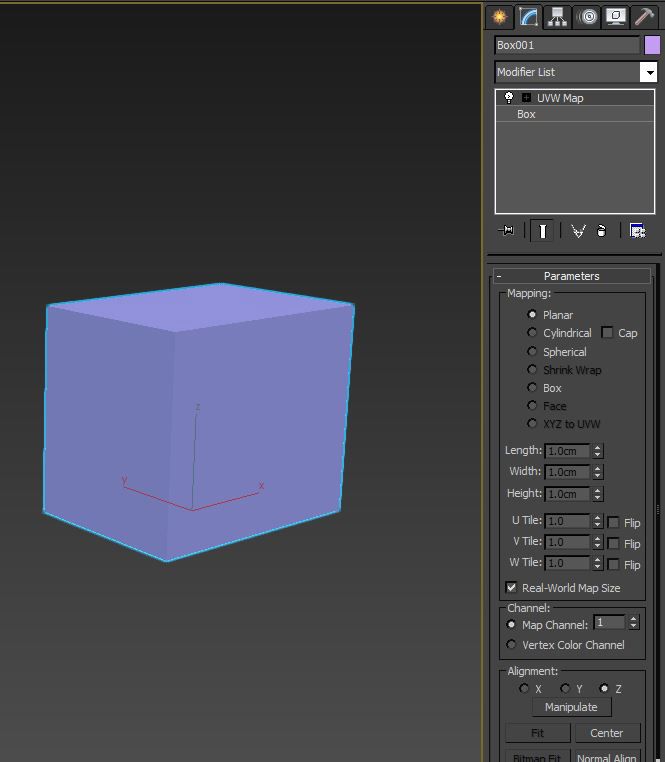


Рисунок 12. Модификатор подходит для большинства объектов

Шаг 2: Виды UVW развёртки

В начале свитка Parameters расположены виды развёртки Mapping. Среди них плоские развёртки Planar, которые могут покрыть только 2d координаты в пространстве U и V, как например коробка, когда координаты развёртки создаются по форме коробки в системе координат UVW. Существует также цилиндрический и сферический вид развёртки. Если есть довольно простой объект, то вполне можно выбрать тип Box. Для более сложных объектов необходимо использовать более сложные функции, которые называются unwrapping, но об этом чуть позже.

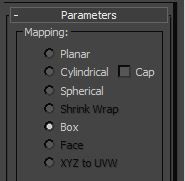


Рисунок 13. Выбор типа развертки

Шаг 3: Реальные размеры UVW развёртки

Мы можем задать текстурам реальные размеры. К примеру, есть узор ткани, и мы знаем, что в реальном масштабе он занимает 1 на 2 метра, и хотим, чтобы он покрывал объект в виртуальном мире, необходимо просто указать это в параметрах. Если использовать реальные размеры, то надо отметить флажок Real-World Map Size в модификаторе UVW Mapping.

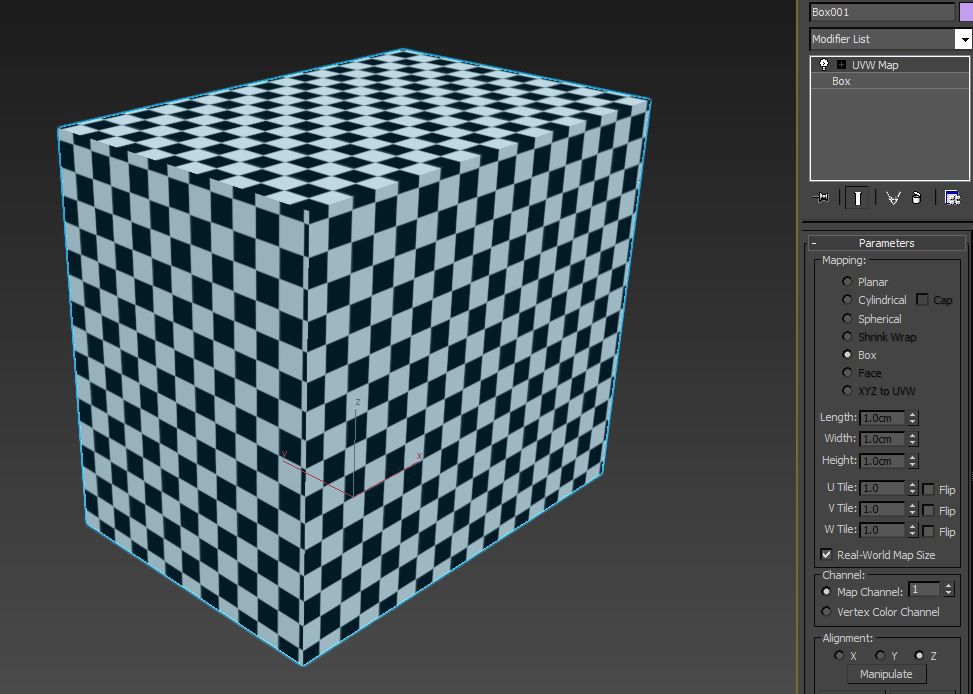


Рисунок 14. На этом примере каждая клетка равна квадрату в 5 см.

Шаг 4: Каналы развёртки UVW.

Прокрутив немного вниз, будет виден способ изменить канал развёртки. Чтобы понять, что это, представьте, что нам нужна определённая развёртка для нашей диффузной текстуры (цвет), а затем понадобилась другая развёртка для карты бампа (рельеф) или дисплейсмента (смещение). Это просто осуществить путем настройки текстур на разные каналы развёртки, а затем применения несколько модификаторов UVW Mapping, каждый из которых установлен на разный канал развёртки с различными свойствами.

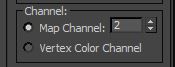


Рисунок 15. Каналы развертки позволяют иметь разные развертки для разных текстур

Шаг 5: UVW выравнивание.

Осталось рассмотреть ещё одну вещь, перед тем как перейти к такому страшному монстру, как развёртка. Модификатор UVW Mapping позволяет быстро применить выравнивание для развёртки в определённом направлении, X, Y или Z. Это особенно полезно при использовании типа развёртки Planar и определении особой плоскости.



Рисунок 16. Инструмент быстрого выравнивания позволяет быстро настроить ориентацию развертки

Шаг 6: Введение в модификатор Unwrap UVW.

Теперь следует перейти к следующему шагу. Ранее были ограничены несколькими параметрами, которые идеально подходили для простых моделей. Теперь же будем использовать процесс под названием «развёртка», который даст гораздо больше контроля, для расположения фейсов на текстуре. Перейдем к набору модификаторов и добавим модификатор Unwrap UVW.

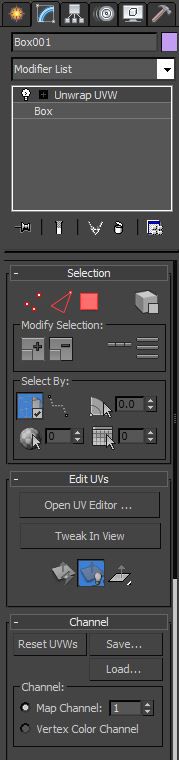
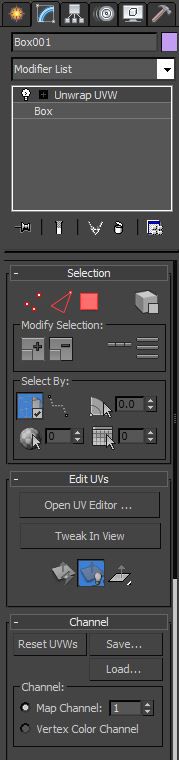


Рисунок 17. Более сложные объекты требует развертки, что бы текстуры правильно применились к геометрии

Шаг 7: Режим редактирования Unwrap UVW

Выбрав этот модификатор, пролистаем вниз к свитку Parameters и нажмем кнопку Edit. Откроется отдельный интерфейс с меню вверху и иконками вверху и внизу. Здесь важно понимать, что текстуры будут располагаться в квадрате, очерченном жирной линией. Это UV пространство от 0 до 1. Следует размещать фейсы объектов внутри этого пространства.

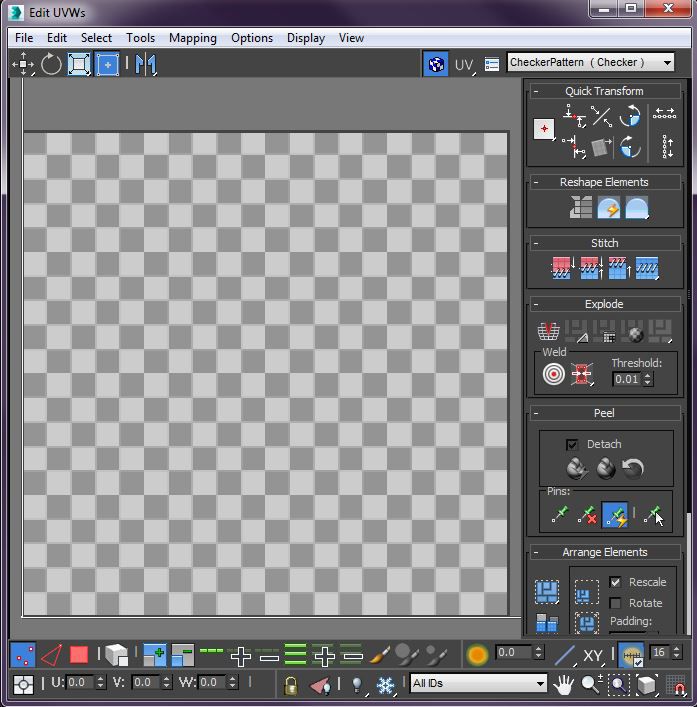


Рисунок 18. Режим редактирования дает все необходимые инструменты для развертки

Шаг 8: Отображение текстур Unwrap UVW.

По умолчанию отображается шахматный узор. Это может быть полезно, чтобы видеть есть ли растяжения на развертке. Для добавления настоящей текстуры необходимо перейти в выпадающий список сверху и выбрав текстуру Bitmap. Найти требуемое изображение и два раза кликните по нему. В результате изображение откроется в редакторе. Заметьте, если изображение не квадратное, то оно будет сдавлено в данном UV пространстве. Вот почему большинство текстур имеют квадратные пропорции, например, 1024x1024 пикселя.

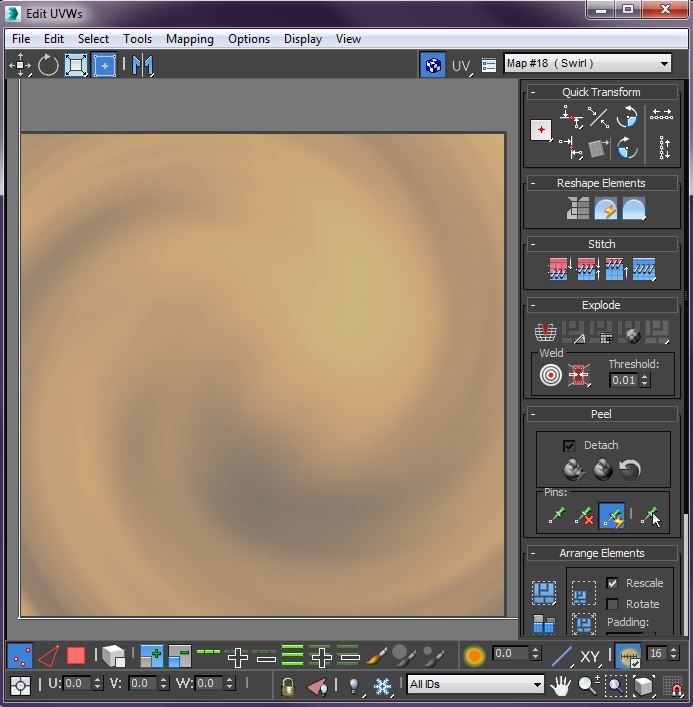


Рисунок 19. Отображение любой текстуры, поможет разложить развертку

Шаг 9: Flatten Mapping для Unwrap UVW.

Теперь рассмотрим наиболее востребованное решение для автоматической развёртки объектов. Оно называется Flatten Mapping и его можно найти в меню Mapping, выбрав Flatten Mapping. Прежде чем это сделать, используйте режим выделения полигонов внизу и нажмите комбинацию Ctrl+A, чтобы выделить все полигоны. Затем перейдите в Flatten Mapping. Выберите OK и увидите, как все грани объекта аккуратно распределились в пространстве.

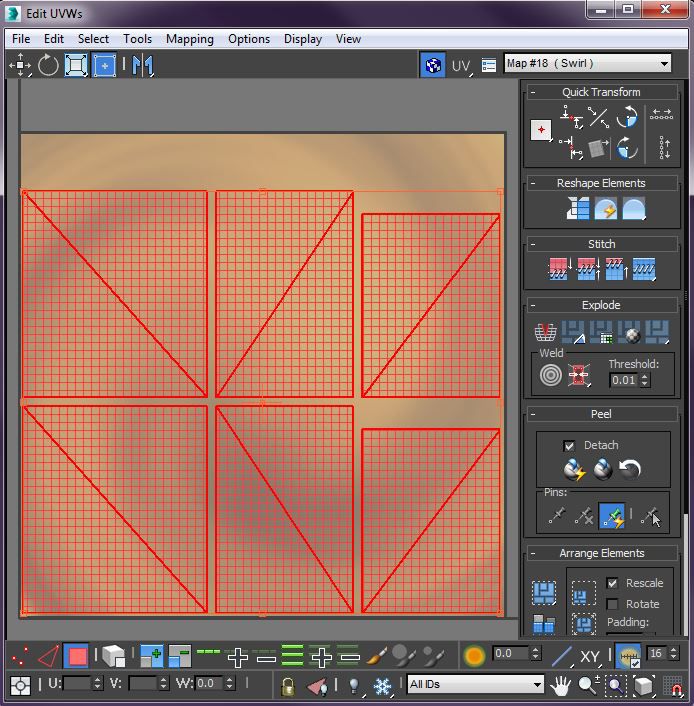
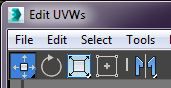


Рисунок 20. Быстро распределить грани в UV пространстве можно при помощи автоматического инструмента

Шаг 10: Ручные настройки Unwrap UVW.

После того, как фейсы размещены, скорее всего, захочется их переместить, повернуть и масштабировать по-своему. Это легко сделать при помощи значков в верхней панели. Они работают таким же образом, как и при действии с объектами при редактировании их в пространстве модели. Заметьте, что вы также можете воспользоваться другими режимами выделения, такие как вершины или рёбра, чтобы переместить ваши фейсы в нужные положения и придать им правильный размер.



ф

* + 1. **Материалы трехмерного объекта**

Рассматривается два подхода «классический» и «физически корректный рендеринг».

Классический – важной составляющей является «художественная выразительность». Результатом является «красиво», насколько талантливо художник может нарисовать, настолько и будет правильным. Такой подход приводит к тому, что одно и то же необходимо делать несколько раз, под разные условия. Лежащий предмет на земле в солнечный день – одна текстура. Тот же предмет, но под проливным дождем – другая.

При классическом подходе можно добиться высококлассного результата, но для этого требуется масса ручной работы. Прекрасно нарисованная текстура великолепно смотрится при одних условия, но кошмарно в любых других.

Физически корректный рендеринг (Physically Based Rendering) – означает что мы моделируем то, что свет делает в реальном мире, на основе реальных физических параметров. Конечным результатом является более точный и как правило, более естественный вид. PBR одинаково хорошо работает во всех средах освещения.

Параметры материалов. «Физическими» параметрами являются четыре свойства:

* Базовый цвет (Base color)
* Шероховатость (Roughness)
* Металичность (Metallic)
* Отражение (Specular)

Базовый цвет просто определяет общий цвет материала. Он принимает в значение Vector3 (RGB), и каждый канал автоматически варьируется между 0 и 1.

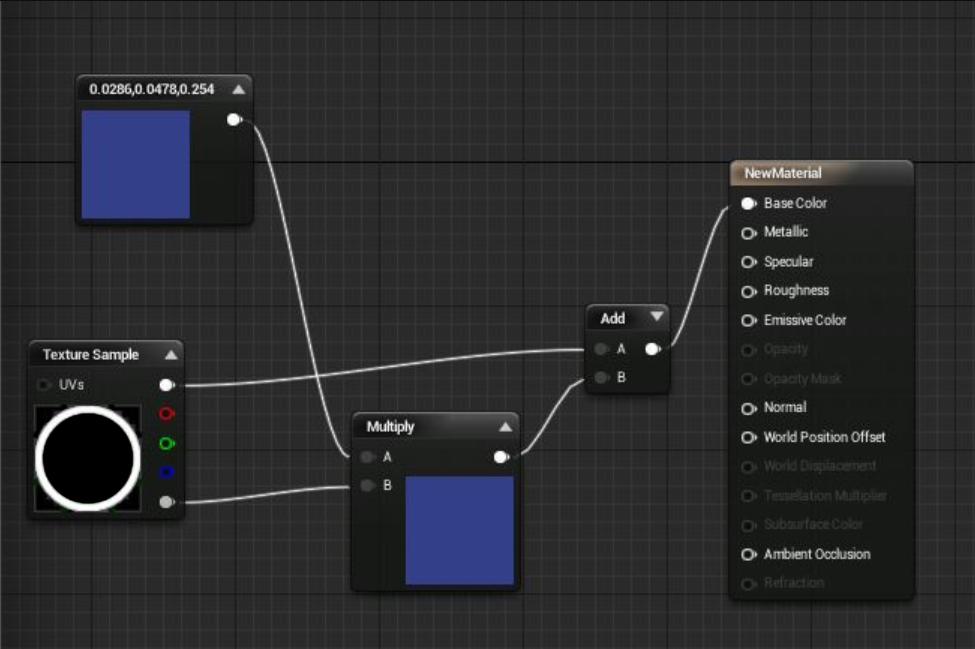


Рисунок 21. НОДа базового цвета

Измеренные значения Base Color для неметаллов (только интенсивность):

Таблица 3. интенсивность неметаллов

|  |  |
| --- | --- |
| **Материал** | **Интенсивность** |
| Древесный уголь | 0,02 |
| Свежий асфальт | 0,02 |
| Изношенный асфальт | 0,08 |
| Голая почва | 0,13 |
| Зеленая трава | 0,21 |
| Песок пустыни | 0,36 |
| Свежеуложенная бетонная смесь | 0,51 |
| Океанический лёд | 0,56 |
| Свежий снег | 0,81 |

Измеренные значения Base Color для металлов:

Таблица 4. Интенсивность металлов

|  |  |
| --- | --- |
| **Материал** | **Цвет** |
| Железо | (0,560, 0,570, 0,580) |
| Серебро | (0,972, 0,960, 0,915) |
| Алюминий | (0,913, 0,921, 0,925) |
| Золото | (1,000, 0,766, 0,336) |
| Медь | (0,955, 0,637, 0,538) |
| Хром | (0,550, 0,556, 0,554) |
| Никель | (0,660, 0,609, 0,526) |
| Титан | (0,542, 0,497, 0,449) |
| Кобальт | (0,662, 0,655, 0,634) |
| Платина | (0,672, 0,637, 0,585) |

Шероховатость буквально управляет тем, насколько шероховат материал. Грубый материал будет рассеивать отраженный свет в нескольких направлениях. Roughness 0 (гладки) является зеркальным отражением и Roughness 1 (грубый) полностью матовый.

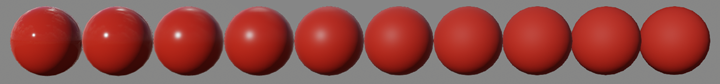


Рисунок 22. Roughness неметалл

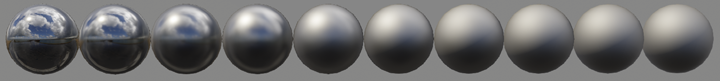


Рисунок 23. Roughness металл

Metallic управляет тем, насколько металлическая поверхность. Значение 0 для неметаллов и значение 1 для металлов. Для чистых поверхностей, таких как чистый металл, натуральный камень, чистый пластик и т.д. Это значение будет 0 или 1, а не что-то между ними. При создании гибридных поверхностей, например, коррозийных, пыльных и ржавых металлов, можно обнаружить, что нужно некоторое значение в диапазоне от 0 до 1.

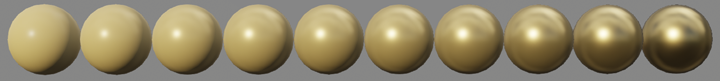


Рисунок 24. Metallic от 0 до 1

Specular как правило не используется, значение по умолчанию 0,5 для большинства случаев. Значение от 0 до 1 используется для масштабирования, текущего количество зеркальности на неметаллических поверхностях. Этот параметр не оказывает никакого влияния на металлы. Обычно, если изменить Specular, добавиться микрозатенение.

Таблица 5. Значение Specular

|  |  |
| --- | --- |
| **Материал** | **Specular** |
| Стекло | 0.5 |
| Пластик | 0.5 |
| Кварц | 0.570 |
| Лёд | 0.224 |
| Вода | 0.255 |
| Молоко | 0.277 |
| Кожа | 0.35 |

* + 1. **Рельефное текстурирование**

Рельефное текстурирование очень напоминает обычный процесс наложения текстуры на полигон. Только при обычном наложении текстур мы работаем с цветом и изменяем его цветовое восприятие, а вот при рельефном текстурировании мы добавляем ощущение рельефа, объемности плоскому полигону. Эта техника может добавить детализацию объекта без создания дополнительных полигонов. Полигон по-прежнему останется плоским, но создастся ощущение его выпуклости.

Рассмотрим куб на рисунке 22, создается впечатление что он состоит из множества мелких полигонов, обрисовывающих чеканку и клепки. Но в действительности этот куб состоит из 6 плоских четырехугольных полигонов. Вы можете задать вопрос, а чем же это отличается от обычного текстурирования. Ответ - рельефное текстурирование отражает реальное положение источника света в сцене и даже изменение его местоположения.

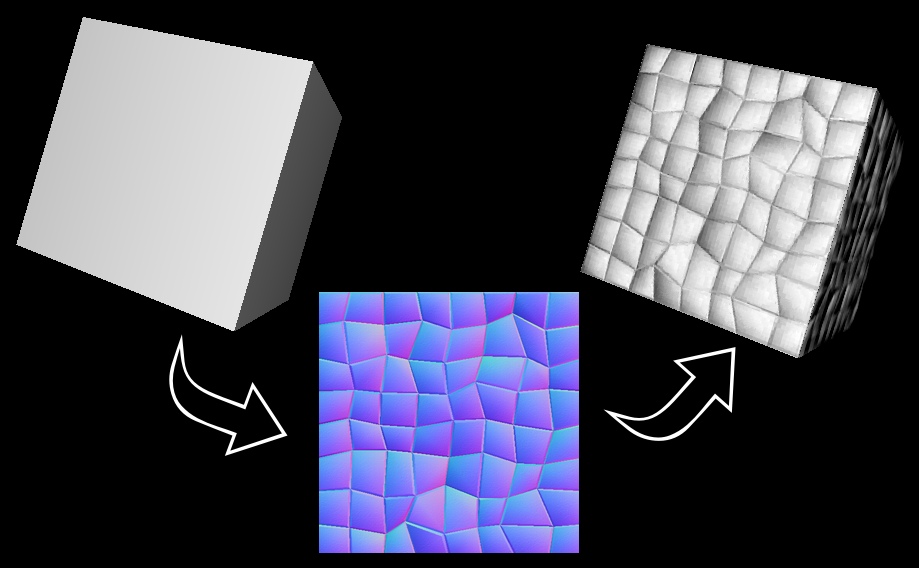


Рисунок 25. Рельефная текстура

Рельефная карта (текстура) — это обычная текстура, только в отличие от первой, несущей информацию о цвете определенных участков, рельефная карта несет информацию о неровностях. Самый распространенный способ представить неровности — это применить карту высот. Карта высот — это текстура в оттенках серого, где яркость каждого пикселя представляет, насколько он выдается из базовой поверхности. Очень простой и удобный метод. Его легко реализовать.

Так же имеется несколько способов достижения данного результата.

Bump mapping - простой способ создания эффекта рельефной поверхности с детализацией большей, чем позволяет полигональная поверхность. Эффект главным образом достигается за счёт освещения поверхности источником света и чёрно-белой (одноканальной) карты высот, путём виртуального смещения пикселя (как при методе Displacement mapping) как если бы там был вертекс (только без физического и визуального сдвига), за счёт чего таким же образом изменяется ориентация нормалей использующихся для расчёта освещённости пикселя ([затенение по Фонгу](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D1%82%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%A4%D0%BE%D0%BD%D0%B3%D1%83)), в результате получаются по-разному освещённые и затенённые участки. Как правило Bump mapping позволяет создать не очень сложные бугристые поверхности, плоские выступы или впадины, на этом его использование заканчивается. Для более детальных эффектов впоследствии был придуман Normal mapping.

Normal mapping — техника, позволяющая изменять нормаль отображаемого пикселя основываясь на цветной карте нормалей, в которой эти отклонения хранятся в виде текселя, цветовые составляющие которого [r,g,b] интерпретируются в оси вектора [x, y, z], на основе которого вычисляется нормаль, используемая для расчёта освещённости пикселя. Благодаря тому, что в карте нормалей задействуются 3 канала текстуры, этот метод даёт большую точность, чем Bump mapping, в котором используется только один канал и нормали, по сути, всего лишь интерпретируются в зависимости от «высоты».

Карты нормалей обычно бывают двух типов:

object-space — используется для недеформирующихся объектов, таких как стены, двери, оружие и т. п

tangent-space — применяется для возможности деформировать объекты, например, персонажей, органику.

Для создания карт нормалей обычно используется высокополигональная и низкополигональная модели, их сравнение даёт нужные отклонения нормалей для последней.

Parallax mapping - Данная технология также использует карты нормалей, но, в отличие от normal mapping, она реализует не только освещение с учётом рельефа, но и сдвигает координаты диффузной текстуры. Этим достигается наиболее полный эффект рельефа, особенно при взгляде на поверхность под углом.

Parallax occlusion mapping является усовершенствованной и в то же время одной из наиболее вычислительно сложных разновидностей Parallax mapping. Фактически представляет собой форму локальной [трассировки лучей](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D1%81%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BB%D1%83%D1%87%D0%B5%D0%B9) в пиксельном шейдере. Трассировка лучей используется для определения высот и учёта видимости текселей. Иными словами, данный метод позволяет создавать ещё большую глубину рельефа при небольших затратах полигонов и применении сложной геометрии. Недостаток метода — невысокая детализация силуэтов и граней.

Реализовать Parallax occlusion mapping возможно в рамках функциональности API [DirectX 9](https://ru.wikipedia.org/wiki/DirectX_9" \o "DirectX 9) [Shader Model](https://ru.wikipedia.org/wiki/Shader_Model" \o "Shader Model) 3, однако для получения оптимальной производительности видеокарта должна обеспечивать надлежащий уровень скорости исполнения ветвлений в [пиксельном шейдере](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%88%D0%B5%D0%B9%D0%B4%D0%B5%D1%80). На данный момент Parallax occlusion mapping используется в некоторых [компьютерных играх](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), например, [Crysis](https://ru.wikipedia.org/wiki/Crysis" \o "Crysis), [Metro 2033](https://ru.wikipedia.org/wiki/Metro_2033" \o "Metro 2033) и [ArmA 2](https://ru.wikipedia.org/wiki/ArmA_2" \o "ArmA 2). Также эта технология используется в популярном [бенчмарке](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BD%D1%87%D0%BC%D0%B0%D1%80%D0%BA_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)" \o "Бенчмарк (компьютер)) 3DMark Vantage.

Displacement mapping – Эта техника, в отличие от описанных выше, изменяет геометрию поверхности по заданной карте высот, обычно передающейся в вершинный [шейдер](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%B9%D0%B4%D0%B5%D1%80) через текстуру. Преимущество в том, что [освещение](https://ru.wikipedia.org/wiki/%D0%9E%D1%81%D0%B2%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5" \o "Освещение)считается обычным способом (пиксельный шейдер может быть практически любым), но требует высокую детализацию модели. Вероятно, будет использоваться в будущем.

* 1. **Результаты исследования**

На основе проведенного исследования методик и средств для создания трехмерного объекта. Выявлен комплекс необходимых программных инструментов для достижения поставленных задач в рамках формирования методики создания трехмерного объекта игрового окружения.

Мною был проанализирован и составлен перечень требуемых модулей:

* Blender - универсальный редактор трехмерного моделирования
* Моделирование
* UV – развертка
* Проверка Faces (вывернутые нормалей)
* Проверка Double vertex (задвоение вершин)
* ZBrush – цифровой скульптинг
* Цифровое скульптирование
* Ретопология
* Substance Painter – создание и нанесение материалов
* Текстурирование
* CryEngine 5.5 – игровой движок
* Функция импорта
* Функция экспорта
* Источник света
* Физические свойства объекта

Обоснование выбора программных средств:

1. Blender – бесплатный, имеется все необходимое
2. ZBrush – стабильный, многофункциональный
3. SubstancePainter – удобный в использование, не ресурсозатратный
4. CryEngine 5.5 – приятная картинка при настройке сцены

Стек технологий продемонстрирован на рисунке 22.

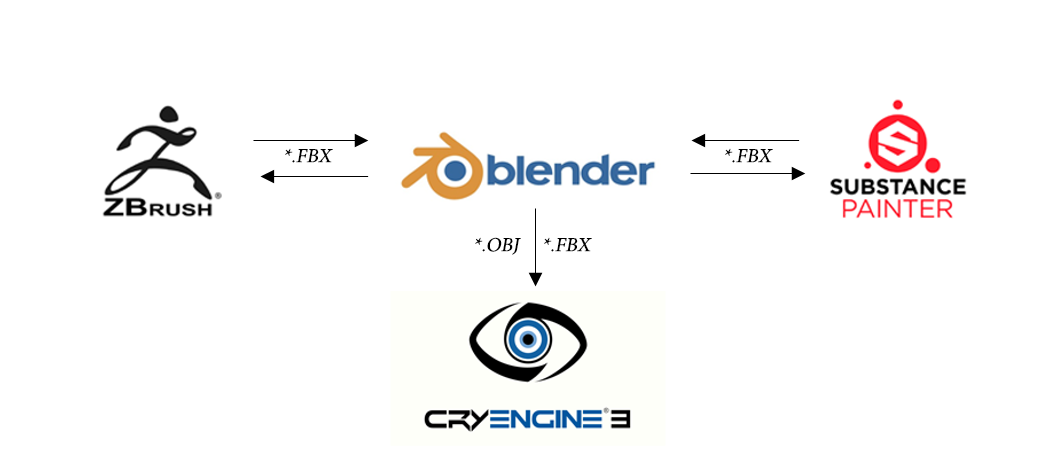


Рисунок 26. Стек используемых программ

С данным набором программных средств, мы сможем выполнить поставленные задачи.

1. **Формирование методики создания трехмерного объекта игрового окружения**
   1. **Выбор методики построения геометрии трехмерного объекта**

Возьмем для примера демонстрации работы методов при использовании разных средств достижения визуально одинакового результата фигуру с большим количеством граней.

В Blender создаем пустую сцену и добавляем объект Icosphere.

Исходная фигура – icosphere.



Сглаживание фигуры без использования средств:



При сглаживании исходной фигуры методом Subsurface, Edge Crease и методом Subsurface+Bevel получаем:





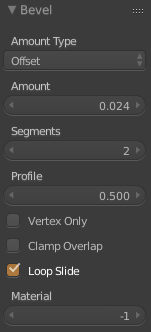
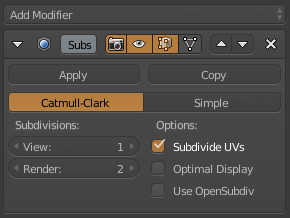
  

Рисунок 27. Параметры Bevel, Subsurf, Edge Crease

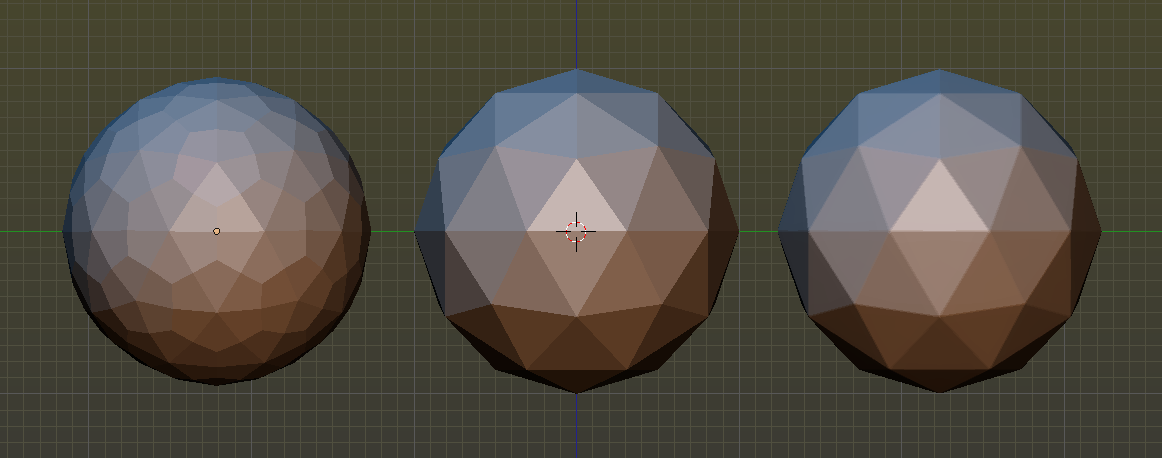


Рисунок 28. Icosphere

Запекание высокополигональной модели в низкополигональную для получения карты нормалей и работа напрямую с геометрией объекта:



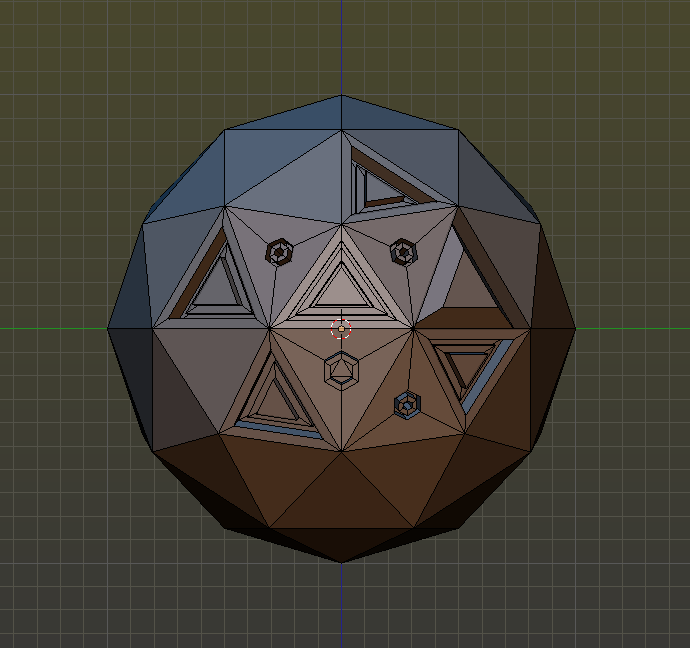
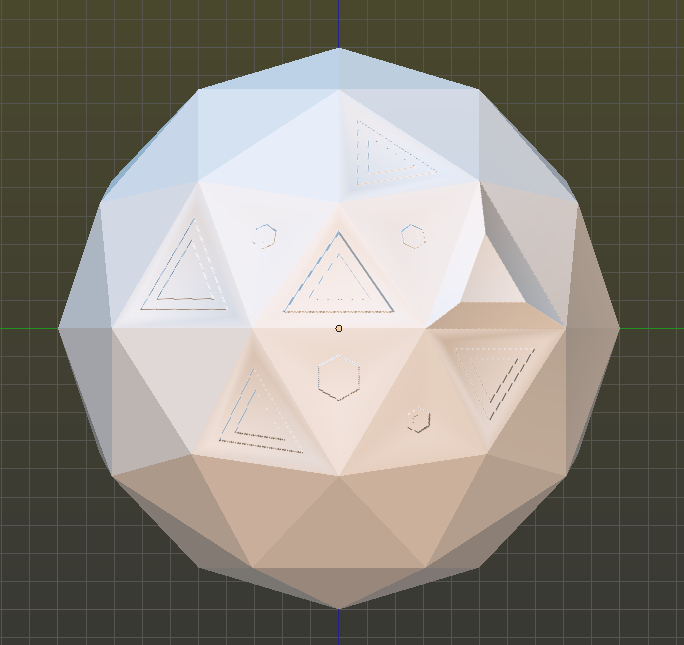


Рисунок 29. Icosphere с bump mapping

При запекании более сложной формы объекта и наложении на реалистичный материал, результат при визуальном восприятии одинаков.

Общий итог использования двух методов с разными средствами:



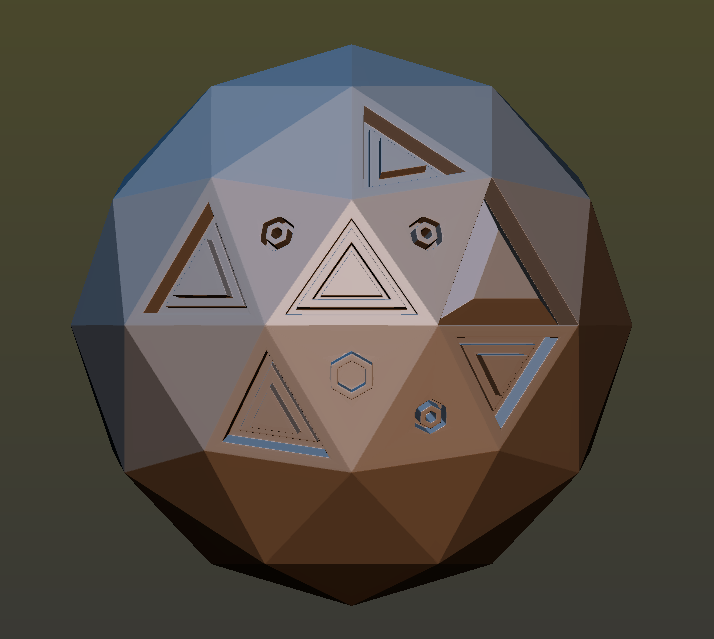
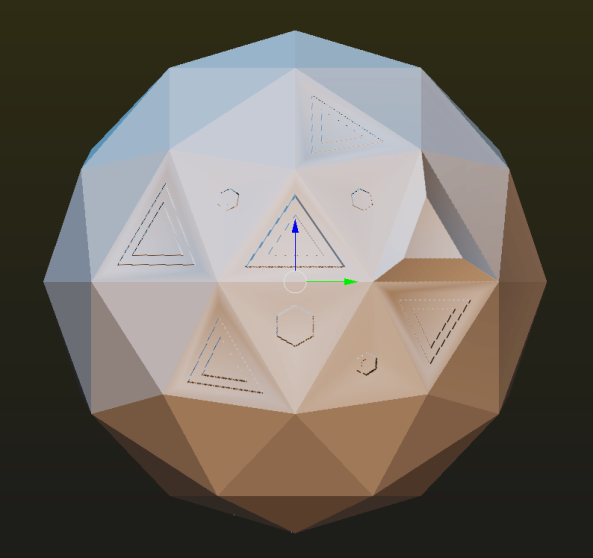


Рисунок 30. Normal map icosphere

Наглядное представление сеток геометрии объектов:

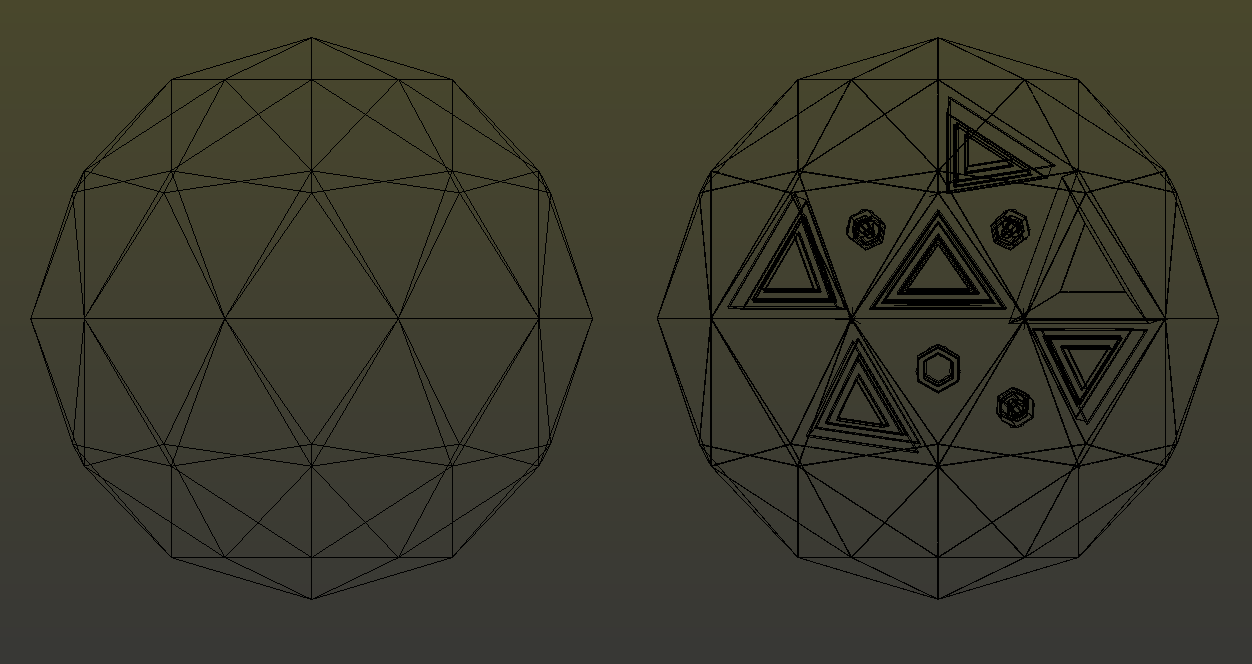


Рисунок 31. Полигональная сетка Icosphere

Количественное сравнение двух методов и их средств достижения результата:

|  |  |  |
| --- | --- | --- |
|  | Грани модели 1 | Грани модели 2 |
| Исходная модель | 120 | 120 |
| Грани | 480 | 4320 |
| Геометрия/ PBR | 120 | 588 |
| Грани+геометрия/PBR | 480 | 10584 |

Вывод.

При использовании разных средств для одного и того же метода мы можем получить требуемый результат, а именно хорошее визуальное представление, при снижении нагрузки на вычислительную технику, в нашем случае посредством уменьшения количества граней при сохранении геометрии объекта и использования PBR (normal map).

Самым экономным для ресурсов вычислительной техники оказался метод Edge Crease. Он позволяет держать грани объекта острыми, при общем сглаживании объекта, не искажая общую геометрию.

* 1. **Выбор методики построения UV-развертки трехмерного объекта**

Для выбора методики позаимствуем трехмерный объект другого художника.

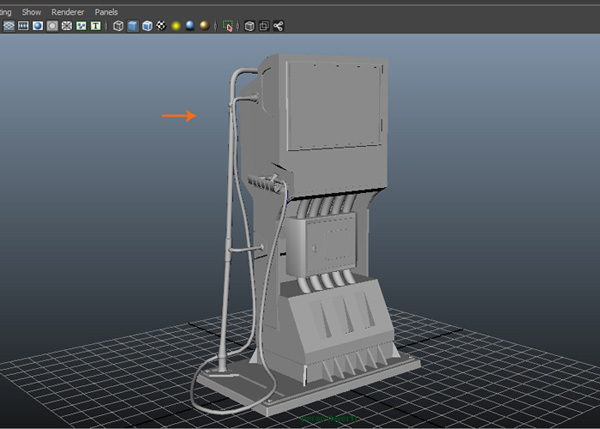
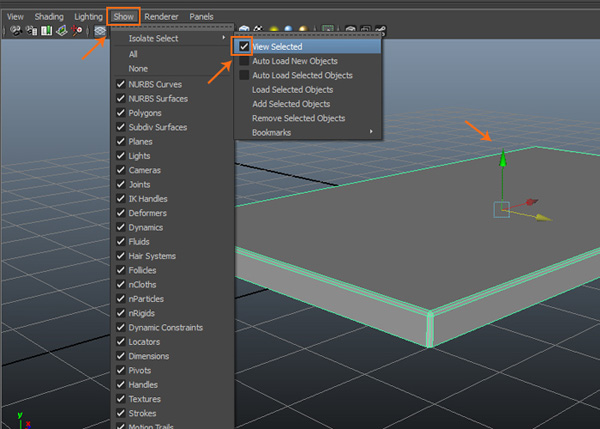


Рисунок 32. Бензаколонка

Выберем опорную плиту и изолируем ее, для того что бы не мешались составные части объекта. Show > Isolate Selected > View Selected



С выбранной плитой перейдем в режим редактирование UV-развертки. Edit UVs > UV Texture Editor и откроем UV редатор.

В окне редактора UV Texture Editor можем увидеть развертку UVs по умолчанию.

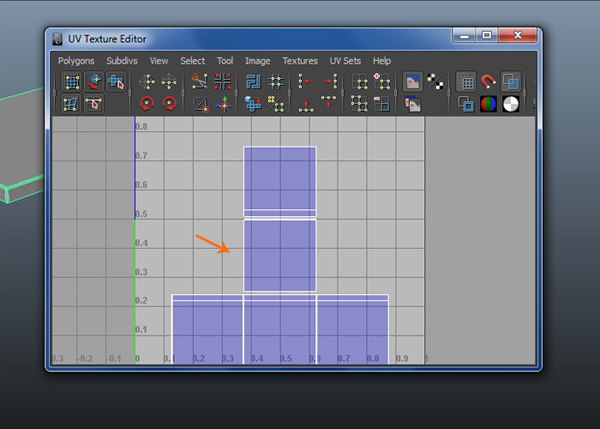


Рисунок 33.UV Texture Editor

С выбранной деталью перейдем в опцию CreateUVs > Planar Mapping

В окне Planar Mapping Options, переключим значение Fit projection to Best Plane и нажмем кнопку Project, чтобы применить наложение текстуры.

После применения команды планарного отображения в окне просмотра появится плоское отображение gizmo. Его можно регулировать по размеру, дергая за цветные маркеры.

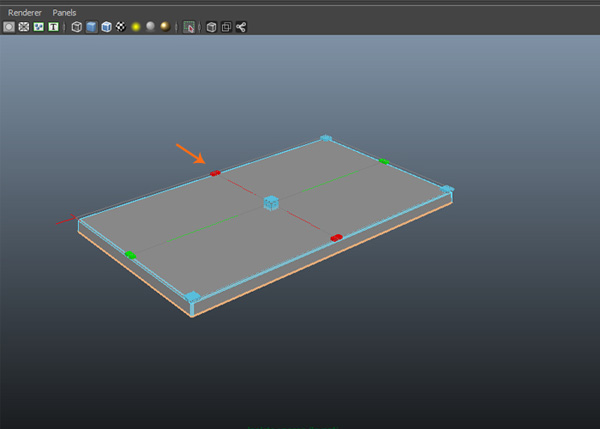
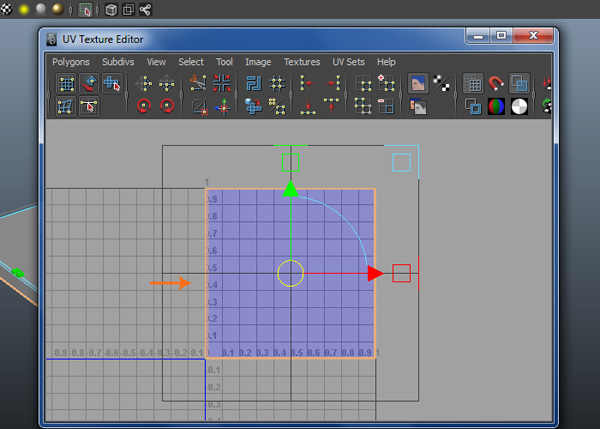


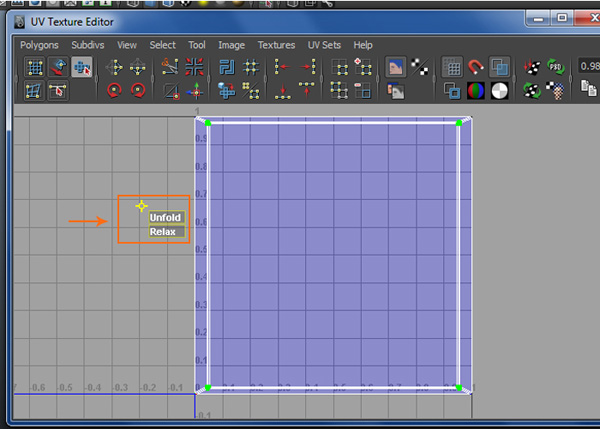
Рисунок 34. Gizmo

После этого можно увидеть нанесенное плоское отображение внутри окна UV Texture Editor в диапазоне X и Y координат 0,1.



С выбранной UV перейдем в панель опций меню Tool > Smooth UV Tool. В окне опций Smooth UV Tool Options, поставим галочки Pin Borders и нажмем Apply.

Теперь в окне UV Texture Editor будут доступны параметры Smooth UV Tool. Перетащим значение Unfold и Relax. Таким образом мы автоматизировано смягчили UV развертку для дальнейшего равномерного распределения текстуры. На этом завершим работу с данным объектом.



Вывод.

При ручном построении UV – развертки можно добиться хороших результатов, но это займет колоссальное количество времени при больших и геометрически сложных объектах. Поэтому лучше воспользоваться автоматизированными средствами построения Unfold и relax.

* 1. **Выбор методики рельефного текстурирования**

Рельефное текстурирование представлено в виде:

* Bump mapping
* Normal mapping
* Parallax mapping
* Parallax occlusion mapping

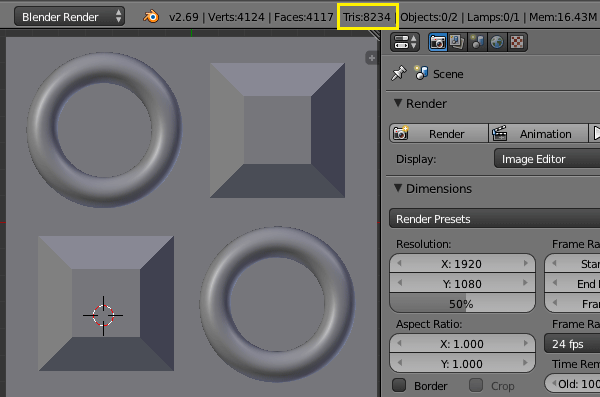
Bump mapping хорошо передает карту высот, но уже уступает другим методам получения карты нормалей.

Передовой технологией считается Parallax occlusion mapping, но также и требовательной к ресурсам вычислительной техники.

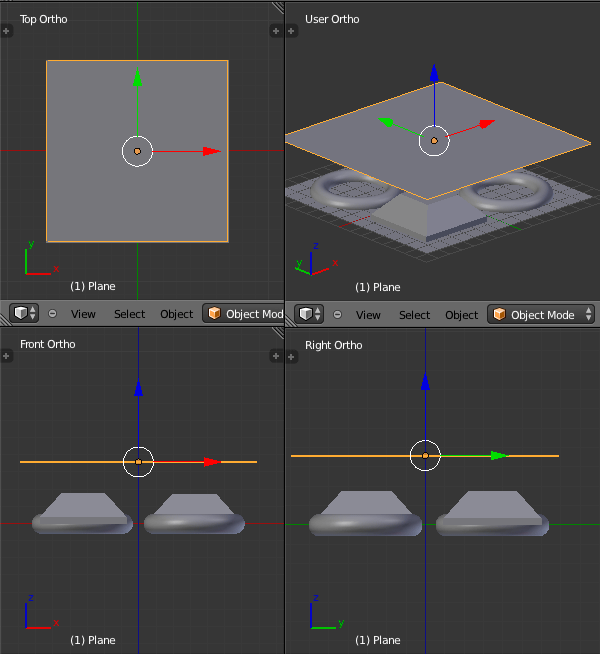
Normal mapping передает все необходимые детали объекта, но при этом держит потребление ресурсов на незначительном уровне по сравнение с Parallax и чуть больше чем Bump (Normal mapping можно использовать в связке с Bump для большей детализации, но в нашей задачи этого не требуется).

Создание Normal Map в Blender

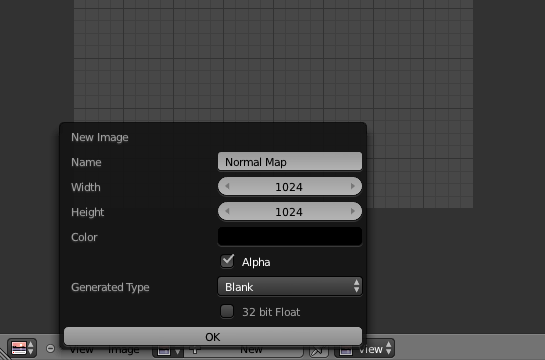
Возьмите любую высокодетализированую поверхность. Для примера, я буду использовать плоскость, на которой расположено несколько простых геометрических объектов. Как видите, она состоит из **8234** треугольников.



Теперь создадим плоскость таких же размеров и расположим ее над высокодетализированым объектом:

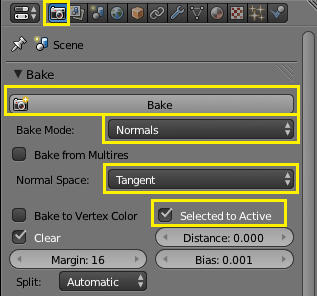


Разделим окно 3D вида на две части и в левой выберем **UV/Image Editor**. Создадим в нем новое изображение с настройками по умолчанию.

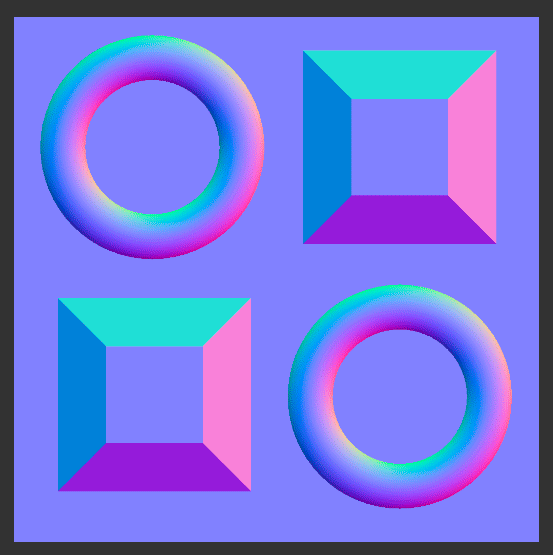


Выделим нашу плоскость, перейдем в режим редактирования и создадим для нее UV-развертку:

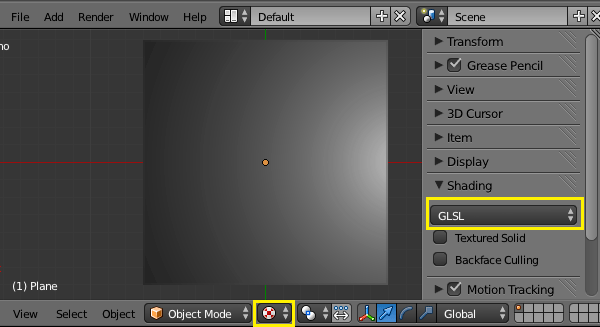
Выделим нашу высокополигональную модель, а затем удерживая **Shift** низкополигональную. На вкладке **Render** в самом низу у находится меню **Bake**. Выставим для него следующие настройки и нажмём кнопку **Bake**.



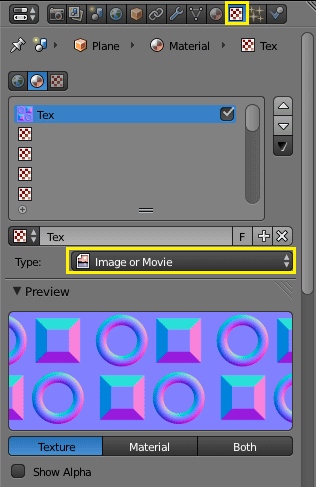
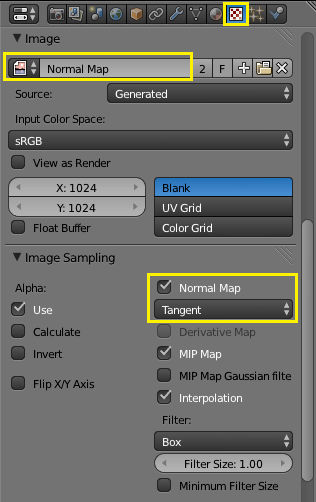
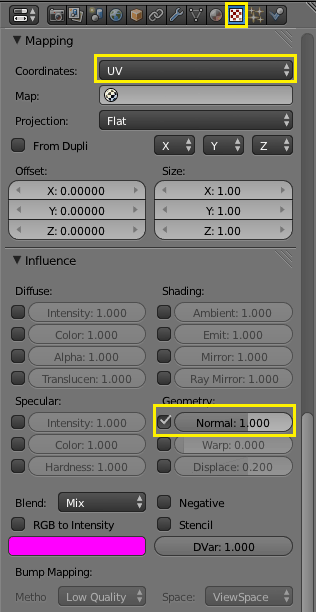
По завершению процесса запекания, у в окне **UV/Image Editor** должна появится текстура карты нормалей. Выглядит следующим образом:



Чтобы увидеть результат работы данной техники, на панели свойств включим режим отображения **GLSL** и режим отображения текстур:



Теперь создадим для плоскости новый материал с настройками по умолчанию, затем перейдем на вкладку текстур и выставим следующие настройки:

Вывод.

Normal mapping являет хорошим методом переноса деталей, метод замечательно подходит при методики высокополигонального моделирования или скульптинга с использование ретопологии низкополигональной модели.

* 1. **Выбор методики нанесения материалов трехмерного объекта**

Виды PBR-текстурных карт

1. Растровая карта (Diffuse map или Albedo map)

Основополагающая карта, которая задает внешний вид 3д-модели. С помощью нее формируется цвет объекта и подчеркивается рельеф элементов модели. Растровая карта создается, как правило, вручную, потому она требует особого внимания и наибольшего времени для ее изготовления

1. Карта нормалей (Normal map)

Карта, которая позволяет в зависимости от того, какой цвет выбран в палитре RGB, изменять угол между поверхностью полигона и нормалью освещения. Генерирование этой карты происходит путем переноса детализации с высокополигональной модели на UV-развертку низкополигональной.

Существует два типа карты нормалей:

* Tangent space – подходит для геометрии, которая должна деформироваться при анимации, представляет собой градиент лилово-синего изображения
* Object space – подходит для статичных объектов, не испытывающих деформаций, представляет собой карту, окрашенную в различные цвет (в том числе и лилово-синию). Просчитывается быстрее tangent space

1. Карта бликов (Specular map)

Карта, которая показывает в какой степени материал модели отражает свет. Она формирует у игрока впечатление от материала и упрощает работу по созданию текстур, так как снижает нагрузку на качество растровой карты. Представляет собой изображения, где светлые тона бликуют сильнее, чем темные

1. Карта затенений (Ambient occlusion map)

Данная карта добавляет реалистичность изображению путем вычисления интенсивности света, доходящего до точки поверхности. Карта имитирует затенения от объектов, блокирующих падающий свет и не зависит от направления света

1. Карта отражений (Reflection map)

Карта отражения показывает в какой степени в материале отображаются другие объекты, её действие аналогично карте бликов, но само отражение добавляется в виде предварительно созданной кубической текстуры. Уровень отражения зависит от яркости участков карты.

1. Карта самосвечения (Selfillumination map)

Данная карта позволяет задавать свечение различным частям 3д-модели, смешиваясь с растровой картой, она влияет на базовый цвет

1. Карта светопроницаемости (Translucency map)

Карта задает степень светопроницаемости материала (например, кожа, листья деревьев, ткань)

1. Карта прозрачности (Opacity map)

Карта в виде черно-белого изображения или с оттенками серого, где белый цвет определяет непрозрачные области, а черный – прозрачные

Физически корректный рендеринг (PBR) можно рассматривать скорее, как методику, а не как жесткий стандарт. Существуют его принципы и инструкции, но не верно считать его стандартом. Таким образом, могут быть различия в его реализации. Эти различия обычно можно найти в типах карт, используемых в процессе шейдинга (создания материалов), трактовке [BRDF](http://3dyuriki.com/2010/03/20/vidy-otrazhenij-v-kompyuternoj-grafike/)(двулучевой функции отражательной способности (ДФОС)) и в том, как значения данных отвечающих за шероховатость / глянцевитость могут быть переназначены в отдельных случаях. Иногда изменяются даже названия карт, но в основе использования PBR заложены одни и те же принципы.

Имеется два наиболее распространенных подхода при создании материалов:

1. на основе карт металличности (metall) и шероховатости (roughness) [будем этот подход дальше называть metall/roughness], а также
2. на основе карт зеркального отражения (specular) и глянцевитости (glossiness) [этот подход будем называть specular/glossiness].



Рисунок 35. Подход metal/roughness и Specular/Glossiness

Инструменты Substance для создания PBR карт, входящие в состав программ SubstanceDesigner, [SubstancePainter](http://3dyuriki.com/2013/12/19/substance-paint-novyj-sposobo-sozdavat-tekstury/" \o "Substance Painter: новый способ создавать текстуры!) и Bitmap2Material 3, поддерживают оба эти подхода. Substance PBR-шейдеры metall/roughness и specular/glossiness используют т.н. GGX-разновидности функции BRDF, в которой не используется переназначение величин шероховатости и глянцевитости. Тем не менее, если пользовательские переназначения все же необходимы – это может быть легко реализовано в Substance материалах. Кроме того, пользовательские шейдеры поддерживаются инструментами Substance, что означает, что вы можете адаптировать Substance инструменты для всех пользовательских [пайплайнов](http://3dyuriki.com/2009/05/31/pipeline-pajplajn-slovar/" \o "Пайплайн (Pipeline) | Словарь).

Подход metall/roughness.

Подход metall/roughness определяется набором каналов, которые передаются в виде определённых текстур в PBR-шейдер. Основными картами при создании материала по подходу metall/roughness являются карты:

* базового цвета,
* металличности и
* шероховатости

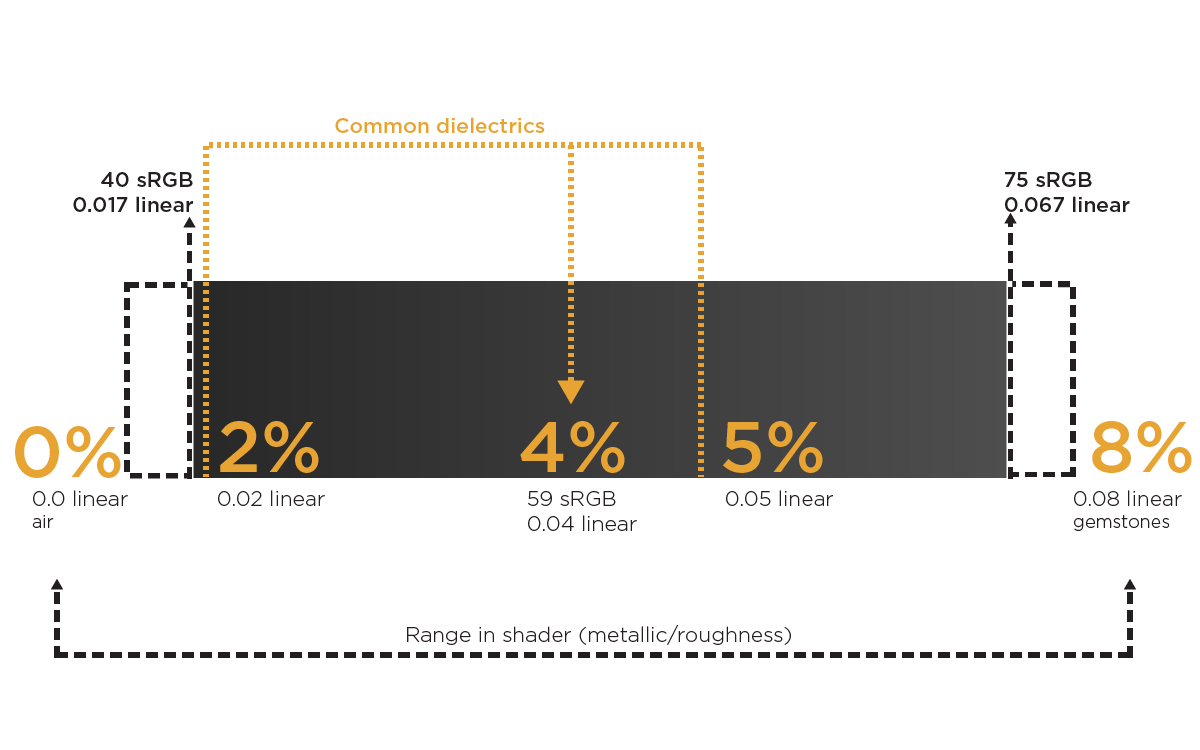
При создании metall/roughness материала, значение отражательной способности материала задается картой базового цвета, а для диэлектриков – также картой отраженного цвета. Отражения при скользящих углах падения определяются BRDF-моделью. Карта металличности работает как маска и используется для того, чтобы дифференцировать (отделить) данные для металлической поверхности от данных для неметаллической, найденные на текстуре базового цвета.

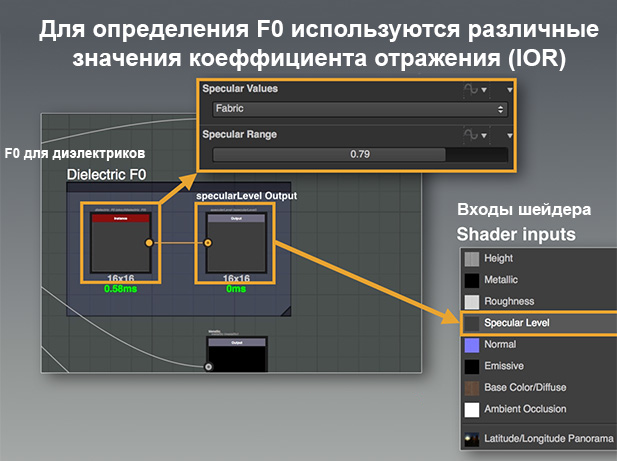


Значения френелевского коэффициента F0 для диэлектриков (неметаллов) не определяются вручную, эту функцию берет на себя шейдер. Когда такой шейдер видит черный цвет на карте металличности, он обрабатывает соответствующие области в карте базового цвета как для диэлектрика и использует значение коэффициента отражения 4% (0,04 в линейном пространстве), как показано на рисунке 04. Как мы уже обсуждали в первой части, значение 4% считается наиболее распространенным для диэлектрических материалов. Важно отметить, что все значения, такие как: F0 для диэлектриков, отражающая способность, диапазон яркостей для альбедо (т.н.  рассеянного цвета или чистого диффузного цвета), устанавливаются на основе реальных измерений.

Константа F0 для диэлектриков.

Инструментарий Substance или UnrealEngine 4 для metall/roughness материалов дает возможность контролировать зеркальность или блеск материала с помощью изменения значение константы F0 (коэффициента отражения по френелю) для диэлектриков. Это значение называется "specularLevel" в Substance. Значения константы F0 представлены интервалом 0,0 – 0,08, как показано на рисунке 05. Если вам в Substance Designer нужно вручную установить F0 для диэлектрика, вы можете сделать это, используя выход "specularLevel", как показано на рисунке 06. Мы подробно обсудим значения F0 для диэлектриков далее, при обсуждении подхода specular/glossiness.



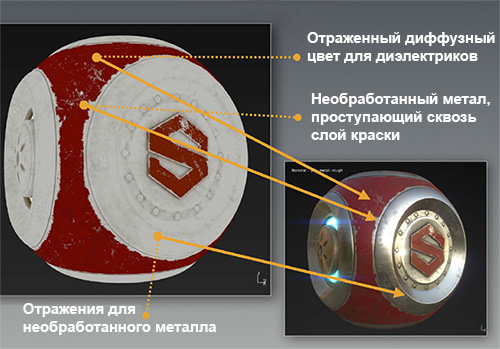


Базовый цвет (RGB - sRGB).

Карта базового цвета – это RGB карта, которая может содержать 2 типа данных:

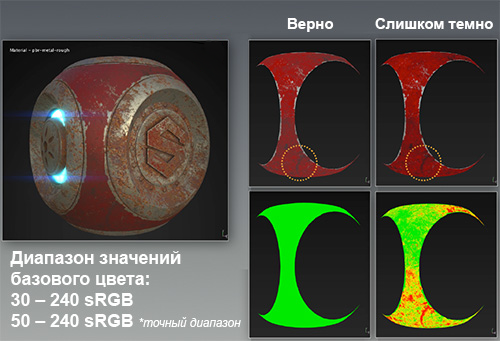
* отраженный цвет для диэлектриков и
* силу отражения для металлов,

Цвет, представляющий на этой карте неметаллическую поверхность, означает отражённую длину волны (как было описано в первом томе). А силу отражения эта карта базового цвета показывает тогда, когда область поверхности обозначена как металл на карте металличности.



Рекомендации по созданию текстуры базового цвета.

Карта базового цвета (альбедо) может рассматриваться как изображение более равномерное по тональности, т.е. менее контрастное, чем традиционная диффузная текстура. Такой подход нужен, чтобы избежать слишком светлых или слишком темных значений. С точки зрения тона, объекты, на самом деле, намного светлее, чем они нам кажутся. Видимый диапазон тональной яркости можно представить себе разбросом от самого тёмного материала - древесного угля с одной стороны и свежим белым снегом с другой. Мы знаем, что уголь темный, но это не 0,0 значение черного. Значения цвета, которые мы выбираем должны оставаться в пределах диапазона яркости. В отношении диапазонов яркости, в основном, следует ссылаться на значения отраженного цвета для диэлектриков. На рисунке 08, можно увидеть пример, где значение загрязнений находятся ниже правильного диапазона яркости. Для темных участков, значения яркостей должны находиться в диапазоне 30 – 50 sRGB. При этом точность значений должна возрастать: при 30 sRGB они могут быть менее точными и более точными при 50 sRGB. Для ярких цветов значения яркостей не должны быть выше 240 sRGB.



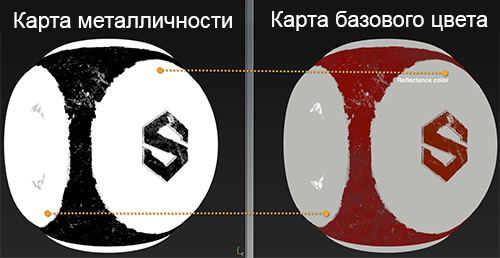
Выше упоминалось, что карта базового цвета (альбедо) содержит данные для отраженного света в терминах диэлектрических материалов и, таким образом, эта карта должна быть лишена информации об освещении, например, такой, как т.н. контактные затенения (Ambient occlusion). Однако, из этого правила могут быть исключения: например, для добавления микро-затенений, когда шейдер не сможет обеспечить этот уровень детализации только посредством канала Ambient occlusion, как показано на рисунке 09. Тем не менее, если микро-затенения добавлены к карте, она по-прежнему должна подчинятся рекомендациям касательно диапазона яркостей.

Значения яркостей для карты, которая описывает отражательную способность для металлов, должны быть получены реальными измерениями значений. Значения коэффициента отражения находятся в диапазоне 70 – 100%. Этот диапазон может быть отображен значениями sRGB 180 – 255. В разделе *Substance PBR Utilities* мы обсудим инструменты, которые обеспечивают предустановленные значения F0 для обычных материалов.

Карта металличности (metallic map, в градациях серого, в линейном пространстве).

Карта металличности используется, чтобы определить, какие участки материала обозначают чистый необработанный металл. Карта металличности – это изображение в оттенках серого или, грубо говоря, чёрно-белая текстура. Она работает как маска для карты базового цвета, и указывает шейдеру, как необходимо интерпретировать данные, найденные в карте базового цвета. Данные в карте металличности не содержат реальных данных, которые непосредственно описывают качества металла. Эта карта просто указывает шейдеру, какие области на карте базового цвета должны быть поняты как диффузный цвет диэлектрика, а в каких областях – как отражающие участки для металла. В карте металличности, значение 0,0 (черный – 0 sRGB) представляет неметаллические области, а 1,0 (белый – 255 sRGB) – чистый металл.

С точки зрения определения понятия чистого металла и неметалла, карта металличности часто бинарная т.е. черная или белая, "металлическая" или "неметаллическая". На практике, когда шейдер "видит" белый цвет на металлической карте, он проверяет соответствующие области в карте базового цвета, чтобы получить значения коэффициента отражения для металла.



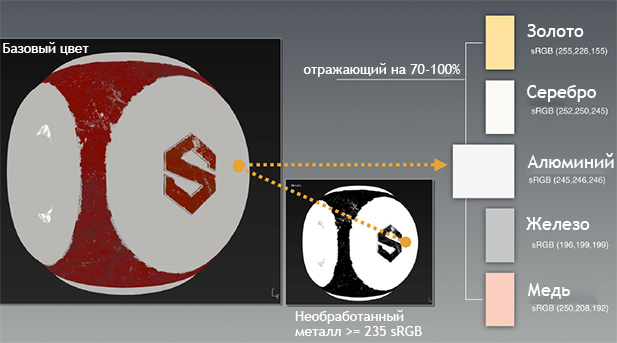
Рекомендации по созданию текстуры металличности.

При создании карт для металлов следует учитывать две важные особенности:

1. отражательная способность металлов имеет высокие значения в диапазоне 70 – 100%;
2. некоторые металлы могут подвергаться коррозии.

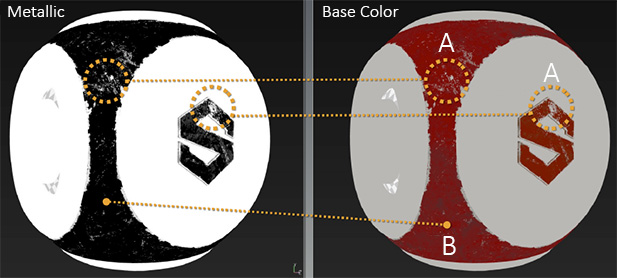
Необработанный металл (raw metal).

Карта металличности заполняется 0 и 1 - металл или нет, - и используется, чтобы определить необработанный полированный металл. Как правило, интервал по шкале серого цвета для необработанного металла составляет 235-255 sRGB для карты металличности. Металлические участки, попадающие в этот диапазон, должны обладать величиной отражательной способности от 70 до 100%.

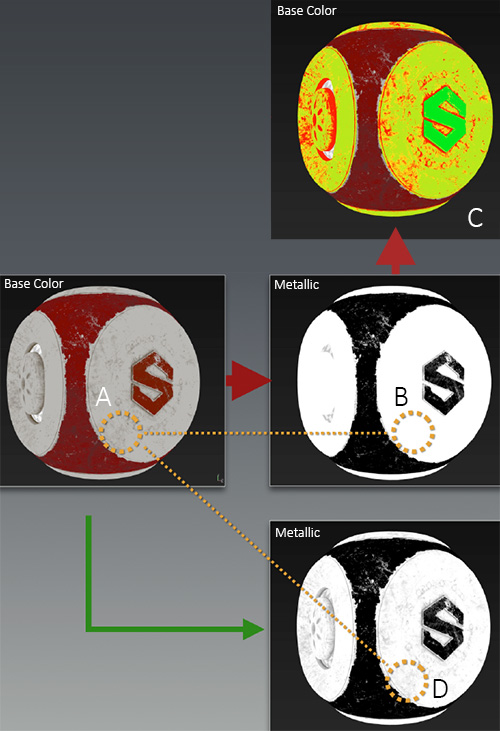


Слой ржавчины или диэлектрик.

Когда поверхность подвергается воздействию окружающей среды, может появиться необходимость показать металл как окислившийся или учесть влияние других факторов, таких как слой грязи или сажи . К металлу, подвергшемуся окислению, т.е. ржавому, надо относиться как к диэлектрику. То же самое верно и для окрашенных металлических поверхностей. Например, если вы присмотритесь к участкам окрашенной металлической поверхности, где краска была поцарапана или счищена, то увидите “необработанной” металл (белый цвет на карте металличности)



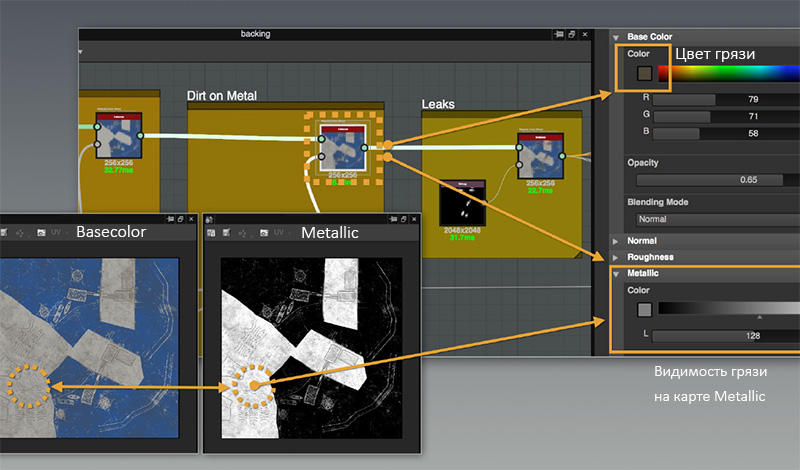
Карта металличности может представлять смешанное состояние между металлом и неметаллом, которое обозначается переходными значениями серого цвета. Главное, что если значения серого цвета на карте металличности ниже 235 sRGB, то вам придется понизить значение величины отражения для базового цвета. Например, посмотрите на слой грязи, который частично покрыл участки чистого металла.



Грязь - это диэлектрик. И если вы оставите карту металличности просто белой, то участки, покрытые грязью, будут обладать тем же коэффициентом отражения, что и металл. Цвет грязи на карте Base Color гораздо темнее, чем цвет металла (цвет которого должен быть светлым, чтобы получить 70-100% отражения для полированных металлов). Затемняя значения карты металличности на грязных участках, вы создадите соответствующий переход между коэффициентами отражения диэлектрика и металла.

Видимость слоя грязи на карте металличности показывает на сколько сильно будет приглушено отражение. Здесь нет четко установленных правил. Вы просто смещаетесь от сильноотражающей поверхности (проводник) до слабоотражающей поверхности (диэлектрик).

Пакеты Substance позволяют легко имитировать влияние погоды и контролировать, как эти эффекты будут влиять на разные каналы (карты) благодаря поддержке многоканальности (multi-channel). В Substance Designer и [Substance Painter](http://3dyuriki.com/2013/12/19/substance-paint-novyj-sposobo-sozdavat-tekstury/" \o "Что такое Substance Painter: новый способ создавать текстуры!) вы просто меняете настройки эффекта, а это автоматически ведет к изменению карт. Например, в Substance Designer вы можете использовать ноду *Мaterial Color Blend*, чтобы применить эффект грязи сразу в нескольких каналах. Чтобы контролировать степень загрязнённости металла, достаточно подвигать слайдер значения металличности в ноде *Мaterial Color Blend*



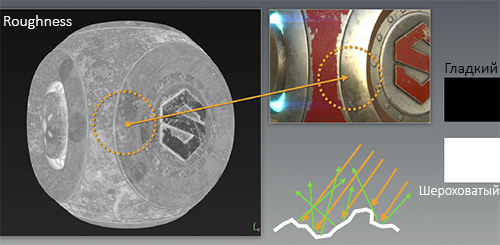
Важно: металл, подвергшийся окислению, т.е. ржавый, необходимо рассматривать как диэлектрик. То же самое верно и для окрашенных металлических поверхностей.

Черный (0.0) = неметалл, а белый (1.0) = металл. Могут использоваться промежуточные значения серого цвета для учета ржавчины или грязи.

Если цвет на карте металличности ниже значения 235 sRGB, то цвет на Base Color нужно затемнить (уменьшить коэффициент отражения).

Шероховатость (Roughness, читается *рафнес*).

Карта шероховатости описывает неровности поверхности. Из-за этих микронеровностей происходит рассеивание света



На карте шероховатости черный (0.0) представляет гладкую поверхность, а белый (1.0) - шероховатую. Эта карта является самой творческой, поскольку она позволяет художнику визуально показать характер поверхности. Вдобавок, она позволяет творчески рассказать историю поверхности/объекта:

Состояние поверхности может многое рассказать о том, в какой среде находилась/жила модель, а значит позволит модели быть лучше интегрированной в тот мир, который вы создаёте.

Карта шероховатости не может быть правильной или неправильной. У художника полный творческий контроль.

**Хорошей базой для карты шероховатости может послужить карта нормалей.** Карта нормалей зачастую содержит ключевые детали поверхности, которые также должны быть представлены и на карте шероховатости.

Вывод.

Всё идёт к тому, любая презентация продукта для конечного клиента будет проходить в реальном времени и с красивой картинкой (а то и вовсе с помощью виртуальной или дополненной реальности). Разработать дизайн - это половина работы. Нужно же ещё продать его клиенту. А для этого лучше всего подойдёт какой-то игровой движок (Unreal Engine или CryEngine), где мы сможем в реальном времени показать задумку в реалистичном стиле, а значит - с PBR шейдерами.

* 1. **Описание сформированной методики создания трехмерного объекта игрового окружения**

Продуманный технологический процесс создания модели игрового окружения позволит существенно сократить сроки разработки за счет сокращения времени, затраченного на исправление ошибок.

Ниже представлена схема этапов создания модели игрового окружения в графическом редакторе.

Моделирование. Выполняет оператор, относится ко всем трехмерным редакторам.

1. Избегать многоугольников в сетке, так как те имеют большее количество вариантов разбиения при триангуляции, а также треугольников при непосредственном моделировании 3д-объекта, так как треугольные полигоны препятствуют образованию петли ребер. При снятии карты нормалей наличие многоугольников (больше четырехугольника) в сетке не желательно, а в некоторых случаях – недопустимо.
2. Правильная топология полигональной сетки в дальнейшем позволяет корректно деформировать модель во время анимации, а также использовать минимальное количество полигонов для описания задуманной формы. Желательно чтобы топология конечной низкополигональной модели повторяла форму высокополигональной модели.
3. Плотность сетки модели должна напрямую зависеть от возможности приблизить камеру к объекту. На участках сложно доступных для обозрения, не стоит делать плотную сетку
4. Следует соблюдать правило – при уменьшении размеров геометрии плотность сетки следует уменьшать, а при увеличении увеличивать. Так же на участках пересечения разной геометрии следует повысить плотность сетки, чтобы не было проглядывания геометрии.
5. Участки геометрии, легко сводимые в карту нормалей, не следует моделировать на низкополигональной модели. Лучше направить свободный вершины на улучшение формы на другом участке геометрии.
6. Следует удалять скрытую геометрию для упрощения модели.

Снятие карты нормалей. Выполняет оператор, относится ко всех трехмерным редакторам.

1. Следует сохранять лишь те вершины и ребра, которые являются формообразующими, защищают карту нормалей от искажений и обеспечивают симметрию развертки.
2. При запекании цилиндрической формы разница между низкополигональной и высокополигональной моделью создает волнистый край. Чем точнее высокополигональная модель повторяет низкополигональную по силуэту, тем меньше будет кривизны и дефектов в карте нормалей.
3. Создавая детали на высокополигональной модели, следует избегать фасок под прямым уголом, так как они плохо запекаются в карту нормалей
4. Низкополигональная модель должна по возможности точно повторять высокополигональную по размеру и форме.
5. При создании фаски в карте нормалей на сложной округлой поверхности следует учитывать растягивания UV-развертки на этих местах при неверной раскройки, из-за чего на карте нормалей могут образоваться изломы. Данная проблема может решаться разными способами:

* Перекраивание UV-развертки
* Повышением полигонажа вдоль проблемной области
* Создание фаски методом прорисовки на растровой карте и снятия с неё затем карты нормалей

Создание UV – развертки. Выполняет программа после автоматизации процессовю

1. Созданная UV-развертка не должна выходить за пределы пространства UV координат
2. Пространство UV координат необходимо заполнять максимально плотно с целью получения лучшего разрешения текстуры, но при этом соблюдать минимальный отступ между частями UV-развертки, чтобы избежать смешивания текстур. При работе с файлом текстуры размером 2048x2048 пикселей промежутки составляют от 6 до 8 пикселей. При создании текстур в Adobe Photoshop по возможности следует делать отступы больше, чтобы визуально легче было избегать наложения соседних текстур при изготовлении карты.
3. Размер полигонов на UV-развертке должен быть приближен к размеру полигонов на геометрии 3д-объекта, иначе текстура, примененная к поверхности модели, может исказиться (сжаться/растянуться).
4. Если у модели есть идентичные по виду части, то целесообразно нарисовать одну текстуру поверх наложенных друг на друга частей UV-развертки
5. Плотность элементов текстуры должна быть равномерной. Удобнее всего регулировать плотность пикселей частей UV-развертки с помощью временного наложения материала в виде повторяющегося рисунки (клетки)
6. Соответствующие друг другу части следует располагать рядом на поле UV координат, чтобы при дальнейшем раскрашивании созданной UV-развертки не было необходимости возвращаться в графический редактор для уточнения элемента.
7. Важно придерживаться ориентации проекционных UV координат, как у модели. Если повернуть грани в обратную сторону или повернуть по вертикали, это немедленно отразиться на текстуре модели, которая станет перевернутой. Для отображения материала использовать физически корректный рендеринг.

**Заключение**

Проведены следующие исследования:

* Моделирования геометрии трехмерного объекта
* Методы построения UV-развертки трехмерного объекта
* Материалы трехмерного объекта
* Рельефное текстурирование.
* Сравнительный анализ трехмерных редакторов
* Сравнительный анализ игровых движков

Результатом исследований является сформированная методика создания модели игрового окружения c критерием сокращения используемых вычислительных мощностей.

Формирование методики создания трехмерных объектов игрового окружения и является оригинальным результатом проведенной работы.

Полезность методики была доказана в сравнении двух моделей, а именно игровой модели из игры и усовершенствованной модели в соответствии с нашей методикой. Эти объекты были загружены в игровой движок и сопоставлены друг с другом по количеству полигонов, визуальной составляющей и нагрузке на систему.

Также в ходе работы был проведен анализ игровых движков и 3D-редакторов, рассмотрены основные этапы создания модели, сравнены и обобщены сведения о методах работы с 3д-моделями, сформированы правила по данным этапам и сделаны выводы.

Сокращение потребляемых ресурсов дает огромный потенциал для использования качественной и количественной визуализации игрового окружения

Немаловажным является и то, что сокращение вычислительных ресурсов в теории может позволить улучшить качество симуляции виртуальной реальности (VR), так как на сегодняшний день необходимый уровень для комфортного времяпрепровождения в очках виртуальной реальности составляет 90 кадров в секунду (FPS) в каждый экран на каждый глаз. Что достаточно затратное по вычислительным ресурсам для представления насыщенных виртуальных миров. Как пример, виртуальная симуляция может быть использована на учении отработки инструкции правил безопасности жизнедеятельности, как дополнительный источник образовательного процесса.

Теоретически по проработанной методике можно добиться быстрого развития и вовлечения новых специалистов в относительно новую сферу, что обеспечит новые варианты использования технологий трехмерной графики и улучшения существующих.

**Литература**

1. 3Ds Max, Autodesk, руководство пользователя. <https://www.autodesk.ru/products/3ds-max/>
2. Maya, Autodesk, руководство пользователя. <https://www.autodesk.ru/products/maya/>
3. Cinema 4D, Maxon, руководство пользователя. <https://www.maxon.net/ru/>
4. Foundry MODO, Luxology, руководство пользователя. <https://www.foundry.com/products/modo>
5. Houdini, SideFX, руководство пользователя. <https://www.sidefx.com/>
6. Blender, Blender foundation, руководство пользователя. <https://www.blender.org/>
7. ZBrush, Pixologic, руководство пользователя. <http://pixologic.com>
8. Mudbox, Autodesk, руководство пользователя. <https://www.autodesk.com/products/mudbox>
9. Foundry Mari, Luxology, руководство пользователя. <https://www.foundry.com/products/mari>
10. 3d Coat, Pilgway, руководство пользователя. <https://3dcoat.com/home/>
11. Substance painter, Allegorithmic, руководство пользователя. <https://www.allegorithmic.com/products/substance-painter>
12. Unity 3D, Unity, руководство пользователя. <https://unity3d.com/ru>
13. Unreal Engine 4, EpicGames, руководство пользователя. <https://www.unrealengine.com>
14. Cry Engine 5.5, CryTek, руководство пользователя. <https://www.cryengine.com>
15. [«Программирование игр и головоломок»](https://vk.com/doc10903696_276779582?hash=bdde6b824bf88779cc&dl=a15e51fa39cc64d374). Ж. Арсак
16. [«Программирование трёхмерных игр для Windows. Советы профессионала по трёхмерной графике и растеризации»](https://vk.com/doc10903696_277061858?hash=6501863862994ebadc&dl=7d94b749d3fd40b3fe). Ламот А.
17. Unity Community. [Unity 5 Tutorial: How to create a Maze/ Adventure Game C#](https://www.youtube.com/watch?v=3khBcfXH__c)
18. NVidia. [GPU Gems](http://developer.nvidia.com/object/gpu_gems_home.html)
19. [Алексей Боресков](https://www.ozon.ru/person/275687/), [Евгений Шикин](https://www.ozon.ru/person/275686/). Компьютерная графика. Полигональные модели
20. Blender: 3D-моделирование и анимация. Руководство для начинающих Год: 2009 Автор: Прахов А.А.

Основы Blender (3-е издание). Год: 2010 Автор: Кронистер Дж.