МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
(НИЯУ МИФИ)

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ В ИНЖЕНЕРИИ

Учебное пособие

Авторы: Толстов М.С., Елагина А.Н., Бородин К.Ф., Синица Д.А., Рябов П.Н.

Москва 2023

ВВЕДЕНИЕ

Материалы учебно-методического пособия подготовлены на базе проектов тренажеров в виртуальной реальности «VR 3D-принтер AnyCubic 4 Max Pro» и «VR станок токарно-винторезный повышенной точности универсальный 1И611П». Проекты планируются к использованию в качестве вспомогательных в рамках преподаваемых инженерных дисциплин во НИЯУ МИФИ, а также для подготовки лаборантов и техников.

Применение технологий виртуальной реальности в инженерии требуют от разработчика навыков точного и полигонального моделирования, а также умения работы в так называемых «игровых движках». В пособии в кратком виде описывается необходимый минимальный теоретический материал и представлены примеры их практического применения. Приведены ссылки на актуальные и относящиеся к рассматриваемой теме на 2023 год российские нормативные акты и стандарты. Также приведены примеры конструкторских разработок, выполненных на кафедре «Конструирование приборов и установок».

Авторы пособия надеются, что представленное пособие поможет читателям получить уникальные для индустрии навыки, способные сделать тех высоко востребованными специалистами на рынке труда.

Системы виртуальной реальности (VR) с каждым днём всё больше становятся частью нашей повседневной жизни и находят применение в большом количестве различных сред- от гейминга до строительства и инженерии. Применение этих технологий позволяет как предотвратить возможные ошибки при проектировании, а значит и последующие убытки за счёт наглядного представления результатов работы, так и положительно влияет на вовлеченность в процесс разработки моделей.

Данные системы обладают высокой интерактивностью и позволяют намного полнее погрузиться в созданный на компьютере мир. За счёт этих свойств стало возможно создавать виртуальные тренажёры, основанные на цифровых двойниках комплексных киберфизических систем и установок, готовящие специалистов широкого профиля.

Вся описанная работа выполнена в средах Blender и Unreal Engine 5.1, большая часть данного пособия посвящена специфике работы именно в этих средах. Тем не менее, при желании читатель сможет перейти на аналогичное программное обеспечение (3Ds Max, Maya, Unity и др.). Вышеописанные программы выбраны для данного пособия ввиду их свободного

распространения на момент написания, а также ввиду их удобности. Для более глубокого изучения функционала этих программ требуется изучать профильные материалы, однако представленная информация позволит освоиться и начать делать самостоятельную работу. Ввиду того, что в пособии рассказывается о работе с результатом конструкторской деятельности с момента разработки 3D-модели в точном формате, предполагается, что читатели успешно прошли такие курсы как «Инженерная графика в системе автоматизированного проектирования КОМПАС-3D», а также «Детали машин и основы конструирования».

Оглавление

ГЛАВА 1. Особенности инженерной VR разработки.

Поговорим немного о процессе разработки цифрового двойника комплексного оборудования в виртуальной реальности, и о нюансах, которые могут возникнуть, и на которые стоит обращать внимание.

Начнём с перечисления перечня компетенций, которыми необходимо обзавестись, и о которых в той или иной степени пойдёт речь в этом пособии. Во-первых, на момент написания этого текста, не существует программных решений, позволяющих в полной мере воссоздать полный функционал цифрового двойника хоть сколько-нибудь комплексного оборудования. Ввиду этого, от разработчика требуется изучать специальные среды, в которых возможно запрограммировать поведение механизма, настроить физику, и, наконец, реализовать виртуальную реальность. Такие программы зовутся «игровыми движками». В данном материале будет рассматриваться среда Unreal Engine версии 5.1.

Во-вторых. Игровые движки практически неспособны обрабатывать привычные инженерам модели, выполненные в САПР системах. Вместо этого, они работают с полигональными форматами. Перевод точных моделей в полигональный формат возможен, однако требует от исполнителя знания нюансов конвертации, основ топологии и умения работы с сеткой. Существует множество программных пакетов, способных работать с полигональными форматами. В пособии представлен материал, подготовленный в среде Blender.

Наконец, при создании любого проекта в виртуальной реальности от разработчика требуется понимания нюансов подобной разработки. Особенности исполнения интерактивных элементов, передвижения, рендера изображения, формирования качественного «эффекта погружения» и многого другого. Ввиду чего требуется также изучить основы игрового дизайна: нарративный дизайн, дизайн уровней, понятия о UI и UX. Давайте со всем этим разбираться.

1.1. Декомпозиция задачи

Очевидно, что перед тем, как начать создавать тренажер в виртуальной реальности, необходимо узнать, как работает та установка, которую вы будете переносить в виртуальную среду и определить какой функционал необходим в первую очередь. Более того, если вы хотите создать по-настоящему хороший тренажер, крайне важно уделить этому пункту особое внимание. И вот почему.

Дело в том, что на текущий момент создание полноценного цифрового двойника абсолютно любой установки, насколько бы простой она ни была, попросту невозможно. Связано это с двумя ключевыми факторами: крайне тяжело написать математический аппарат, способный описать всё то великое взаимосвязанное многообразие происходящих физических процессов и явлений, протекающих в процессе работы установки, и решительно

невозможно найти компьютер, способный потянуть задачу всё это обрабатывать в реальном времени. К счастью, подробнейшая физически корректная реализация поведения каждого из лепестков отцветшей сакуры где-то на периферии игровой зоны зачастую и не требуется. Достаточно добиться ощущения правдоподобности. Ваша задача, как разработчика, заключается в том, чтобы определить, какой опыт должен получить пользователь и чему научиться, и исходя из этого понять, насколько подробную логику необходимо подготовить для реализации той или иной механики.

Данная задача может быть весьма непростой и, вообще говоря, для её полного раскрытия одного пособия не хватит. С чисто практической точки зрения мы советуем подход, заключающийся в следующем. Определите для себя минимально устраиваемую техническую реализацию проекта. Далее, пропишите тезисно основные умения и навыки, которыми должен овладеть пользователь в результате выполнения тех задач, что перед ним стоят. Ими могут быть: овладение техникой безопасности, обучение технике работы за установкой, ознакомление с внутренним устройством установки и так далее в любой необходимой комбинации. После чего дайте ответ на подобную серию вопросов:

1. Требуется ли соблюдение точной формы поверхности для тех или иных составных узлов, компонентов и деталей?
2. Требуется ли физически корректная модель поведения тех или иных составных узлов, компонентов и деталей при различных отрабатываемых условиях?
3. Влияет ли качество визуализации тех или иных узлов, компонентов, событий и так далее на уровень итогового обучения?
4. Требуется ли подробное воссоздание взаимодействия с интерактивными элементами установки по аналогии с реальным прототипом?

Положительный ответ на любой из представленных выше вопросов и наметит основной путь дальнейшей разработки сверх намеченного минимально удовлетворительного уровня.

Например, при реализации обзорного тренажера о строении установки не требуется точное воссоздание геометрии изделия, зато понадобится воссоздать качественный визуал, включающий в себя материалы и работу со светом. С другой стороны, для реализации тренажёра, призванного обучить пользователя работы за ним вероятно понадобятся тщательная работа с виртуальными органами управления установки и написание более приближенной к реальности физики работы отдельных компонентов, в свою очередь зависящие от более глубоко воссозданной геометрии деталей. Ну и так далее. Не пытайтесь сделать всё и сразу, особенно по началу, определите для себя основные приоритеты. В дальнейшем при необходимости проект

можно доработать. В этом сильно поможет иерархическая система построения взаимодействий, или дерево. Таким образом внесение коррективов в более вышестоящие элементы передастся зависящим элементам без необходимости работать с каждым индивидуально.

Для получения же информации о функционале установки не обойтись без профильной литературы, инструкций, ну и конечно же не помешает наличие специалиста под боком, под опытным надзором которого вы сможете потрогать установку своими руками и которому зададите все необходимые вопросы.

В процессе набора знаний вполне возможно стать более чем квалифицированным оператором изучаемой установки. Весьма неплохое повышение квалификации.

Чтобы сделать качественный тренажер, хорошим вариантом будет послушать как учат работать человека, который никогда не был с ней и подобными установками связан. Идеально, когда вы сами являетесь таким человеком, поскольку для вас не будет очевидных моментов, и вы на собственном опыте поймете, как направлять пользователя в нужные места, основываясь на том, как учили вас.

В рассматриваемом методическом пособии в качестве примера будут описаны два проекта тренажеров виртуальной реальности, разрабатываемых нами: обзорный тренажер устройства 3D-принтера Anycubic 4 MAX Pro, а также тренажер по обучению работе за станком токарно-винторезным повышенной точности универсальным 1И611П, оба из которых располагаются в мастерских корпуса НЛК МИФИ.

1.2. Anycubic 4 MAX Pro

Данный тренажер виртуальной реальности необходим для того, чтобы рассказать человеку по какому принципу работает 3D-принтер, за что отвечают его составные части. Помимо этого, он нужен, чтобы у пользователя была возможность заглянуть внутрь принтера, разобрать и собрать его, обучиться порядку действий по подготовке к печати, ну и понять принцип его работы. Обозначенный тренажёр показан на рисунке 1а. Его реальные прототипы показаны на рисунке 1б

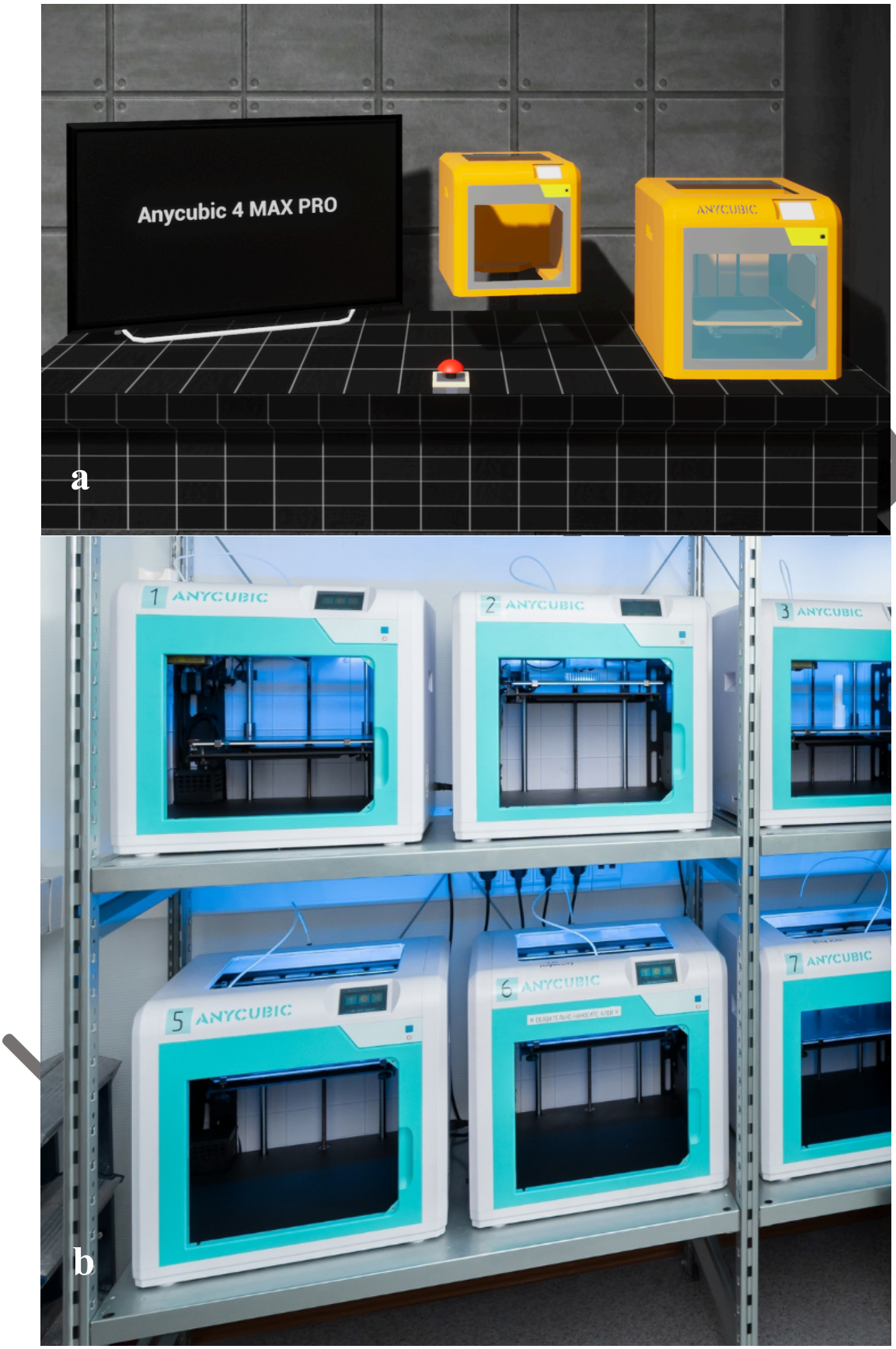


Рисунок 1 – 3D-принтер Anycubic 4 MAX Pro

a- Разрабатываемый виртуальный тренажёр; b – реальный прототип.

1.3. 1И611П

Данный тренажёр создан с целью помочь будущему сотруднику мастерской овладеть технике работы за ним. Тренажёр позволяет ознакомиться с устройством, на его базе планируются обучение технике безопасности. Тренажер обозначен на рисунке 2а, его реальный прототип на рисунке 2б

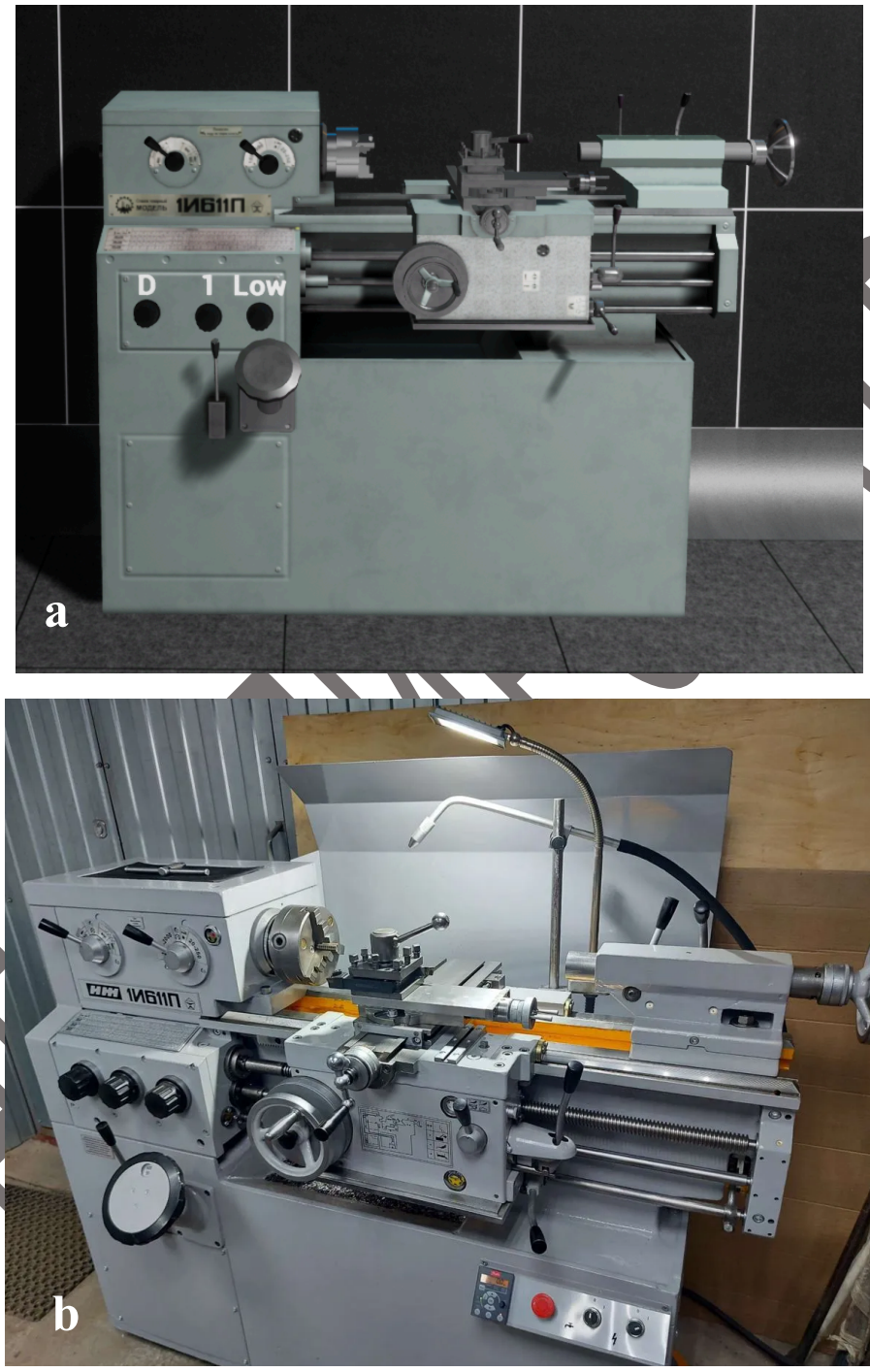


Рисунок 2 – Станок 1И611П

а- Разрабатываемый виртуальный тренажёр; b – реальный прототип.

1.4. Взаимодействие, наиболее приближенное к реальности

Раз мы ведем речь про тренажёры в виртуальной реальности, в первую очередь рассмотрим типичные применяемые контроллеры, на примере комплекта HTC Vive Pro. На рисунке 3 показаны основные кнопки и прочие органы управления, традиционно используемые при создании логики.

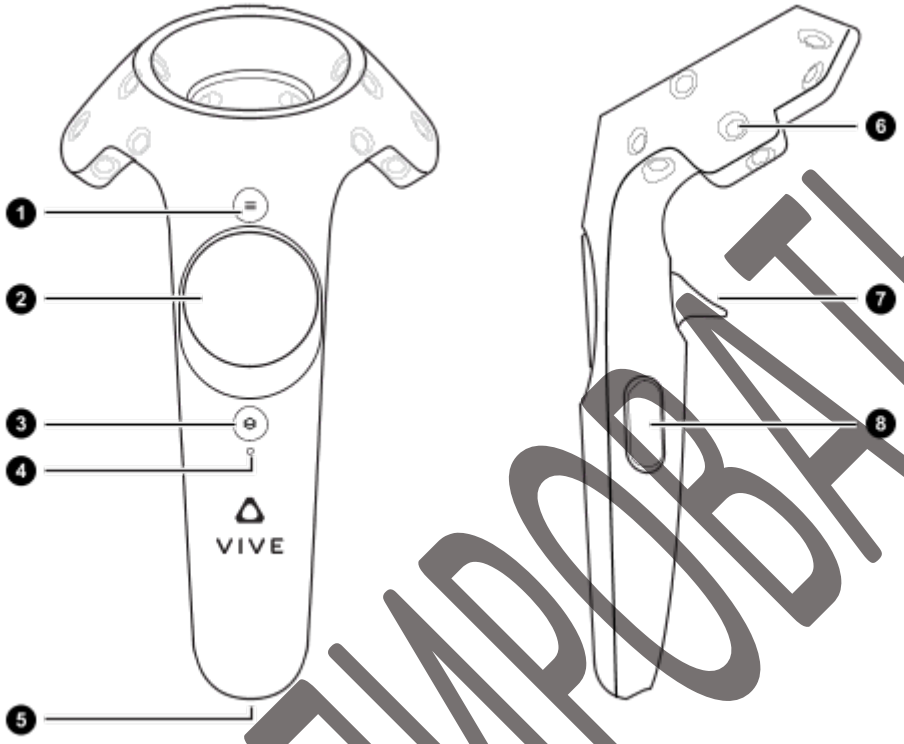


Рисунок 3 – Контроллеры Vive

1 - Кнопка «Меню»; 2 - Сенсорная панель; 3 - Кнопка «Система»; 4 - Индикатор состояния; 5 - Порт micro-USB; 6 - Датчик отслеживания; 7 – Курок; 8 - Кнопка «Захват»

Обозначенные индикаторы и кнопки призваны имитировать действия, обычно совершаемые нашими руками. При разработке тренажера виртуальной реальности, важно сохранить взаимодействие с объектами настолько приближенное к реальности, чтобы человек, впоследствии, подойдя к реальной установке не потерялся и для него не было необходимости учиться заново на ней работать. Дадим несколько примеров того, как можно выполнить различные взаимодействия на примере токарного станка. На рисунке 4 представлены рукоятки переключения подач и резьб.

Рисунок 4 – Ручки переключения подач и резьб

Важно отметить, что для представленных органов управления справедливы два ключевых момента: угол их поворота никак не ограничен, а также, стоя за станком, эти ручки обхватываются пальцами оператора станка так же, как круглые дверные ручки. Наилучшим вариантом будет реализовать максимально приближенный функционал взаимодействия. Следовательно, ручка должна хвататься рукой в виртуальной реальности при нажатии на курок контроллера (рисунок 3, позиция 7) вблизи модели ручек, выполняемый указательным пальцем с последствием поворотом контроллера по или против часовой стрелки.

Теперь давайте посмотрим на маховик выбора скоростей шпинделя, представленный на рисунке 5 и обратим внимание на то, как берется уже он.

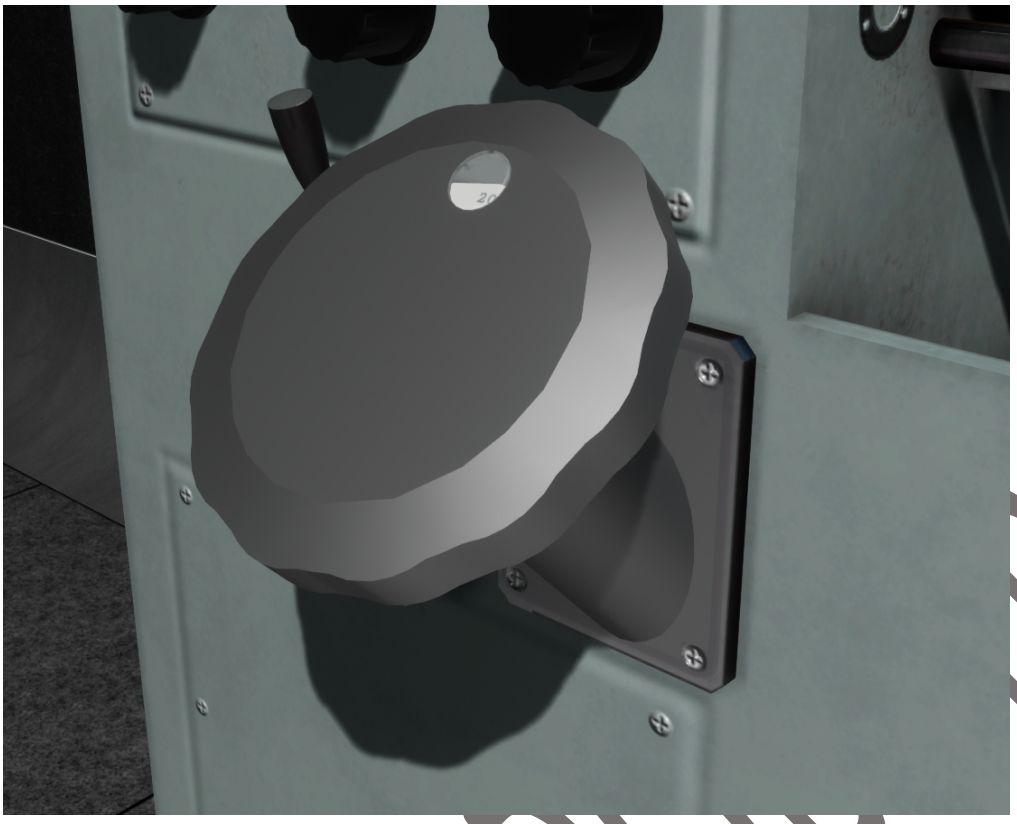


Рисунок 5 – Маховик преселективного выбора скоростей шпинделя.

Хват данного колеса напоминает чем-то то, как водитель держит руль автомобиля. Таким образом, в данном случае имеет смысл реализовать функционал, при котором колесо будет браться в виртуальной реальности при нажатии кнопок контроллера, расположенных по бокам, кнопок «Захват» (Рисунок 3, позиция 8).

Рассмотрим теперь пример с 3D- принтером. Напомним, в данном тренажере виртуальной реальности необходимо дать пользователю возможность ознакомиться с устройством принтера, показать, какая часть за что отвечает и, наконец, реализовать возможность сборки-разборки. Если пытаться реализовать эту механику максимально реалистично, придется заставлять пользователя буквально брать виртуальную отвёртку и раскручивать каждый винтик (а для максимальной реалистичности еще и сделать так, чтобы к моменту сборки у вас магическим образом образовалась лишняя деталь).

Однако, подобная скрупулезность излишня и может негативно сказывается на опыте пользователя. Просто представьте, вы в виртуальной реальности имея натуральные пульты вместо рук и сильно ограниченные возможности по взаимодействию с объектами пытаетесь открутить винтик, даже не чувствуя сопротивления от его кручения. Весьма фрустрирующие ощущения. А в это время первоочередная цель создания данного тренажера состоит в том, чтобы рассказать пользователю о том, как работает принтер и за что отвечают его составные части. Поэтому в данном случае вполне

допускается внести существенные упрощения и позволить пользователю буквально разрывать принтер по частям руками, либо же при помощи некоторой волшебной лазерной указки, стремящийся прямо из виртуальной руки. Поставленная задача будет выполнена, вам, как разработчику, придется делать меньше работы, конечный пользователь получит более приятный опыт.

Ещё один момент, который имеет смысл рассмотреть, это – непосредственное взятие какой-либо части принтера. Например, корпус и отсоединение от общей конструкции. Дело в том, что при предложенной выше схеме неизбежно будут происходить коллизии объектов (то есть объекты будут просто проходить сквозь друг друга), что теоретически может выбить пользователя из ощущения реальности происходящего. Однако, по факту, если вы добавите какую-либо вспомогательную информацию, дающую пользователю понять, что взаимодействие началось (например, цветовое выделение объекта, частицы, звук, текстовую информацию и так далее), то этого будет вполне достаточно, чтобы пользователь мог жить с подлобными допущениями. Но подождите, давайте теперь зададим вопрос, а что будет происходить, если пользователь отпустит этот самый корпус или, что хуже, любой более мелкий объект и он просто упадет на пол. Как лучше поступить, заставить пользователя заниматься физкультурой и физически за ним наклоняться, или выполнить чуть большую работу и реализовать функцию возврата объекта на место или же сделать так, чтобы он остался висеть в воздухе, что звучит еще более странно?

Другими словами, каждое принятое решение в свою очередь ведет к новым проблемным вопросам. А посему еще раз советуем в самом начале как можно более подробно прописать всю работу разрабатываемого приложения и сразу предложить реализацию тех или иных механик. Это значительно сократит необходимость что-то переделывать после. К слову, у нас реализована механика возврата упавшей детали на место.

1.5. Реализация положительного пользовательского опыта

Подведем промежуточные итоги. В прошлой главе мы уже говорили о том, что не всегда нужно реализовывать функционал, наиболее приближенный к реальности. Попробуем классифицировать основные причины:

Первая – тренажер может стать излишне сложным для взаимодействия в виртуальной реальности. При создании тренажера мы хотим отгородить пользователя от этого, поскольку он должен понять, как работает установка и научиться за ней работать. Пользователь должен с легкостью понять, как работать за установкой иначе зачем тогда вообще создавать тренажер.

Вторая – неоправданные затраты на разработку. Необходимо понимать, что на разработку сложных механик затрачивается больше времени, поэтому важно найти оптимальный способ реализации механик.

Вспомним о том, что в прошлой главе мы рассматривали способы того, как можно реализовать сборку-разборку принтера.

В нашем тренажере это реализовано при помощи указки и табло, как показано на рисунках ниже.

Рисунок 6 – Лазерная указка и информационное табло

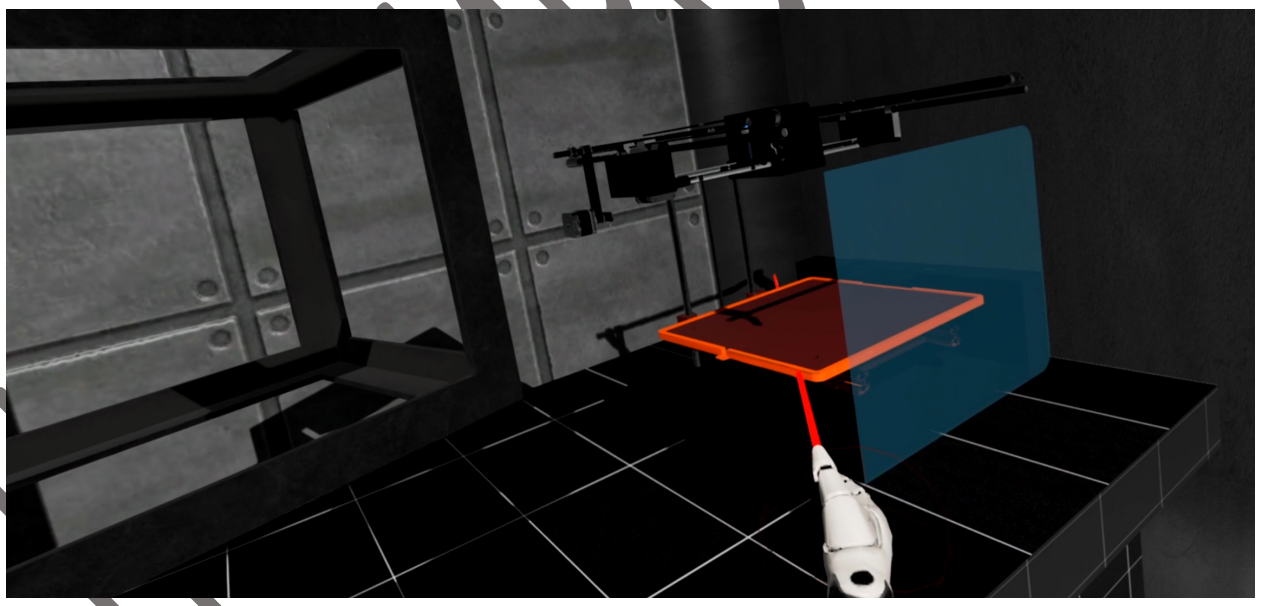


Рисунок 7 – Визуальная подсветка выделенного элемента

Из данных рисунков видно, что у пользователя есть красная указка, вылетающая прямо из руки. Данную указку возможно направлять на разные объекты, что приведет к его подсвечиванию в случае если тот относится к принтеру. Когда пользователь нажимает на курок, направив указку на подсвеченных объект, перед ним появляется вращающаяся увеличенная копия объекта, а на экране телевизора появляется информация о нём. На табло над столом перечислены все составные части принтера, что является вспомогательным способом выбора объекта, информацию о котором хочет

увидеть пользователь. В нижней части этого табло находятся интерактивные кнопки «скрыть» и «показать». Если навести на них указку и нажать на курок, это приведет к соответственно, скрытию и показу выбранных элементов принтера. Таким образом у нас реализуется функция сборки-разборки.

Является ли предложенный способ идеальным? Разумеется, нет, однако с точки зрения поставленного ТЗ описанная механика более чем достаточна.

Во-первых, пользователь имеет в руке указку и уже интуитивно понимает, что её нужно куда-нибудь направить.

Во-вторых, при направлении на какую-либо часть принтера появляется цветовое выделение, и пользователь понимает, что с ней можно взаимодействовать. Останется только предварительно объяснить пользователю, что при нажатии на курок появится увеличенная копия выбранного объекта, которая крутится, чтобы можно было рассмотреть со всех сторон, а на экране телевизора появится соответствующая информация о выбранном объекте.

Таким образом, уже сейчас мы получили достаточно понятный для знакомого с системами виртуальной реальности пользователя интерфейс. Тем не менее для вас, как для разработчика не повредит сбор обратной связи от целевой группы, чтобы те смогли объяснить удобные и неудобные моменты. Нам с этим повезло, потому как нашей целевой группой являются гости и абитуриенты, приходящие на разнообразные профориентационные события в НИЯУ МИФИ, где мы и демонстрируем свои наработки.

Теперь же давайте снова вспомним о маховике для выставления скорости оборотов шпинделя на токарном станке.

С одной стороны, использование кнопок «захват», нажимаемые пальцами со среднего по мезинец, должно создавать ощущения от взаимодействия наиболее приближенные к реальности. Однако нажатие на данные кнопки контроллера крайне неудобно, особенно для людей, у которых мало опыта взаимодействия с виртуальной реальностью и обычно реализуется, если «курок» чересчур перегружен другими механиками. В связи с этим, данные кнопки контроллера лучше использовать для других целей и, по возможности, сокращать количество их использования.

Для хватов хорошо подойдет «курок». Как правило, пользователь с первых секунд ознакомления с функцией «хватания» начинает интуитивно использует именно его. В данном случае теряется реалистичность взаимодействия, но повышается качество пользовательского опыта, поскольку нет разных механик «хватаний» и не придется разбираться как взаимодействовать с тем или иным предметом. Зато отрабатывается порядок действий, который актуален в том числе и для реального станка. Перейдя же к

реальному станку, пользователь скорее всего и сам возьмется за колесо правильным образом.

Также важно уметь находить способы взаимодействия, в случаях, когда наиболее приближенные к реальным по разным причинам невозможно реализовать.

Например, при создании тренажера для работы на токарном станке возникла проблема: некоторые управляющие органы являются рычагами, которые должны поворачиваться в продольном и поперечном направлениях. Данные рычаги показаны ниже.

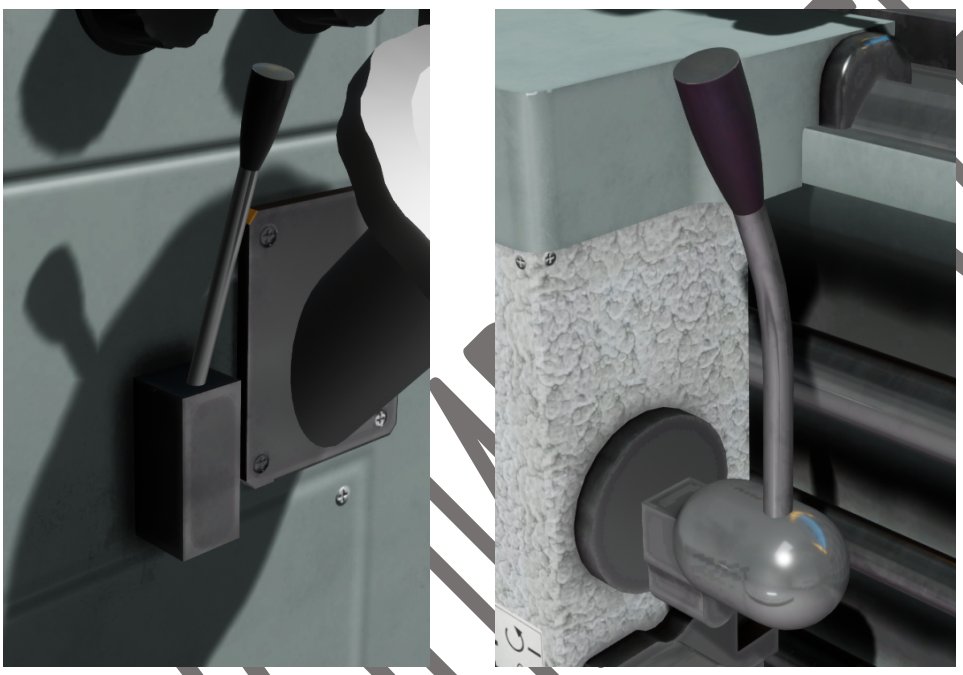


Рисунок 8 – Проблемные рычаги

Слева - рычаг сцепления; Справа – рукоятка направления автоподачи.

На первый взгляд, какие могут возникнуть проблемы? Механик реализации подобных рычагов очень много, осталось лишь правильно настроить взаимодействие объектов друг с другом, ограничить углы отклонения и все готово. Однако, на ровном месте возникают баги, из-за которых объект, который должен вращаться вокруг одной точки в одной плоскости, крутится так как он хочет.

Причина безусловно в чем-то то кроется, однако бывает такое, что найти её самостоятельно решительно не получается. Особенно при недостатке опыта. Приходится искать другие способы реализации взаимодействия. В нашем случае подошло отслеживание положения руки в двух состояниях. Первое – пользователь взялся за ручку сцепления или подачи (зажал курок при касании виртуальной рукой любого из рычагов). Второе – пользователь перемещает руку с зажатым курком в том направлении, в которое он хочет переключить рычаг, после чего отжимает курок контроллера. В результате

ручка переходит из исходного положения в установленное пользователем, если это возможно.

В данном подходе безусловно тоже есть минусы. Один из них- всё то же возможное ощущение нереалистичности происходящего. Тем не менее, пользователю достаточно один раз усвоить правила работы вашего виртуального мира, после чего он продолжит взаимодействовать с интерактивными объектами уже инстинктивно. Ну и самое главное, удалось реализовать механику, которая опять же соответствует поставленным требованиям технического задания. Кстати, до сих пор не понимаем, откуда возник тот баг, прям спать не даёт.

1.6. Замена интерактивных элементов

Порой разрабатывая тренажер виртуальной реальности, вы столкнетесь с тем, что в реальности какие-то элементы могут быть хорошо видны, однако при попытке создания фотореалистичных текстур в виртуальной реальности смотреть на это будет невозможно.

Для пояснения данного пункта снова обратимся к станку, а, точнее, к ручкам переключения передач и резьб, рассмотренных ранее и сравним их с реальными ручками. Рисунки 9а и 9б соответственно.

Рисунок 9а – Ручки в VR тренажере

Рисунок 9б – Ручки на реальном станке

Обратим внимание на буквы. В реальности (рисунок 9) они нанесены на саму ручку и могут быть прочитаны пользователем, но в виртуальной реальности (рисунок 8) эти буквы становятся плохо читаемыми в силу разных причин, включающих, например, сравнительно малое разрешение линз шлема, а также размер самих надписей на ручках.

Теоретически, можно пытаться сделать и нанести невероятно хорошие текстуры, однако данном случае намного логичнее просто заменить нанесение букв на ручки на одну большую, которая меняется в зависимости от положения ручки.

Данный подход обеспечил для пользователя удобный и читабельный интерфейс.

ГЛАВА 2. Обработка инженерных моделей

Прежде чем переходить к самому полигональному моделированию, для начала разберем некоторые его принципы.

2.1. Принципы полигонального 3D-моделирования

2.2. Объемность

Мэш не должен состоять из плоскостей и объемных элементов одновременно. Если сам объект – рисунок/плоскость, то есть его форма изначально не имеет объема и в нем нет необходимости, то можно экспортировать и плоскость. Примером может служить плоский спрайт травы, изображенный на рисунке 10.

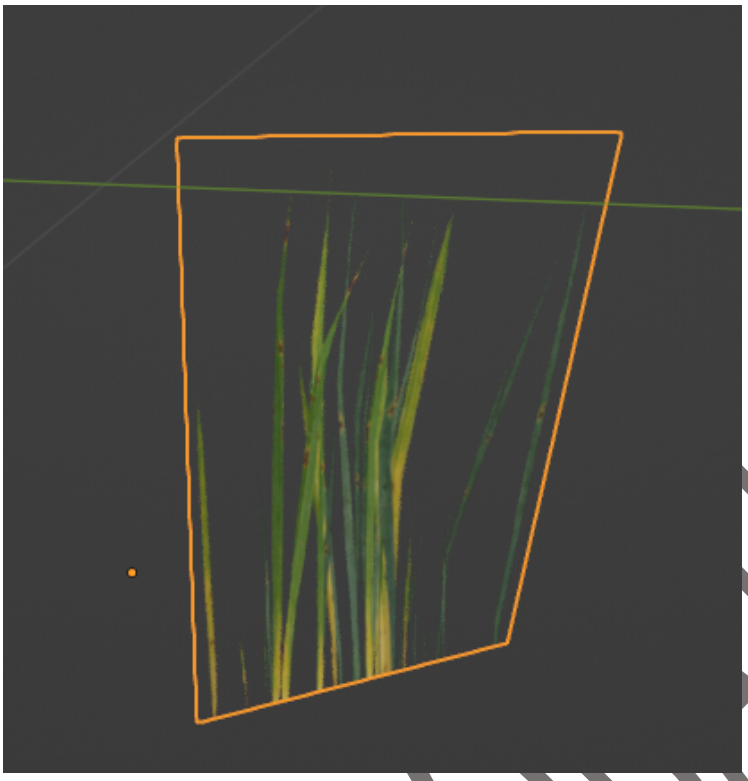


Рисунок 10 – Текстура травы на плоскости

Если же создаваемый объект должен обладать объемом, то он не должен содержать плоскостей. В примере на рисунке 11 моделируются крышку коробки и создаваемые «крылья» также должны обладать объемом, не важно на сколько тонкими они должны быть.

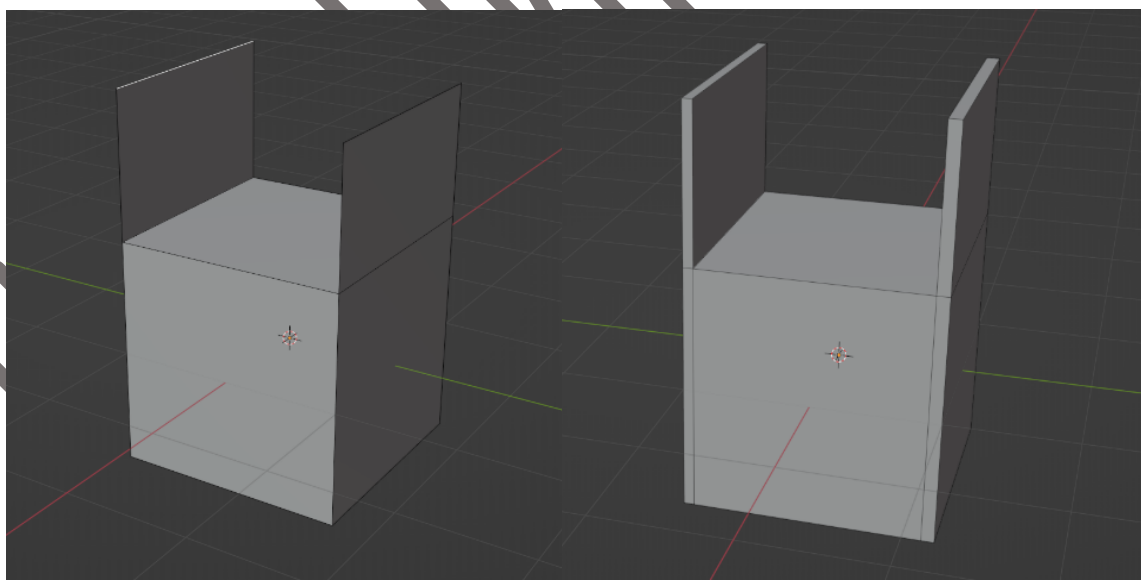


Рисунок 11 – Неправильное (слева) и правильное (справа) моделирование коробки

2.3. Форма полигонов

Сетка модели должна состоять из треугольников, либо плоских четырехугольников. В таком случае результат моделирования будет предсказуем и корректен. Имеющиеся многоугольники следует разбивать на

трех- или четырехугольники. Исключением может служить случай, когда плоский многоугольник расположен не на сгибах поверхности.

При моделировании объектов можно использовать модификаторы симметрии и массивов для экономии времени моделирования и избегания ошибок.

2.4. Формообразование

Существует следующий принцип: подавляющее большинство имеющихся в модели полигонов должно участвовать в образовании её формы. Связано это банально с оптимизацией, ведь чем больше полигонов у объекта, тем он массивнее и тем сложнее его обрабатывать. Пример показан на рисунках 12а и 12б.

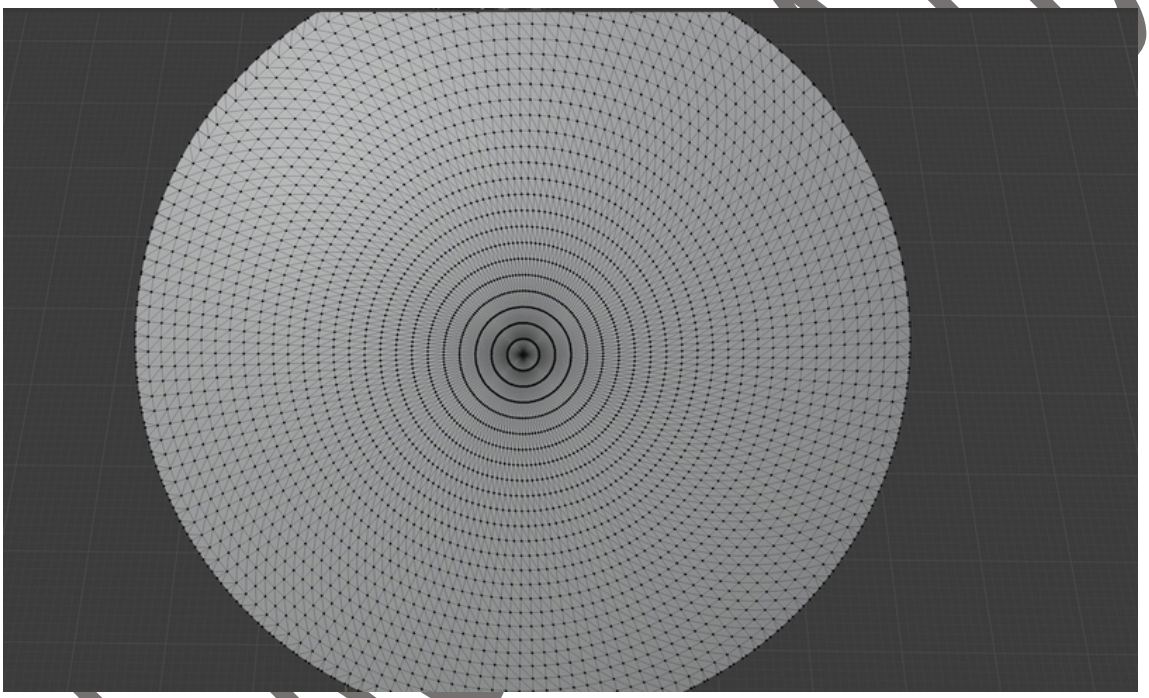


Рисунок 12а – Переусложнённый круг, выполненный из множества полигонов

Рисунок 12б -Круг, выполненный одним многоугольным полигоном.

И там и там плоская окружность, которые практически идентично выглядят в проекте, но в первом случае общее количество полигонов приближается к пятистам, в то время как во втором используется всего один полигон

Понять о наличии лишних полигонов можно следующим образом: если при удалении грани между двумя соседними сомнительными полигонами форма модели не изменяется, значит их можно объединить в один полигон.

2.5. Обработка точной модели

В целях удобства восприятия приведём пример моделирования 3D-принтера Anycubic 4 MAX Pro. Работу можно поделить на следующие задачи:

1) понимание принципов работы каждого блока;
2) создание базы референсов - фото составных частей и их расположение внутри принтера;
3) деление деталей на три категории: стандартные (те детали, модели которых можно скачать и отредактировать), уникальные (детали, создаваемые с нуля, которые невозможно скачать либо создание которых не является стандартным) и неважные (не отображающиеся в итоговой сборке или не несущие полезной информации);
4) поиск стандартных моделей;
5) обработка найденных стандартных моделей:

   а) редактирование сетки;

б) создание UV-развертки;

в) текстурирование;

6) создание уникальных деталей:

а) создание сетки по референсам;

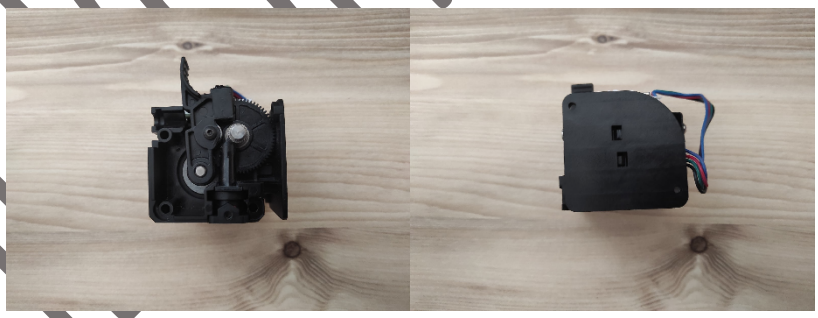
б) создание UV-развертки;

в) текстурирование;

7) Создание итоговой масштабной сборки.

Для создания базы референсов выбран принтер, находящийся в ремонте. Благодаря чему составляющие получилось сфотографировать со множества ракурсов в собранном и разобранном виде. Детали фотографировались со всех сторон для будущего сопоставления с найденными моделями либо для грамотного моделирования по референсам с нуля. Так, например, чтобы создать уникальную сборочную модель механизма подачи, пришлось фотографировать с 5 сторон (рисунок 13).

а) б) в)



г) д)

Рисунок 13 - Механизм подачи

а – сверху; б – снизу; в – сзади; г – спереди; д - справа

Затем нужно понять, какие детали важны и на сколько при создании модели 3D-принтера в виртуальной реальности. Например, так как модель нужна была для объяснения принципа работы механики 3D- принтера, точные модели электронных компонентов не были нужны. Также, хоть многие составные части и скреплялись винтами, их описание функционала и

детализация не сопоставимы с детализацией и описанием того же корпуса принтера. А раз так, то не имело смысла бы скачивать CAD модели винтов, обрабатывать их и подготавливать комплект текстур.

Для разбиения деталей по степени важности возможно ввести критерии выбора. Она может быть довольно простой. Пример используемой матрицы показан в таблице 1:

Таблица 1. Матрица определения важности деталей

	габаритные размеры детали или узла много меньше габаритных размеров блока, в котором она используется	Деталь или узел влияет на функционал механической работы принтера	найдена готовая модель либо визуально похожая, но требующая доработки
неважные	✓		✓
стандартные		✓	✓
уникальные		✓	

При обработке точной модели модели необходимо проверить на следующие пункты:

- соответствие геометрии реальной детали;
- содержание лишних вершин, ребер, плоскостей;
- «угловатость» изначально цилиндрических или сферических частей;

Для рассматриваемого 3D- принтера очень яркий пример- двигатель линейного перемещения каретки со шкивом. Можно моделировать каждый зубчик отдельно, как показано на рисунке 14, или же создать лишь видимость присутствия зубьев, нарисовав их на плоскости. Это не только упрощает работу, но и уменьшает вес итоговых файлов.

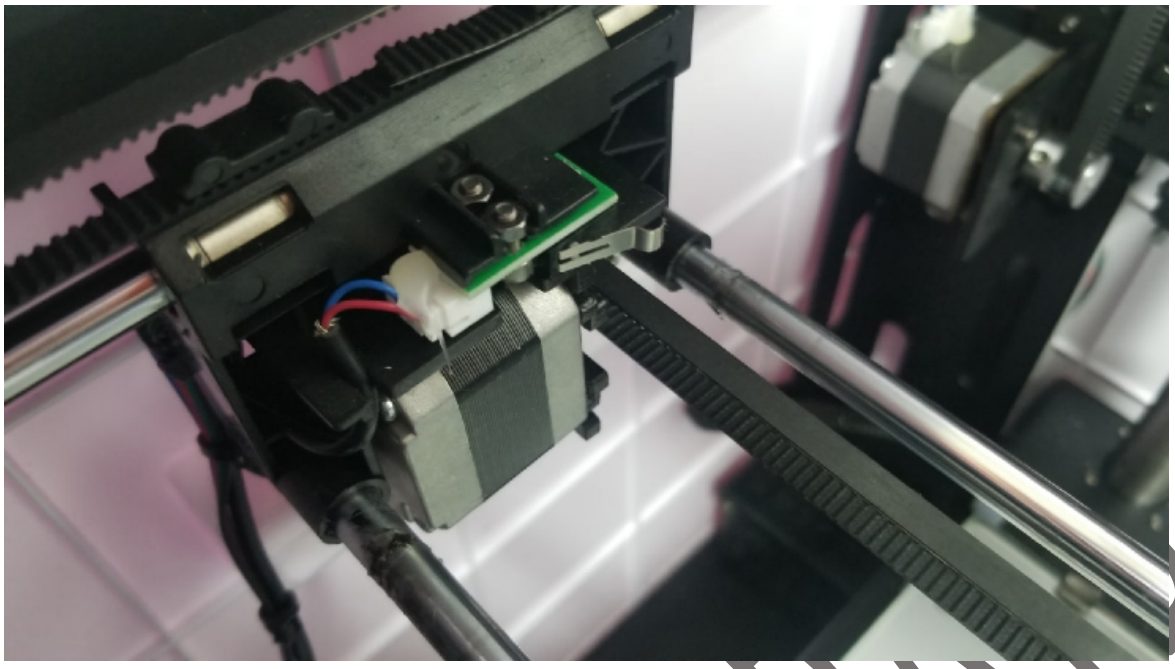


Рисунок 14 - двигатель линейного перемещения каретки со шкивом и ремнём

Полигональное моделирование означает, что в файле созданного вами объекта будет храниться информация о положении каждой его вершины. То есть память тратится под хранение трех координат - **x,y,z**. А значит, что чем больше полигонов, тем больше вершин. тогда и весить файл будет больше. Поэтому в полигональном моделировании нет таких понятий как окружность, цилиндр, сфера, потому что они представляются в виде многоугольников и многогранников. Если слишком увлечься оптимизацией веса файла, внешне напоминающий цилиндр многогранник станет двенадцатигранной призмой, что видно на рисунке 15.

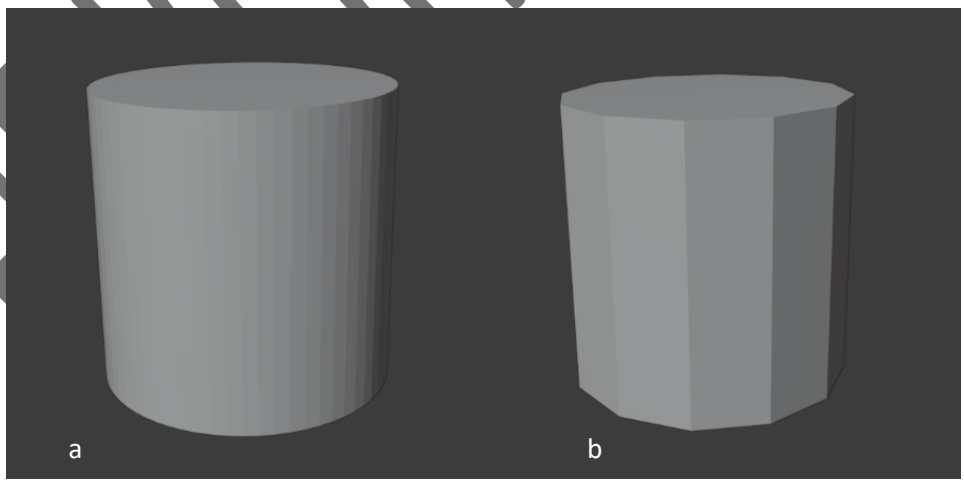


Рисунок 15 – визуальное сравнение количества граней у цилиндра в процессе оптимизации

a – 128 граней; b- 12 граней

Ввиду этого скачиваемые CAD или отсканированные модели перед использованием обязательно нужно проверять и редактировать. Так,

найденная модель корпуса принтера не полностью соответствовала внешне реальному корпусу, а также в некоторых местах присутствовали лишние вершины. Например, плоскость, описываемая четырьмя вершинами, содержала внутри еще множество ненужных вершин, которые следовало удалить. Пример показан на рисунке 16

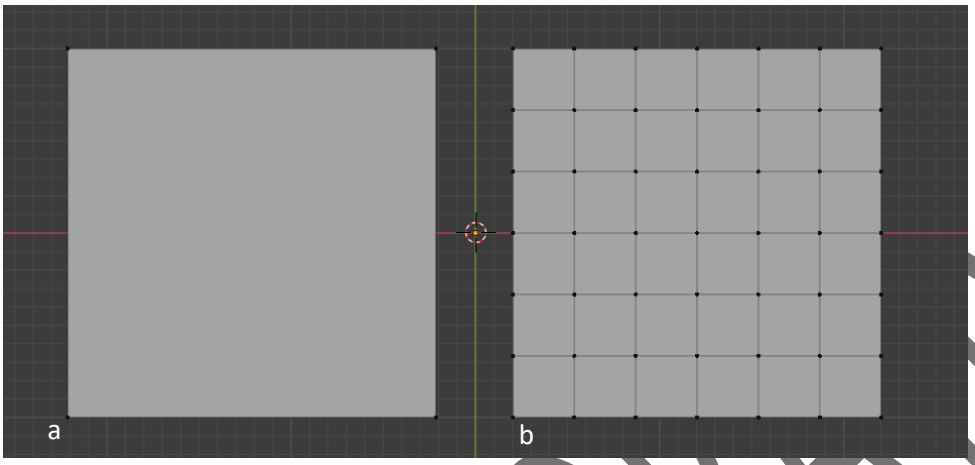


Рисунок 16 - чрезмерное определение плоскости вершинами

а – плоскость, определённая одним полигоном, b- та же плоскость, переопределённая двадцатью пятью полигонами

Говоря о соответствии реальной детали, в пример можно привести дверь корпуса принтера. Скачанная модель корпуса была без двери, и, как по времени, так и по затратам времени, проще сделать ее самостоятельно, нежели искать подходящую по габаритам, а потом еще и редактировать под настоящую (рисунок 17).

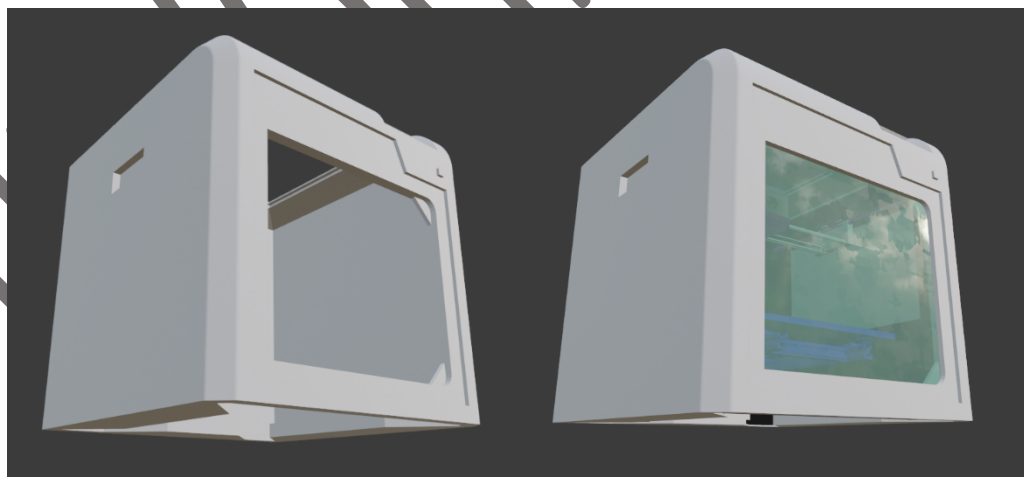


Рисунок 17 - Скачанная модель корпуса без и с дверью.

Для этого скопированы крайние вершины «дверного проема», после чего соединены в одну плоскость, состоящую из нескольких треугольников, которой после придано утолщение в соответствии с габаритами реальной двери.

Плоскость же разбита на четырёхугольники, так как игровые движки и другие приложения для полигонального моделирования преимущественно работают с объектами, состоящими из треугольных либо из четырехугольных полигонов. Иначе, при передаче им объекта, содержащего многоугольники, они сами разбивают его на треугольники, и не всегда делают это правильно (рисунок 18).

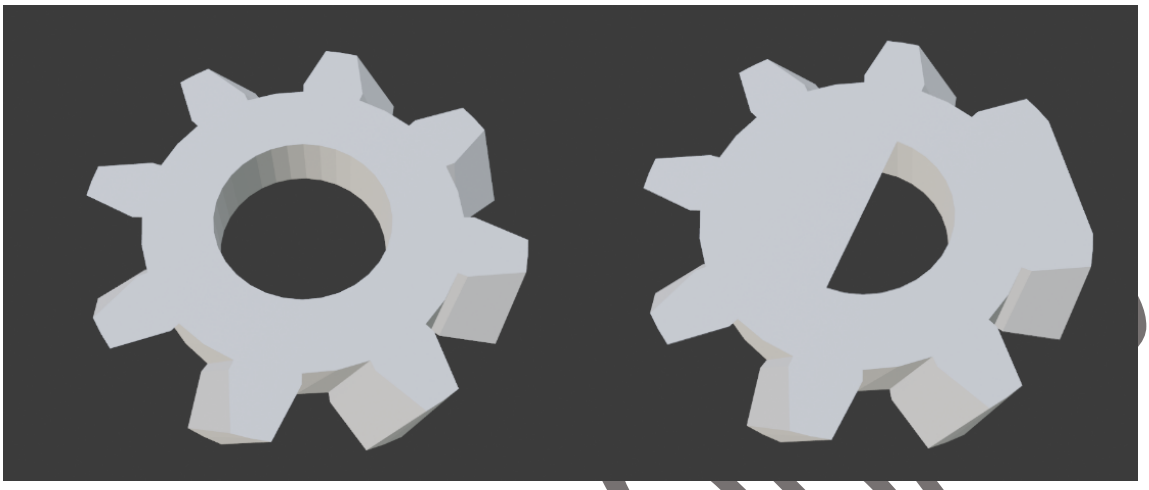


Рисунок 18. – Некорректный экспорт CAD модели шестеренки сразу в игровой движок

Детализацией с помощью полигонов можно пренебречь за счет использования текстур. Например, при редактировании скачанной модели шпильки оси Z, показанной на рисунке 19.а, являющейся изначально CAD моделью, была создана малополигональная (**low poly**) модель из цилиндра (рисунок 19.б), что в разы уменьшало вес итогового **FBX** файла, а визуальное соответствие с реальной деталью достигалось за счет запекания карты нормалей с высокополигональной (**high poly**) на **low poly** модель.

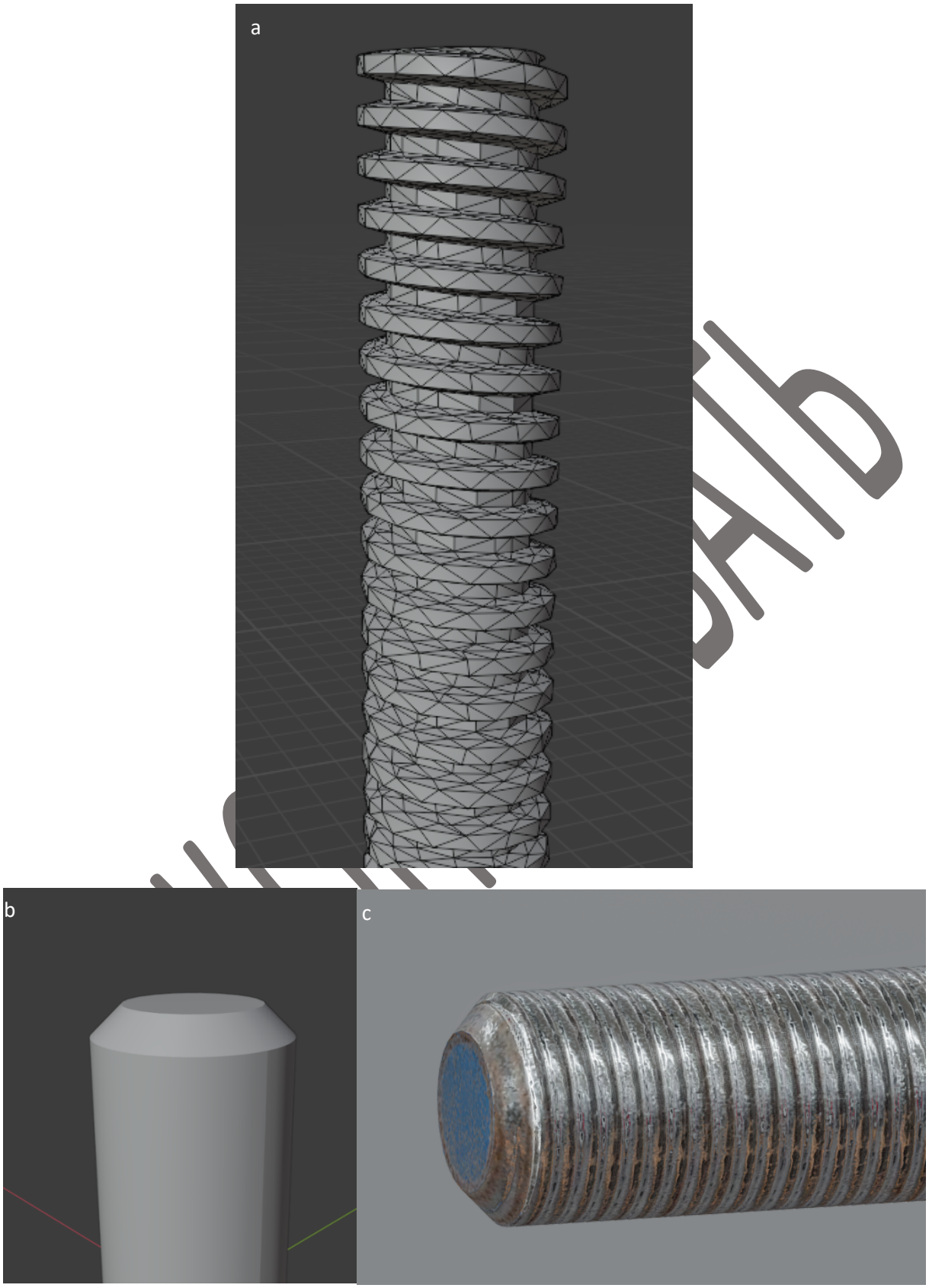


Рисунок 19 - Пример оптимизации модели за счет текстурирования

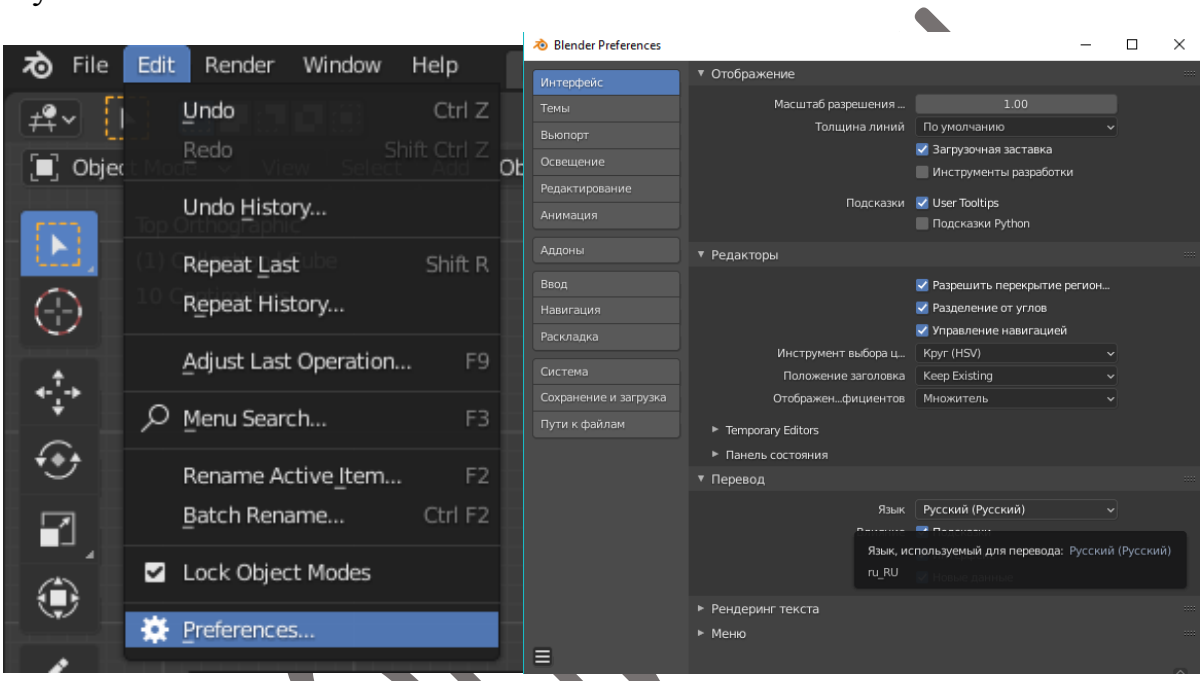
а - скачанная модель шпильки; б - уменьшение количества полигонов и редактирование сетки; в - итоговый результат благодаря использованию текстур

При создании уникальных объектов по референсам необходимо руководствоваться теми же принципами, что и при точном моделировании CAD моделей. Необходимо использовать несколько ракурсов, максимально приближенным к проекциям или чертёжным видам, чтобы отследить все габаритные и линейные размеры или как минимум сохранить масштаб. Местами необходимо создавать контур детали по фото, а после выдавливать на нужное расстояние. Также модель можно собрать из примитивов, а потом их объединить в один меш.

Глава 3. Основы работы в Blender

3.1. Интерфейс Blender

Познакомимся с окружением программы, которая позволит нам обрабатывать полигональные форматы: Blender. Для начала давайте выставим более удобные для нас настройки работы и сохранения. Для этого найдем в левом верхнем углу **edit** (Рисунок 20а). В выпадающем окне найдем опцию preferences и выберем их. Во вкладке **interface** выставим настройки как на рисунке 20б.



а б

Рисунок 20– Окна в Blender: а – Edit; б – настройки Blender

Остальные настройки в будущем каждый сможет выставить под себя в зависимости от удобства или особенностей проекта. Далее будет подробнее рассматриваться лишь вкладка Аддонов.

Теперь основное окно – **view port** или окно просмотра (Рисунок 21). Здесь расположена сцена, на которой в будущем будут располагаться детали проекта.

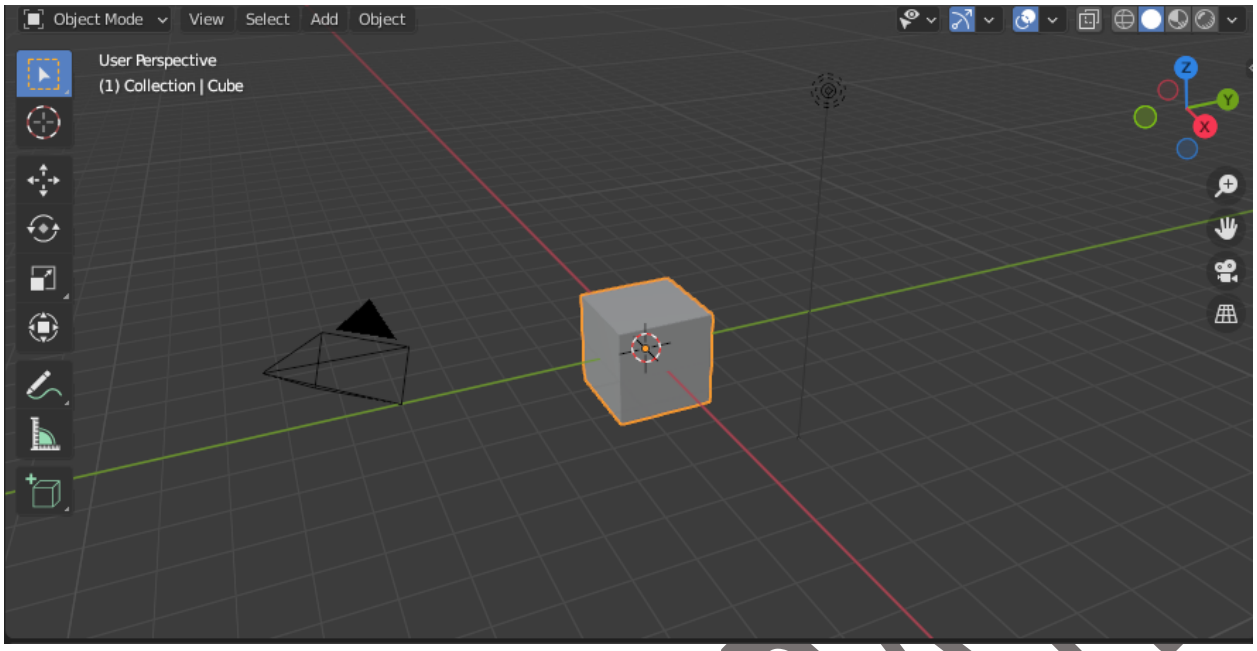
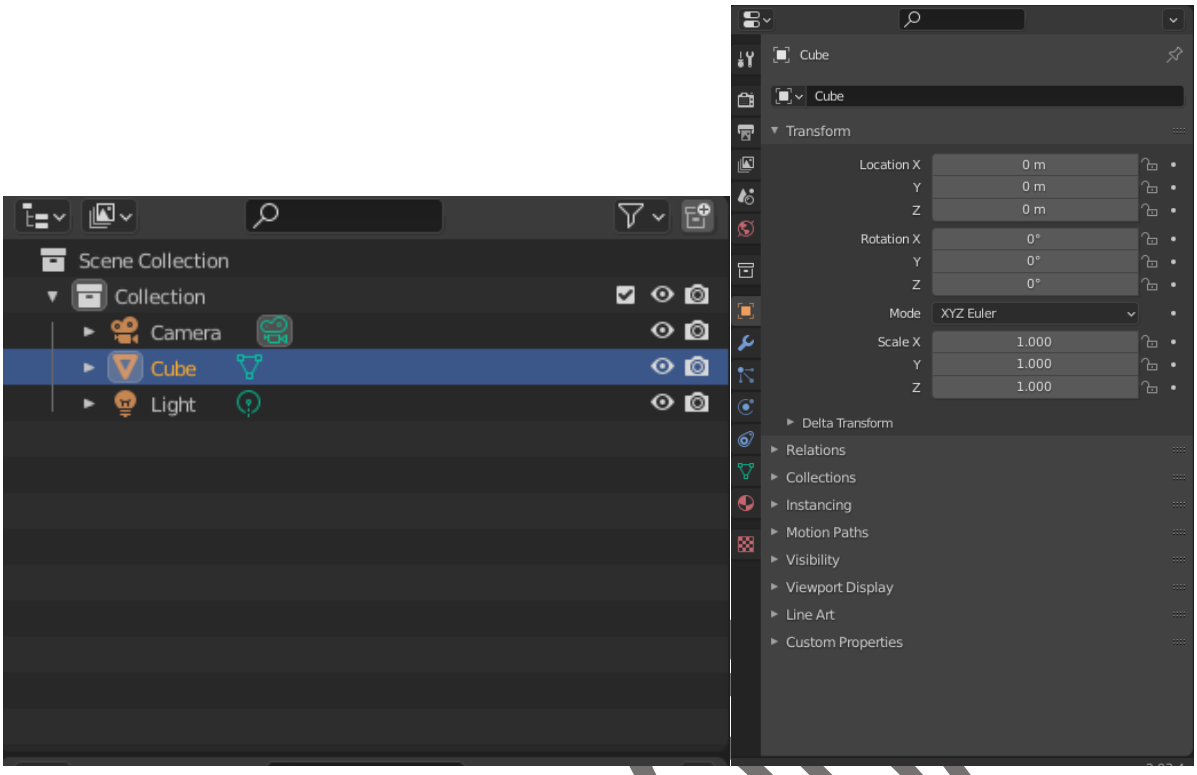


Рисунок 21 – окно View port

При создании нового файла на ней автоматически создаются и отображаются следующие объекты: стартовый куб, источник света, имитирующий Солнце и камера. Для перемещения по сцене можно либо зажать комбинацию Shift + колесико мыши и перемещать мышь в нужном направлении, либо зажать иконку «ладошки», расположенную в правом верхнем углу окна просмотра по осями координат, и также перемещать мышь. Для вращения либо зажимаем колесико мыши и перемещаем ее, либо зажимаем одну из осей **x,y,z** и также перемещаем мышь. При вращении колеса мыши происходит приближение/удаление от центра окна просмотра, также можно зажать иконку лупы со знаком плюс для аналогичного эффекта. При нажатии иконки камеры сцена начинает отображаться с точки обзора камеры и с ее настройками (которые сейчас выставлены по умолчанию). Последняя иконка с сеткой включает/выключает ортогональное отображение. Также его можно включить, нажав клавишу «5» на NumPad'е.

При нажатии клавиши «1» на NumPad'е произойдет перемещение на вид спереди, при нажатии клавиши «2» на вид сбоку и наконец, если нажать клавишу «7», то произойдет перемещение на вид сверху. Если же необходимо перейти быстро на вид снизу/сзади/другой бок, то нужно нажать те же клавиши, но уже с зажатым Ctrl.

Справа на экране расположено дерево объектов (Рисунок 22). В нем будут отображаться все объекты, которые присутствуют на сцене.



а б

Рисунок 22 – Окна в Blender:

а – Дерево объектов; б – Настройки объекта и рендера

При большом количестве предметов целесообразнее их группировать и сортировать по коллекциям. Сейчас в дереве есть только одна коллекция с аналогичным названием. Для создания новой нужно выбрать «Коллекцию сцены» и нажать на иконку коробки с плюсом. При наведении курсора высвечивается подсказка «новая коллекция». Таким образом появится «collection 2», не связанная с уже существующей «collection». Если вы хотите отсортировать объекты внутри конкретной коллекции, вы можете также создать новую коллекцию в нужной вам папке, выполнив ту же последовательность действий, с тем исключением, что необходимо выбрать в начале нужную вам коллекцию вместо «Коллекции сцены». Важно отслеживать какая коллекция у вас выбрана на момент создания объекта – именно в ней он и появится.

При нажатии правой кнопки мыши по выбранной коллекции появиться окно с доступными действиями. Можно копировать коллекцию, вставлять, вырезать, дублировать. Также при выборе пункта **Select Object** выберутся все элементы, которые находятся в этой коллекции.

Ниже дерева объектов расположены дополнительные настройки объекта и рендера (Рисунок 2.1.3 б), о которых мы расскажем чуть позже.

3.2. Object mode

В Blender существует два основных режима работы с объектом - **Object** и **Edit mode** или объектный режим и режим редактирования. По умолчанию при создании проекта включается объектный режим.

Выберем курсором куб на сцене. Теперь он подсвечивается оранжевым. Также у него появляется оранжевая точка – это pivot или точка опоры. В объектном режиме слева расположены инструменты воздействия на объект (Рисунок 23).

Рисунок 23 – Инструменты воздействия на объект

Первая иконка – прямоугольное выделение. При нажатии на нее и последующем зажатии левой кнопки мыши и выделении определенной области, все, что попало в нее, будет выбрано и подсветится оранжевым. Если зажать эту иконку, то появятся скрытые инструменты выделения, а именно

выделение «Лассо» и выделение окружностью. Переключать режимы выделения можно на клавишу W. Также вызывать элементы выделения можно на горячие клавиши. B – это выделение прямоугольником. C – выделение кругом. Радиус круга можно менять, крутя колесико мыши. Посредством левой кнопки мыши выделяются объекты. Если зажать колесо мыши и проводить по объектам, то выделение будет сниматься. Выйти из режима выделения окружностью можно, нажав правую кнопку мыши. Наконец, клавиша J позволяет выделять объекты лассо.

Следующая иконка позволяет переместить 3D курсор – нужно только нажать на иконку инструмента и после щелкнуть в нужном месте. Это нужно для использования продвинутого функционала blender при моделировании, который будет рассмотрен ниже.

Отдельным блоком расположены инструменты перемещения, поворота, изменения размера и трансформации (являющийся суммой трех предыдущих). При их выборе у объекта возникают оси соответствующего цвета, при захвате и перемещении/повороте которых соответственно меняются координаты, угол поворота или размер вдоль выбранных осей. Для ускорения работы можно воспользоваться горячими клавишами. При нажатии G появляется возможность перемещать объект в плоскости наблюдения с помощью мыши. Если же его нужно переместить относительно конкретной оси, нужно вместе с G нажать соответствующую клавишу x,y,z. При двойном нажатии x,y,z перемещение происходит не вдоль глобальных осей(осей сцены), а локальных (осей объекта). Для поворота нужно нажать клавишу R. Для вращения вокруг осей работают те же клавиши, что и для перемещения. Для изменения размера нажимается клавиша S, в этом случае параметры меняются вдоль всех осей. Чтобы увеличить/уменьшить объект вдоль конкретной оси можно воспользоваться теми же клавишами, что и для перемещения. Также, при нажатии клавиш g/r/s внизу слева возникает окно данного инструмента, позволяющее точно выставить параметры изменений (Рисунок 24). Для того, чтобы переместить, повернуть, увеличить объект на определенную величину нужно нажать соответствующую горячую клавишу и на клавиатуре написать значение цифрами перемещения, поворота или масштаба.

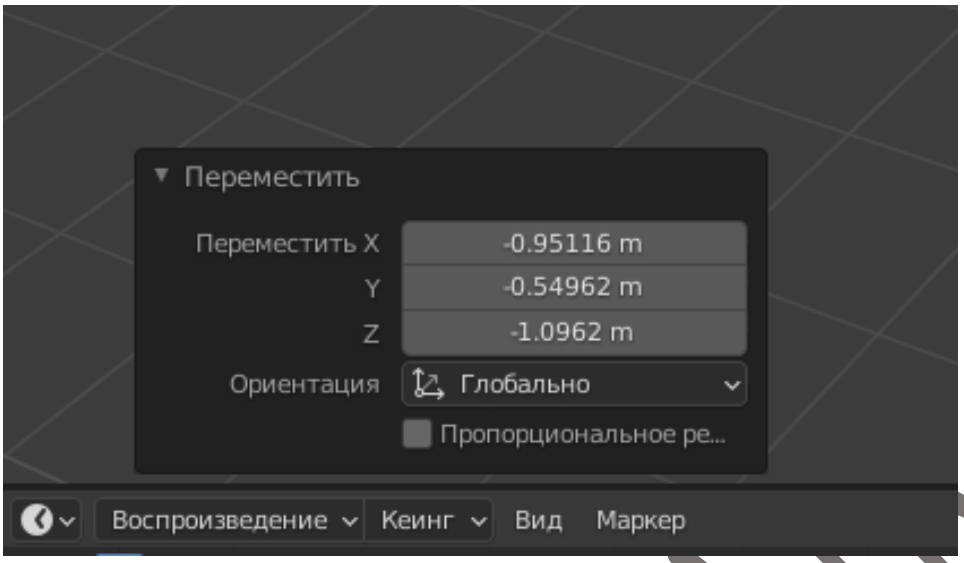


Рисунок 24– Параметры трансформаций объекта

Пример задачи. Переместить куб на 2м вдоль оси X. Нажимаем клавишу G, затем выбираем ось X, нажав по клавише, и вписываем значение 2 на клавиатуре, жмем Enter. Поставленная задача выполнена.

Также иногда появляются ситуации, в которых объект нужно перемещать, вращать, масштабировать вдоль двух осей. Сейчас это можно сделать за два действия, изменив координату перемещения, поворота или масштаба по двум осям. Однако сочетание клавиш Shift + X/Y/Z позволяет устранить одну из осей и совершать действие по двум другим.

Замечание: перед исключением одной из осей нужно выбрать само действие – перемещение(G), вращение(R), масштаб(S).

Для создания нового объекта нужно нажать Shift+A и в выпадающем меню выбрать меш→нужный объект (Рисунок 25).

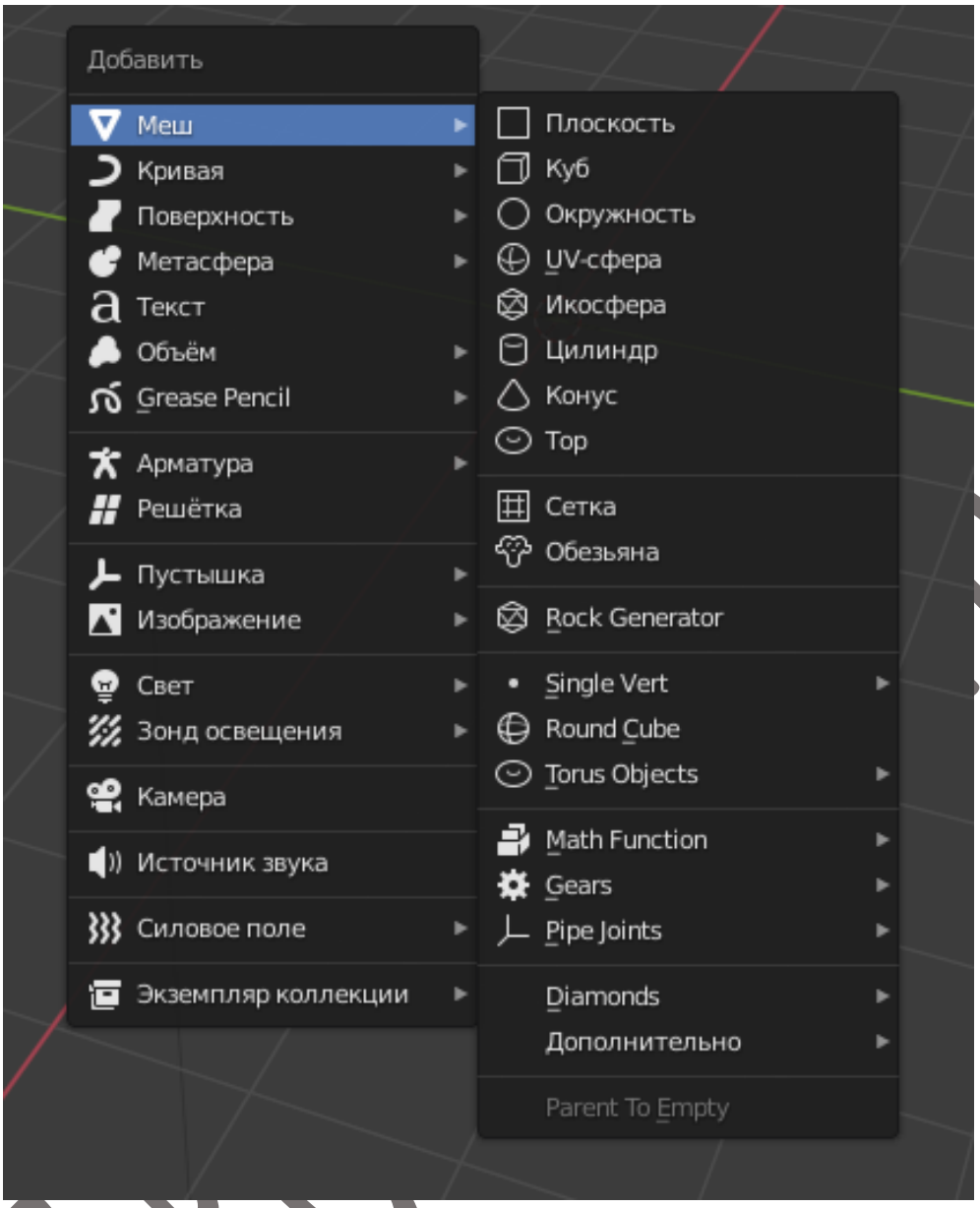


Рисунок 25– Создание нового объекта

Для удаления объекта существует несколько способов:

1) Нажать клавишу Del

2) В дереве проекта выбрать объект, нажать правую кнопку мыши, выбрать пункт Delete

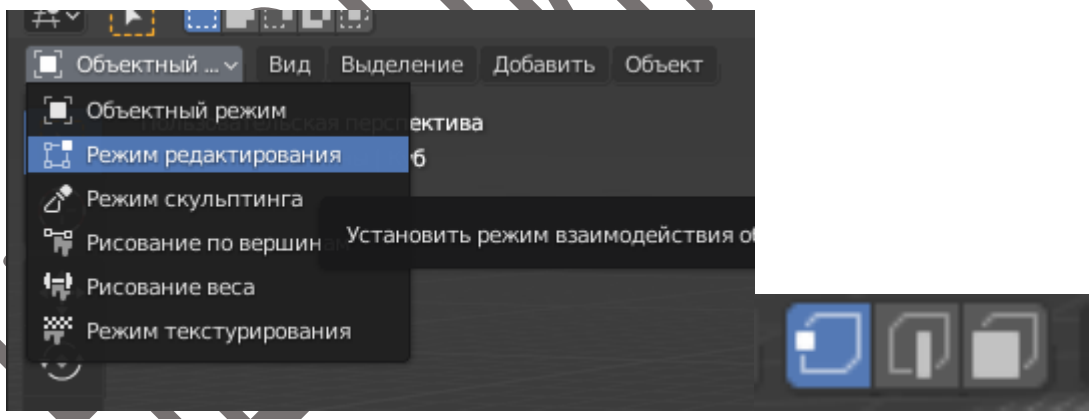
3) Нажать клавишу X и правую кнопку мыши

Иногда на сцене очень много объектов и с ними нужно выполнить какую-либо операцию, и за один раз инструментами выделения все не охватить. Для такого случая есть клавиша A. Она позволяет выбрать все объекты на сцене разом. Для снятия выделения нужно нажать левой клавишей мыши на пустое место в 3D Viewport.

Рассмотрим еще один случай, когда на сцене много объектов и они мешаются, но являются нужными. Для такой ситуации есть два решения. Первое – скрывать объекты. Для этого нужно нажать клавишу H. Для того, чтобы вернуть скрытый объект, нужно нажать Alt + H. Также эти действия можно совершить в дереве проекта. За скрывание отвечает глазик напротив каждого объекта. Второе решение – переход в изолированный режим. Он начинает действовать при нажатии клавиши «/». Для выхода из этого режима необходимо снова нажать эту же клавишу.

3.3. Edit mode

Теперь выберем куб и перейдем в режим редактирования. Для этого в левом верхнем углу окна просмотра нажмем на объектный режим и в выпавшем меню выберем нужный нам пункт (Рисунок 26а). Это же действие можно осуществить, нажав горячую клавишу Tab. В отличии от объектного режима здесь изменяются не общие параметры объекта, а лишь частные, так как тут мы воздействуем не на объект, а на его сетку. Слева к предыдущим инструментам (горячие клавиши которых работают и здесь) добавились новые. Для их вызова также существуют быстрые клавиши: Экструдирование (**Extrude**) – E, выдавливание внутрь (Inset) - I, создание фаски (**Bevel**) -ctrl+B, Loop Cut – ctrl+R , нож – k(использование вдоль осей остается то же, что и для перемещения). Однако в edit mode важно отслеживать к какой части объекта вы применяете инструмент. Это могут быть вершины, ребра и грани. Также есть три варианта выделения – точки, ребра, грани (Рисунок 26 б).



а б

Рисунок 26 – Окна в Blender:

а – переход в режим редактирования; б – варианты выделения

Описанные выше инструменты будут применяться лишь для выбранных частей объекта.

Рассмотрим работу каждого из инструментов более подробно.

Extrude или экструдирование – это инструмент, который позволяет создать новую вершину, ребро или грань. Для примера рассмотрим куб.

Выбрав одну из граней и нажав клавишу E, появилась ось выдавливания новой грани (Рисунок 27). Перемещая мышь, меняется расстояние выталкивания нового элемента. Также это расстояние можно вписать на клавиатуре цифрами. По аналогии выталкиваются новые граи и вершины.

Выдавливание внутрь или **Inset** – позволяет создать подразделение на выбранной грани схожей формы (Рисунок 28).

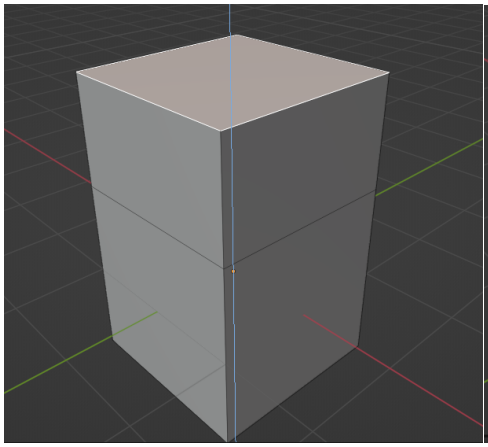


Рисунок 27– Extrude грани

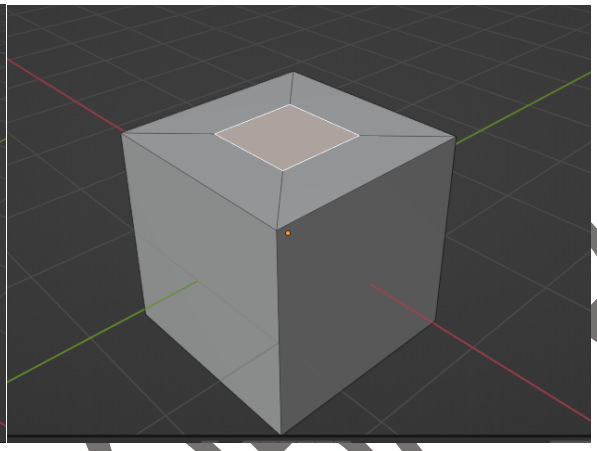


Рисунок 28 - Inset грани

Применив после **Inset** экструдирование, можно получить башню (Рисунок 29).

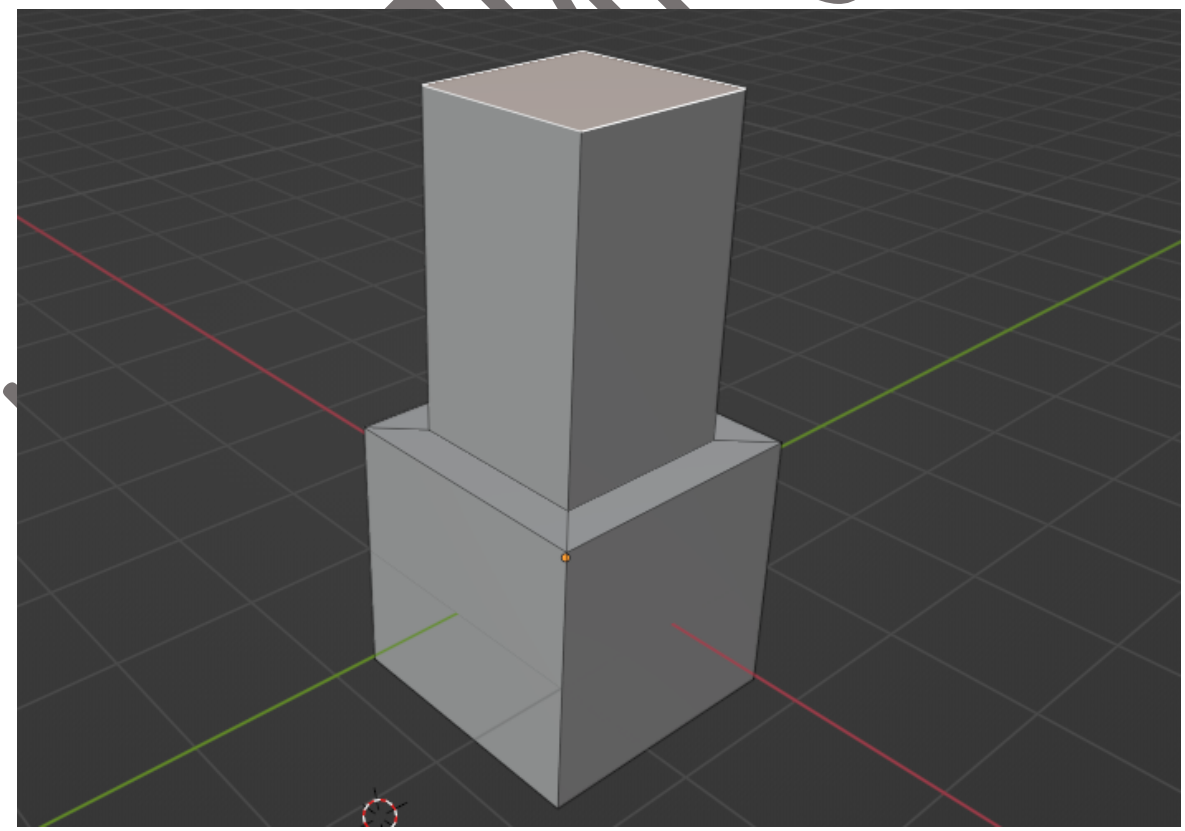


Рисунок 29– Пример работы двух инструментов редактирования меша

Создание фаски – **Bevel**. Создает фаску на выбранных элементах. Имеет много параметров. Кручение колесика мыши позволяет менять количество подразделений. Это позволяет делать скругления. Перемещение мыши дает возможность определить зону действия инструмента. Все эти настройки возможны только после нажатия сочетания клавиш Ctrl+B. Чтобы применить настройки, нужно нажать левую клавишу мыши. Однако если произошла ошибка и настройки выбраны не те, то не обязательно возвращаться назад с помощью Ctrl + Z. После применения ваших настроек появиться окошечко, расположенное в левом нижнем углу 3D Viewport. Развернув его, можно внести и задать более тонкие изменения (Рисунок 30). **Width** – это радиус или зона действия. **Segments** – это количество подразделений. Shape отвечает за то, выпукла или вогнута поверхность. **Clamp Overlap** позволяет изменять радиус действия with только в конечных пределах, не допуская самопересечений. **Vertices** позволяет подразделять вершины.

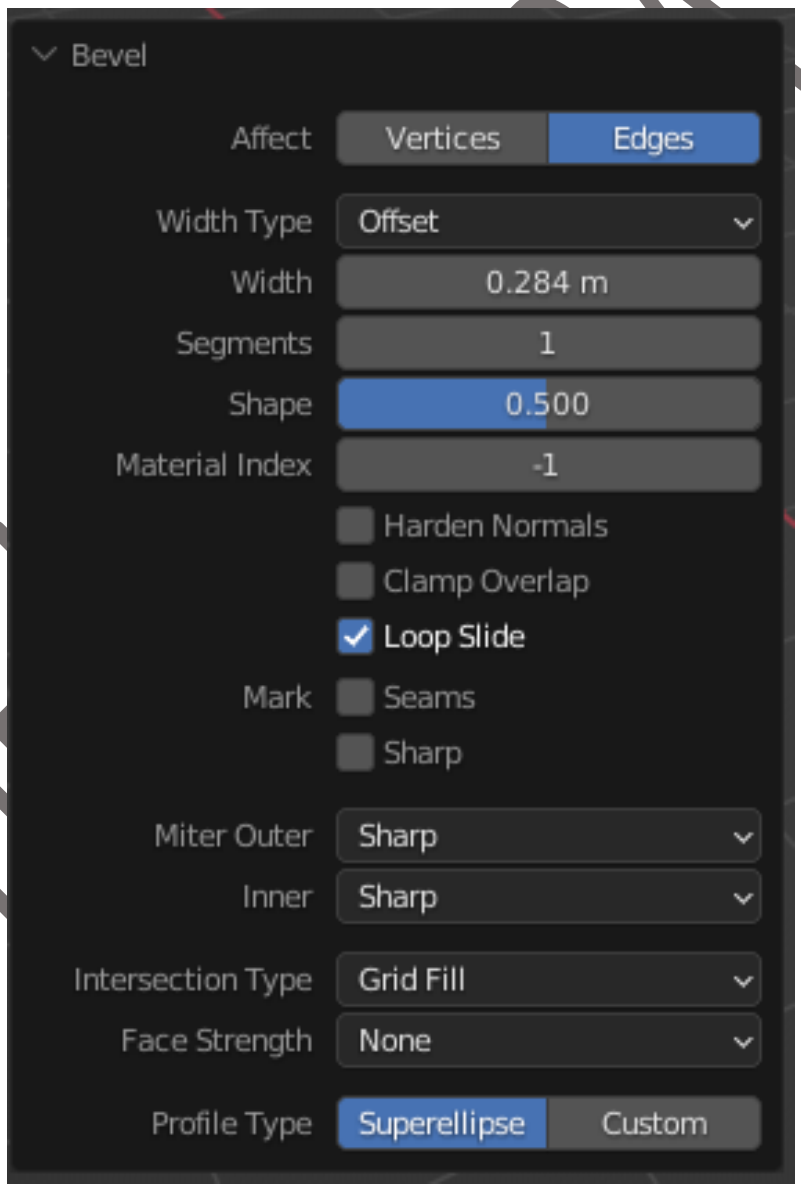


Рисунок 30– Настройки инструмента Bevel

Замечание: если выбрать только одну вершину. То первичные настройки с помощью мыши установить не получится. Задавать из нужно только на вкладке vertex в меню **Bevel**.

Инструмент **Loop Cut**. Он позволяет подразделить поверхность на равные петли. Количество петлей задается колесиком мыши. Затем, нажав левую кнопку мыши, выбирается положение, где эти петли могут быть расположены. Снова нажимается левая кнопка мыши, для применения действия инструмента. В левом нижнем углу также появится меню с дополнительными настройками инструмента.

Нож – **Knife**. Позволяет разрезать сетку модели в нужных местах. Чтобы сделать первый разрез, нужно нажать левой кнопкой мыши в нужное место меша. Далее вести нож в другое место и повторять действия. Чтобы закончить и применить действие инструмента, нужно нажать клавишу Enter. Для отмены действия нужно нажать клавишу Esc.

Дополнительный функционал ножа. Иногда возникает необходимость разрезать ребра строго по их центру. Для этого нужно резать ребра с зажатой клавишей Shift, в таком случае нож будет резать строго по центру ребер. Чтобы резать меш на сквозь, нужно нажать клавишу C. Она включает и выключает этот режим.

Наконец, есть случаи, когда разрезать нужно с учетом определенного угла. И для этого есть решение. Нажав клавишу A нож будет резать с привязкой к углу. Для записи конкретного угла нужно просто ввести его значение цифрами после нажатия клавиши A.

Поговорим о способах выделения. При нажатии клавиши A выделяются все элементы, как и в объектном режиме. Можно использовать прямоугольное выделение, выделение кругом или лассо, как и в объектном режиме. Важно отслеживать также в каком типе отображения окна просмотра вы находитесь (Рисунок 31).

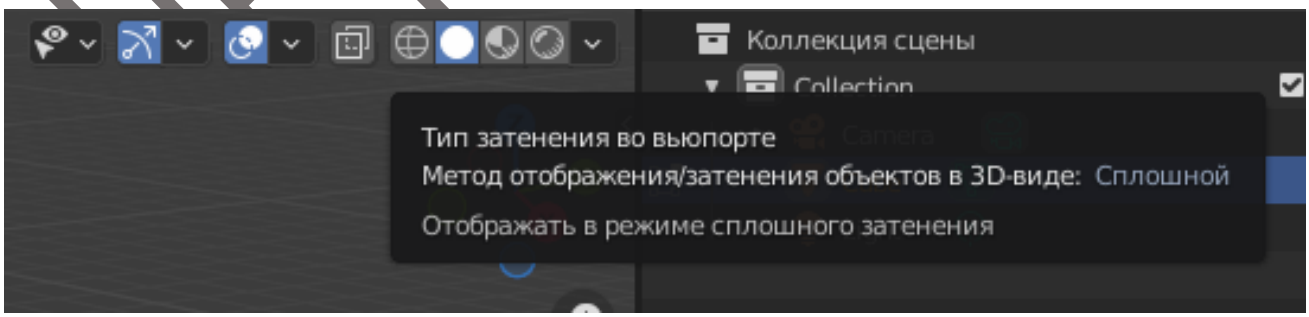


Рисунок 31– Тип отображения окна

Если вы используете сплошной, предпросмотр материала или рендер, то выделится только видимая часть с вашей точки просмотра. И только в типе сетка или **Toggle X-Ray** будет выделятся все, что попало в область выделения.

Чтобы понять остальные возможности выделения, добавим в edit mode еще один объект – сферу. Да, в режиме редактирования также можно добавлять объекты. Для этого используем то же сочетание клавиш, что и в объектном режиме - Shift+A. Важно помнить, что при последующем переходе в object mode созданное в edit mode будет одним объектом и центром для инструментов перемещения, поворота, изменения размера будет все тот же pivot куба (Рисунок 32).

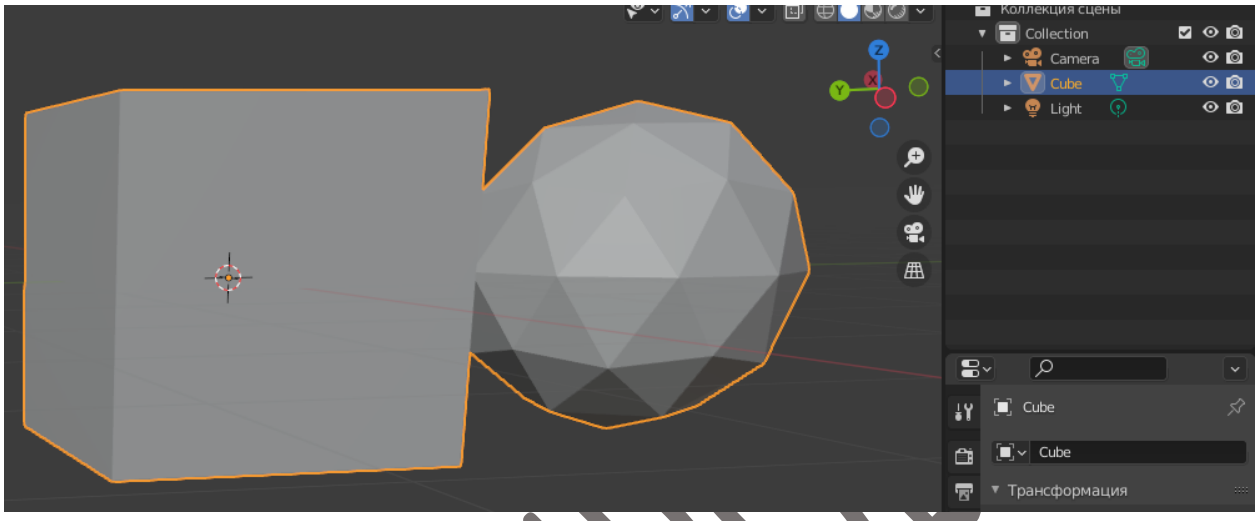


Рисунок 32– Pivot куба

Вернемся к выделению. Если же в edit mode нужно выбрать только один предмет, например, куб, можно навести на него курсор и нажать L.

Также можно выбрать одну вершину, зажать ctrl и выбрать другую, не соседнюю вершину. Тогда выделятся все вершины, которые расположены на кратчайшем пути от первой вершины до второй.

Если выбрать одно ребро, затем рядом с ним расположенное и нажать Ctrl+Shift+ «+», то далее ребра начнут выделяться в указанном направлении (за направление отвечают первые два ребра).

Если выбрать одну грань, затем нажать Ctrl + «+», то выделятся следующие грани в радиусе одной грани, затем следующие и так далее.

Двойной клик по замкнутому контуру позволяет сразу выбрать его.

И последнее, можно, зажав shift, выбрать нужные части сетки (в object mode также работает с объектами).

3.4. Работа с Pivot и 3D курсором

Вернемся к 3D курсору. Он отвечает за место, где происходит появление объектов. Меняя место 3D курсора, меняется и место появления объектов на сцене. То, где он будет расположен, отвечает не только инструмент Cursor, расположенный слева в окне 3D Viewport. Нажав сочетание клавиш Sift + S, появится меню (Рисунок 33).

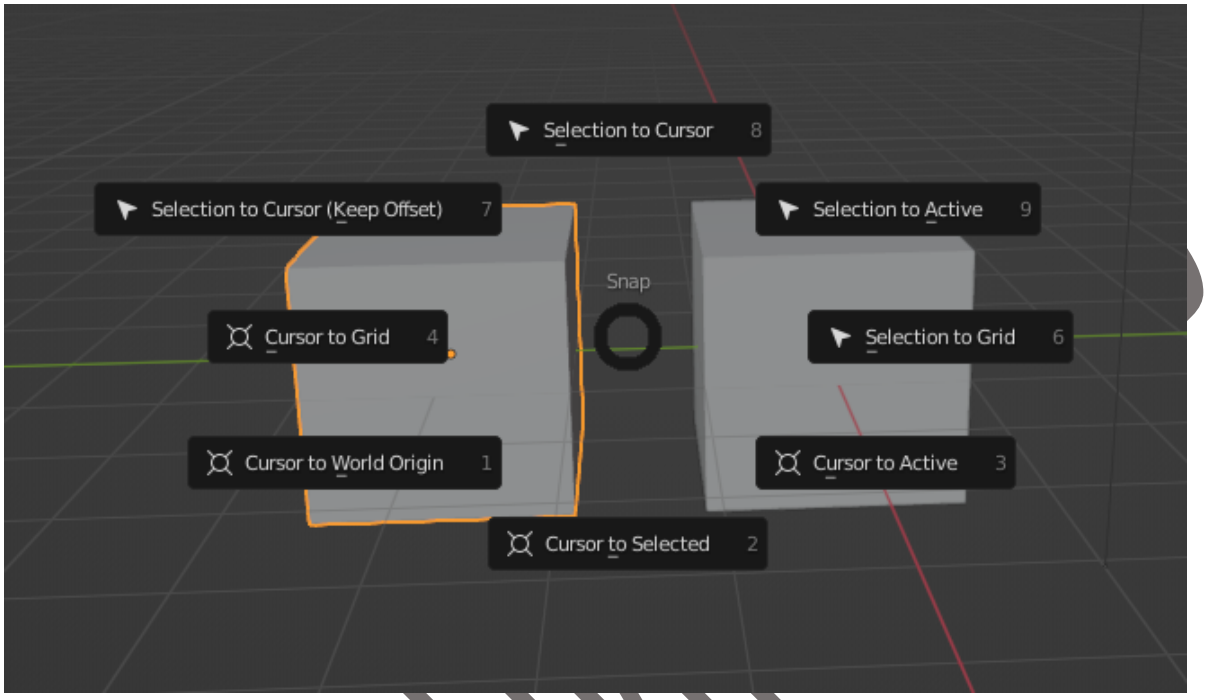


Рисунок 33 – Выпадающее меню

Из рисунка 33 видно, что курсор можно переместить:

1) Обратно в центр координат – **Cursor to World Origin**.
2) К центру выбранных объектов – **Cursor to Selected**.
3) К центру объекта -**Cursor to Active**.
4) К ближайшему пересечению сетки на сцене – **Cursor to Grid**.

Можно заметить, что такие же действия можно совершать и с обычными объектами.

Центром у объекта считается его **Pivot**. Рассмотрим управление Pivot-ом (его еще называю **Origin**). В объектном режиме можно перемещать Origin к 3D курсору и возвращать в геометрический центр объекта (или в другое место, указанное в меню). Для этого нужно нажать правую клавишу мыши и выбрать Set Origin -> Origin to 3D Cursor или Origin to geometry (или другой нужный вам пункт, наведя на него выскочит подсказка с описанием его действия).

А что, если необходимо, чтобы Pivot находился в одной из вершин куба или в центре какой-либо грани? Для такого случая задача разбивается на 2 этапа. Первое – это перемещение 3D курсора в нужно место для Pivot.

Решение:

Необходимо перейти в режим редактирования объекта. Выбрать необходимый элемент: вершину, ребро, грань, или разные их совокупности. Далее переместить вначале 3D курсор в центр выбранных элементов.

Второе – переместить Pivot в объектном режиме на место 3D курсора.

Решение:

Возвращаемся в объектный режим. Нажимаем правой кнопкой мыши в 3D Viewport. Выбираем Set Origin -> Origin to geometry.

3.5. Модификаторы

Модификаторы — это инструменты, которые позволяют облегчить процесс моделирования и осуществлять некоторые операции над объектами. В данной главе будут рассмотрены только самые базовые модификаторы.

Array - позволяет создавать копии объекта, на который наложен этот модификатор (Рисунок 34).

Настройки:

- Count - количество копий объекта.
- Relative Offset - расстояния между копиями объекта по трем осям координат. Расстояние отсчитывается от габаритных размеров объекта.
- Constant Offset - это постоянное дополнительное смещение между объектами.
- Object Offset - смещение относительно выбранного объекта. Выбор объекта осуществляется с помощью пипетки.
- Merge - позволяет “склеивать” объекты, если они расположены на заданном в Merge расстоянии.
- UVs - настройки UV развертки.

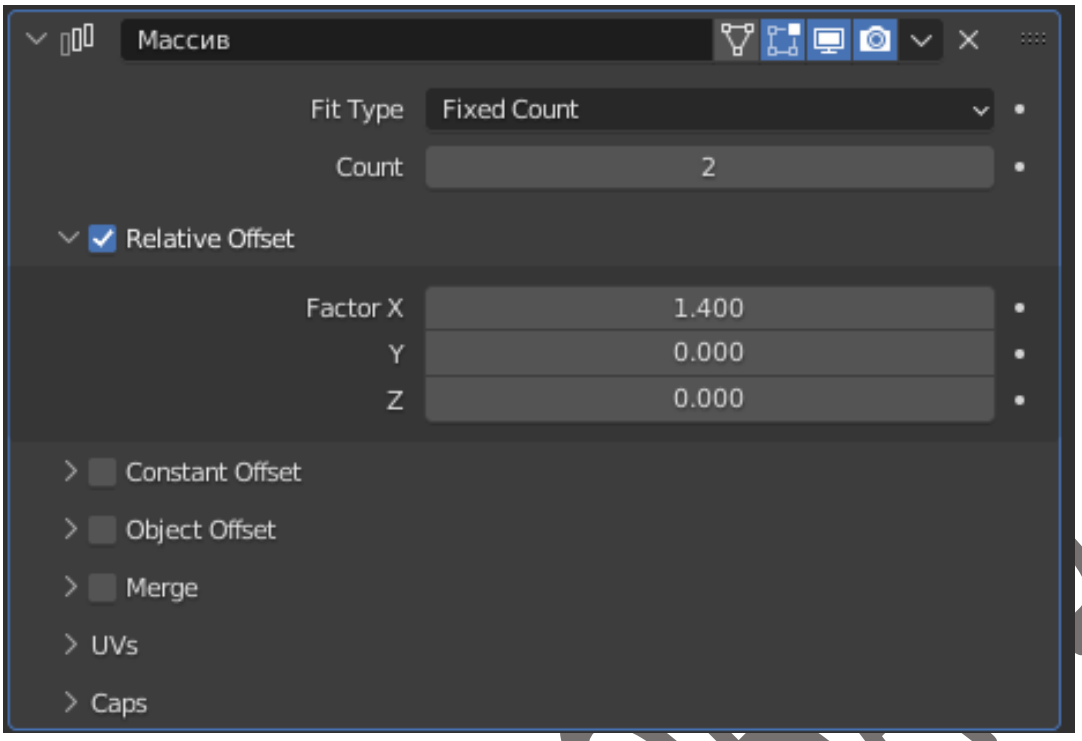


Рисунок 34 – Модификатор «Array»

Bevel - добавляет фаски на объект (Рисунок 35).

Настройки:

- Amount - радиус фаски.
- Segments - количество подразделений в фаске.
- Profile - позволяет настраивать изгиб фаски. Она может иметь как выпуклость, так и вогнутость.
- Geometry - настройки геометрии фаски.

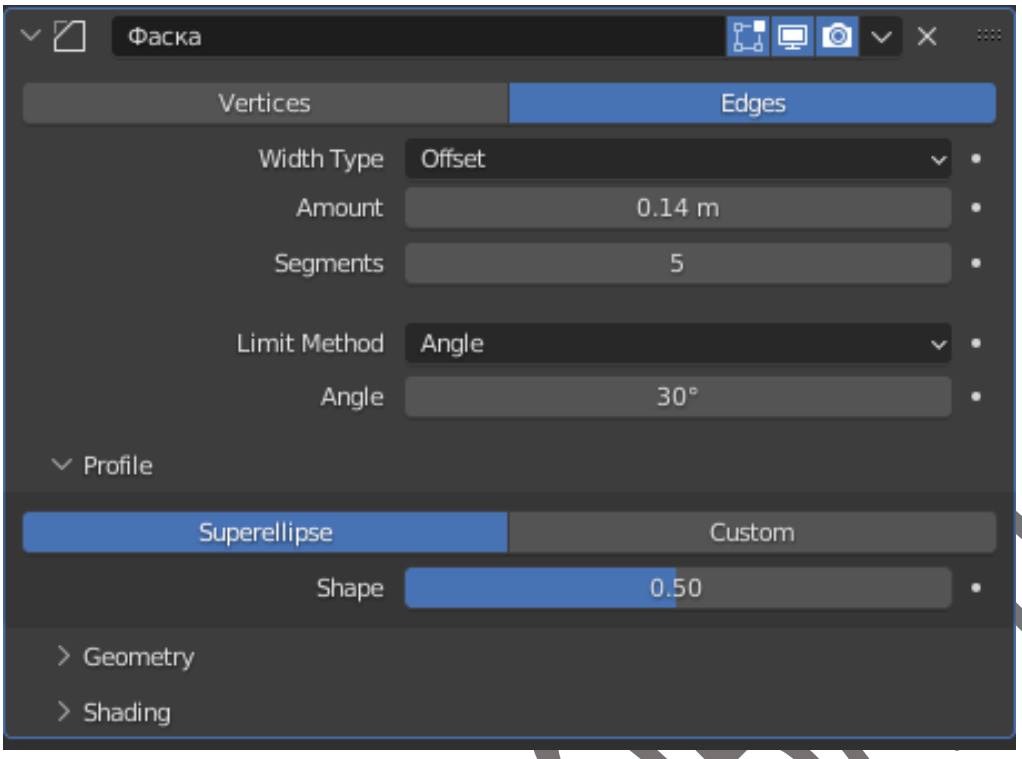


Рисунок 35 – Модификатор «Bevel»

Boolean - позволяет объединять объекты, вычитать геометрию одного объекта из другого, оставлять только пересечение объектов (Рисунок 36).

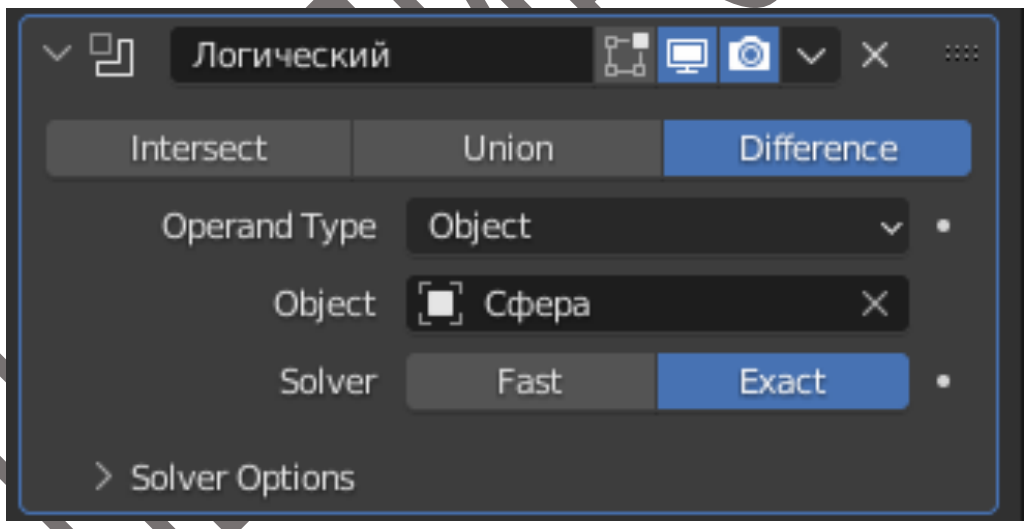


Рисунок 36 – Модификатор «Boolean»

Decimate - уменьшает количество полигонов у модели. Настраивается ползунком Ration.

Mirror - позволяет симметрично отражать объекты по выбранным осям относительно своего пивота (Рисунок 37). Есть возможность выбирать в качестве симметрии любой объект (пункт **Mirror Object**).

Настройка «**Clipping**» отвечает за сшивание геометрии близко расположенных исходного объекта с его симметричным.

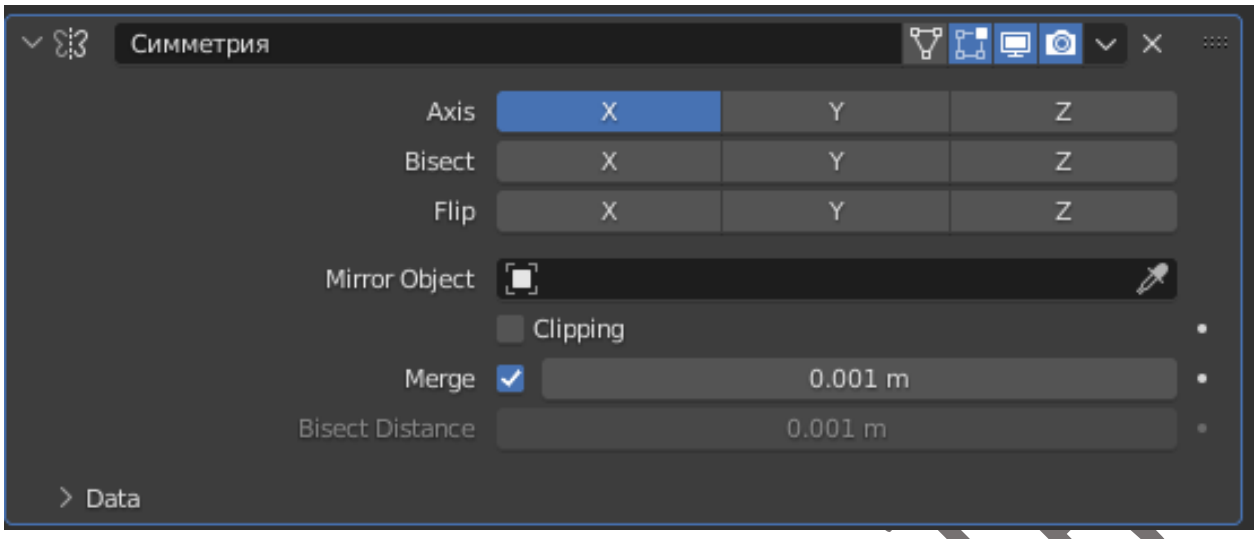


Рисунок 37 – Модификатор «Mirror»

Solidify - добавляет толщину объектам (настройка - **Thickness**).

Subdivision Surface - сглаживание объекта. **Levels Viewport** – отвечает за количество подразделений поверхности.

Wireframe – модификатор, создающий меш вокруг полигональной сетки объекта. Рисунок 38.

Параметр толщины сетки – **Thickness**.

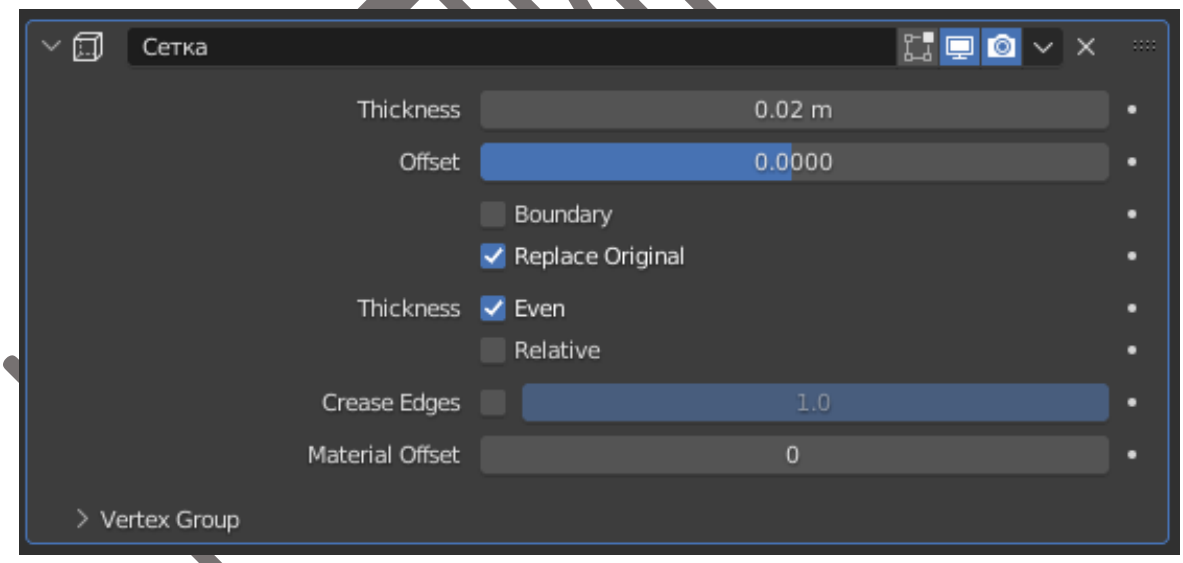


Рисунок 38 – Модификатор «Wireframe»

3.6. Адаптация сетки

В некоторых случаях возникает необходимость уменьшить количество полигонов в плоскости. Основные виды переходов изображены на рисунке 39.

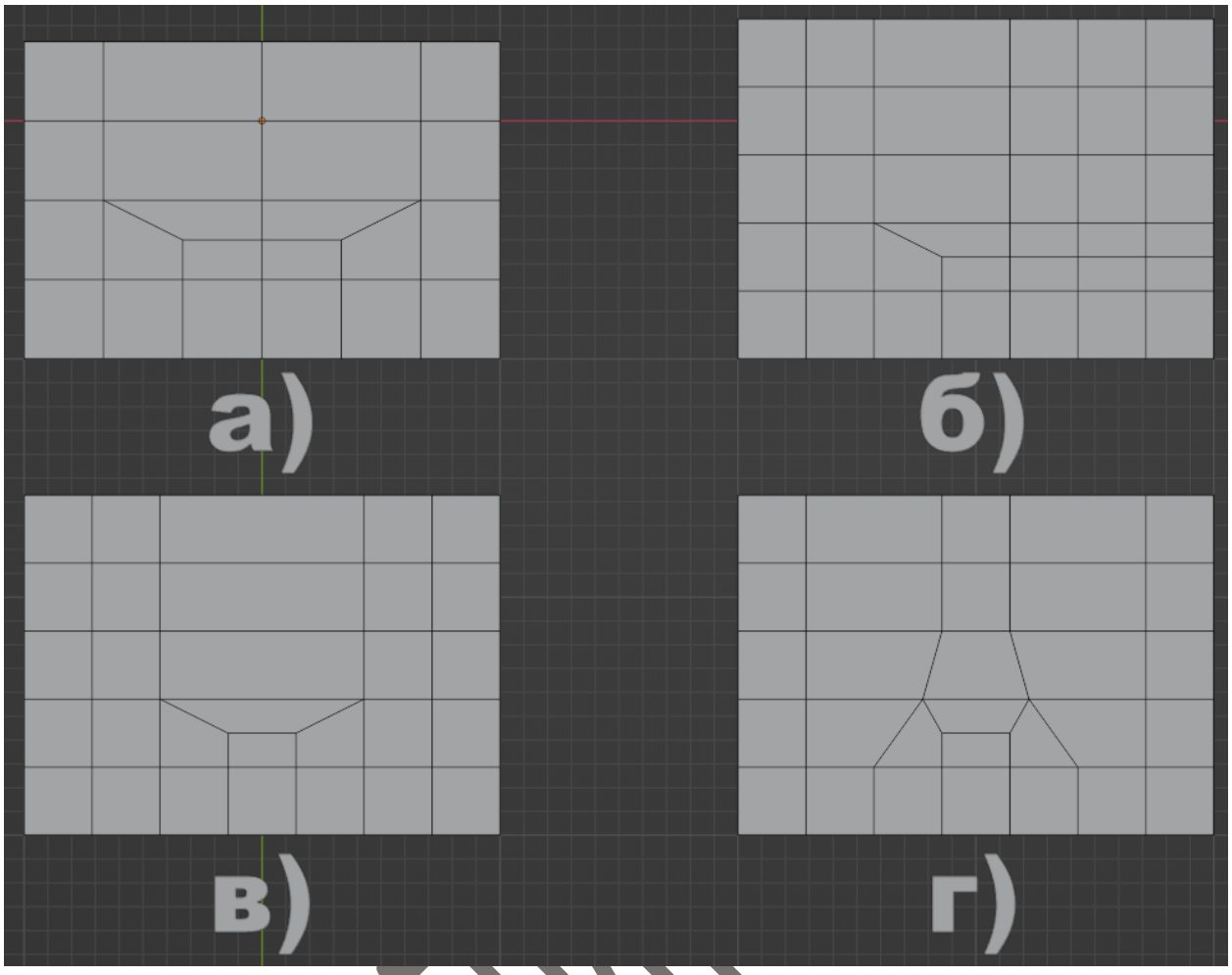


Рисунок 39 - а) Переход от четырех полигонов к двум б) переход от двух полигонов к одному в) Переход от трех полигонов к одному г) Переход от пяти полигонов к трем

Разберем по порядку процесс создания каждого из вариантов в картинках. Введем обозначения. Зеленая линия – ребра, которые нужно добавить. Красная линия – ребра, которые нужно растворить. Обратите внимание на рисунок 40.

Примечание: растворение ребер происходит путем нажатия клавиши **X** и выбора пункта **Dissolve edges**

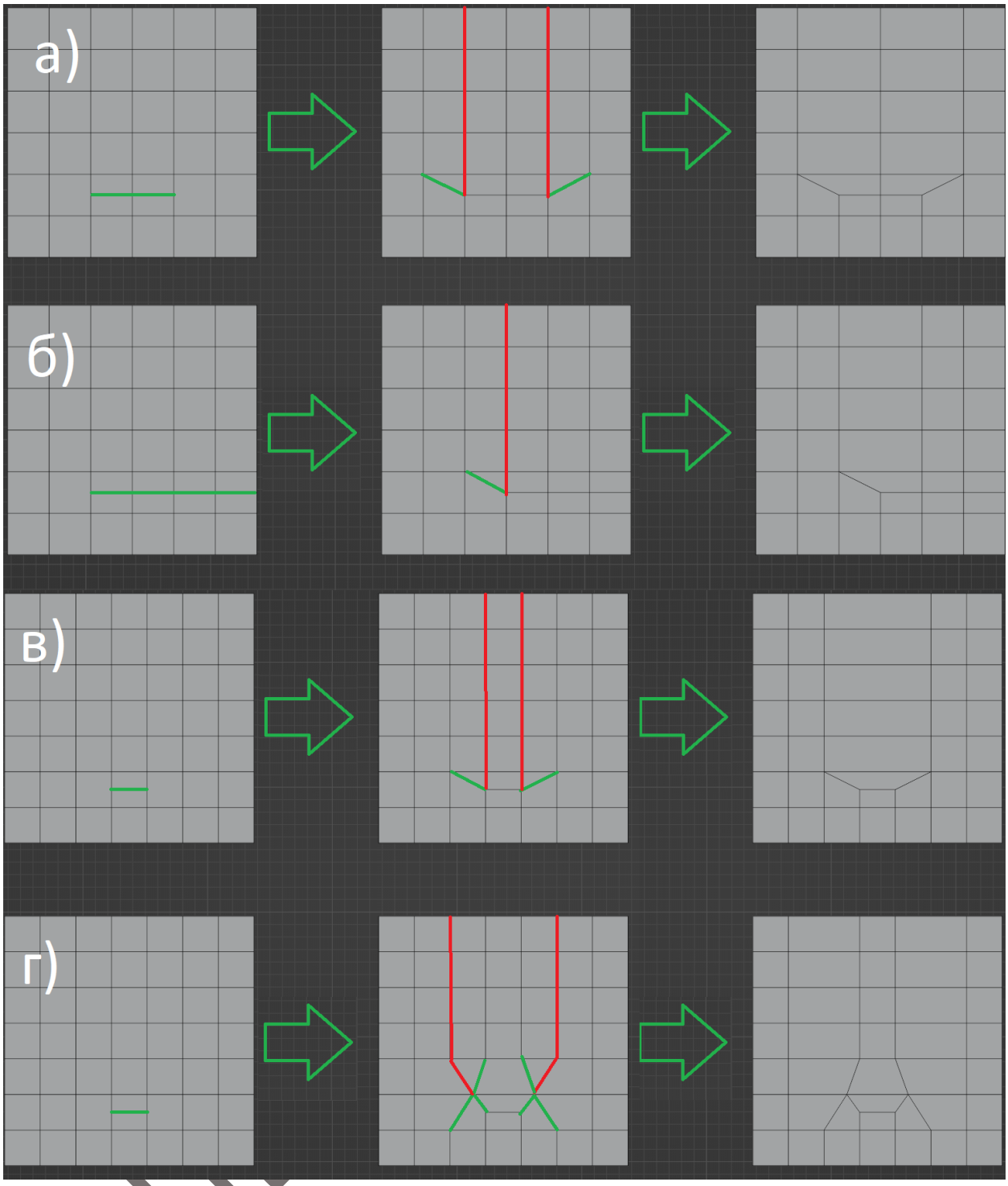


Рисунок 40 - Поэтапное уменьшение сетки а) Переход от четырех полигонов к двум б) переход от двух полигонов к одному в) Переход от трех полигонов к одному г) Переход от пяти полигонов к трем

3.7. Триангуляция модели

Что же необходимо делать при моделировании 3D- деталей, чтобы избежать неприятных ситуаций с наложением текстур и правильным отображением поверхности у них?

Мы уже говорили про направление нормалей у объекта и про триангуляцию сетки.

Приведем пример непредсказуемости результата, и как вариант, наличия шва на модели при отсутствии разбиения сетки модели на треугольники. В blender полигоны (в общем случае многоугольники) могут быть не плоскими. Для простоты рассмотрим случай для четырехугольника. На рисунке 41 можно увидеть сглаженную поверхность и ее сетку. Нас интересует фигура, отмеченная галочкой. Видно, что она не разбита на треугольники и вся поверхность не имеет угловатостей.

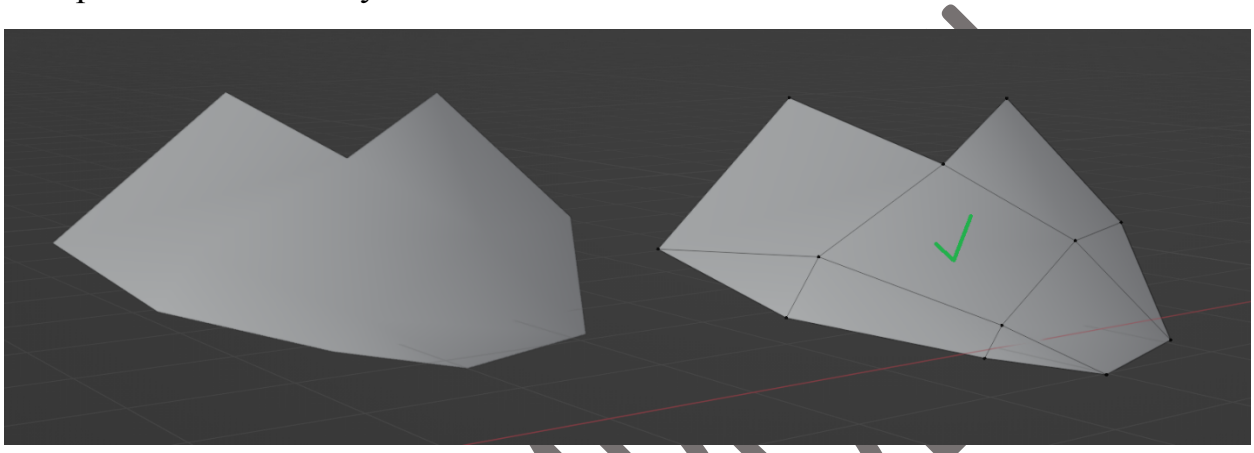


Рисунок 41 - Поверхность с корректным сглаживанием без разбиения на треугольники центрального полигона

Теперь допустим предположение, что при экспорте объекта в FBX формат, полигон будет разбит на треугольники, как на рисунке 42. Видно, что поверхность все такая же гладкая.

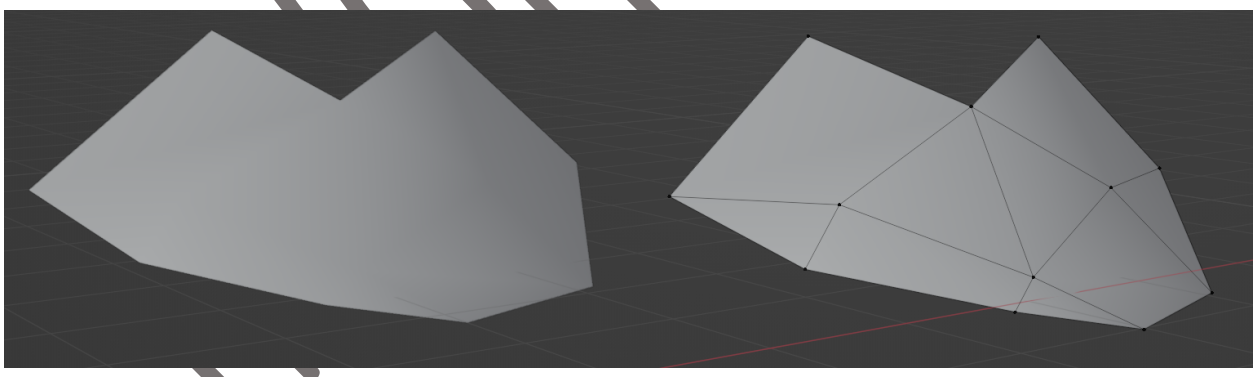


Рисунок 42 - Поверхность с корректным сглаживанием с разбиением на треугольники центрального полигона одним из двух способов

Вторым предположением будет разбиение полигона, как на рисунке 43. Как раз в этом случае у поверхности появляется артефакт.

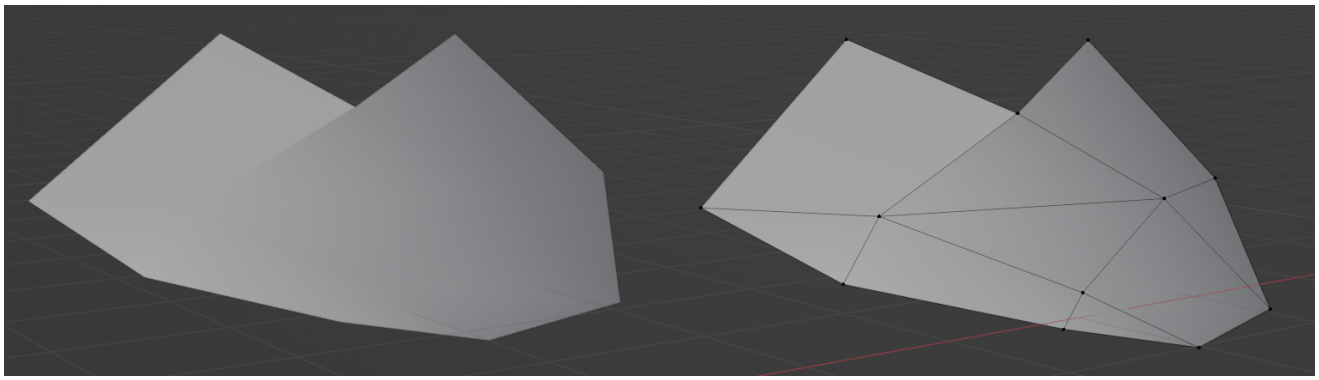


Рисунок 43 - Поверхность с некорректным сглаживанием с разбиением на треугольники центрального полигона одним из двух способов

На этом примере ярко видно отрицательное воздействие наличия случайности. Поэтому перед экспортом объекта в другую программу, необходимо триангулировать сетку самому и проверять наличие артефактов.

Однако это не единственная причина наличия некорректного отображения поверхности модели. Очень важно моделировать объекты с правильной сеткой. Далее будем использовать понятие топологии сетки.

Топология в 3D графике – это построение поверхности по геометрической сетке объекта. Поверхность 3D модели состоит из полигонов.

Создавая модель с правильной топологией, можно быть уверенны в ее корректном отображении после триангуляции.

3.8. Двойные вершины

При моделировании объекта или при редактировании сетки у САПР модели может возникнуть ситуация, когда две вершины расположены в одной точке пространства. Это может произойти по множеству причин. Из-за этого могут появится проблемы отображения модели после сглаживания. Пример можно посмотреть на рисунке 44 и 45. На первом изображении видно, что оранжевая вершина находится посередине горизонтального ребра и сглаживание происходит корректно. На рисунке 45 оранжевая вершина была перемещена специально в соседнюю левую. Из-за этого появилось некорректное сглаживание объекта.

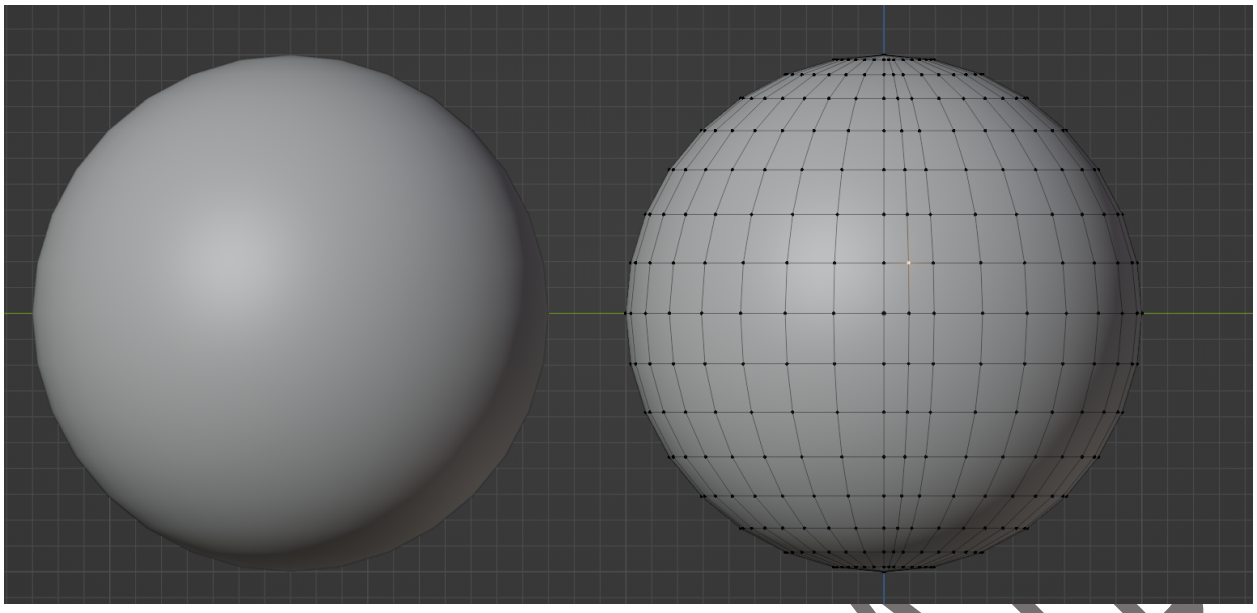


Рисунок 44 - Сглаженная сфера без двойной вершины.

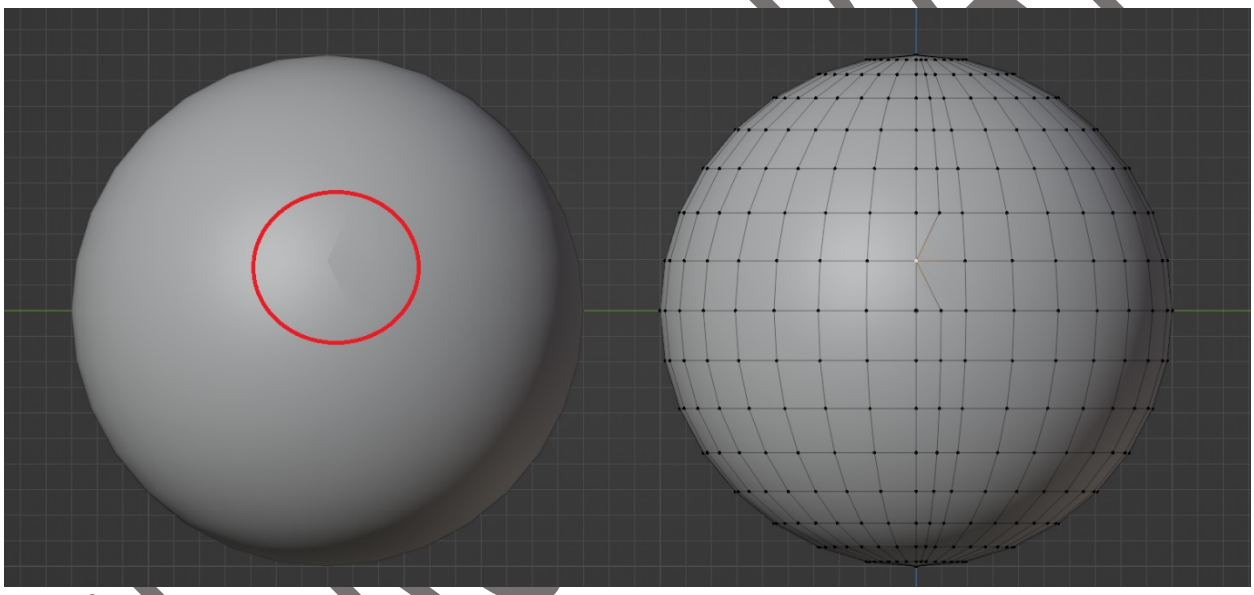


Рисунок 45 - Сглаженная сфера с артефактом из-за наличия двойной вершины.

А что, если нам не нужно сглаживание? В таком случае проблема все равно проявится, но позже. Появятся артефакты при наложении текстур независимо от того сглажена поверхность у объекта или нет. Это можно проверить по UV развертке. На рисунке 46 видна перетяжка.

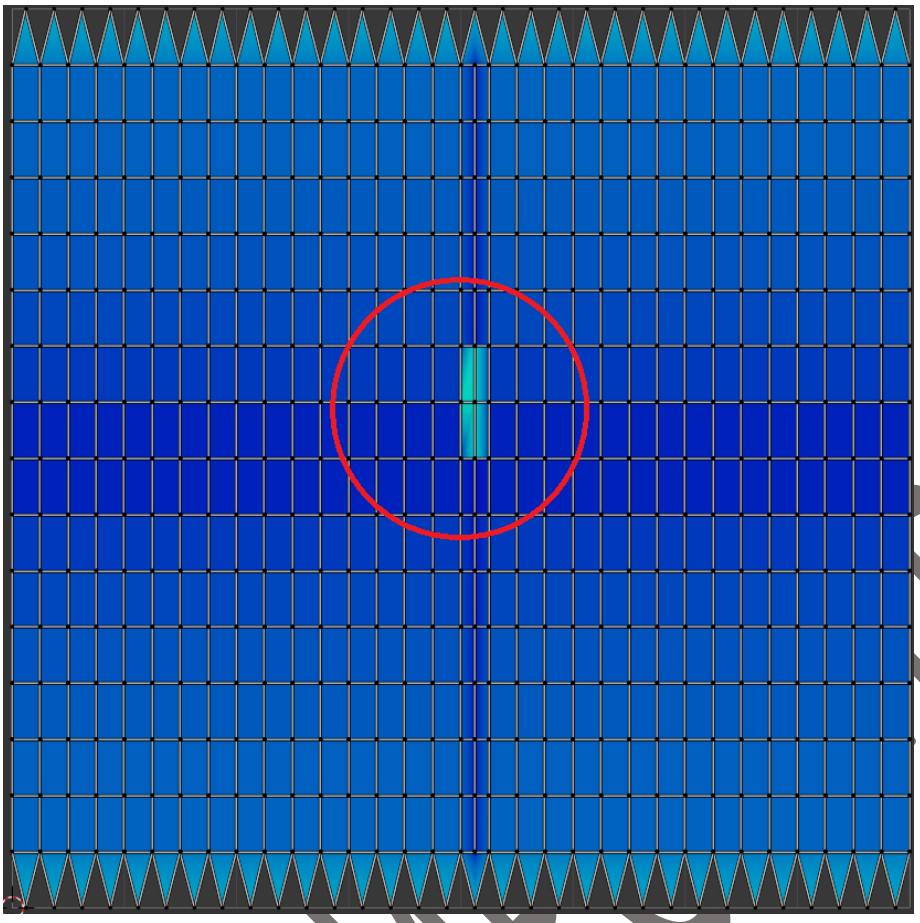


Рисунок 46 - UV-развертка сферы с перетяжкой из-за двойной вершины.

Для того, чтобы удалить двойные вершины, необходимо в режиме редактирования выбрать весь объект, нажать клавишу **M** и в списке выбрать пункт **By distance**.

3.9. Создание окружности в фигуре квадратной формы

В моделировании одними из сложных случаев являются наличие отверстий в модели.

Предположим, что у нас есть плоскость как на рисунке… Как можно сделать корректную сетку с окружностью в центре? Ведь по определению, окружность состоит из бесконечного числа точек, равноудалённых от общего центра. Как нам представить это в виде многоугольника?

На самом деле, всё проще, чем может показаться на первый взгляд. Разберём общий принцип создания окружностей. Первым шагом является определение достаточного количества ребер в окружности. Для простоты возьмем восемь ребер. Подразделяем плоскость три раза с помощью subdivide. Для этого выбираем в режиме редактирования всю плоскость клавишей **A**, кликаем левой клавишей мыши и выбираем в появившемся меню пункт **Subdivide**. Далее в настройках инструмента снизу слева во вьюпорте вписывает цифру три в параметр **Number of cuts**. Все выбранные элементы изображены на рисунках 47.

Далее необходимо в центре сделать звезду из ребер. Для этого нужно добавить ребра, как на рисунке… Это можно сделать несколькими способами. Разберем один из них. Выбираем вершину «1» и «0» в режиме редактирования и нажимаем J. Такую операцию проделываем для вершин «2» и «0», «3» и «0», наконец для «4» и «0». Номера вершин и результат можно увидеть на рисункт 48.

Последним этапом является фаска на центральной вершине. Для ее создания необходимо нажать сочетание клавиш Ctrl + B и в настройках инструмента выбрать Vertices с единичным подразделением. Настройки для примера на рисунке 48 можно посмотреть на рисунке 49.

Все настройки не стоит понимать буквально. Это лишь иллюстрация примеров.

Далее необходимо удалить центральный полигон и анализировать результат.

В данном примере сетка получилась равномерной и никаких ошибок в этом случае возникать не будет.

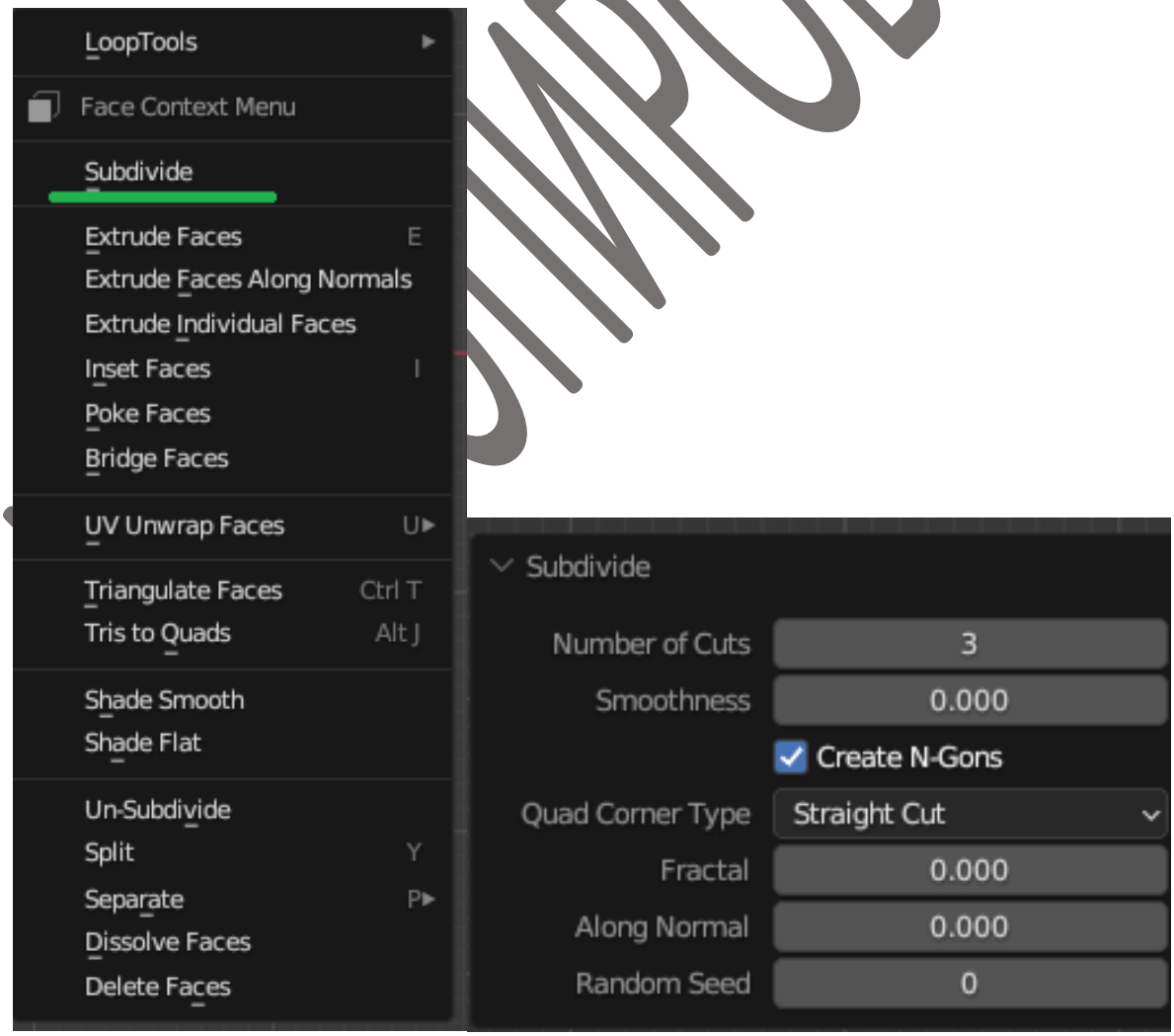


Рисунок 47 - слева выбор подразделения поверхности, справа – его настройки

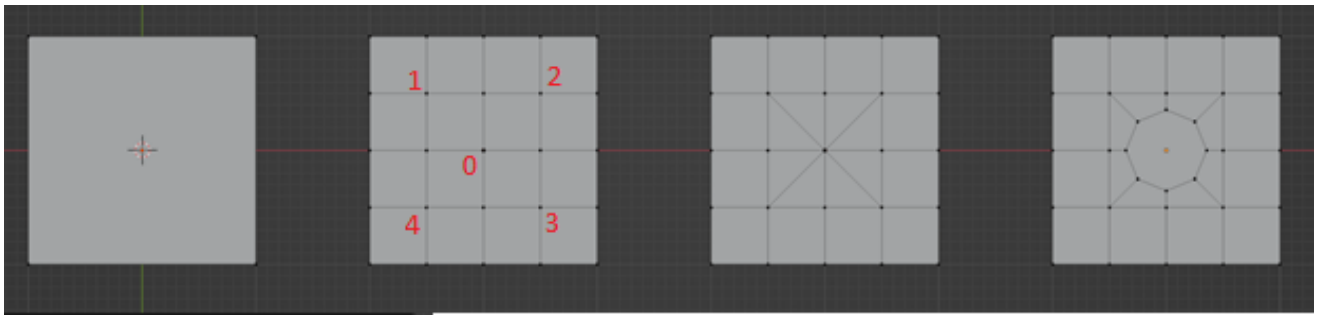


Рисунок 48 - Последовательность создания «круглого» отверстия в квадратной плоскости

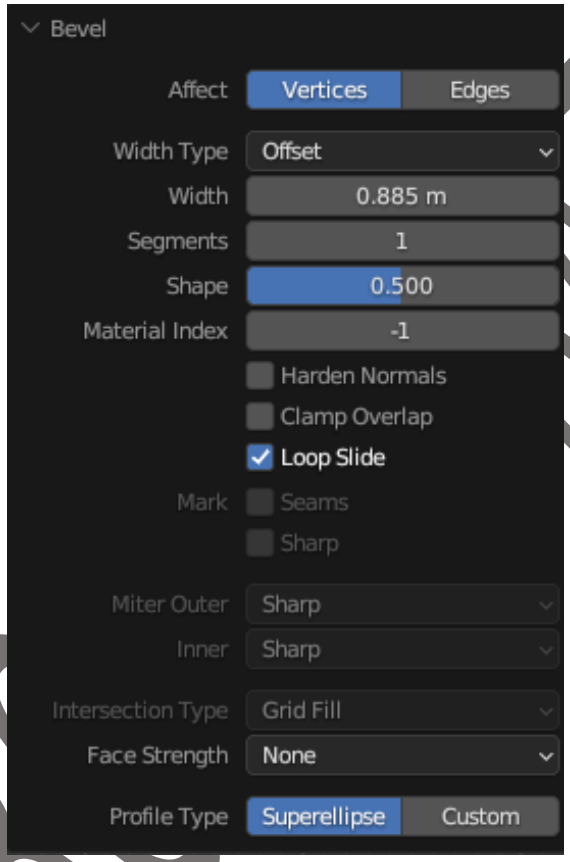


Рисунок 49 - Настройки инструмента Bevel

3.10. Скелетная анимация

Порой бывает такое, что комплексные механизмы также обладают большим числом движущихся частей. Анимировать каждую по отдельности может быть весьма трудоёмко. В таких случаях на помощь приходит скелетная анимация. Её реализация позволяет значительно сэкономить время, затраченное на разработку.

Важным инструментом при создании анимации являются **Bone** *(кости)* (*Рисунок 50*).

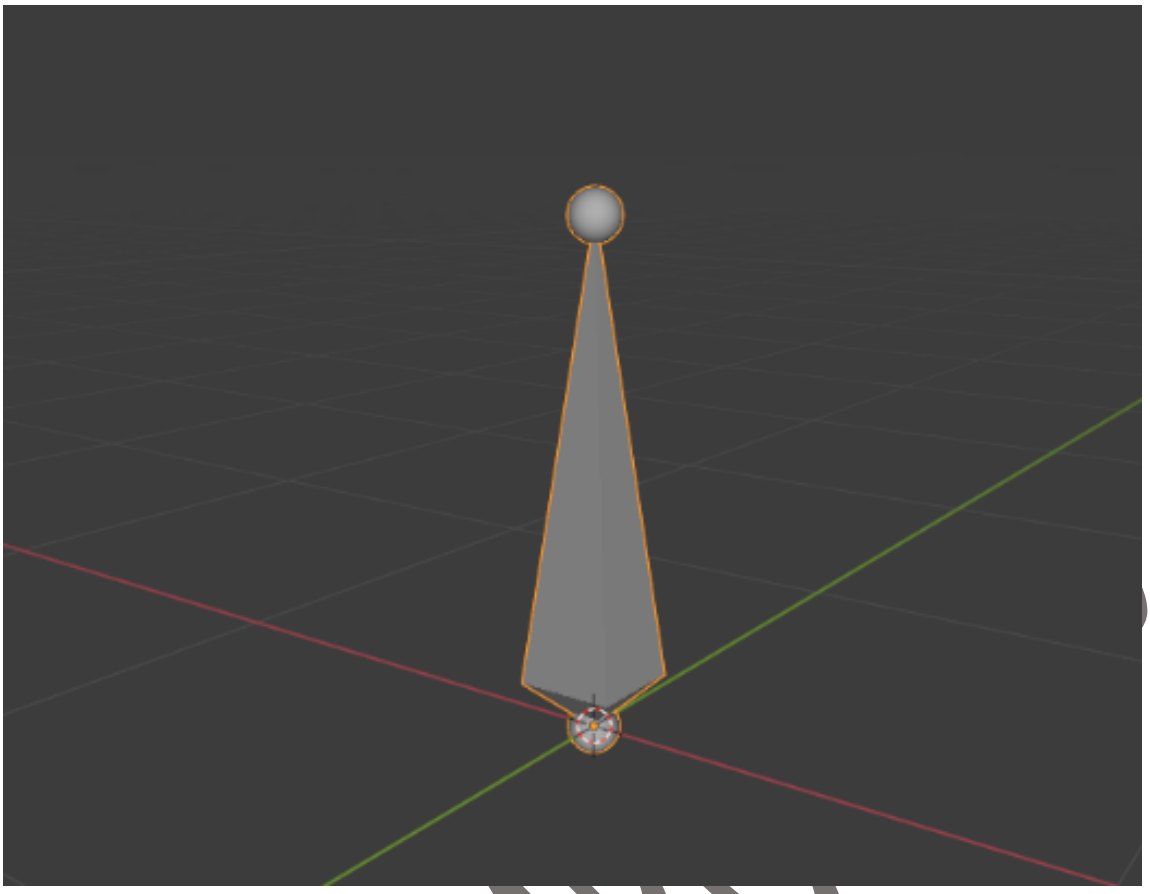


Рисунок 50 – Отображение кости

Чтобы появилась кость, нужно нажать Add -> **Armature** в 3D Viewport или нажать сочетание клавиш Shift+A и также выбрать Armature. Из них можно делать различные скелеты. Именно они будут полезны при создании анимации. Однако несколько костей для одного скелета мы будем создавать из одной кости, а не из меню **Add** или по нажатию сочетания клавиш Shift+A. Почему именно так? Потому что при создании костей из меню они будут считаться разными скелетами (даже одна кость считается скелетом).

Как сделать скелет из нескольких костей? Для этого, выбрав одну кость, нажимаем клавишу Tab для входа в режим редактирования объекта. Как и с обычными объектами, так и с Bone мы можем их перемещать, вращать и изменять масштаб, но также мы можем делать **Extrude**. Для **Extrude** необходимо выбрать верхнюю часть кости (Рисунок 51). Можно выбрать и нижнюю, но тогда не будут выполняться связи, которые очень важны.

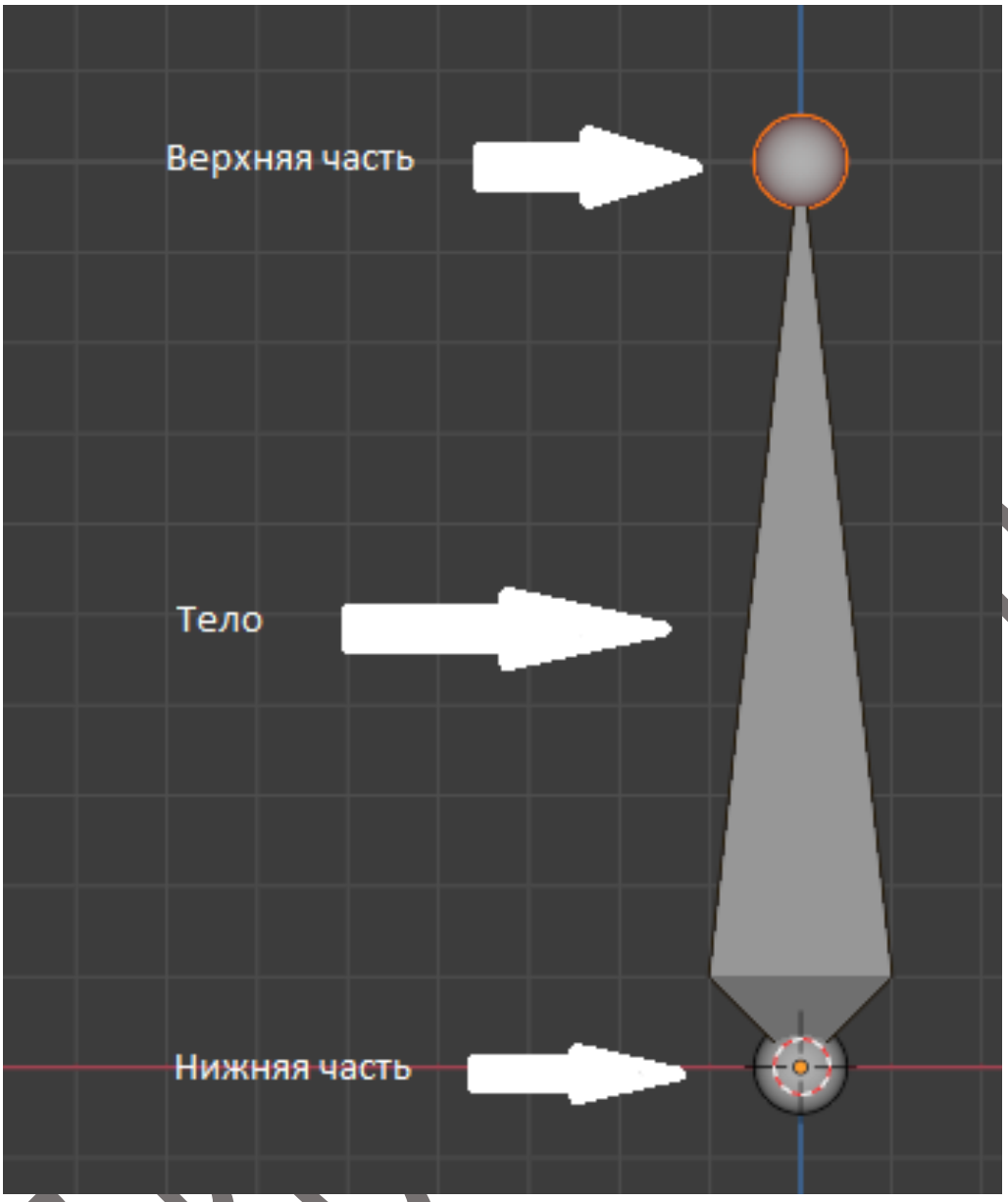


Рисунок 51 – Части Bone

О каких связях идет речь? Для ответа на этот вопрос обратимся к устройству дерева. Дерево имеет один ствол, и он самый главный в дереве. От него растут, например, две ветки. Эти две ветки не влияют друг на друга, то есть если «шатать» одну ветку от ствола, то вторая двигаться не будет. Однако если перемещать ствол, то будет перемещаться все. Предположим, что от одной ветки, растущей от ствола, растут еще две ветки. Пусть они называются «Ветки2». Аналогично предыдущим рассуждениям, «шатание» одной из «веток2» не влияет на другую и не влияют вообще ни на что. А вот влияние на ветку-родителя будет давать изменения и тд.

Далее, создав необходимый скелет под модель, можно им управлять. Однако в объектном режиме перемещаются все кости одновременно. Чтобы управлять каждой косточкой по отдельности нужно перейти в режим Pose Mode. Он находиться в той же вкладке, что и объектный режим, и режим

редактирования. Перейдя в него, появляется возможность выбрать любую кость или кости в скелете. Сделав это, можно, как и с обычными объектами вращать и перемещать кости, а также изменять их масштаб. Это дает нам удобную возможность анимировать.

Далее можно нужно установить прямые связи между скелетом и нашим объектом(и). Это значит, что одной части модели или одному объекту будет принадлежать одна управляющая кость.

Рассмотрим пример с двумя кубами, которые расположены один под другим. Сделаем под него скелет, состоящий из двух костей, чтобы верхним кубом управляла верхняя кость, как показано на рисунке 52.

1. Выбираем куб;
2. Через Shift выбираем скелет;
3. Переходим в режим Pose Mode в окне 3D Viewport;
4. Выбираем кость, которая будет управлять кубом;
5. Нажимаем сочетание клавиш Ctrl+P -> **Bone.**

Проделав те же операции со вторым кубом и второй костью, были установлены прямые связи. Теперь в режиме **Pose Mode**, перемещая кости, перемещаются и кубы.

Конечно же можно делать более сложные конструкции, которые затем можно будет анимировать.

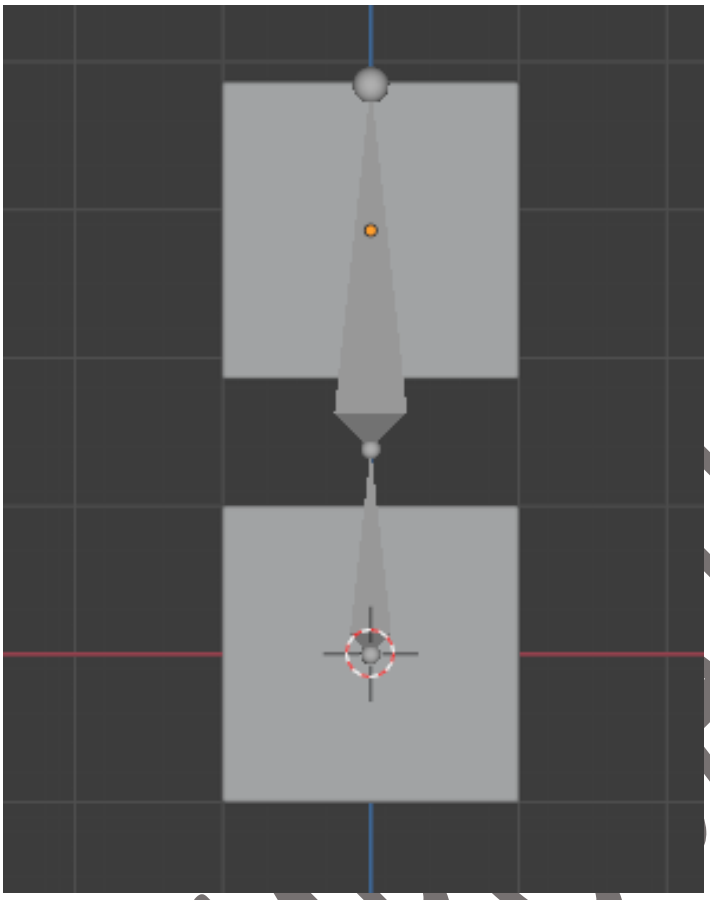


Рисунок 52 – Скелет для пары кубов

Замечание: на рисунке кости видны перед кубами, но это не так. Они расположены внутри. Это сделано для удобства работы. Для этого в правом окне Properties на вкладке данных объекта необходимо найти меню **Viewport Display** и поставить галочку **In Front**. Перед всем этим нужно выбрать сами кости.

Анимация скелета происходит по таким же правилам и принципам, как и для обычных объектов. Может появиться вопрос: почему скелетная анимация удобнее? На самом деле все зависит от задачи и от потребностей.

А как отменить привязку объекта к кости? Для этого выбираем объект и нажимаем сочетание клавиш Alt+P -> **Clear Parent.**

Привязывать объекты можно не только к скелету к конкретной кости, но и к другим объектам. Например, на нашем примере с двумя кубами можно верхний куб привязать к нижнему. Тогда нижний куб будет «стволом», а верхний «веточкой». Для такой связи выбираем вначале тот объект, который будет зависимым, потом через Shift выбираем объект, который будет главным.

Нажимаем:

Ctrl + P -> Object.

Меню Ctrl + P для привязки объектов к костям и к другим объектам отличаются. Все дело в том, что у Bone больше возможностей.

Все это время рассматривались привязки для разный, не цельных объектов. Но анимировать можно и обычный меш, например, змею. Змея не состоит из отдельных кусочков, благодаря которым она движется. Она извивается. И для того, чтобы анимировать змею нам как раз понадобится скелет, так как анимировать движения без него не получится. В этом и раскрываются дополнительные возможности привязок скелета.

Рассмотрим пример с длинным цилиндром и подразделенным на множество частей, показанный на рисунке 53.

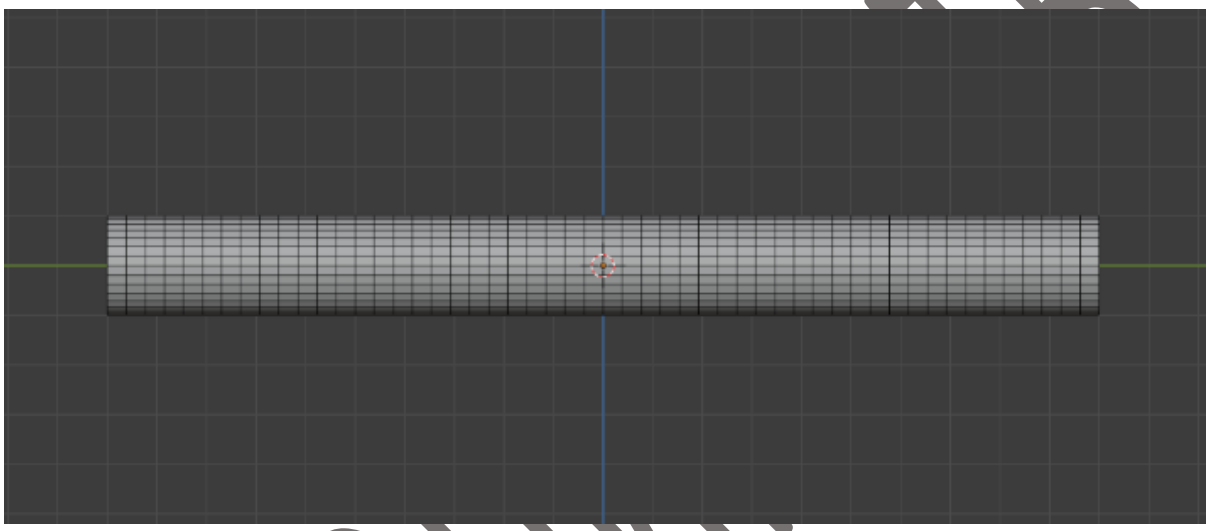


Рисунок 53 – Цилиндр, подразделенный на множество частей

Вдоль него сделан скелет, состоящий из 6-ти костей. Выбрав меш, затем через Shift кости и нажав Ctrl + P -> **Armature Deform With Automatic Weight** были расставлены автоматически связи между костями и объектом. Чтобы их посмотреть, нужно зайти в режим **Weight Paint** для объекта. В окне **Properties** -> Настройки данных объекта можно выбирать какую-либо кость, сделанного скелета и видеть его влияние (*Рисунок 54*).

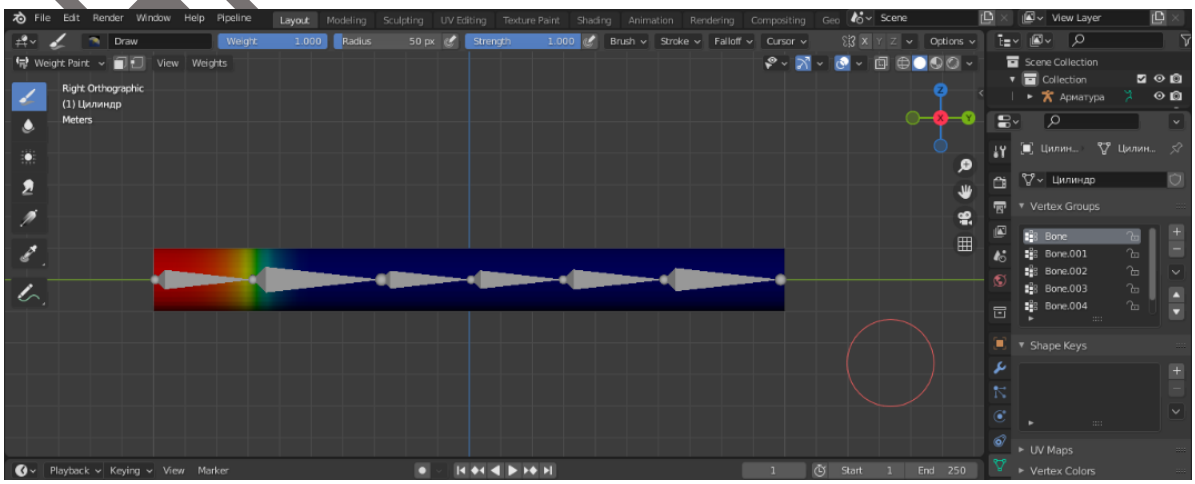


Рисунок 54 – Влияние кости на участок меша

Например, выбрав кость Bone, виден градиент от красного к синему. Это влияние кости на участки меша. Красный – сильное влияние, синий – его отсутствие. Теперь благодаря такому скелету можно анимировать цилиндр, он может изгибаться и принимать разные формы, как это показано на рисунке 55.

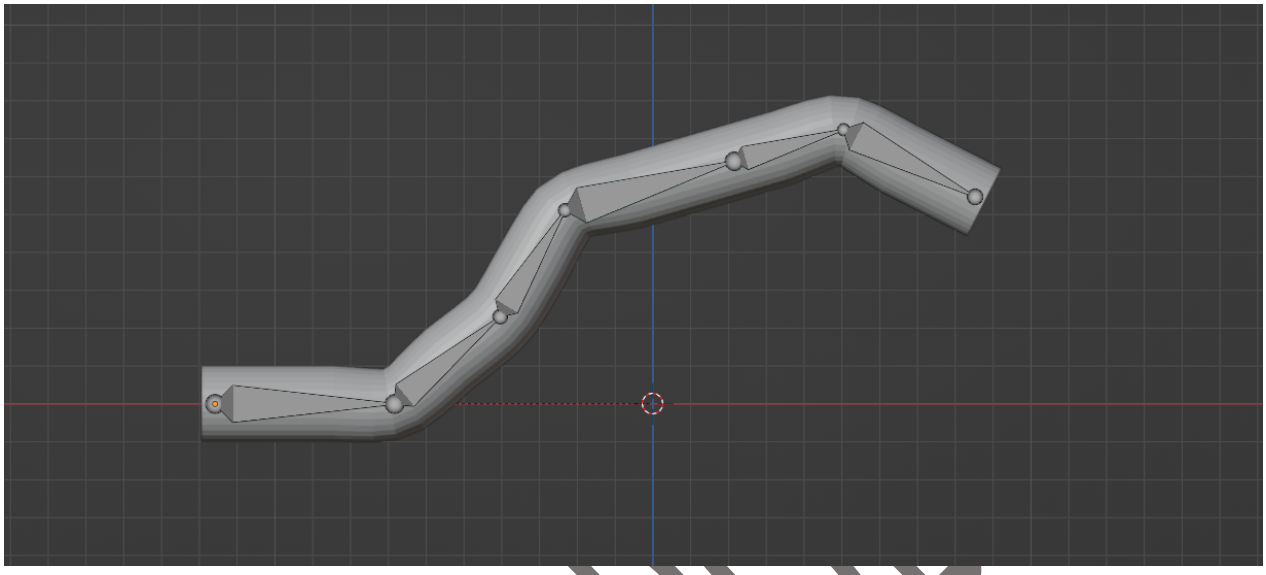


Рисунок 55 – Пример изменения формы меша скелетом

Замечание: Расставление влияния костей на участки одного объекта называются весами.

3.11. Функционал режима Weight Paint

Слева расположена панель инструментов: **Draw, Blur, Average, Smear, Gradient, Sample Weight**.

Расшифруем:

- **Draw** – позволяет рисовать веса по вершинам. Для этого необходимо нажать левую клавишу мыши и вести курсор по необходимым местам;
- **Blur** – размывает влияние. Совершаем те же действия, что и для инструмента Draw;
- **Average** – усредняет значение влияния. Совершаем те же действия, что и для инструмента Draw;
- **Smear** – размазывает вес по направлению движения зажатой мыши;
- **Gradient** – градиент;
- **Sample Weight** – берет значение влияния в месте, где был совершен клик мыши.

Веса имеют значения от 0 до 1. Отсутствие влияния или синий имеют значение 0, а полное влияние или красный имеют значение 1.

Некоторые инструменты имеют радиус действия и силу влияния. Этими параметрами можно управлять в верхней шторке 3D viewport или по нажатию правой кнопки мыши. Также на верхней шторке справа есть возможность включить симметрию по нужной оси.

На рисунке 56 рассмотрен пример с одним объектом, состоящем из трех объединённых кубов и его скелетом.

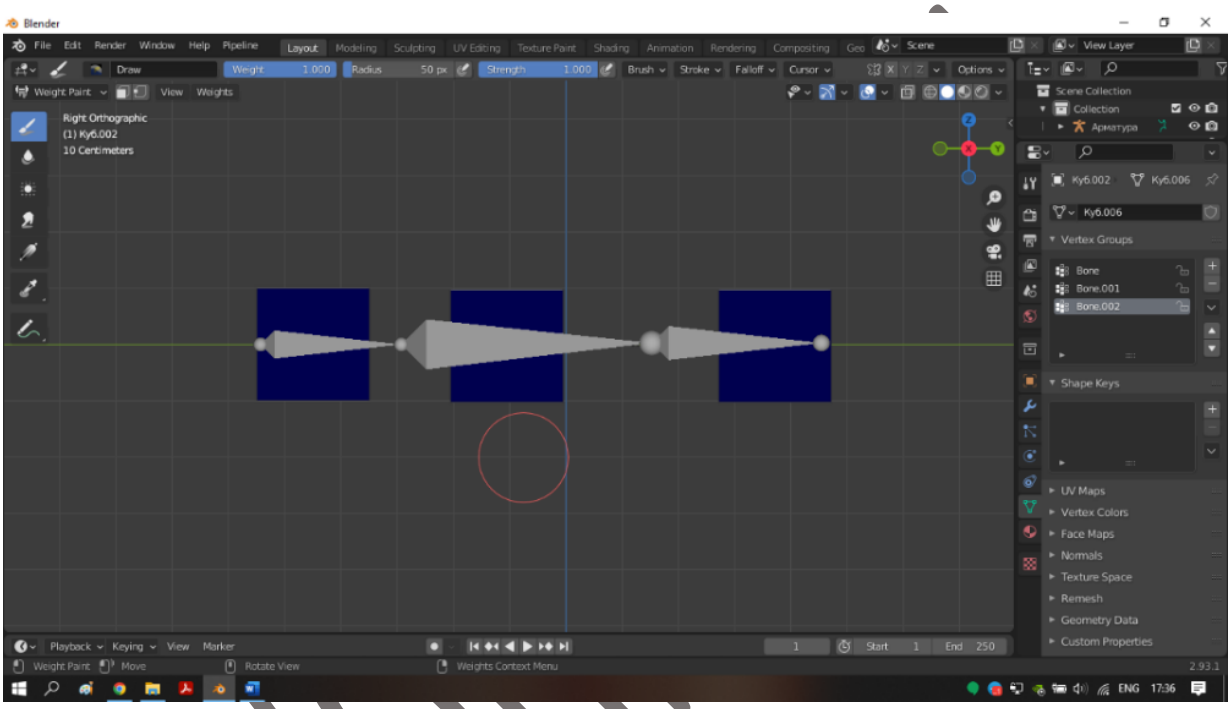


Рисунок 56– Скелет для трех кубов

Выбрав меш, затем через Shift его скелет нажимаем Ctrl + P -> **Armature Deform With Envelope Weight**. Это действие привязало кости к мешу, но их влияние нужно настроить. В окне **Properties** на вкладке данных объекта можно увидеть все три созданных кости. Выбираем одну из них. Начнем снизу-вверх. Нижняя кость, (в данном случае она является левой) не должна влиять на правый куб. Для определения влияния будем использовать инструмент **Draw**. Необходимо, чтобы значение **Weight** было равно 1. Проводим инструментом по вершинам куба. Он должен стать полностью красным. Далее выбираем среднюю кость в списке. Проводим те же действия, но уже с центральным кубом. И все тоже самое для первой кости. У нас получилось расставить веса для костей, чтобы они могли управлять объектом. Значение **Weight** бралось равным единице для того, чтобы влияние было полным. Теперь в **Pose Mode** для костей можно приступать к анимации.

3.12. Окно Graph Editor

На рисунке 57 изображено окно **Graph Editor**. Это окно также помогает видеть действия, которые мы делаем для анимации и редактировать их. Изначально окно пустое. Самое главное в Graph Editor – это его пустое поле, в котором будут появляться графики для каждого зафиксированного действия для анимированного объекта. На этом поле по горизонтальной оси расположены кадры, а по вертикальной – значение какого-либо параметра. Слева расположено меню со всеми зафиксированными действиями.

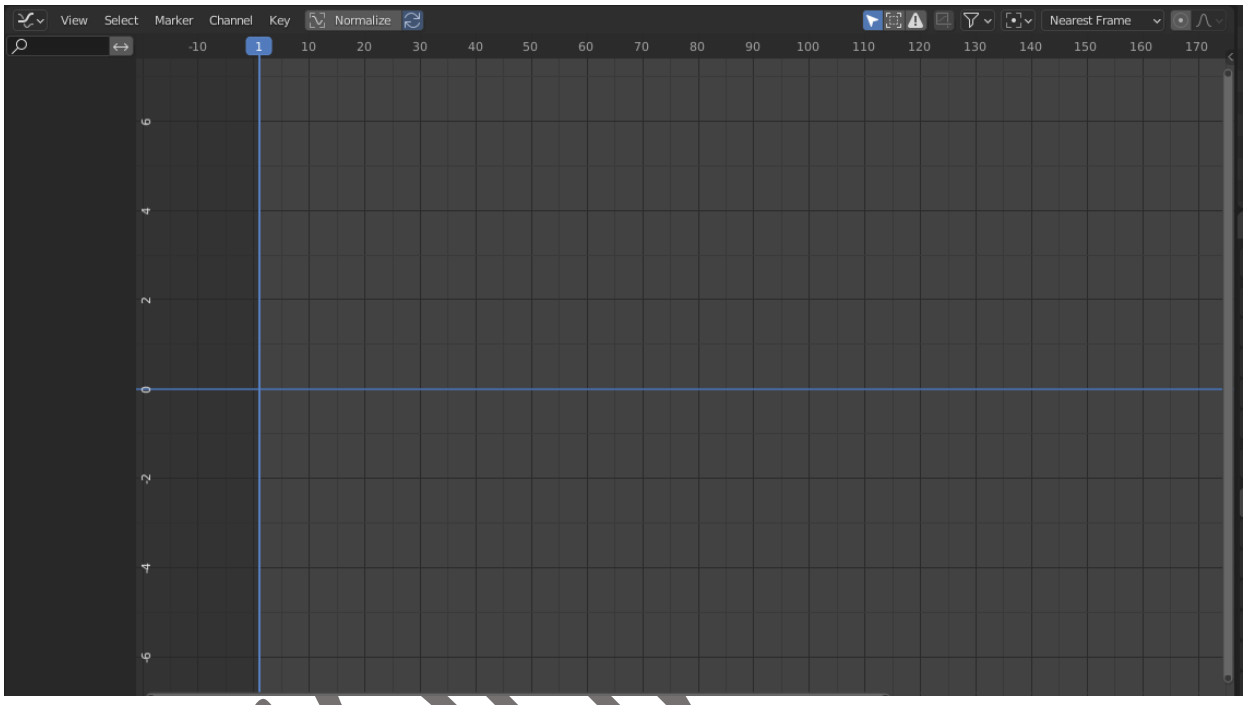


Рисунок 57 – Окно Graph Editor

Для того, чтобы в окне появилась какая-либо информация, необходимо выбрать объект с заготовленной анимацией или сделать ее. Для примера, будем запоминать Location объекта. На первом кадре зафиксируем его Location, а на 30 кадре переместим объект по оси X на небольшое расстояние, и также запомним это положение.

Что появилось после первого шага? В **Graph Editor** появилась прямая линия, а в левом меню появились данные. Раскрыв их, можно увидеть, что появилась информация о трех осях **X, Y, Z**. Прямая линия для каждой оси имеет свой цвет. Также прямая означает, что больше с этими данными изменений не происходило. На втором шаге изменяется координата по оси X. После запоминания этого изменения в окне **Graph Editor** по оси X изменится график, а по осям Y и Z изменений не происходило, поэтому они остаются без изменения, то есть прямыми. Также на графиках появились управляющие элементы, точки, на кадрах, где происходила фиксация положения объекта. Эти точки можно перемещать на поле, это будет влиять на анимацию. Перемещение происходит при нажатии клавиши G. Можно одновременно

выбирать несколько точек и перемещать их по графику. Также, выбрав несколько точек, их можно масштабировать, нажав клавишу S. Те же действия можно делать и для основных точек, например, чтобы сместить перемещение на другой кадр.

Что означает график по оси X? Он отображает как будет происходить переход с 1 по 30 кадр объекта с координаты 0 до координаты 2 (Рисунок 58). Переход будет происходить с параболической плавностью с обоих сторон. Да, график отображает и отвечает за плавность изменений параметров.

Параболическая плавность установлена по умолчанию. При необходимости её тип можно менять. Для этого в настройках Blender в меню **Animation** во вкладке **Default Interpolation** нужно поменять **Bezier** на необходимый переход. Но значение по умолчанию иногда нужно оставить то же, однако плавность нужно изменить. Для этого, выбрав основную точку на графике (не управляющую), нужно нажать правую кнопку мыши, выбрать **Interpolation Mode** и нужный переход. Если нужных переходов нет, то можно с помощью вышеупомянутых точек для управления изменять график, перемещая эти точки. Кстати говоря, у них тоже есть свои свойства, расположенные в настройках Blender в меню **Animation** во вкладке **Default Handles** или при выборе основной точки при нажатии правой кнопки мыши и выборе пункта **Handle Type**.

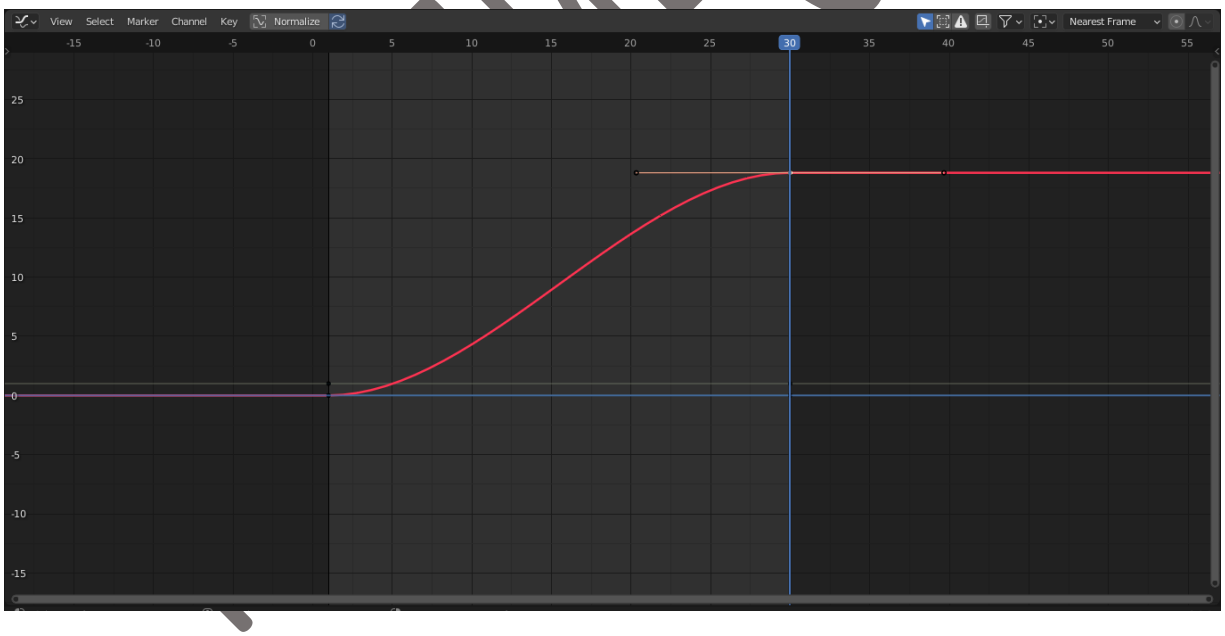


Рисунок 58 - График перехода объекта из точки А в точку Б

При нажатии в окне **Graph Editor** клавиши N, появляется дополнительное окно справа. Оно позволяет также управлять графиками. На вкладке **F-Curve** осуществляется настройка поведения графика и управляющих точек. На вкладке **View** осуществляется управление курсором. Наконец, на вкладке **Modifiers** можно добавить свойство графику. Например,

свойство **Curve** позволяет проигрывать кусок анимации бесконечное или конечное количество раз.

3.13. Constraint

Constraint (или ограничители) – это инструменты, которые позволяют упрощать или создавать анимацию. Ограничители есть как для объектов, так и для костей. Они имеют минимальные отличия. Рассмотрим Constraint для объектов. Они находятся в окне **Properties** на вкладке **Object Constraint Properties** (настройки ограничителей объекта). Обратите внимание на рисунок 59.

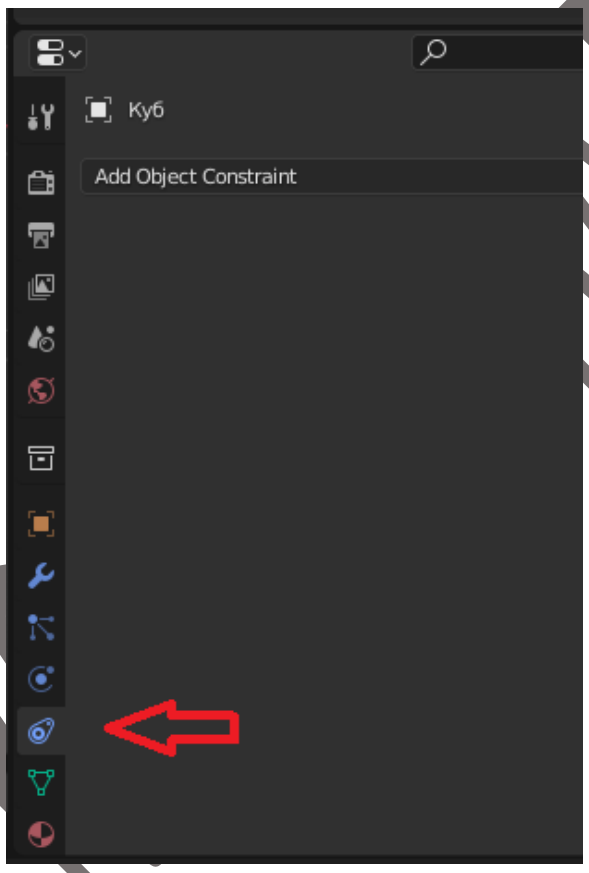


Рисунок 59 – Меню Object Constraint Properties

Рассмотрим работу ограничителей на примере двух кубов, расположенных на некотором расстоянии друг от друга. Выбрав один из объектов, нажимаем **Add Object Constraint** -> **Copy Location**. Появилась новое меню, показанное на рисунке 60. Изначально пункт **Target** пустой. В нем мы должны выбрать объект, с которого будет происходить копирование положения. В нашем примере – это второй куб. Далее обратим внимание на пункт **Axis**. Он определяет, какие координаты куб будет брать с управляющего объекта. Если выбраны все оси, то будут скопированы все координаты, если только одна ось, то только одна координата.

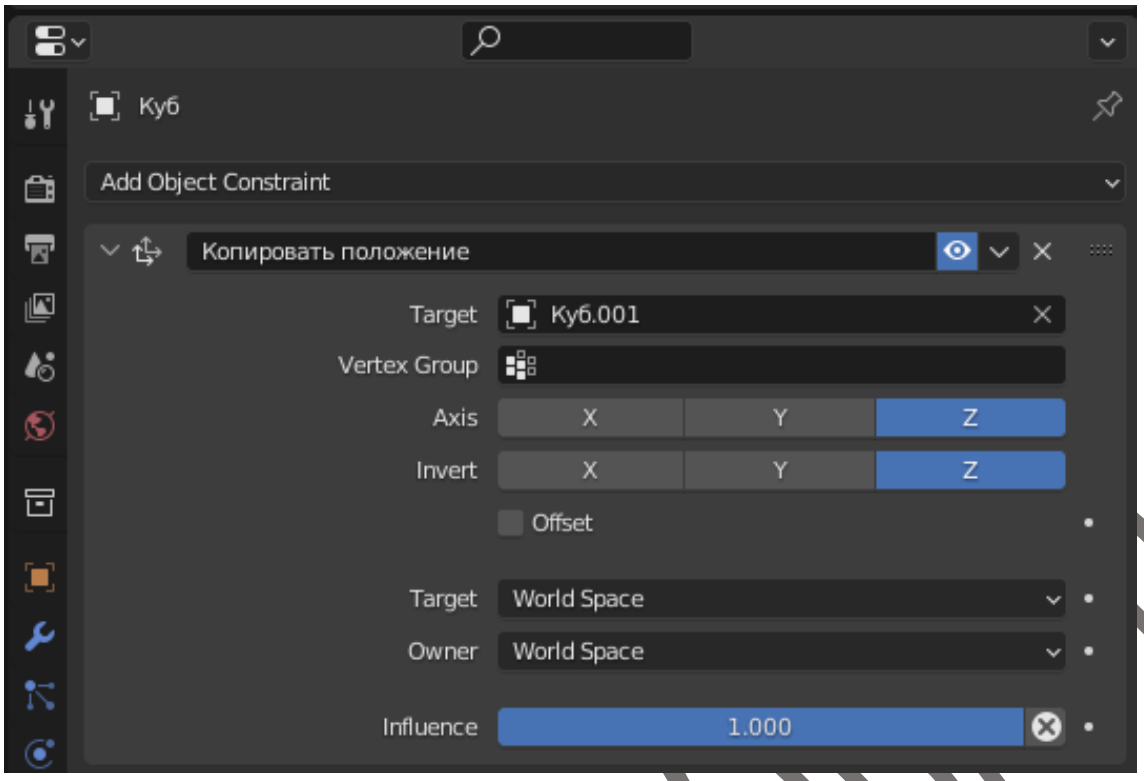


Рисунок 60 – Constraint Copy Location

Замечание: при этом во вкладке **Invert** должны быть выключены все оси.

Зачем нужна вкладка **Invert**? Если мы дублируем, например, только положение координаты Z, то кубы будут всегда на одной высоте. При этом. Если включена ось Z на пункте Invert, то координата будет инвертирована, то есть отражена относительно нуля (координата 10 скопируется как -10, -42 перейдет в 42 и тд). Если поставить инвертирование координаты, которая не выбрана в Axis, то ничего не сработает.

Кубы изначально могут быть расположены где угодно друг относительно друга. Если нужно сохранить их взаимное расположение и копировать изменение

Нажатая галочка Offset позволяет устанавливать дистанцию между объектами по выбранным осям. Например, можно задать дистанцию 2 между объектами по всем трем осям, и она будет сохранятся.

Influence позволяет определить степень влияния объекта, с которого копируют координаты. 1 – это полное влияние, 0 – нет влияния. Пример: Нужно при перемещении одного из кубов по оси Z, передавать эту координату другому кубу, но в 2 раза меньше. Для этого необходимо выставить Influence на 0,5 и в зависимом объекте поставить в Axis ось Z.

Также, если в нашем объекте есть **Vertex Group**, то ее также можно выбрать как родителя, с которого нужно копировать положение.

Target и **Owner** позволяют задавать координаты, которые будут служить точкой отсчета для копирования положения объекта.

Далее рассмотрим **Copy Rotation**. В этом окне все тоже самое, что и в **Copy Location**, кроме пункта **Order**, который позволяет определить порядок копирования осей вращения, и пункта **Mix**, который заменяет пункт **Offset**. Окно **Copy Scal**e имеет дополнительный пункт **Power** – степень, в которую нужно возвести копируемый масштаб. Также есть галочка **Make Uniform**, которая делает изменение масштаба равномерным.

Copy Transform делает все в комплексе. Он копирует и положение, и вращение, и масштаб в соответствии с выбранными настройками.

Limit Location. Объект, на который наложено это ограничение, может находиться не дальше заданного значения до выбранной цели или наоборот, не ближе заданного значения до цели (**Target**). Это значение задается в параметре **Distance**. Режим дистанции задается в пункте **Clamp Region**. Также остается пункт со степенью влияния ограничителя **Influence**.

Limit Rotation. Позволяет устанавливать ограничение на вращение каждой оси для объекта. Имеет степень влияния **Influence** и задание порядка осей **Order**. Все остальные настройки также уже встречались в других ограничителях.

Limit Scale – ограничение на объем. Все настройки были описаны выше.

Maintain Volume. Позволяет компенсировать масштабирование по выбранной оси при масштабировании двух других.

Damped Track. Объект с данным ограничителем может «смотреть» на выбранную цель (пункт **Target**). Направление оси выбирается в пункте **Track Axis**.

Locked Track. Позволяет также «смотреть» в сторону выбранной цели, но с ограничением на какую-либо ось.

Stretch To следит за выбранной целью и при этом при удалении цели растягивается.

Track To. Ограничитель очень похож на **Damped Track**, но при этом имеет больше настроек.

У одного объекта может быть несколько **Constraint**. Однако нужно учитывать их совместимость.

Для костей можно добавлять как Constraint для объектов в объектном режиме, так и как для костей в режиме **Pose Mode**. В первом случае кости будут взаимодействовать с объектами. Во втором случае взаимодействие возможно как среди костей, так и среди объектов (**Target**).

3.14 Анимация и Curve

Рассмотрим кривые, как инструмент упрощения анимации. В качестве примера возьмем манипулятор (Рисунок 61). Манипулятор должен иметь одну ветку зависимых элементов.

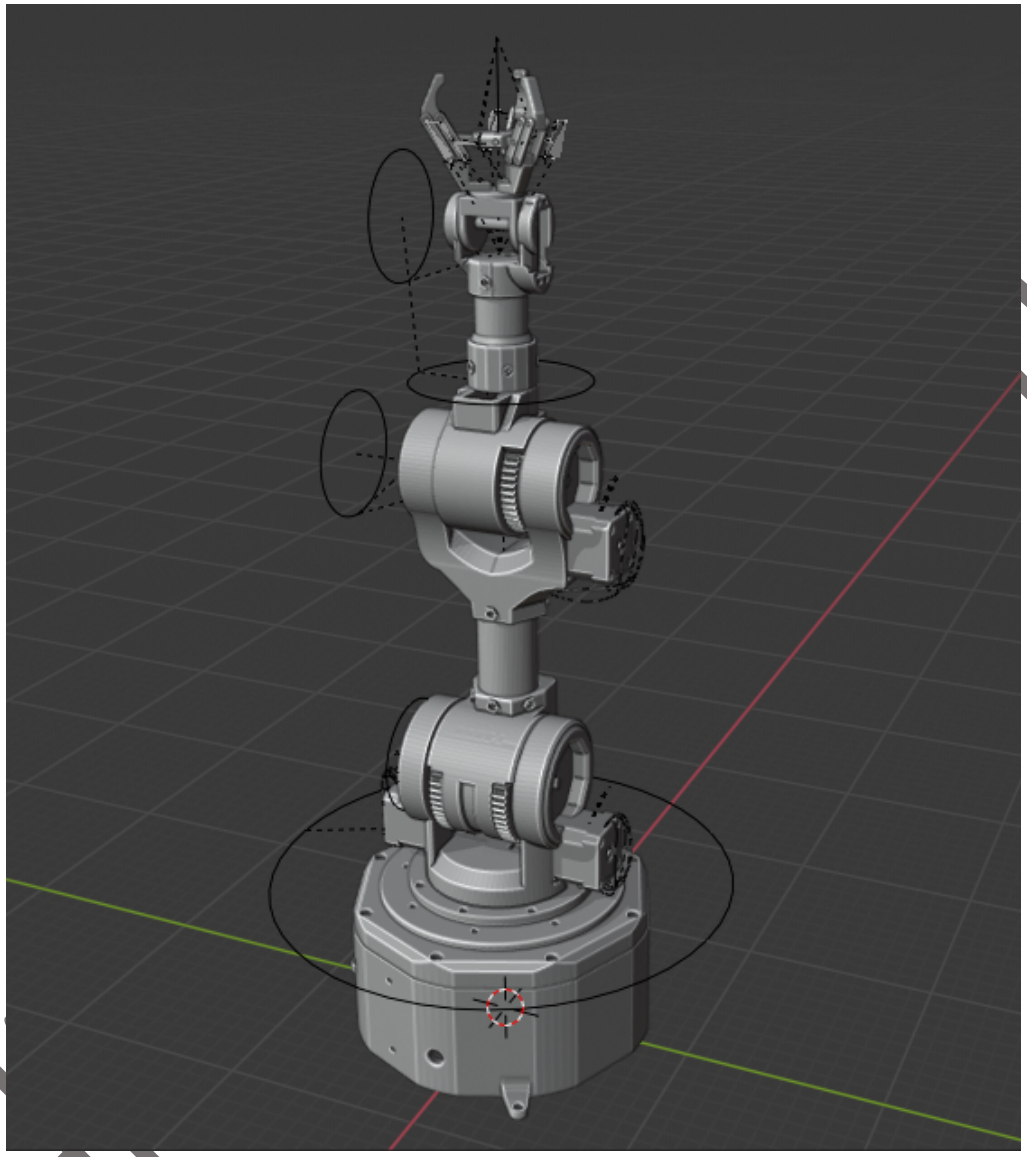


Рисунок 61 – Манипулятор

На рисунке видно, что механизм имеет рядом с собой кривые, в частности окружности. Можно заметить, что они расположены в тех местах, где «Роборука» должна осуществлять сгибы и повороты. Дело в том, что кривые привязаны к этим подвижным частям и являются родителями. Напомним, для того чтобы один объект привязать к другому, необходимо нажать сочетания клавиш Ctrl + P и выбрать тип привязки. Таким образом кривые являются скелетом для манипулятора, с помощью которых им становиться проще управлять.

Также кривые можно использовать как пути для движения объектов. Для этого необходимо добавить анимируемому объекту **Constraint** -> **Follow**

Patch (*Рисунок 62)*. В качестве **Target** выбираем кривую, вдоль которой будет двигаться объект. Далее нажимаем **Animate Path**. При запуске анимации, объект начнет двигаться вдоль кривой.

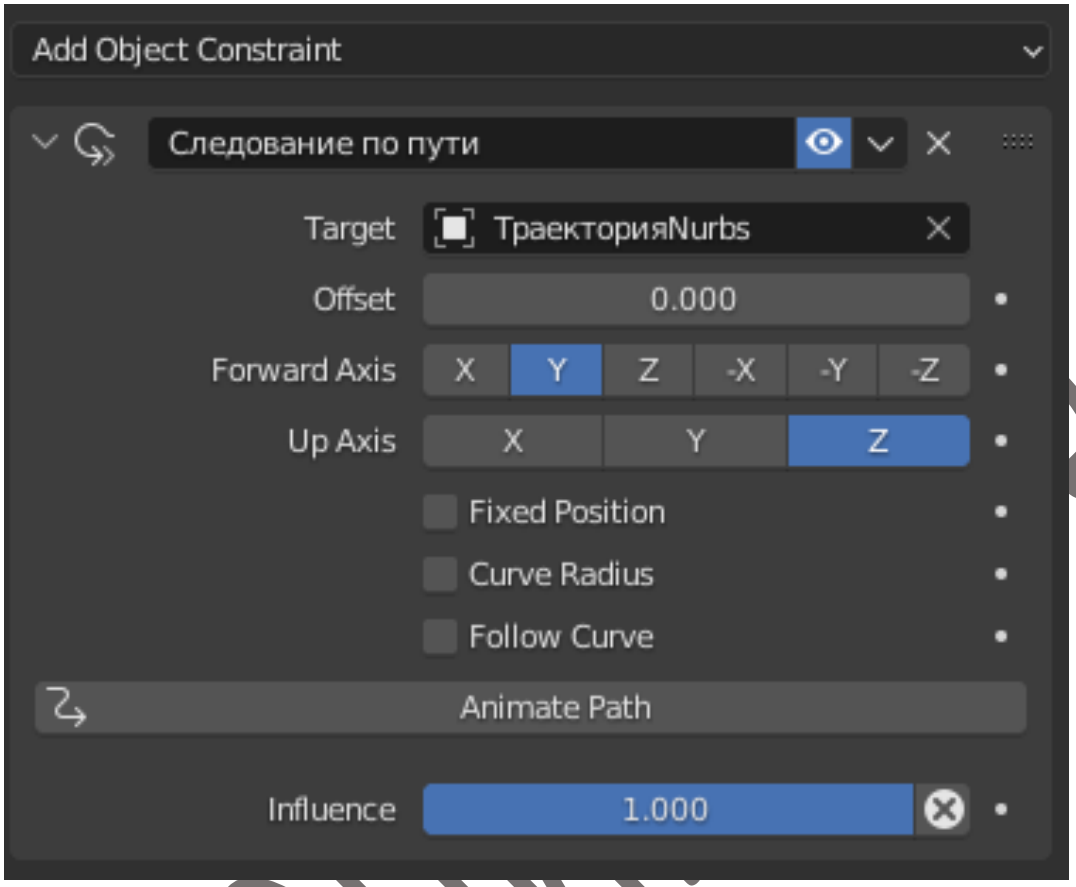


Рисунок 62 – Constraint Follow Patch

Пусть в качестве примера нашим объектом будет обезьянка. Чтобы она смотрела по направлению своего движения, нужно в меню **Follow Patch** выбрать пункт **Follow Curve**. **Fixed Position** останавливает движение обезьянки. Curve Radius масштабирует объект по радиусу кривой. Forward **Axis и Up Axis** позволяют установить направление объекта. **Offset** – смешение объекта. С **influence** мы уже встречались, это степь влияния.

Замечание: если **Pivot** анимируемого объекта и кривой не совпадают, то объект появиться не точно на кривой, которая была выбрана в качестве **Target**.

3.15. Создание UV- развёртки

После того, как все модели найдены или созданы, нужно переходить к следующему этапу. Им является создание UV-развертки.

UV-развертка— это создание 2D-поверхности из готовой 3D-модели для того, чтобы раскрасить эту поверхность и нанести на нее текстуру. Примером может служить создание объемной фигуры из бумаги, что показано на рисунке 63.

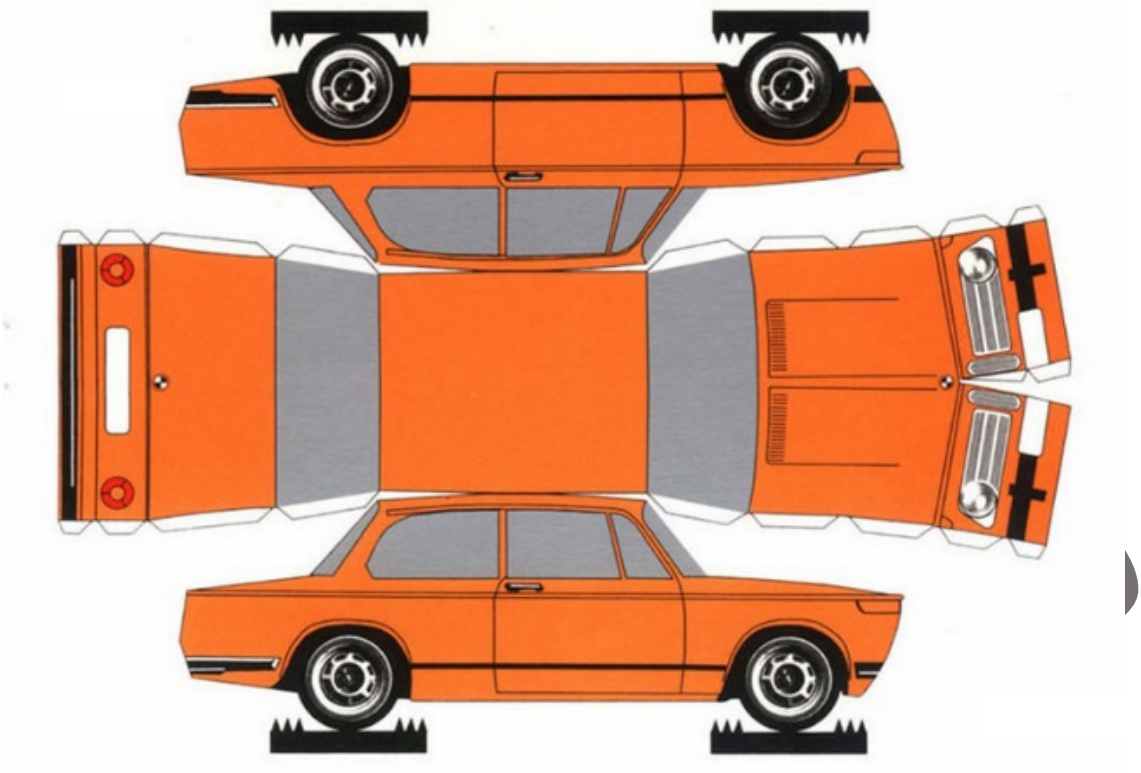


Рисунок 63 – Развертка автомобиля на плоскости

UV-развертка отображает 2D картинку на 3D объект. Каждой координате UV на 2D плоскости однозначно соответствует 3D координата объекта и наоборот, благодаря этому и происходит верное отображение текстуры на объекте. Однако это будет работать в том случае, если UV-развертка сделана правильно и не содержит наложений UV-развертки, это будет рассмотрено ниже.

В качестве примера возьмем модель нагревательного элемента принтера Anycubic, изображенную на рисунке 64.

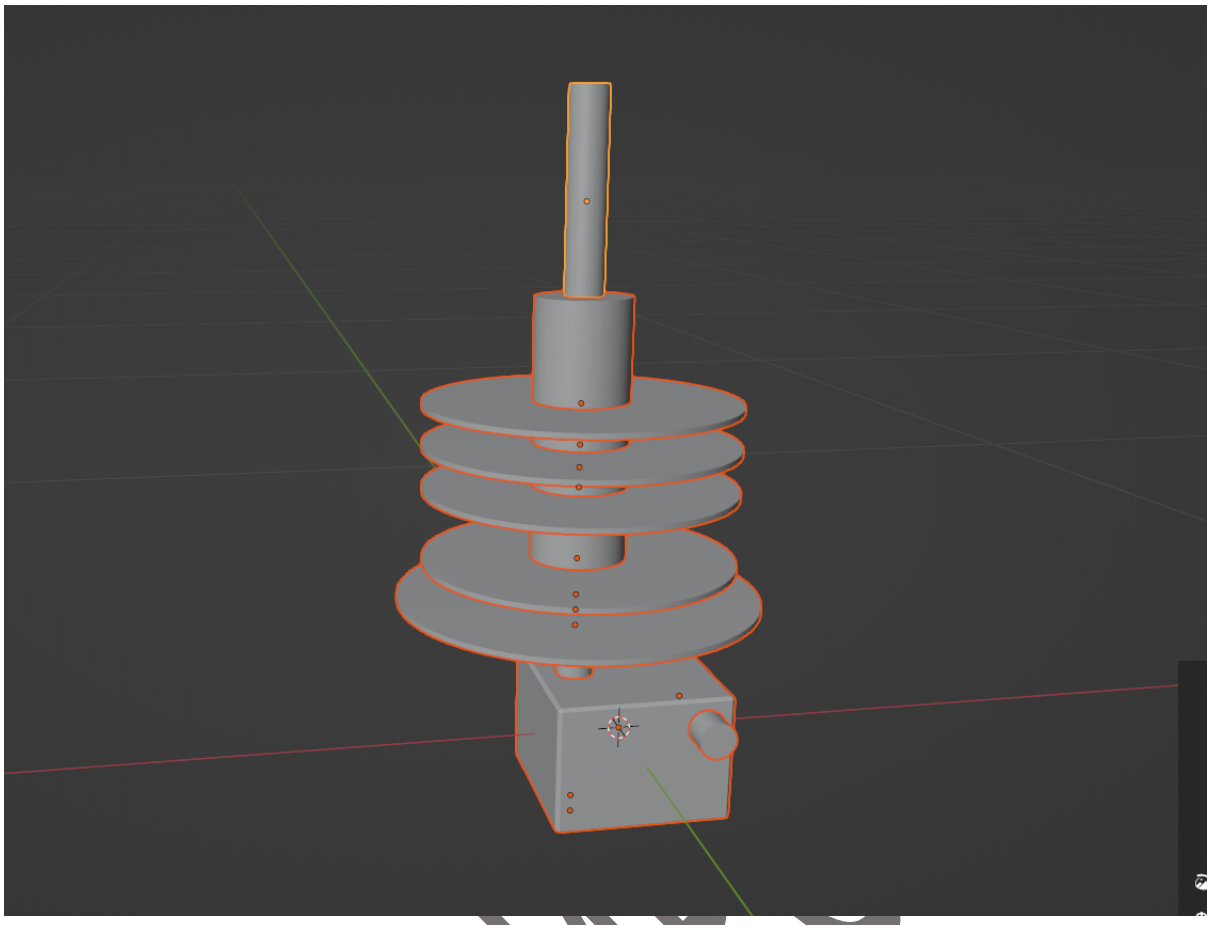


Рисунок 64 – Нагревательный элемент 3D-принтера

В Blender существует два варианта подхода к созданию UV-развертки. Первый – вручную, второй – автоматически. Создавать развертку самим как правило сложнее и дольше. Тем не менее вы сможете полностью контролировать процесс, не допуская перетяжек и прочих ошибок построения развёрток, которые могут возникнуть при автоматическом построении. Выбор того или иного подхода зависит от задачи.

Прежде чем описывать процесс создания UV, кратко опишем используемые инструменты.

Для создания автоматической развертки необходимо перейти в **UV Editing** в **Edit Mode** (режим редактирования), затем перейти во вкладку UV и выбрать **smart UV project**. После подбора параметров и нажатия “Ok”, UV создастся и появится в окне **Editor Type**. Автоматическая развертка может быть полезна для запекания карт нормалей, так как создается очень просто. Однако рекомендуется создавать UV вручную, так как в таком случае будет больше понимания процесса и ожидаемых результатов.

Интересный факт. Речь идет о запекании сглаженных фасок **High poly** моделей на острые углы **low poly** моделей. Например, можно запечь карту нормалей для обычного куба так, чтобы его ребра казались сглаженными. Пример приведен на рисунке 65.

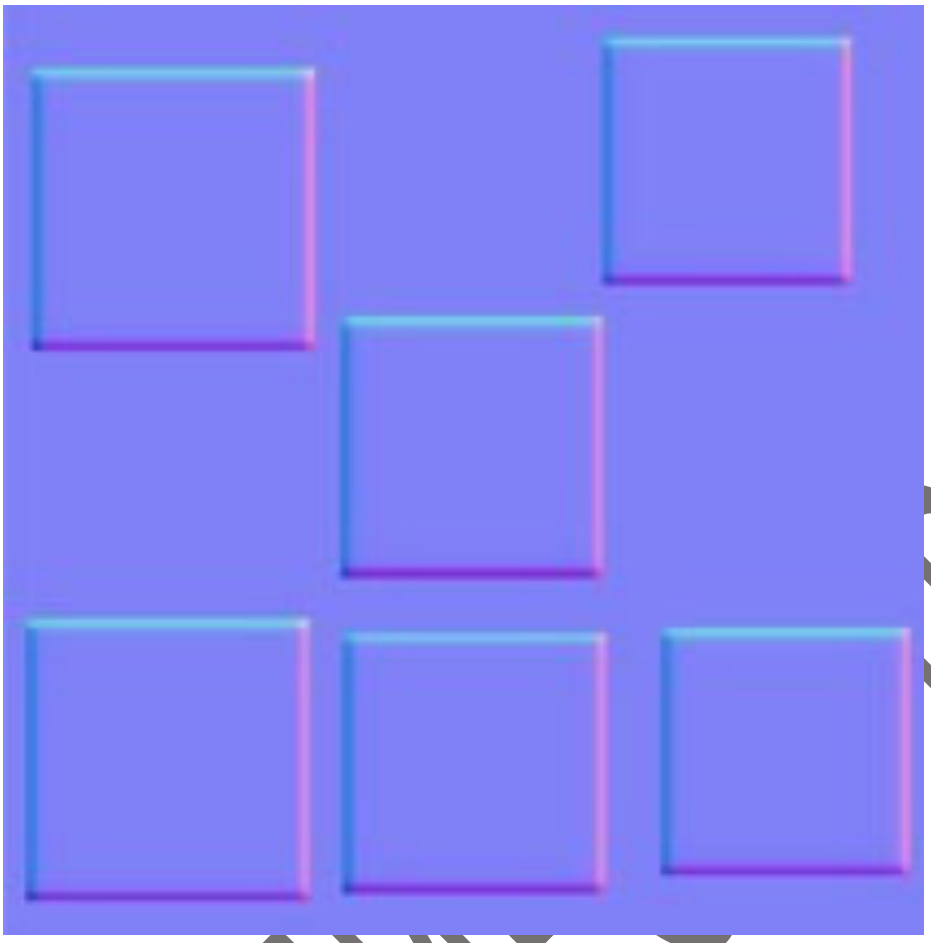


Рисунок 65 – Карта нормалей куба для сглаженных углов

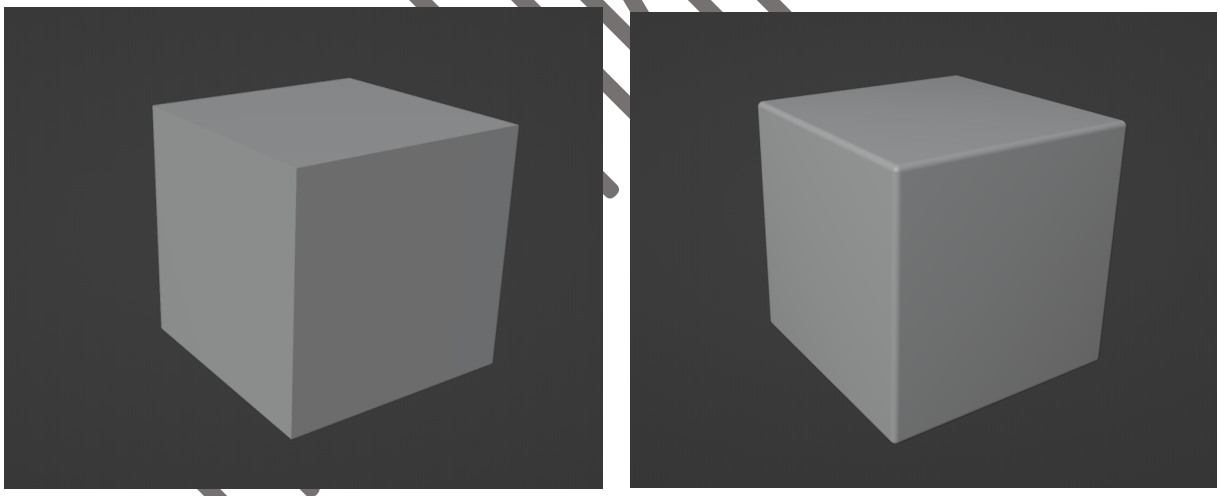


Рисунок 66 – Обычный куб и он же, но со сглаженными углами благодаря карте нормалей

Разберем по порядку процесс создания UV вручную.

Для начала необходимо перейти во вкладку **UV Editing** и включить **Edit**. Необходимо добавить швы объекту. Швы – это места, по которым объект будет разворачиваться на 2D плоскость. Выбрав нужные ребра у объекта, нажимаем правую кнопку мыши и выбираем **Mark seam** (создать шов). Все швы помечаются красным цветом. Пример наложения швов на нагревательный элемент приведен на рисунке 67

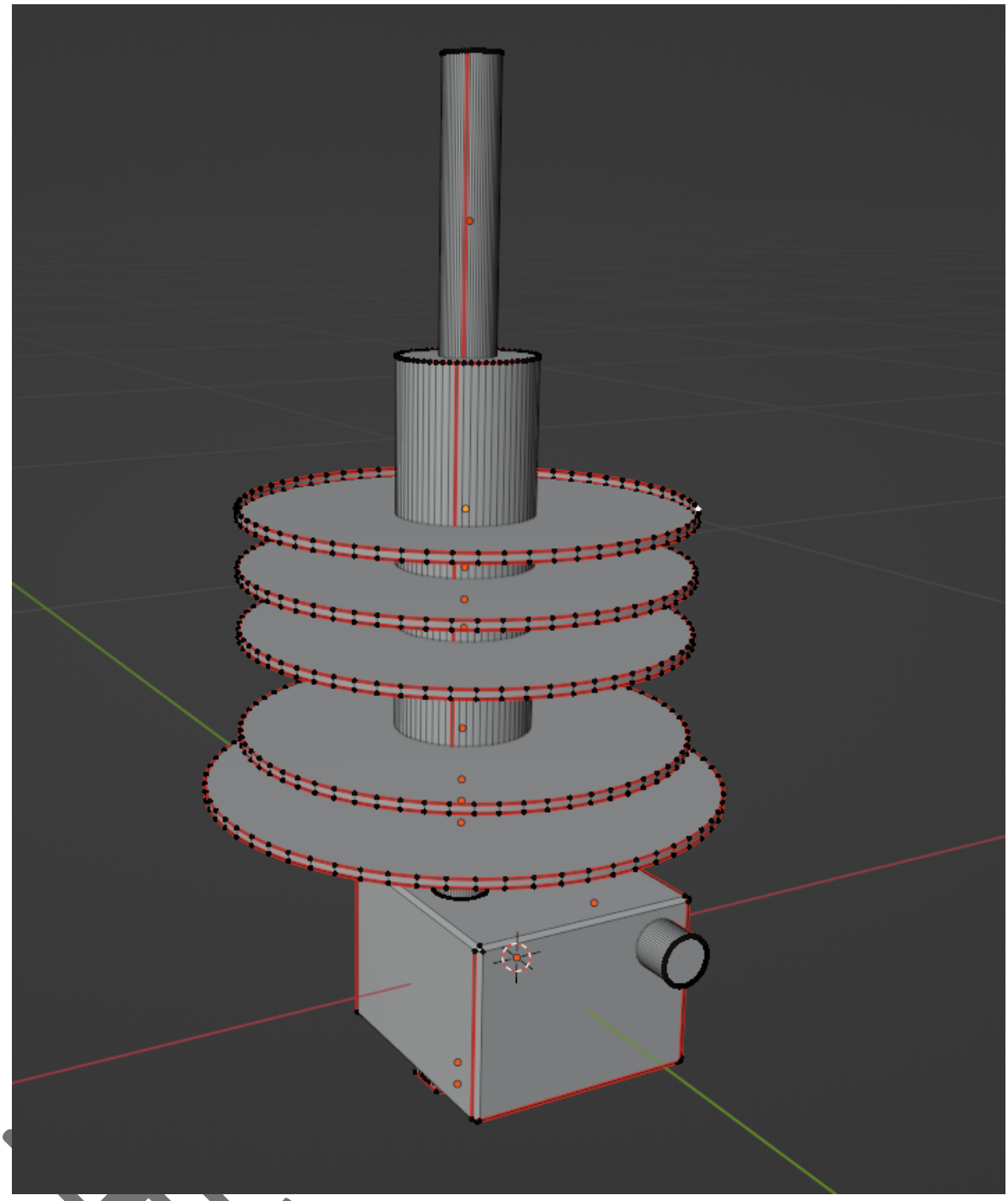


Рисунок 67 – Наложенные швы на нагревательный элемент 3D принтера

Важно, чтобы в режиме редактирования было выбрано выделения ребер. Только в таком случае возможно наложение швов, в противном случае в меню по нажатии правой клавиши мыши не будет **Mark seam**.

Для снятия шва стоит выбирать **Clear seam**. После того, как все разрезы у объекта добавлены, можно нажимать вкладку UV и нажимать Unwrap. Получившаяся развертка появится в окне **UV editor** (важно, чтобы объект в режиме редактирования был выбран, иначе ничего не отобразиться). Пример UV нагревательного элемента приведен на рисунке 68.

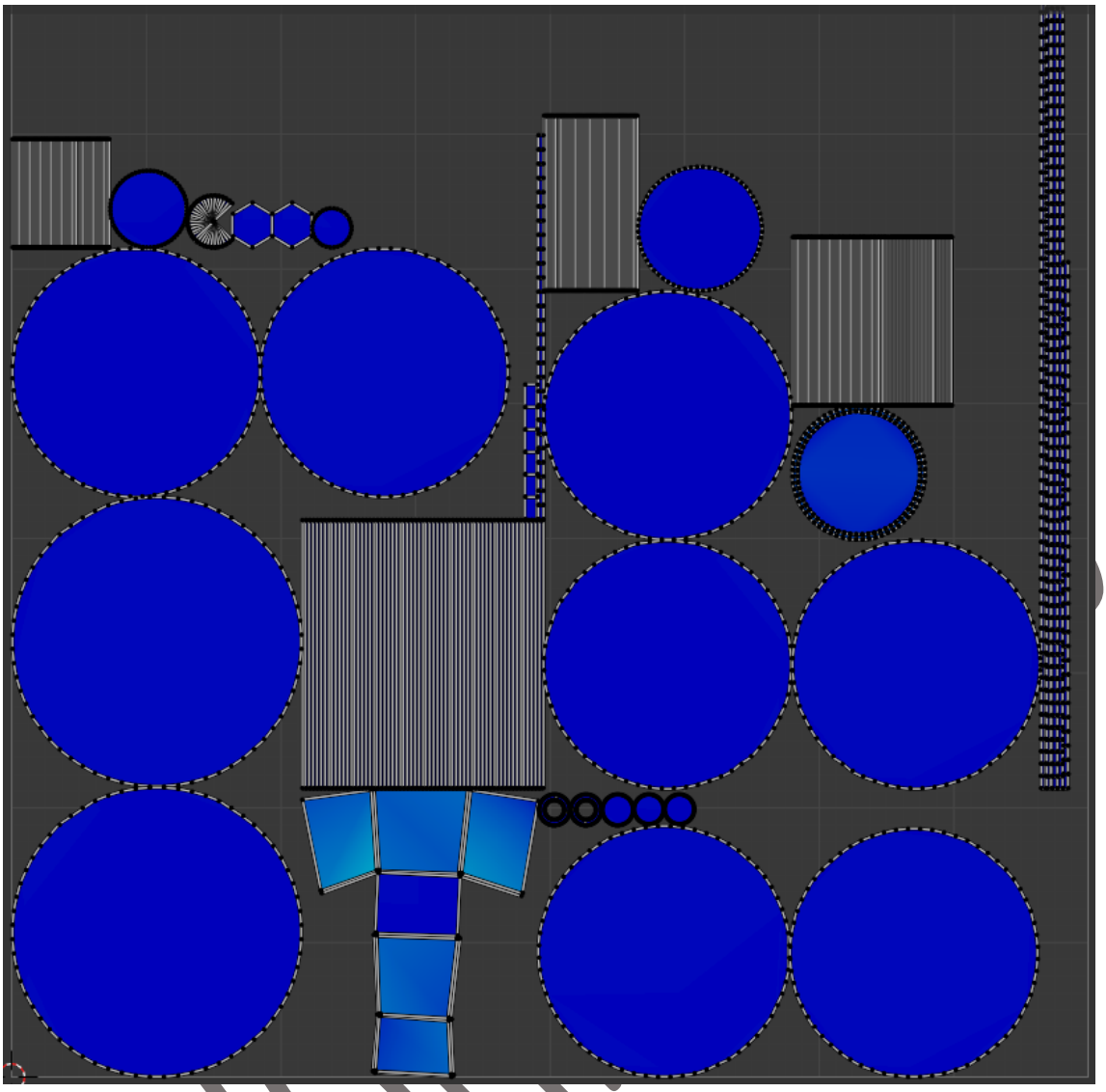


Рисунок 68 – UV развертка нагревательного элемента 3D принтера

После того как UV развертка сделана, ее нужно проверить на качество. Есть две проверки. Первая – это проверка на растяжение островов UV. В окне UV editor нужно включить **Display stretch**. Если все сделано качественно, то UV-развертка вся будет равномерно окрашена в синий цвет. Если же есть перетяжки, то появится градиент цвета в не качественно развернутом месте. Градиент может быть разным. Сильная перетяжка – переход от синего цвета хорошего участка UV к красному цвету плохого.

Допустимым является переход от синего к голубому. В таком случае текстура практически не потеряет своего качества.

Иногда без перетяжек невозможно разложить 3D объект в один остров UV (например, сферу). Именно поэтому легкие перетяжки могут быть допустимыми.

Перетяжкой считается любая ситуация, когда фигура поверхности 3D объекта переход в отличную от оригинала фигуру. Например, сторона куба

(квадрат) переходит в прямоугольник на 2D плоскости. В общем случае этот квадрат может отобразится в произвольный многоугольник или вообще в точку.

Примеры UV с перетяжками представлены на рисунках 69 и 70.

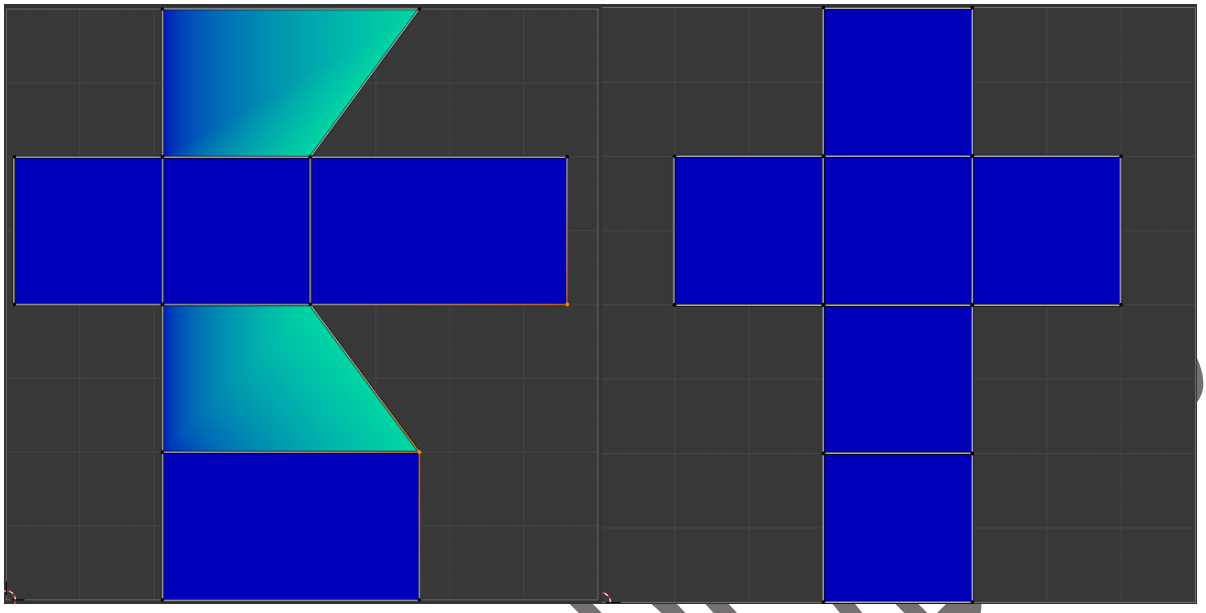


Рисунок 69 – Перетянутая UV-развертка куба – слева, хорошая – справа

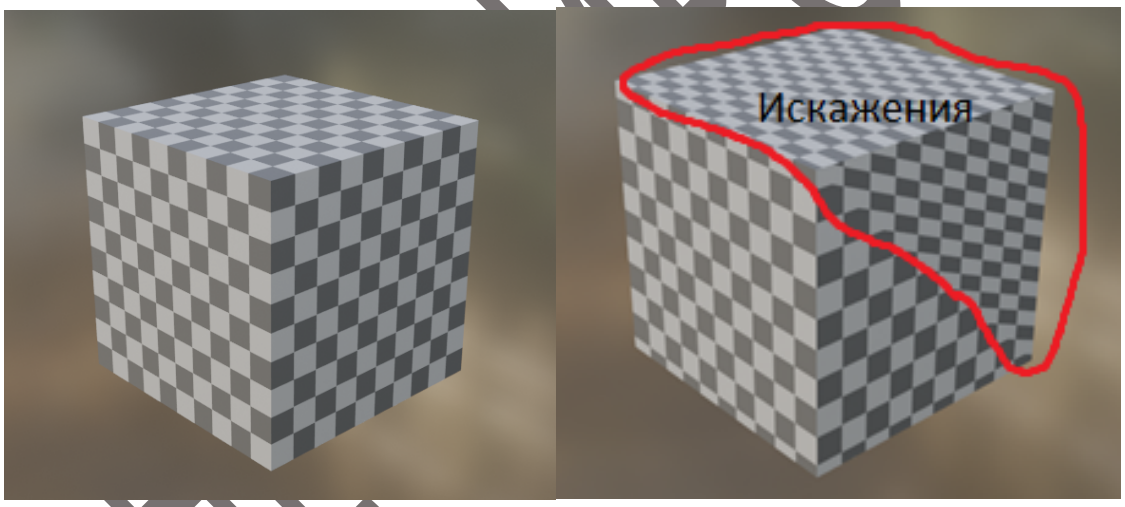


Рисунок 70 – Шахматная сетка на кубе с не перетянутой uv и с перетянутой

На UV нагревательного элемента, показанной на рисунке 68 также есть допустимая перетяжка.

Второй проверкой качества UV является поиск самопересечений/наложений острова/островов. Как раз в таком случае возникает ситуация, когда одной точке 2D плоскости соответствует несколько точек на 3D объекте, что может привести к некачественному наложению текстуры.

Кстати, полное наложение островов UV друг на друга допускается для оптимизации моделей. В таком случае несколько зон объекта будут иметь одинаковые текстуры.

UV- развертка создана. Что же будет если над каким-либо островом UV производить действия? Все просто. Изменяя положение острова, будет изменяться и положение текстуры на объекте. Изменяя масштаб острова, будет изменяться масштаб текстуры. Наконец, вращая остров UV, будет поворачиваться текстура на объекте. Примеры приведены на рисунке 71.

Рисунок 71 – Слева – исходный куб, справа – куб с увеличенной и повернутой текстурой

Несколько объектов могут иметь одну текстуру или материал. В таком случае UV развертка для этих объектов делается общая.

UV-развёртка для 3D-принтера создаётся с соблюдением тех же принципов.

Соблюдались несколько параметров:

- Допустимые перетяжки;
- Отсутствие случайных наложений островов;
- Наличие между островами UV минимального расстояния;
- Как можно большее заполнение области UV-разверткой.

Для чего необходимо минимальное расстояние между островами? Все дело в том, что текстура состоит из пикселей. Количество этих пикселей задает разрешение текстуре. Как пример, для 3D принтера были созданы текстуры с разрешением 2048*2048 пикселей. Если между островами UV не оставлять расстояния, то в таком случае есть шанс минимального наложения цвета одного острова на другой. Это будет создавать риски на модели и выглядеть не красиво.

Наибольшее заполнение пространства островами необходимо для того, чтобы текстура выглядела качественнее. Иногда на одной зоне UV разворачивают сразу несколько объектов, для того чтобы занять как можно больше пространства. Да, в таком случае острова UV могут быть все равно маленькими. Но в таком случае разрешение текстуры делается выше и проблем с качеством отображения не возникает. В больших проектах такой вариант предпочтительнее, так как на него уходит меньше ресурсов компьютера.

Ну и в качестве бонуса, приведенный пример развертки нагревательного элемента, изображенного на рисунке 68, имеет недочеты в виде малой зоны заполнения и отсутствием расстояний между островами.

При создании UV-развертки следует стремиться к следующим показателям:

- заполнение должно быть не менее 75%. Пример корректного заполнения представлен на рисунке 72;

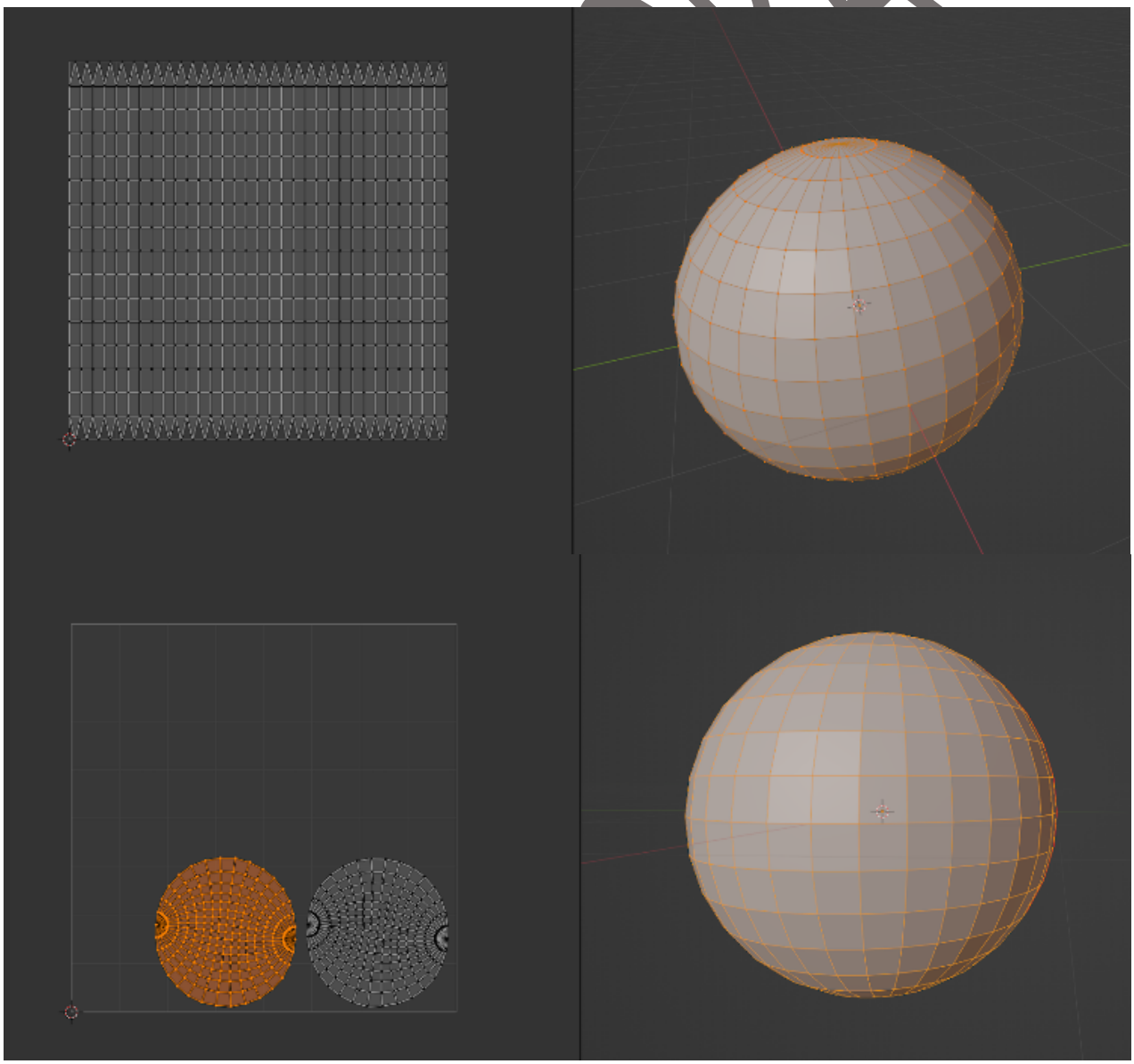


Рисунок 72 – Правильная (сверху) и неправильная (снизу) заполняемость UV

- должны отсутствовать наложения (исключения составляют повторяющиеся элементы, текстуры которых могут быть одинаковыми). Обратите внимание на рисунок 73;

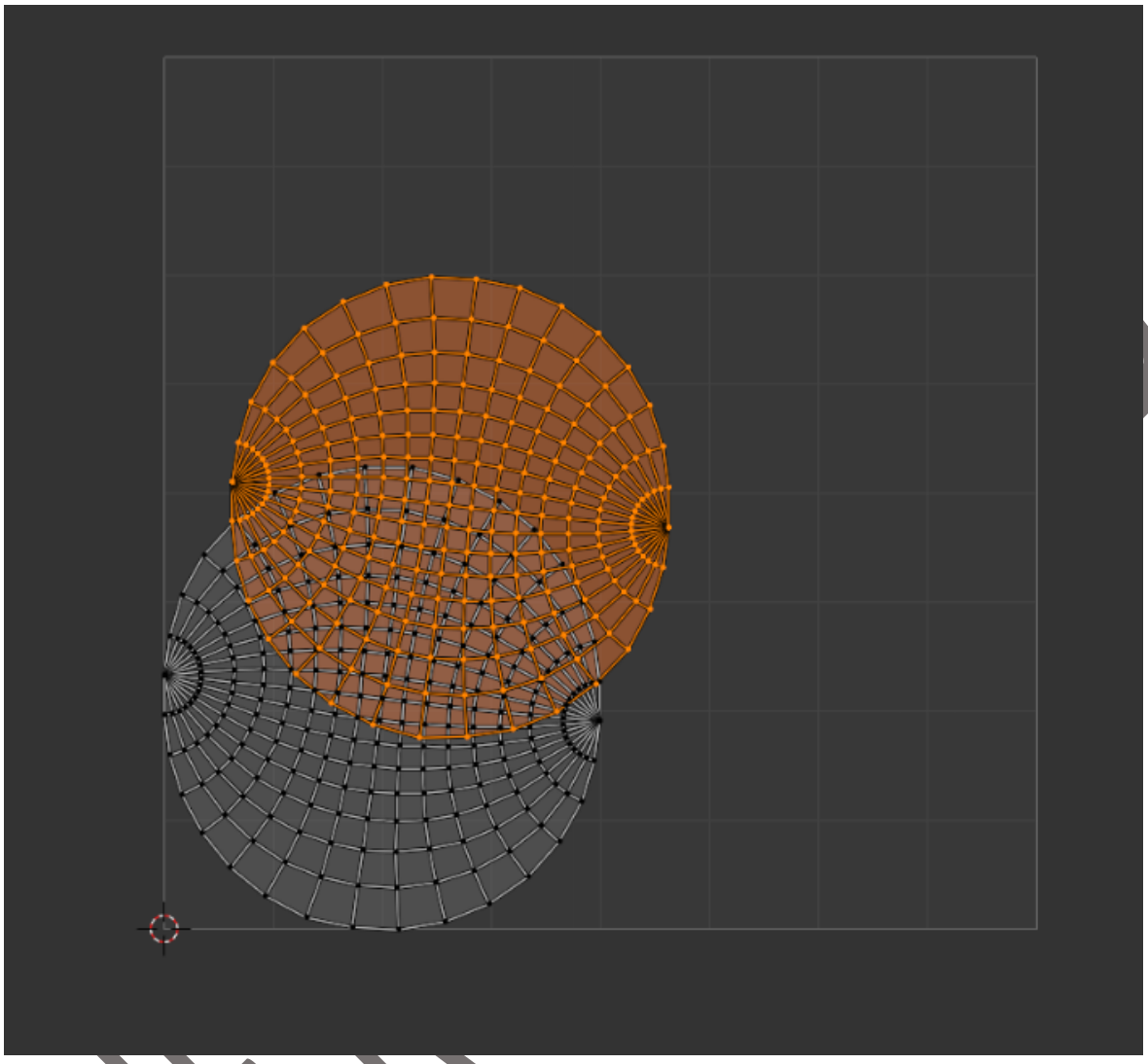


Рисунок 73 – Пример наложения

- должны отсутствовать перетянутые полигоны;

Для отслеживания этого параметра нужно в разделе uv-развертки включить **Display stretch.** Рисунок 74**:**

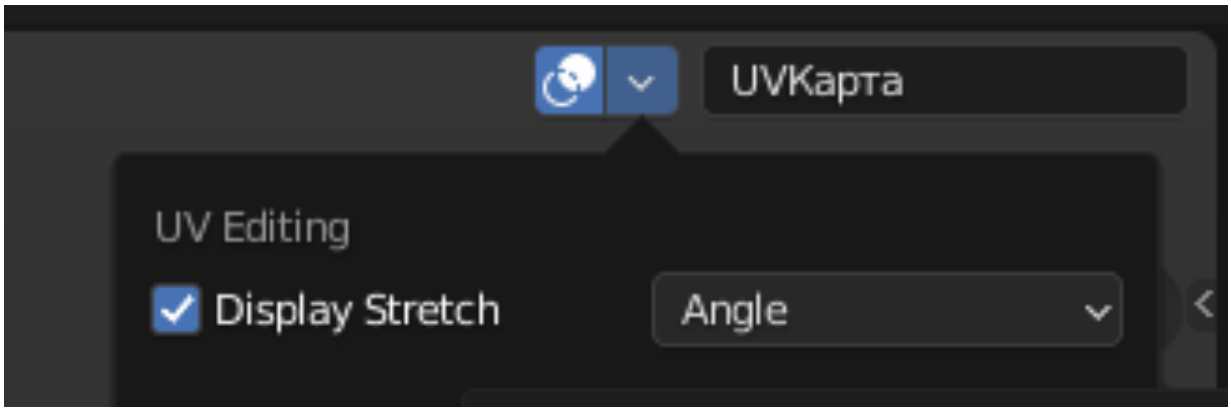


Рисунок 74 – панель Stretch display

У развертки появится цвет, варьирующийся от синего до красного, где синий означает отсутствие растяжек, а чем ближе к красному, тем сильнее перетяжка. Рисунок 75 демонстрирует как это выглядит на развёртке.

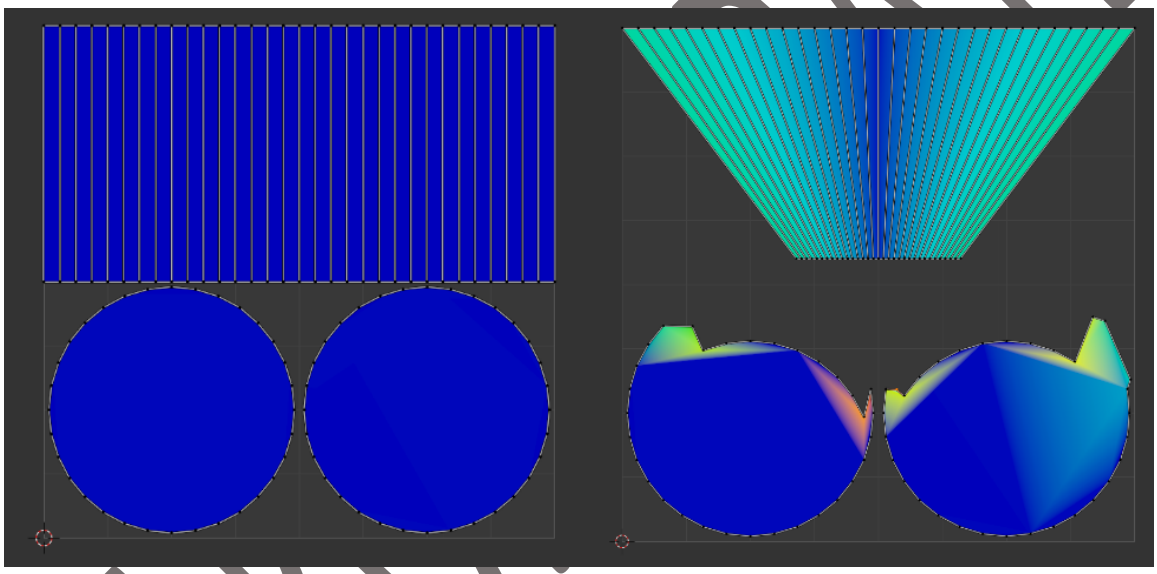


Рисунок 75 – Пример одной и той же развёртки без перетяжек (слева) и с перетяжками (справа). Желтые полигоны перетянуты

Для быстрого создания развертки в **edit mode** нужно выбрать все вершины объекта, нажать **u** и выбрать **smart uv project**, выставив настройки как показано ниже на рисунке 76:

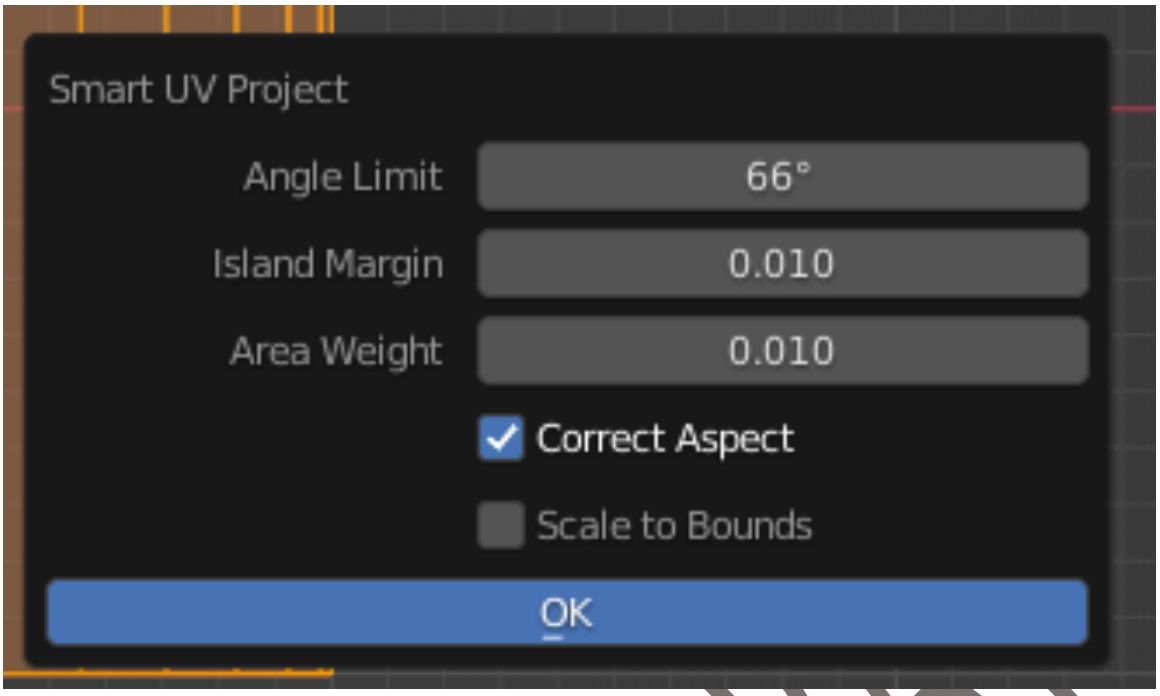


Рисунок 76 – панель Smart UV Project

Но так как автоматическая развертка не всегда хорошо срабатывает, чаще нужно указывать по каким «швам» нужно разрезать объект. Для этого нужно выбрать нужные грани, нажать правую кнопку мыши и в выпадающем меню выбрать **mark seam** (тогда выбранные грани станут красными), после выбрать все вершины объекта, нажать u и выбрать **unwrap**.

ГЛАВА 4 Материалы

Как мы уже знаем, при создании проектов цифровых двойников в виртуальной реальности очень важно сохранять эффект погружения. Очень важной его частью является качественный внешний вид моделей. Поговорим о создании качественных материалов.

4.1. Качество текстур на 3D объектах

UV-развертка имеет характеристику Texel density.

Texel density – это количество пикселей деленное на размер геометрии объекта. То-есть это плотность пикселей. Чем этот показатель выше, тем качественнее будет текстура на данной поверхности.

На рисунке 77 наглядно видно, изображение с одинаковым разрешением имеет разную детализацию. При меньшей площади поверхности текстура будет выглядеть лучше и не кажется растянутой. Чтобы исправить разную детализацию объектов, можно увеличить разрешение текстуры, пример приведен на рисунке 78.

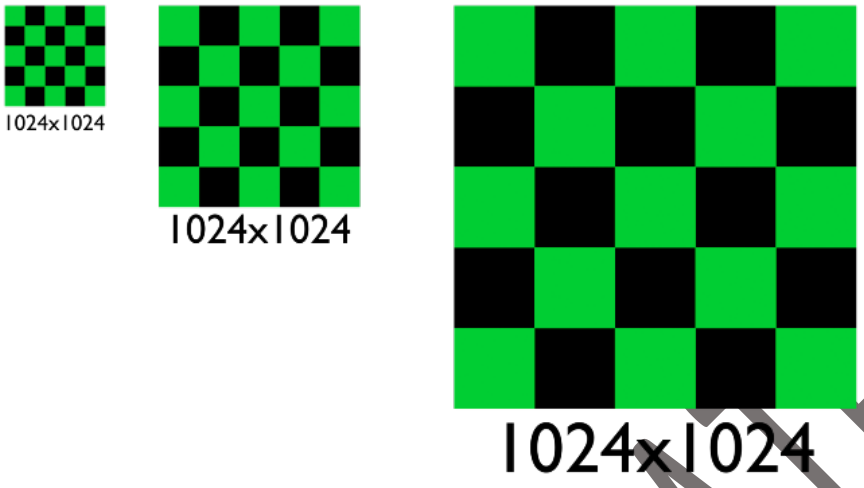


Рисунок 77 – Квадраты с разной площадью поверхности и с одинаковым разрешением текстур

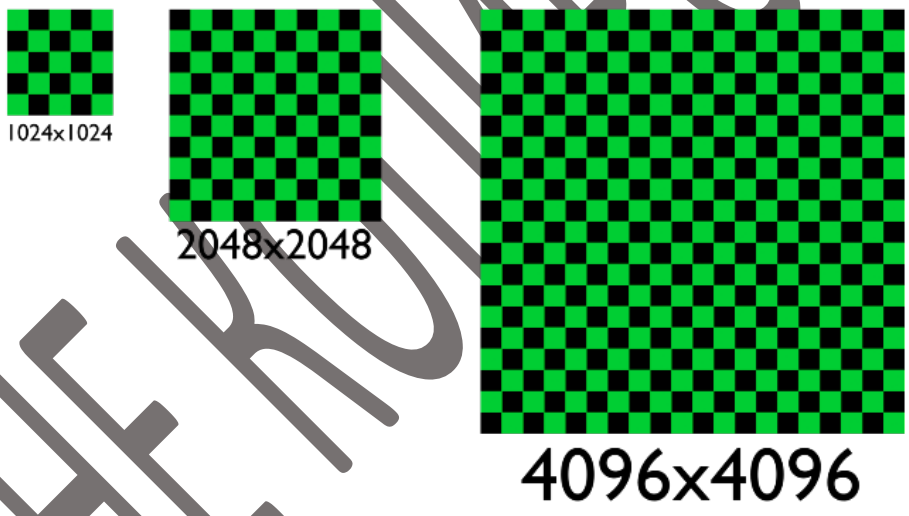


Рисунок 78 – Квадраты с разной площадью поверхности и с разным разрешением текстур

В общем случае **Texel density** должен быть примерно одинаков для всех объектов на сцене. Пример можно посмотреть на рисунке 79. Для 3D-принтера и токарного станка этот параметр отслеживался в Blender по масштабу объекта на сцене. Важно, чтобы Scale всех объектов был одинаков.

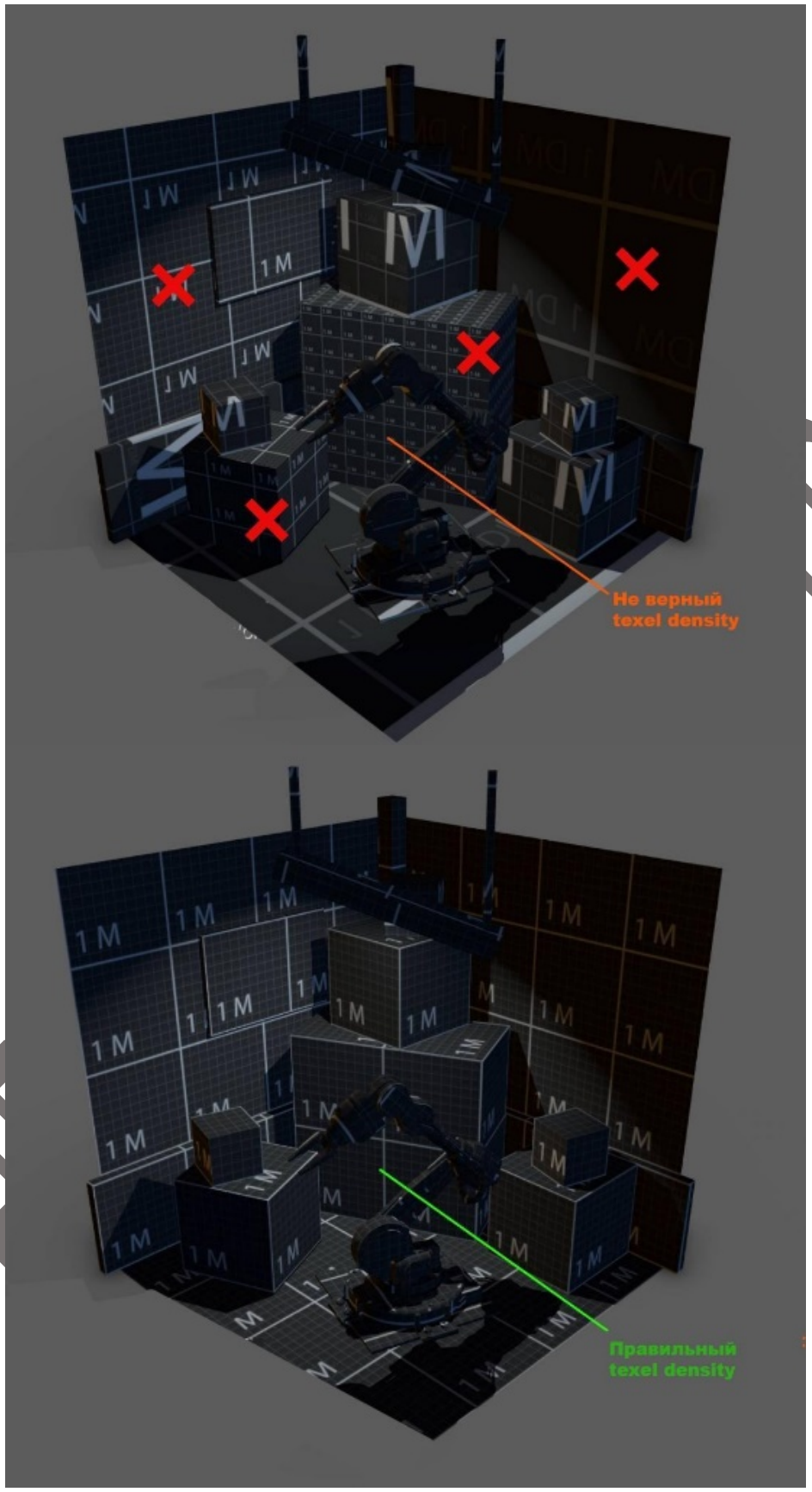


Рисунок 79 – Сверху – разный Texel density, снизу – одинаковый

Есть и исключения из правил. Элементы поверхностей, которые практически не заметны делают с низким **Texel density**, например, подошва

ботинка персонажа или дно машины. И наоборот, если объект является основным, важным, то его плотность пикселей делают выше. Примером может служить лицо персонажа.

4.2. Разрешение текстур

Текстуры могут иметь бесконечное количество разрешений, ограничением может служить только память аппаратного обеспечения.

Почему UV в большинстве случаем разворачивается в квадратной области и соответственно задает квадратную форму изображению? Все дело в аппаратном обеспечении и его особенностях использовать память для обработки изображений.

Представим, что картинка лежит на графической плоскости с двумя перпендикулярными осями Ox и Oy. Пример изображен на рисунке 80. Каждый пиксель (оранжевый квадрат) изображения может задаваться только одной координатой каждой из осей.

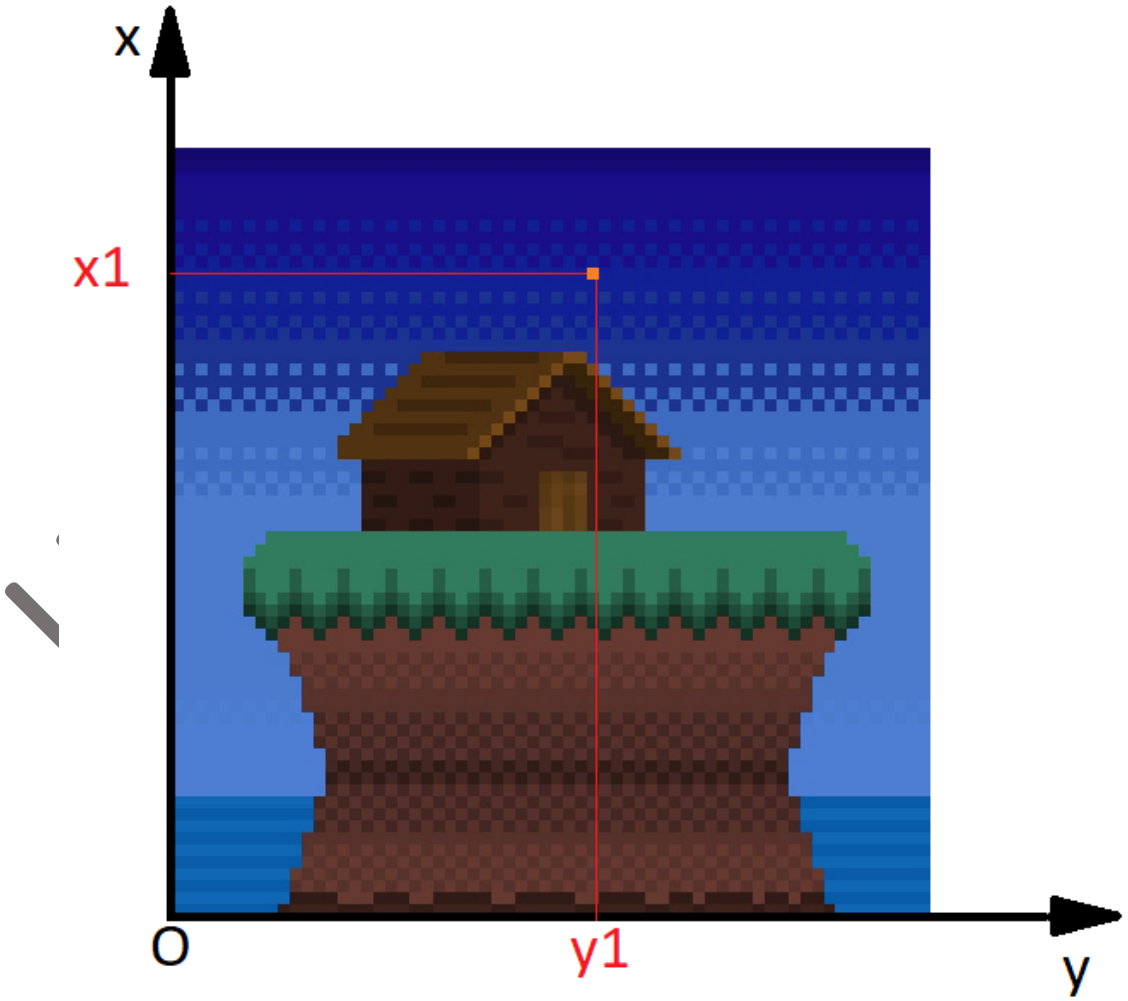


Рисунок 80 – Пиксельное изображение

Существует такой термин, как **swizzling**. Его суть заключается в том, чтобы разместить соседние пиксели изображения так, чтобы они были соседями в памяти. Однако в памяти возможно выделить только определенный конечный блок байтов (32 или 64 байта для intel, это особенность аппаратного обеспечения). Поэтому и картинка будет делиться на эти фрагменты. Если разрешение текстуры не делиться ровно на блоки, в таком случае в памяти будет выделены участки, содержащие остатки пикселей картинки, а остальное место в блоке памяти будет пустое. Это пустое место не может заниматься какими-либо другими процессами, то есть память используется в пустую. Для того, чтобы этого избежать, следует создавать текстуры кратные степеням двойки.

Однако это не отвечает на вопрос, почему текстуры следует делать квадратными. Что мешает сделать разрешение изображения, например, 1024x512? Прямоугольные текстуры с разрешением кратным двойке делать можно, однако в таком случае нужно учитывать растяжение UV- развертки.

4.3. Основные текстурные карты

Материал как правило состоит из текстурных карт. Эти карты, в свою очередь, бывают разных типов. Перечислим основные:

- диффузная (diffuse map/base color)
- карта бликов (specular map), может также содержаться в альфа-канале диффузной текстуры
- карта нормалей (normal map)
- карта высот (height map), может содержаться только в альфа-канале карты нормалей, используется для реализации рельефной поверхности (parallax mapping)
- карта шероховатостей (roughnees)
- карта прозрачности (alpha)
- Карта маталичности (metallic)

Текстурные карты могут влиять на разные свойства материалов:

- задавать цвет материалу;
- влиять на способ отображения этого цвета;
- являться инструментом для работы. В этом случае текстура не обязательно должна принадлежать объекту. Примером может служить карта ID.

Кратко поговорим об основных текстурных картах, которые задают цвет модели.

Диффузная текстура применяется для указания распределения цвета на объекте. Пример приведен на рисунке 81.

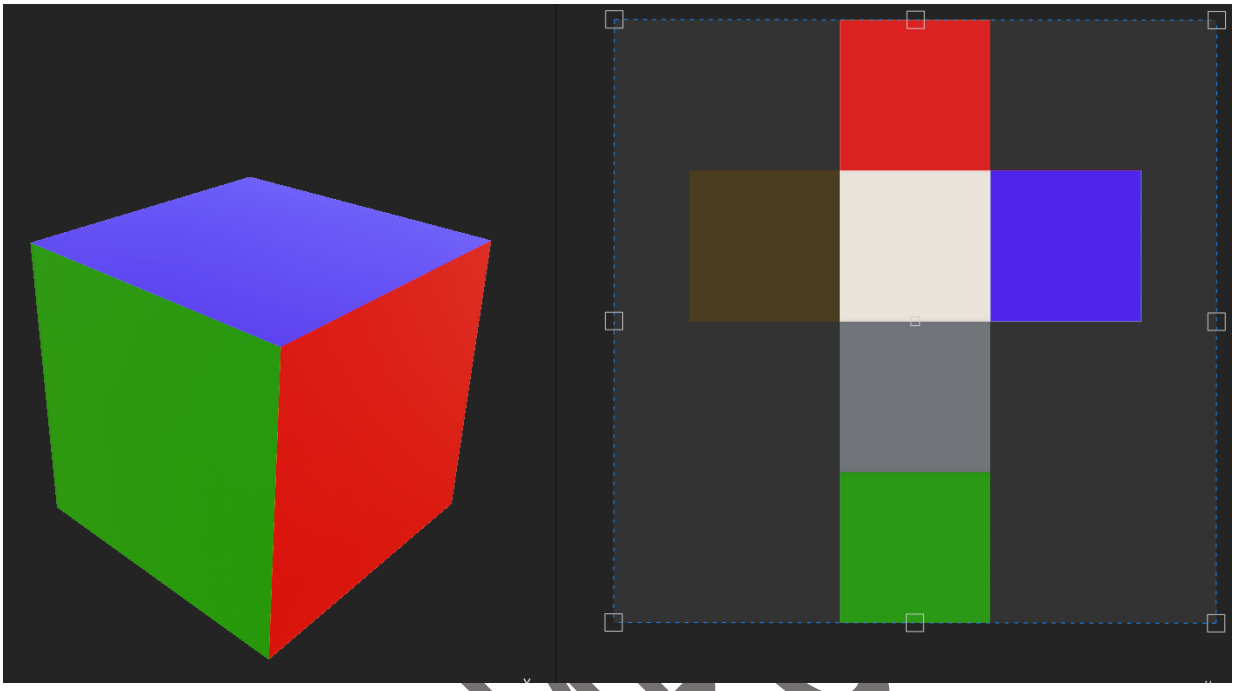


Рисунок 81 – Карта base color и его UV-развертка на примере куба

Карта нормалей используется для имитации деталей поверхности. Это карта RGB, где каждый компонент соответствует координатам X, Y и Z нормали к поверхности. Его можно использовать для сохранения спроецированных деталей модели с **High Poly** модели на модель **Low Poly**. Это очень важно в игровых моделях для оптимизации. Пример приведен на рисунке 82.

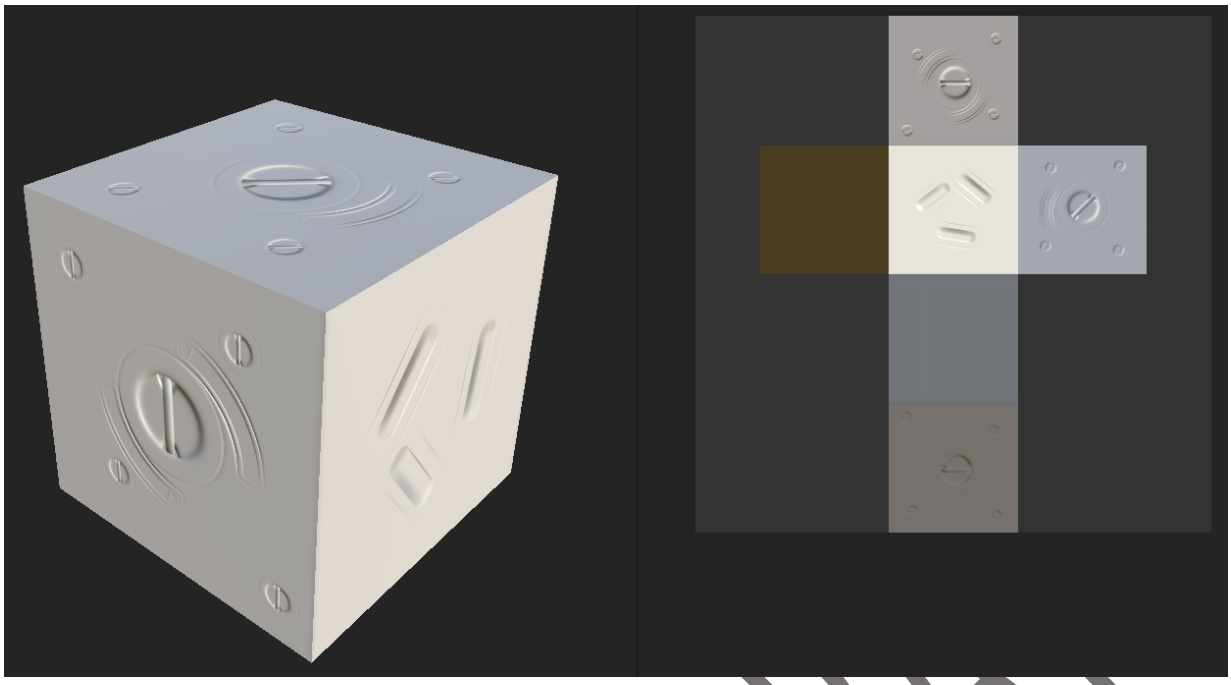


Рисунок 82 – Пример карты нормалей на кубе

Карта высот содержит информацию о распределении относительных высот рельефа. Более высокий уровень поверхности обозначается более светлым цветом. Карта высот в сочетании с картой нормалей требуются в качестве входящих данных для реализации рельефной поверхности (p**arallax mapping**). Карта высот должна содержатся в альфа-канале карты нормалей.

Карта прозрачности (**alpha**) отвечает за прозрачность объекта. Это черно белая карта, в которой белый цвет отвечает за видимую часть объекта, а черный – за невидимую.

Roughness – карта шероховатости. Она описывает неровности поверхности, вызывающие рассеивание света. Направление отражения будет изменяться случайным образом в зависимости от шероховатости поверхности. Это изменяет направление света, но интенсивность света остается постоянной. Более грубые поверхности будут иметь более крупные и тусклые блики. Более гладкие поверхности будут сфокусировать зеркальные отражения, которые могут выглядеть ярче или интенсивнее, даже если отражается то же общее количество света. Примеры объекта с шероховатостью и без приведен на рисунке 83.

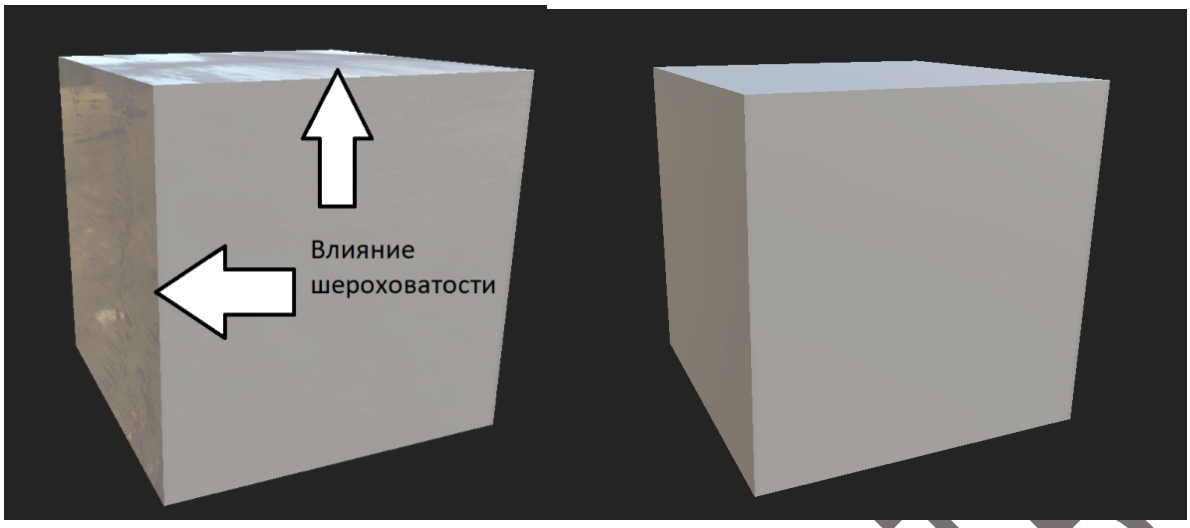


Рисунок 83 – Куб с шероховатостью и без нее

Metallic – это карта, которая задает свойство отображению всему материалу. Это черно белая карта с градациями серого. Черный цвет определяет не металлические свойства, белый – наоборот. На рисунке 84 слева можно видеть объект с однородной максимальной металличностью, а справа – с наложенной черно-белой картой

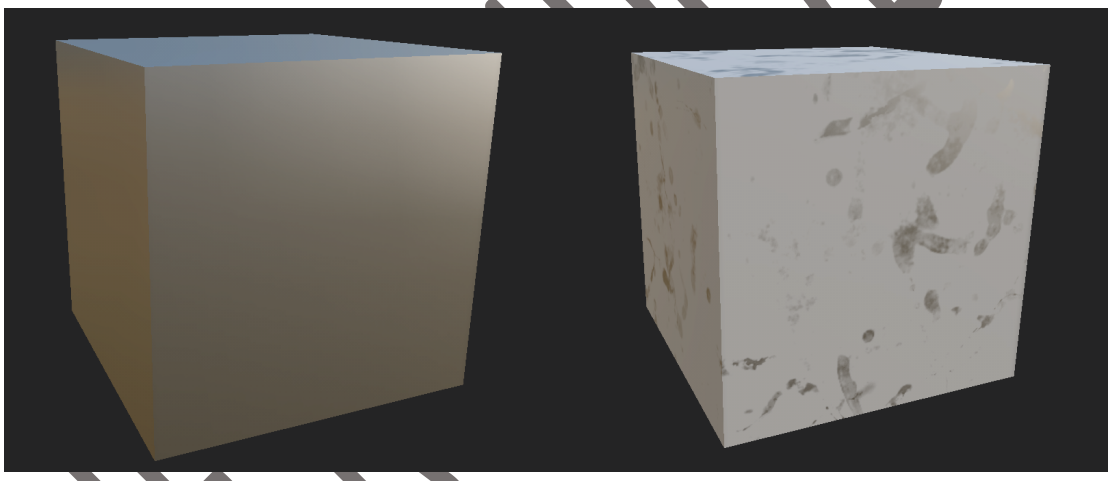


Рисунок 84 – Влияние металлических свойств на объект

4.4. Создание материалов в Substance painter

Substance painter – программа для работы с материалами и быстрого текстурирования 3D-моделей. Она позволяет значительно упростить и ускорить процесс подготовки материалов.

На рисунке 85 демонстрируется главное окно программы.

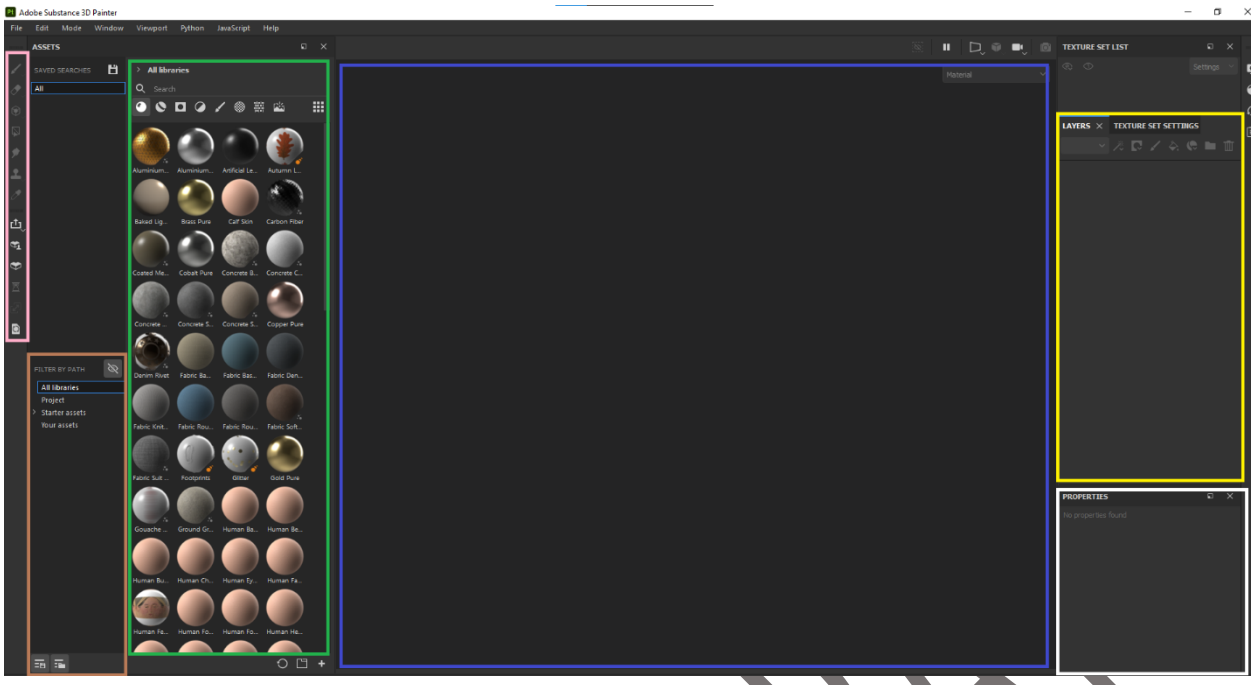


Рисунок 85 - Начальное окно программы после запуска

Разберем, какие области обозначены рамками на рисунке 85.

- Синяя рамка – окно с моделью. При запуске это всегда пустое.
- Зеленая рамка – библиотека асетов материалов и текстур, необходимых для покраски модели.
- Розовая рамка – инструменты рисования.
- Коричневая рамка – позволяет выбрать источник ассетов с текстурами и материалами. Также в этой зоне можно выбрать место сохранения своих текстур, импортируемых в программу.
- Желтая область имеет два меню. Первое – **layers**. Это слои материалов и текстур, назначенных объекту. Второе – **Texture set settings**. В нем происходит переход в окно запекания текстур и материалов.
- Белая рамка. В ней происходит настройка любого активного элемента, инструмента, материала и т. д.

Рассмотрим поближе некоторые области.

Окно **Texture set settings**. В нем самым главным являют элементы, выделенные в прямоугольные рамки. Зеленая отвечает за размер текстур. Синяя рамка включает в себя настройку текстурных карт, их добавление и удаление. Самым главным элементом этого окна является кнопка перехода в окно запекания текстур – bake mesh maps, она выделена красным цветом. Все элементы изображены на рисунке 86.

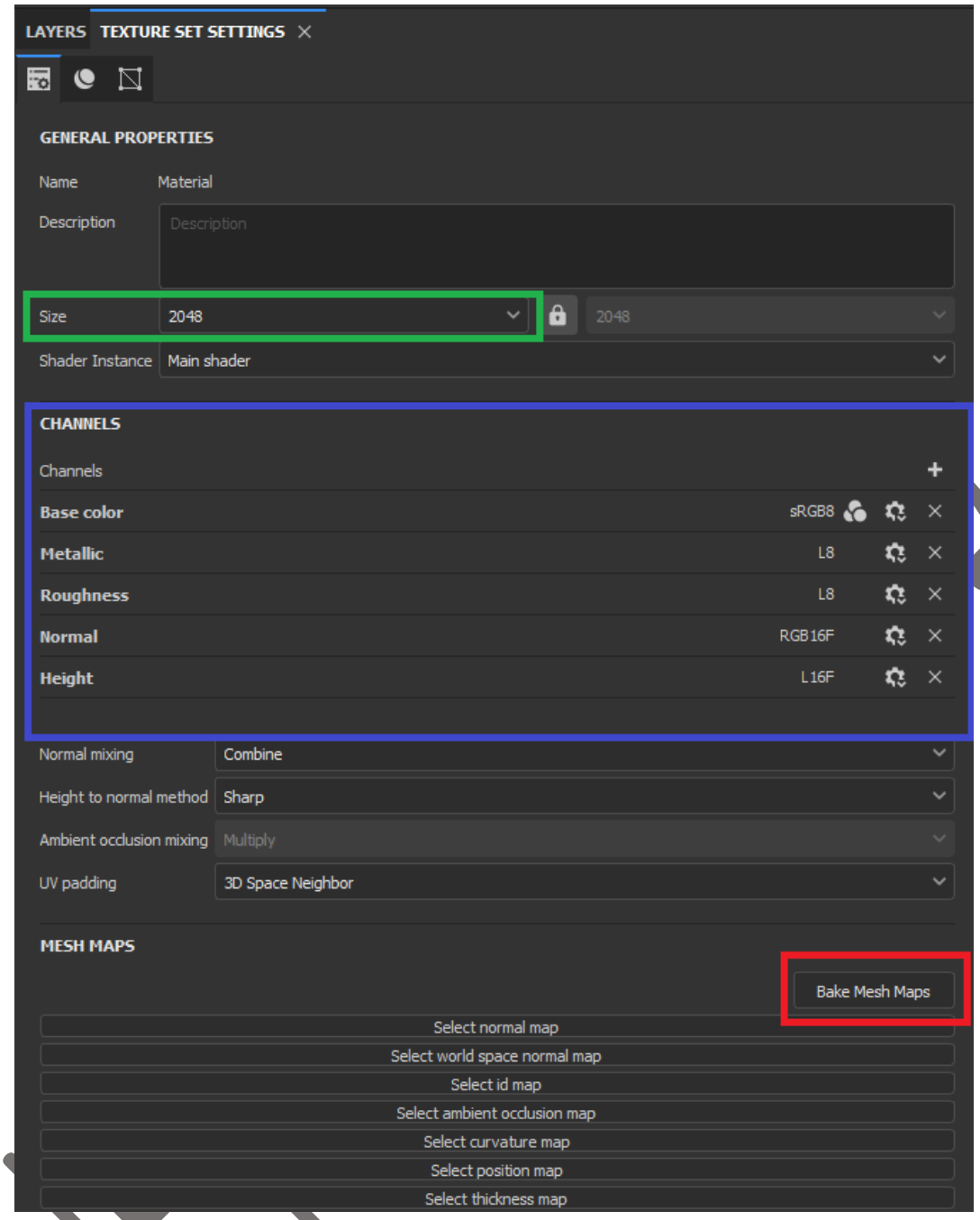


Рисунок 86 - Окно Texture set settings

Перед тем, как начать что-либо текстурировать, нужно экспортировать объект в сцену и выполнить запекание карт. Окно запекания карт было кратко описано в главе экспорта и импорта. Наживаем вкладку **File -> New**. Появится окно **New project**. В нем нужно выбрать путь сохранения файла, а также обратить внимание на галочку **Auto-unwrap**. Она отвечает за то, нужно ли создавать автоматическую UV-развертку или нет. Развертку стоит делать самостоятельно, чтобы не было ошибок и результат на выходе был предсказуем.

На рисунке 87 элементы, упомянутые в предыдущем абзаце, обозначены прямоугольными рамками.\

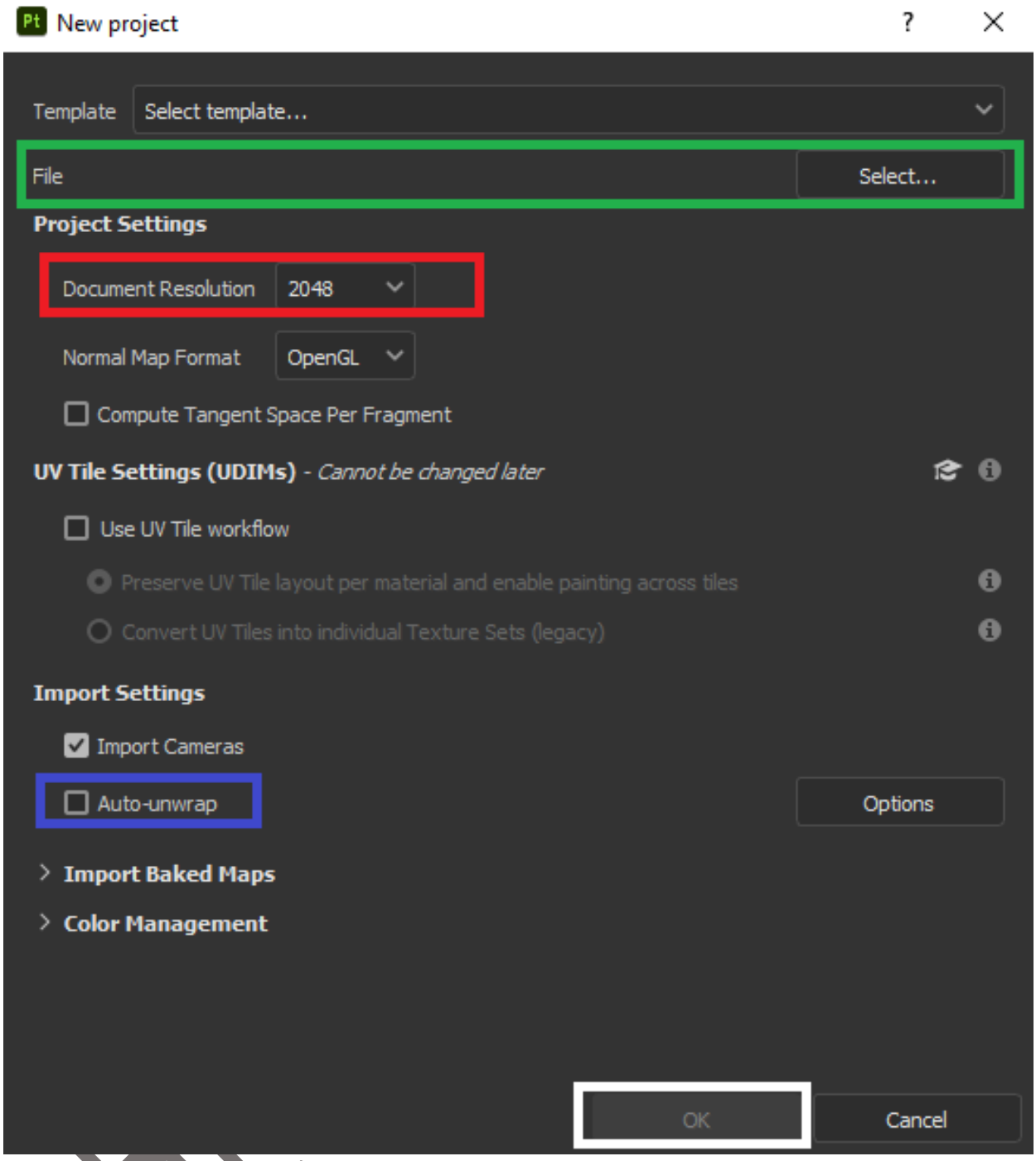


Рисунок 87 - Окно создания проекта

Важно: объект в сцене должен иметь UV-развертку. Без нее текстуры будут накладываться неправильно.

Нажимаем «Ок» и видим, что объект появился на сцене.

Следующим этапом является запекание карт. Можно обойтись и без нее, если требуется только покрасть объект. Однако в таком случае наложение процедурных (комплексных/сложных) материалов становится невозможным или возможным, но некорректно. На рисунке 88 можно увидеть отсутствие необходимых для правильной работы карт у генератора.

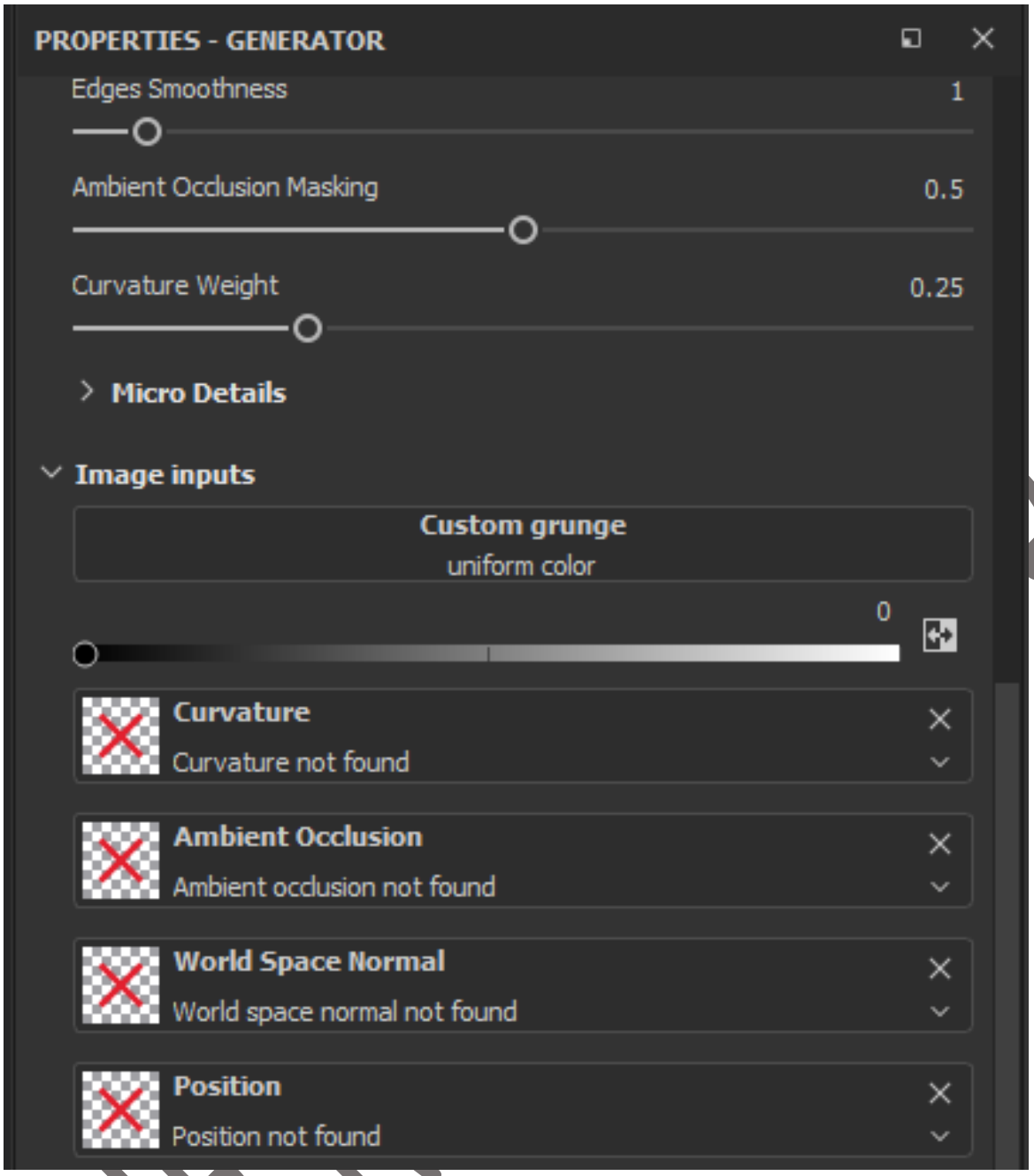


Рисунок 88 - Пример некорректной работы генератора при отсутствии запекания текстурных карт

Выбираем **Texture set settings** -> **Bake mesh maps**. Появляется окно **Baking**

4.5. Библиотеки текстур, материалов для Substance painter

В общем случае для Substance painter нет ограничений по библиотекам текстур и материалов. Для программы можно использовать практически все, что попадется в интернете. Помимо этого, substance painter имеет собственную библиотеку материалов прямо в программе.

Для того, чтобы скачанный материал или текстура появилась в substance painter, ее достаточно просто импортировать. Текстура появится во

внутренней библиотеке substance painter. После этого ее можно накладывать на объект.

Разберем процесс импорта по порядку. Переходим в меню **File -> Import** resources. Нажимаем **Add resources**. Находим необходимый файл. Затем выбираем как будет использован выбранный объект. На выбор представлены четыре варианта:

- **Alpha** - карта прозрачности;
- **Color lut** – цветовой диапазон;
- **Environment** – окружение;
- **Texture** – текстура.

Помимо **Import resources** существуют еще места, где нужно экспортировать данные. Им является окно с запеканием текстур. В нем есть вкладка ID, которая позволяет экспортировать данные о способе покраски модели. К этим способам относятся покраска с помощью созданных материалов в другой программе или покраска по **Vertex color/ID карта**. О ней будет рассказано ниже.

4.6. ID карта

Карта ID является инструментом, помогающим в текстурировании. Это разноцветная карта поверхности 3D или 2D объекта. Каждый цвет является ссылкой на участок поверхности. Экспортировав ее в Substance painter, можно избавится от большого количества материалов у объекта, для каждого из которых будет созданы текстурные карты.

Предположим, что нужно покрасить куб в три цвета. Есть возможность добавить три материала на куб, экспортировать в substance painter и для каждого материала создать карту base color, normal и roughness. Получаем 3 материала * 3 текстурные карты под каждый из материалов = девять карт.

Создав карту ID из трех цветов и экспортировав ее в substance painter, мы получим всего один материал. С помощью масок можно для каждого цвета ID карты наложить карты base color, normal и roughness. При этом на выходе будет экспортировано всего три текстурные карты.

ID может быть создана в blender с помощью Vertex color. Для примера создадим куб и раскрасим его грани в три цвета с помощью vertex color. Для этого перейдем в режим редактирования и выберем произвольные две грани. Затем перейдем в режим Vertex color, выберем выделение гранями, установим произвольный цвет покраски, например, зеленый. Осталось перейти во вкладку **Paint -> Set vertex paint**. Результат представлен на рисунке 89.

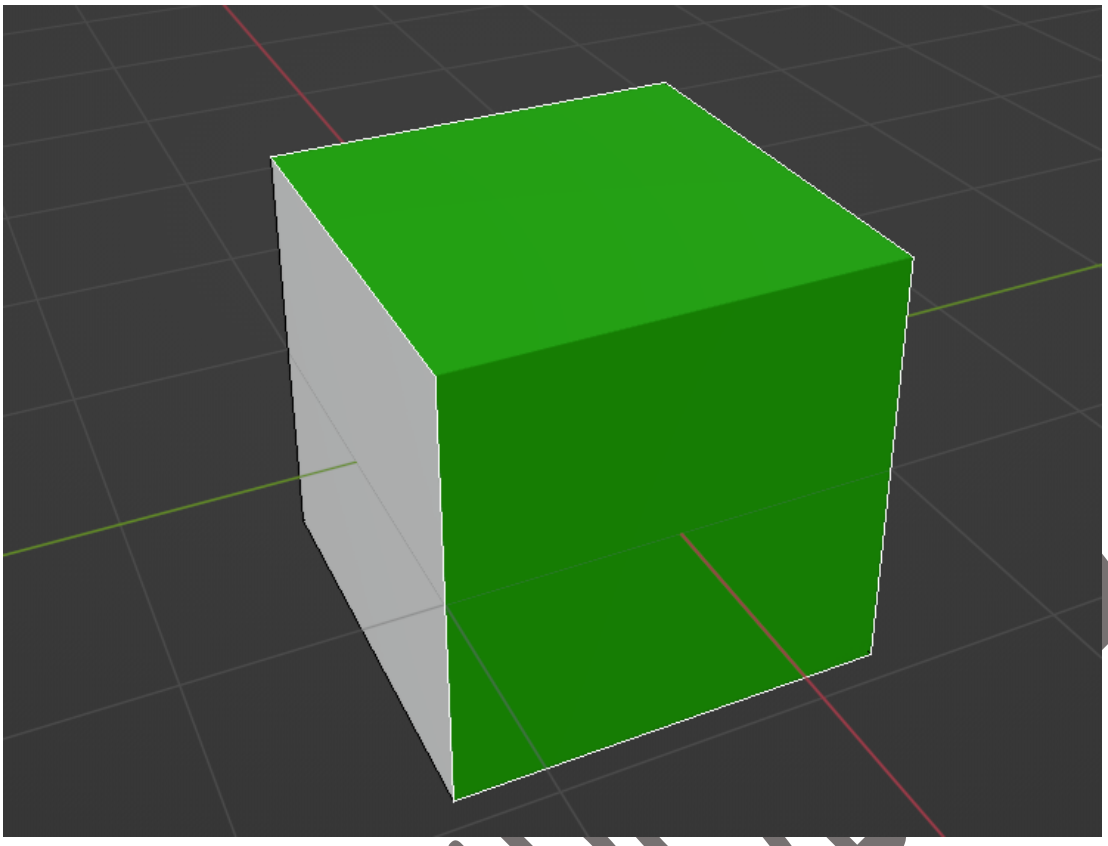


Рисунок 89 - Куб, с белым и зеленым цветом на ID карте

Теперь куб покрашен в два цвета – зеленый и белый. Белый цвет присваивается всему объекту по умолчанию, а зеленый мы настроили сами. Именно эта покраска является нашей ID картой.

Присвоим двум граням красный цвет, не выбирая грани с зеленым цветом. Повторяем действия, описанные выше, и видим результат на рисунке 90.

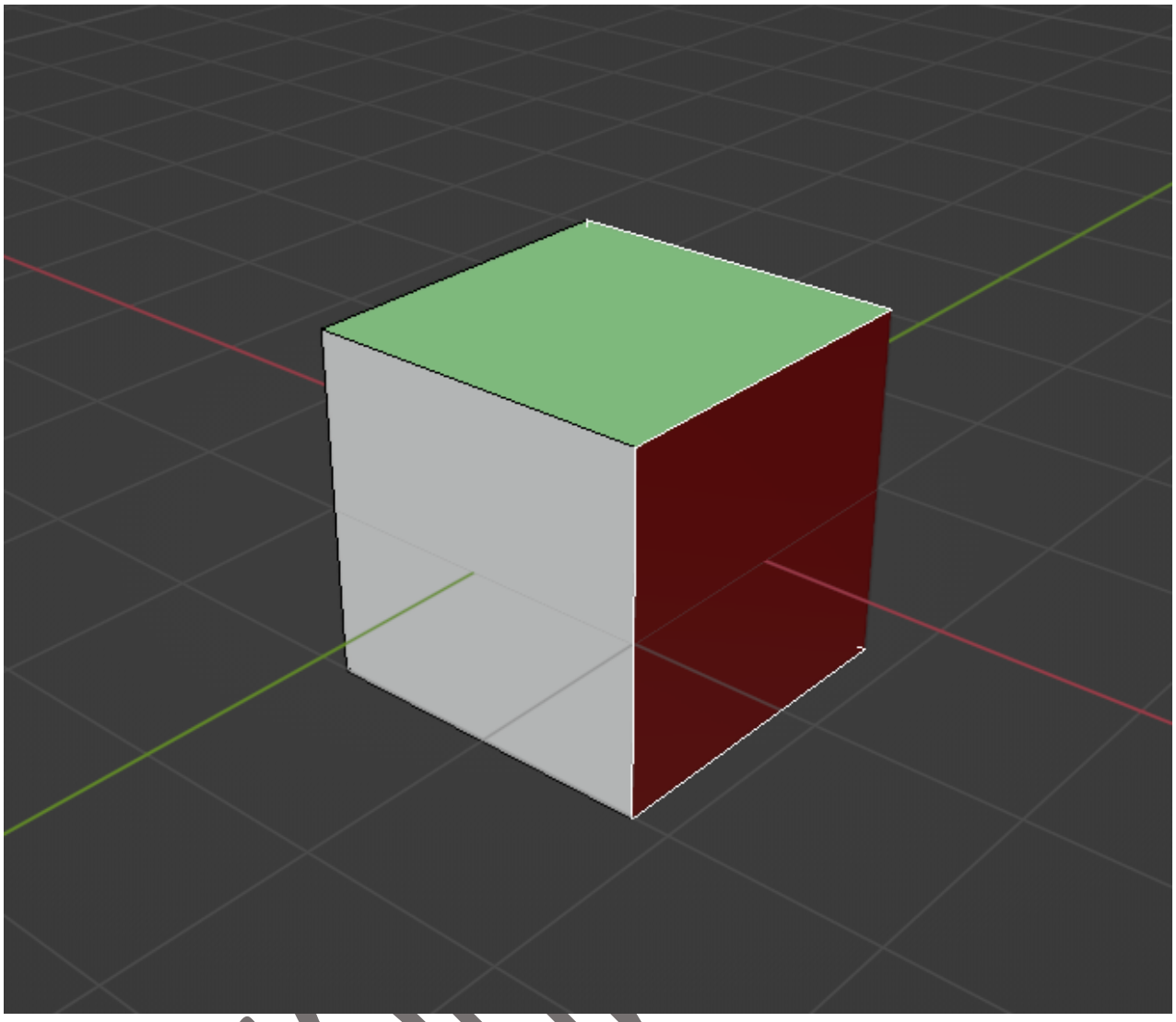


Рисунок 90 - Куб, с тремя цветами на ID карте

Важно, чтобы цвета отличались друг от друга для того, чтобы substance painter мог различить их при импорте.

Экспортировав куб в substance painter, нужно запечь на нем текстурные карты, перед эти выбрав для **ID Vertex color**.

Наложив на меш материал и выбрав **add mask with color selection** программа предлагает пипеткой в параметрах маски выбрать участок меша по цвету ID маски (рисунок) для того, чтобы можно было отдельно настроить для него текстурные карты.

На рисунке 91 красным выделена пипетка, фиолетовым – черная маска.

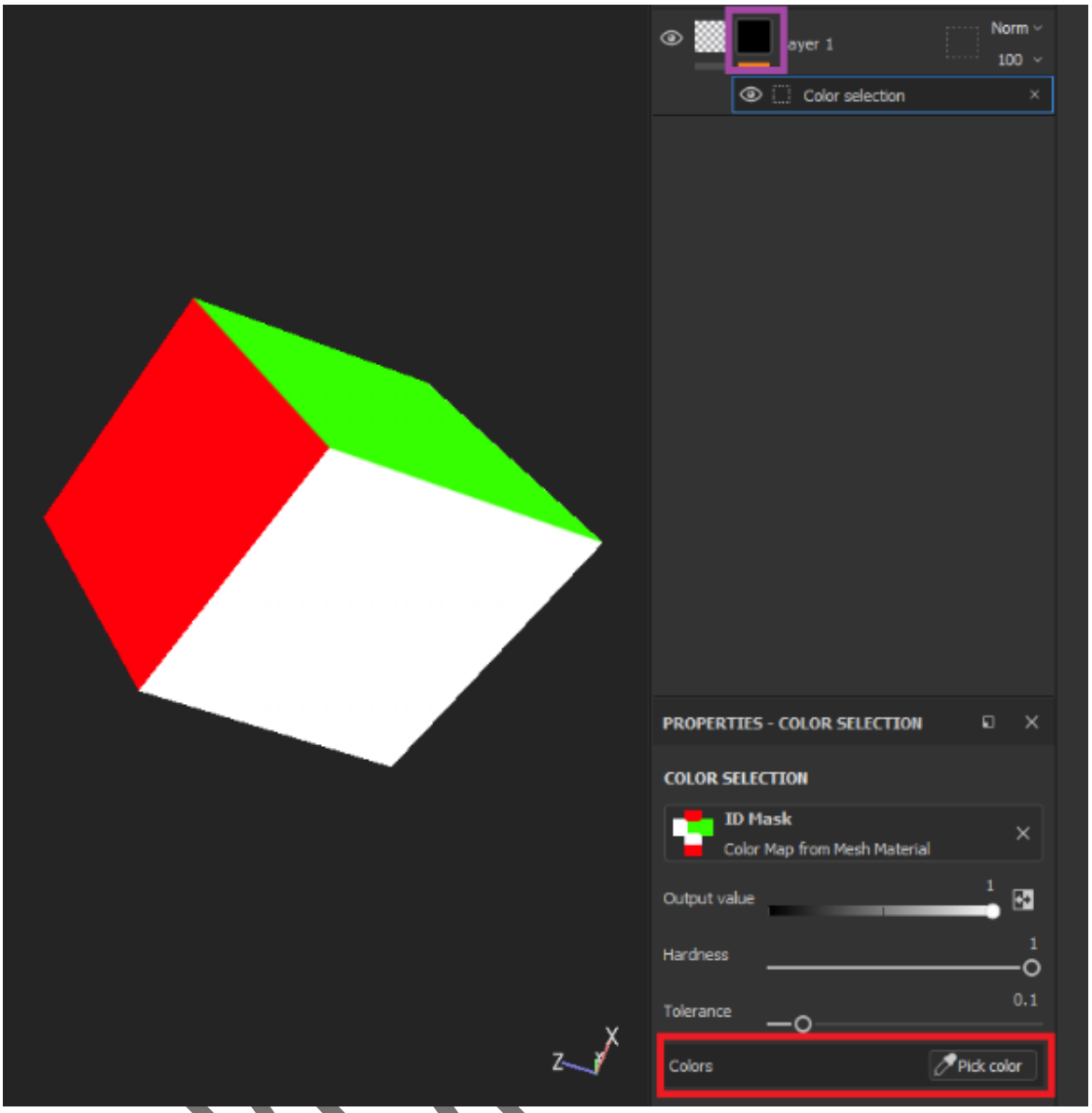


Рисунок 91 - Выбор участка, на который будет назначен материал

Проделав эти операции с тремя участками объекта, substance painter при экспорте текстурной карты объединит все слои на одну карту.

Все это используется для оптимизации ресурсов компьютера и сокращения рабочего времени. Этот способ позволяет подключать на много меньше текстур в игровых движках. Этот метод был применен и для 3D принтера.

Перейдем к непосредственному текстурированию объекта. Возьмем в качестве примера нагревательный элемент 3D принтера. Для него была создана ID карта, изображенная на рисунке 92.

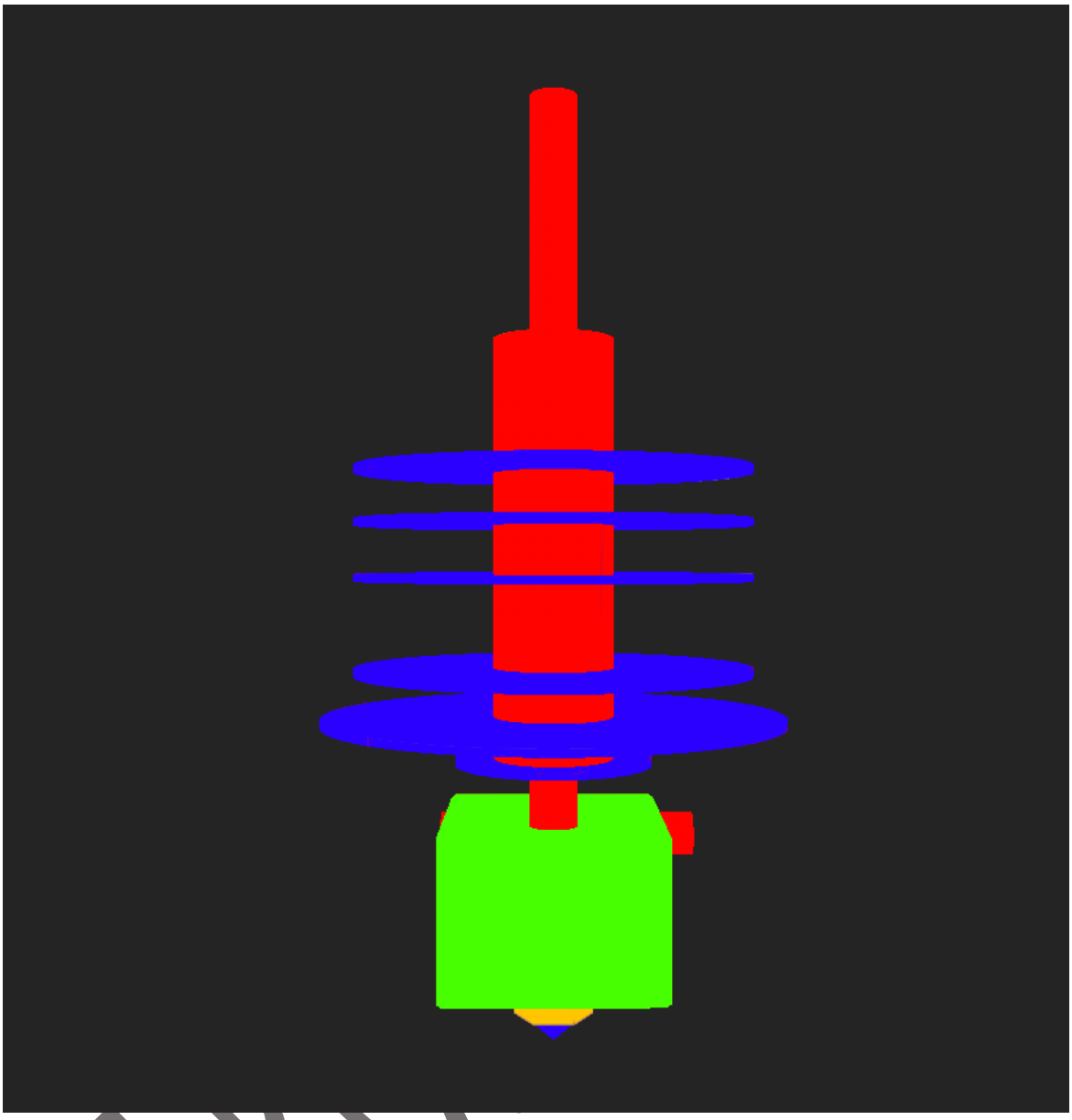


Рисунок 92 - ID карта нагревательного элемента

После того, как объект был импортирован в substance painter, нужно запечь для него текстуры. В **ID** выбираем **Vertex color**.

Далее нужно определиться с материалами. Всего у нас их будет четыре. Для синего цвета ID карты будет создан стальной материал с картой **Roughness**. Для элемента зеленого цвета будет подобран черный материал. Красные цилиндры будут так же стальными, но иметь другой оттенок, с желтой гайкой аналогично.

Начнем со стали. В поиске ассетов введем **iron** или **steel**. Понравившийся материал перетащим левой клавишей мыши в окно layers.

Слои в substance работают так же, как и в photoshop. Их может быть большее количество. Чем выше слой. Тем он главнее. Верхний слой перекрывает нижний. Предположим наличие двух слоев красного и синего

цвета. Если красный цвет расположен выше, значит модель будет покрашена этим цветом.

Каждому слою можно назначать различные свойства, применять маски, сортировать по папкам и много е другое.

Рассмотрим предлагаемые инструменты, применяемые к слоям, изображенные на рисунке 93.

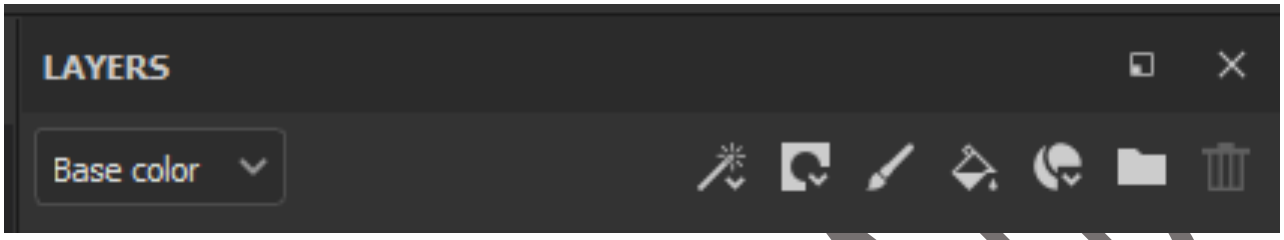


Рисунок 93 - Инструменты, применяемые к слоям

Рассмотрим инструменты справа на лево.

- Мусорка. Позволяет удалить слой;
- Папка. Нужна для того, чтобы создать раздел под слои;
- Таблетка. Добавление заготовленных ассетов в слои;
- Заливка. Она создает слой с однотонным цветом;
- Кисть. Нужна для того, чтобы начать рисовать на слое объекта;
- Маски. Они позволяют настраивать видимость заданных участков материала. Это черно белая карта, в которой черный цвет отвечает за скрытие материала. А белый – за видимую часть;
- Волшебная палочка – это меню с выбором инструментов, которые необходимы слою. Например, можно добавить рисование кистью или заливку слоя через это меню. Так же в нем можно добавить генераторы – это особые инструменты/материалы, которые позволяют накладывать на выбранный материал особые свойства. Примером может служить наложение градиента, наложение карты Ambient Occlusion и многое другое.

Вернемся к текстурированию объекта. После того, как материал металла добавлен, накладываем на него черную маску с возможностью выбрать цвет ID. Далее выбираем необходимый участок. После этого идем искать следующих подходящий материал для других частей объекта. В итоге получаем модель. Изображенную на рисунке 94.

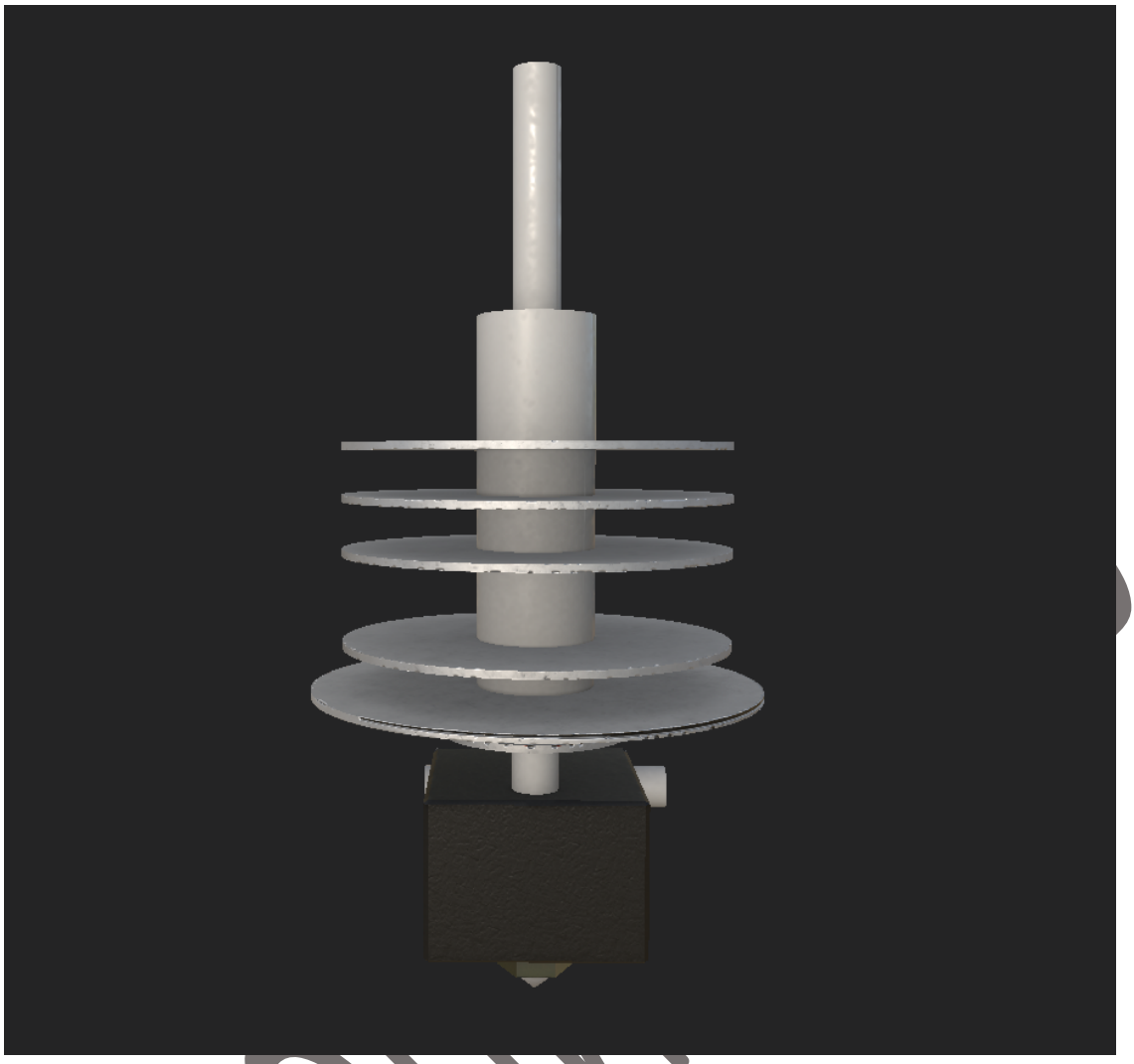


Рисунок 94 - Нагревательный элемент с созданными для него текстурами

После того, как материалы были созданы. Их необходимо экспортировать.

Поговорим о текстурных картах, которые являются отражением качества UV – развертки и без которых процедурные текстуры будут работать некорректно или вовсе не будет работать. К этим картам относиться **Ambient Occlusion** и **Curvature**.

Ambient Occlusion – это черно-белая карта, которая затемняет щели, впадины, отверстия на объекте и осветляет ровные участки. Одним из способов ее применения является дополнение к карте нормалей. Ambient Occlusion создает больший объем там, где его на самом деле нет. Однако это не единственный способ ее применения. Без этой карты создание некоторых процедурных материалов является невозможным.

Curvature - это «карта кривизны», чёрно-белая текстура, на которой яркость пикселя показывает изменение кривизны поверхности.

Обе эти карты запекаются сразу после добавления 3D объекта в сцену Substance Painter.

ГЛАВА 5. Программируем поведение цифровых двойников

5.1. Знакомимся с Unreal Engine

Теперь познакомимся с основной средой, которая позволит нам создавать наши виртуальные тренажёры- с игровым движком Unreal Engine. Основным его преимуществом наряду с Unity и прочими открытыми игровыми движками является его свободное распространение. Замечательно, что UE5 содержит большое число специализированных модулей, предоставляющих разработчику отличный набор инструментов для решения практически любой задачи.

Первое, что вам необходимо сделать, это скачать Epic Games Launcher. При первом запуске появится окно, представленное на рисунке 95

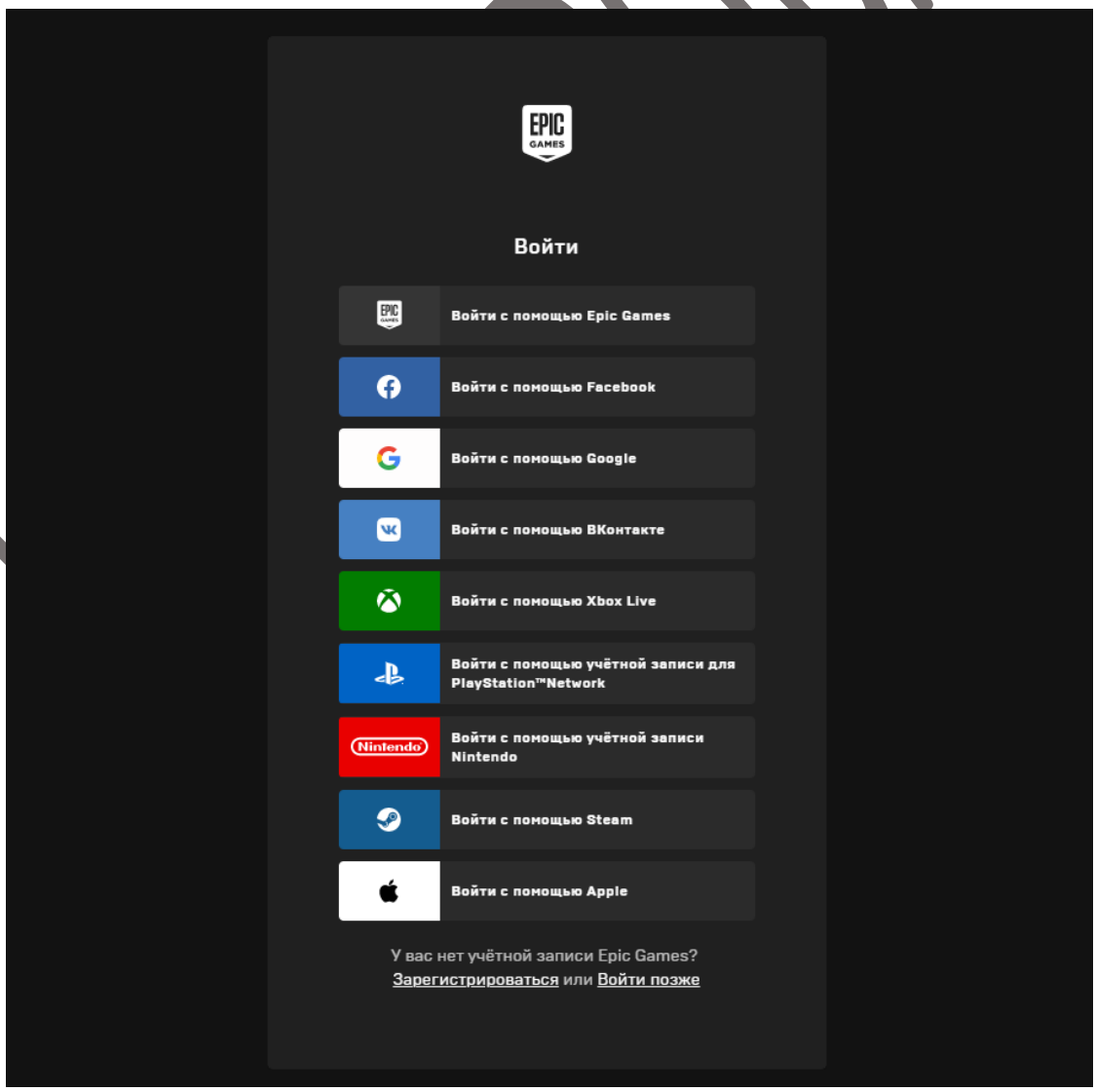


Рисунок 95– Окно авторизации Epic Games

При первом запуске программа предложит обширный список опций создания аккаунта, либо входа при помощи сторонних ресурсов. Подойдёт любой наиболее удобный пункт. После входа в левой части окна вы увидите вертикальную панель (рисунок 96) с тремя позициями в следующем порядке: «Магазин», «Библиотека» и «Unreal Engine». Как нетрудно догадаться, вас интересует нижний пункт.

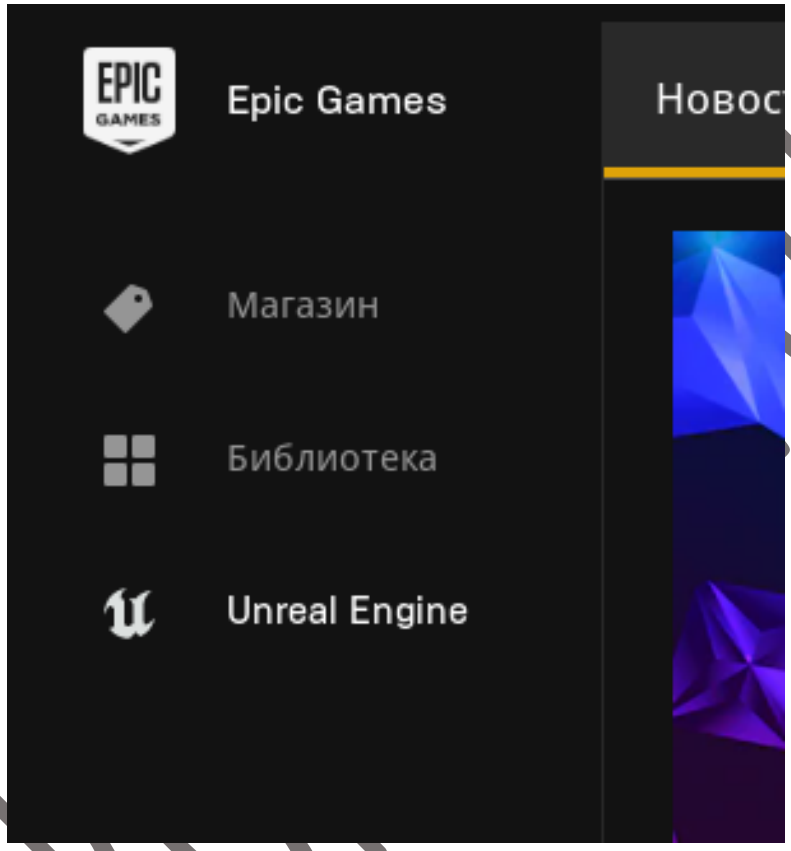


Рисунок 96 – Вертикальная панель в Epic Games Launcher

При переходе вам станет доступна горизонтальная панель сверху, показанная на рисунке 97. Вас интересует четвёртая опция «Библиотека».

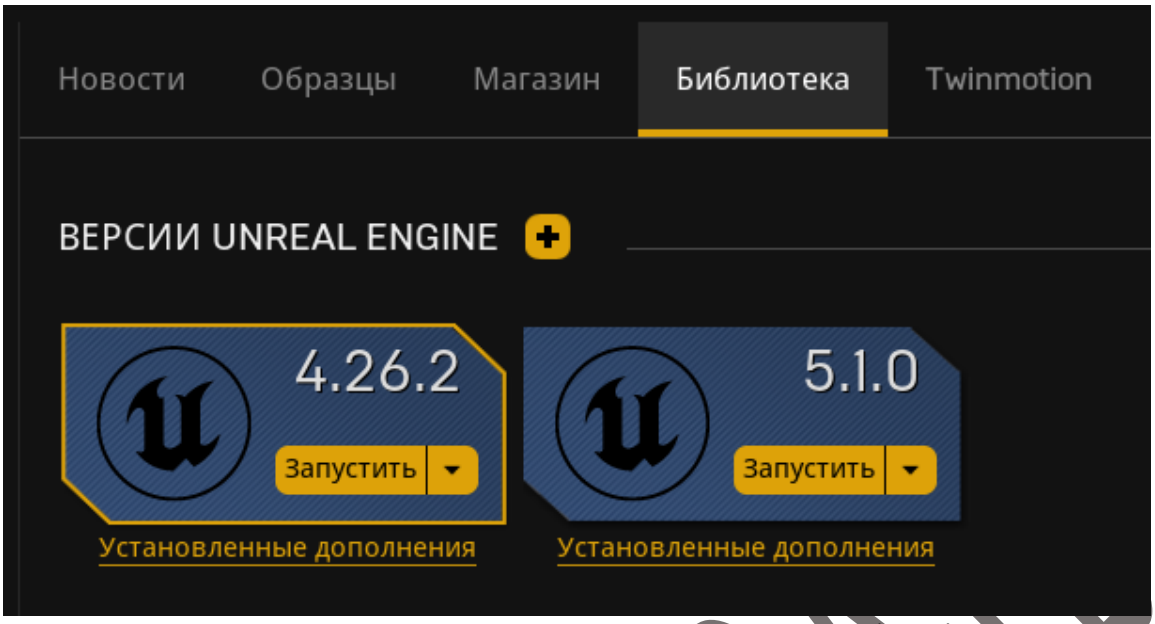


Рисунок 97 – Горизонтальная панель раздела Unreal Engine

Чуть ниже вы увидите чёрный плюс на жёлтом фоне, следующий после надписи «ВЕРСИИ UNREAL ENGINE». По нажатии на значок появится серая панель с манящей кнопкой «установить». В целом можете смело на неё нажимать, однако при желании вы можете установить более ранние версии движка. Для этого требуется нажать на цифры в верхнем правом угле панели и в выпадающем списке выбрать необходимую версию (Рисунок 98).

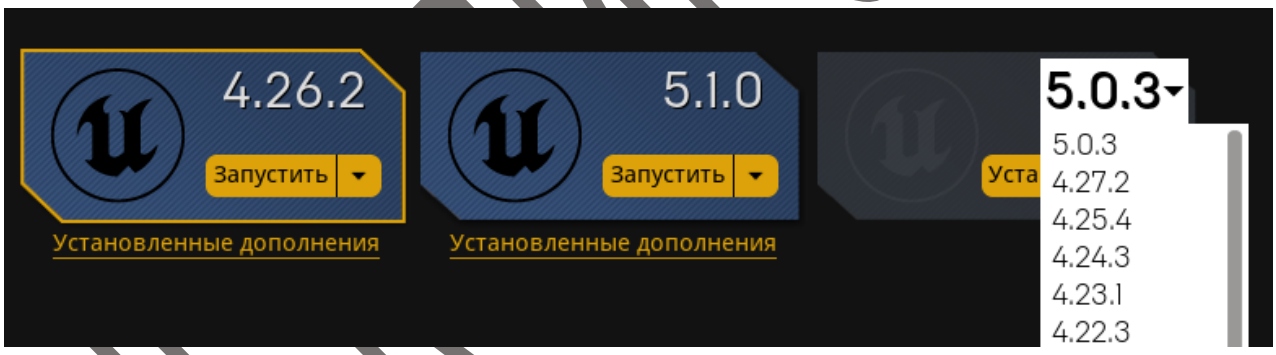


Рисунок 98– Выбор версии Unreal Engine

Текущий материал будет посвящен работе в версии Unreal Engine 5.1.0, последней на момент написания этого текста. Тем не менее показанное здесь будет возможно также воссоздать и в более ранних стабильных версиях.

5.2. Создаём проект

Устанавливаем, запускаем. Первым делом Unreal Engine встретит вас стартовым окном с содержанием, представленным на рисунке 99.

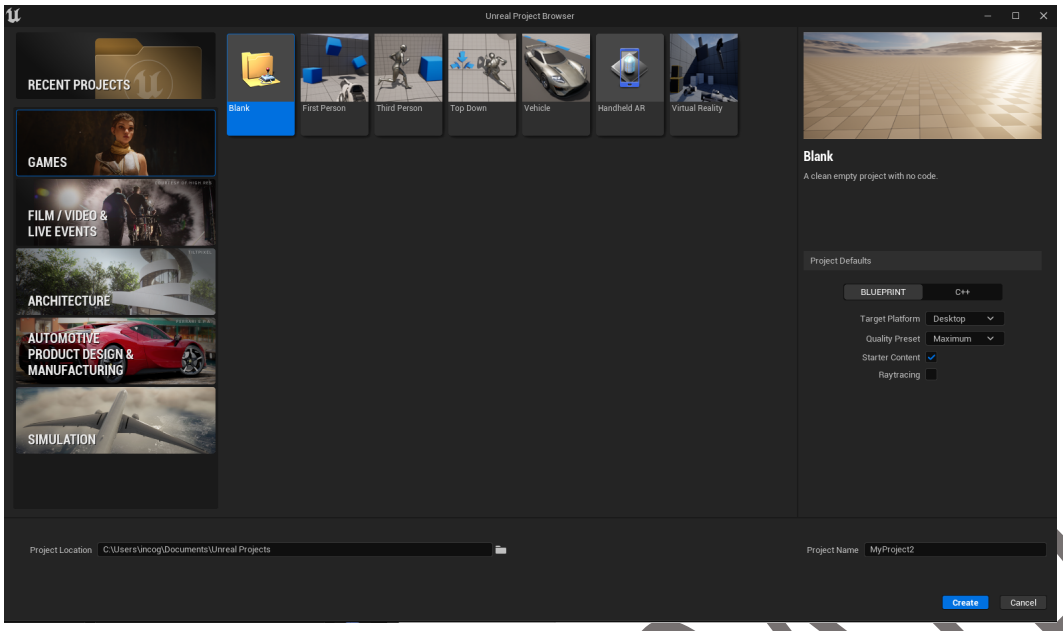


Рисунок 99 – Стартовое окно Unreal Engine

Ни один инженер или программист не любит раз за разом изобретать колесо без особой надобности, и разработчики из Epic Games это понимают. Поэтому они добавили большое количество шаблонов, позволяющих на их основе создавать самые распространённые форматы игр и прочих проектов: от режимов управления персонажем с видом «от первого» и «от третьего лица» до дизайнерских проектов и симуляторов. Кратко пробежимся по основным параметрам создаваемого проекта, которые можно задать при помощи вертикальной панели справа (Рисунок 100).

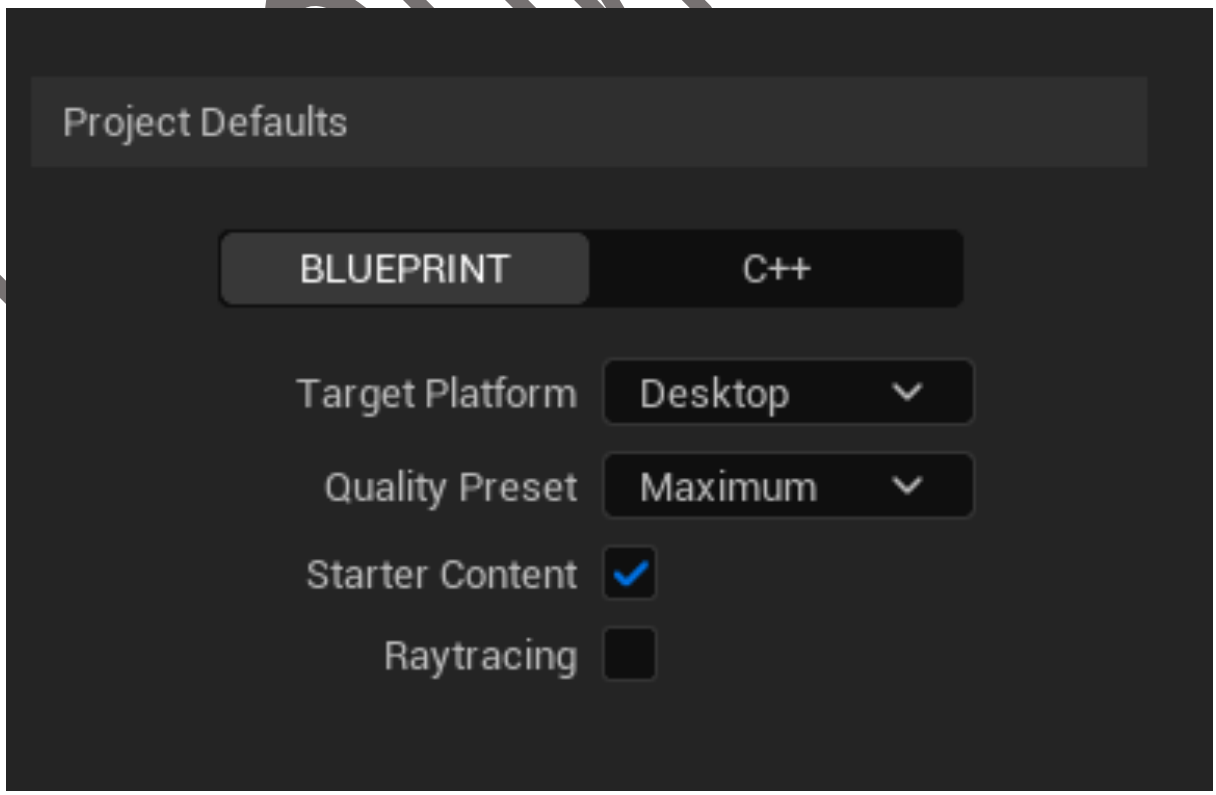


Рисунок 100 – Параметры создания проекта

Для первого запуска рекомендуется выставить параметры в соответствии с примером выше. При необходимости их можно будет поменять позже. Так как мы условились, что будем создавать логику при помощи Blueprints, его и выбираем в первую очередь.

Unreal Engine предоставляет два способа создания логики виртуального мира: язык программирования C++, а также основанный на нём язык визуального программирования Blueprints. Последний во многом создавался с оглядкой на открытость Unreal Engine для широкой публики и потому позволяет быстрее и легче научиться разрабатывать игры. Стоит понимать, что владение C++, да и в целом знание принципов ООП* значительно облегчает процесс обучения, ну и конечно же однозначно является необходимым условием для разработки монументальных AAA проектов. Тем не менее, в чисто прикладном смысле система Blueprints прекрасно справляется со своей задачей, поэтому в этом пособии мы будем пользоваться им.

В окне «**Target Platform**» выбираем опцию «**Desktop**». В окне «Q**uality Preset**» выставим максимальное качество. Оба параметра влияют на характеристики уровня визуализации проекта. Проекты-тренажёры в виртуальной реальности зачастую находятся меж двух огней: с одной стороны, VR является довольно требовательной технологией с точки зрения ресурсов платформы. С другой стороны, для погружения в виртуальную реальность очень важен хороший визуал. В нашем случае высокие настройки качества отображения приведут к более живым теням и освещению, к уменьшению размытия и увеличению четкости картинки, а также позволит раскрыться подготовленным материалам.

Чтобы увидеть различия между настройками, на котором показаны параметры целевого оборудования с соответствующим качеством графики. Как вы можете видеть, настройка масштабируемого качества графики удаляет Auto Exposure и Motion Blur из эффектов постобработки по умолчанию. Она также отключает Anti-Aliasing, когда целевое оборудование настроено на ПК-версию. Однако установка масштабируемого качества графики, когда целевое оборудование является мобильным, просто отключает Bloom и Mobile HDR; автоматическая экспозиция и размытие движения будут отключены, если целевое устройство установлено как мобильное.

Различия в настройках для разных типов устройств показаны на рисунке 101.

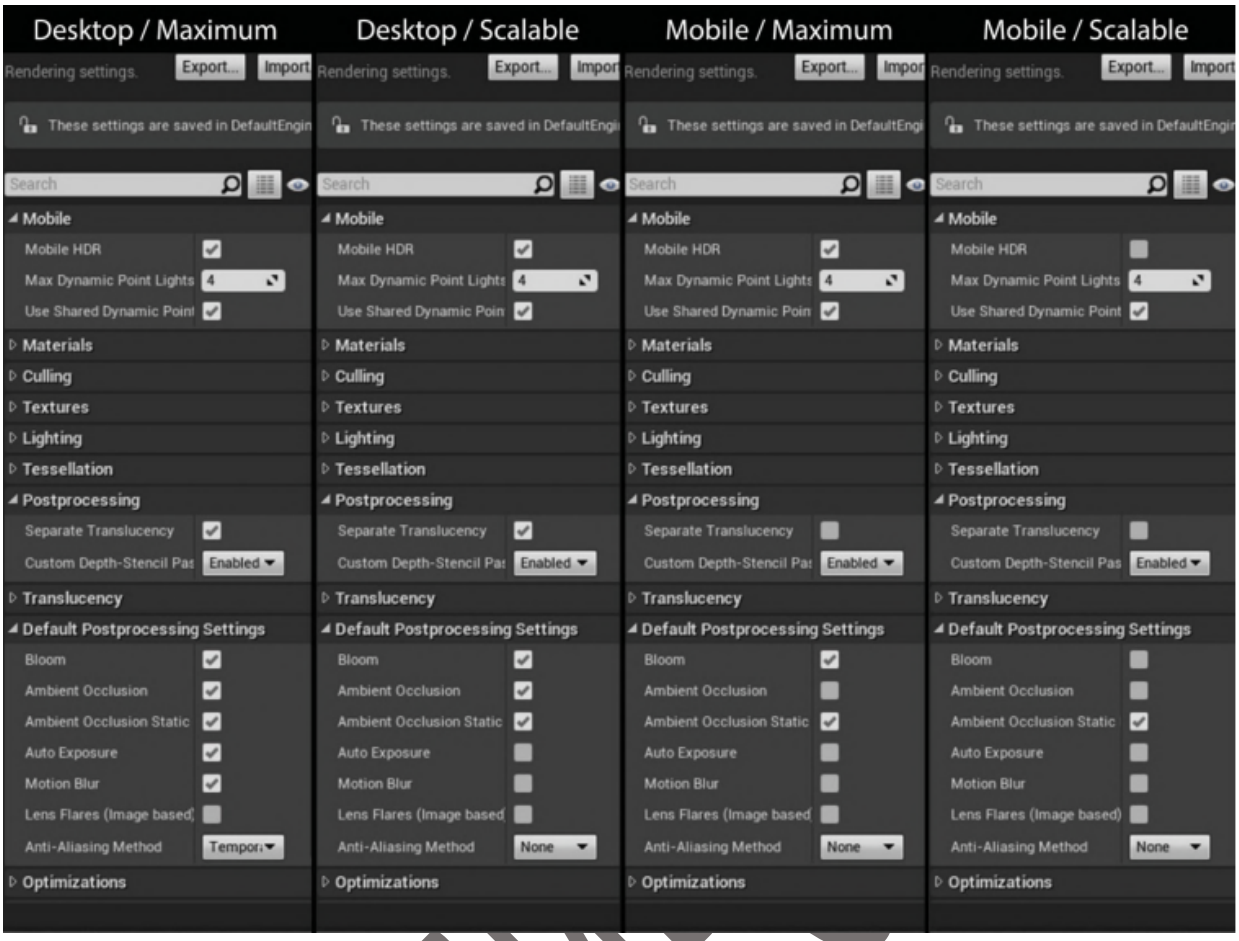


Рисунок 101 – Различные настройки для разных устройств

Следом ставим галку напротив окна «**Starter Content**». Данное окно добавляет в проект некоторое количество уже готовым ассетов из библиотеки Unreal Engine: набор стандартных мешей, текстур и материалов, звуковых и визуальных эффектов. Полезно для размещения оных в качестве «**Placeholder'ов**» («заполнителей», объектов, которые временно используются вместо конечных, но пока неготовых ассетов), а также позволит познакомиться с основными видами игровых объектов. Содержимое «**Starter Content**» всегда можно импортировать заново, поэтому не бойтесь менять свойства ассетов, если хотите поэкспериментировать. Наконец, убирайте галку напротив окна «**Reytracing**». Технология трассировки лучей хоть и является одной из самых совершенных (если не самой) в области имитации освещения, но в то же время она оказывает колоссальную нагрузку на производительность ПК. Да, мы хотим добиться хорошего визуала, но в то же время мы также хотим иметь больше десяти FPS. Выставленных выше высоких настроек графики обычно хватает. Если же в силу каких—то причин вы считаете, что вам всё же нужно добиться физически—корректного освещения, то рекомендуется обратить внимание на пятую версию Unreal Engine, в которой появилась технология «**Lumen**», являющуюся превосходной, менее требовательной альтернативой «**Reytracing**».

В списке шаблонов также присутствует шаблон для создания проектов в виртуальной реальности. Находится он во вкладке Games (или же Simulation) и обычно стоит последним в очереди шаблонов. Отталкиваться будем именно от него. Примечательно, что для этого шаблона в версии 5.1.0 разработчики убрали практически все вышеописанные пункты настроек, оставив только «Starter Content».

Внизу вы можете указать путь хранения проекта, а также его название.

5.3. Интерфейс Unreal Engine

Запустим наш проект. При первом запуске шаблона проекта после небольшой загрузки и компиляции шейдеров в версии движка 5.1.0 нам будет доступна следующая картина, приведённая на рисунке 102.

Рисунок 102 – Окно Unreal Engine при первом запуске

Как и в любом комплексном инструментарии Unreal Engine включает в себя колоссальное количество всевозможных модулей, панелей, настроек и так далее. В рамках этого материала мы воспользуемся лишь некоторыми из них, однако, чтобы лучше ориентироваться сконцентрируем внимание на ключевых элементах. Красной зоной выделена так называемая сцена. Похожую вы могли увидеть в Blender, а также Substance Painter. Отличие в том, что там вы работали с конкретными ассетами. Здесь же мы наконец сможем собрать и увидеть весь проект в целом.

Фиолетовая зона- дерево проекта. Все объекты, размещённые на сцене, также начинают указываться в этой зоне в виде списка. Здесь вы сможете найти конкретный ассет по его названию, а также настроить зависимости и организацию.

При выборе любого объекта на сцене станет доступна оранжевая зона. Это зона свойств конкретного объекта: его положение, скалирование, параметры отображения, физика, коллизия и так далее. При выборе разных ассетов на сцене открывается конкретная зона.

Жёлтая зона, это – «Content Drawer». По умолчанию она будет скрыта, вызвать её можно либо по нажатию на иконку в нижнем левом углу, либо при нажатии сочетания клавиш «Ctrl + Space Bar». Здесь хранятся все ваши «Блупринты», а также все добавленные ассеты в ожидании того, как вы ими воспользуетесь. Рекомендуется не пренебрегать грамотным неймингом и обратить внимание на уже используемое обозначение соответствующих ассетов (Blueprints имеют в своём названии приставку «BP_», Static Mesh- «SM_» и т.д.). Это может оказаться сильно полезным при поиске конкретного файла в крупном проекте.

Вверху, под стандартными для абсолютного большинства профессиональных пакетов набора из настроек программы (file,edit,…), вы сможете найти настройки проекта, выделенные зелёной зоной (Рисунок 103).

Рассмотрим поближе на эту панель на рисунке 4.1.2.2.

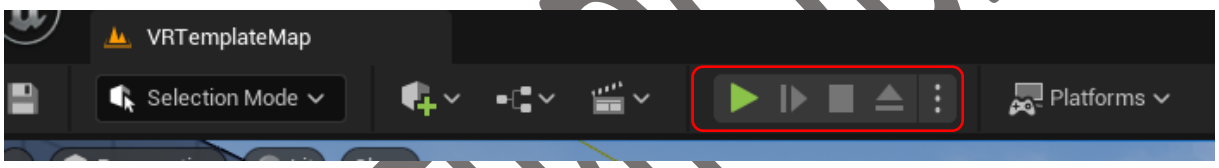


Рисунок 103 – Верхняя панель окна Editor

Здесь вы сможете сохранить свой проект, переключаться между режимами отображения определённой группы элементов на сцене и т.д. Функция, которая интересует нас больше всего, находится в зоне, обозначенной красным цветом. Здесь мы можем запустить наш проект и протестировать логику. По нажатию на три точки появится возможность запуска проекта в режиме симуляции. Таким образом вся логика начнёт функционировать, однако мы сохраним возможность свободного перемещения по сцене вместе с возможностью редактирования. Экономит время при тесте некоторой логики без необходимости постоянно надевать шлем виртуальной реальности.

5.4. Библиотеки ассетов

Любая работа над проектом, связанная с 3D графикой, будь то создание игры, создание ролика, делится на несколько этапов.

Первым является подбор референсов. Он заключается в поиске различных примеров. Примеры могут быть в виде картинок, видео, в виде настоящих предметов. Как правило, чем больше примеров, тем лучше, так как они позволяют хорошо изучить, понять, как можно создать объект, на что следует обратить больше внимания, а на что меньше.

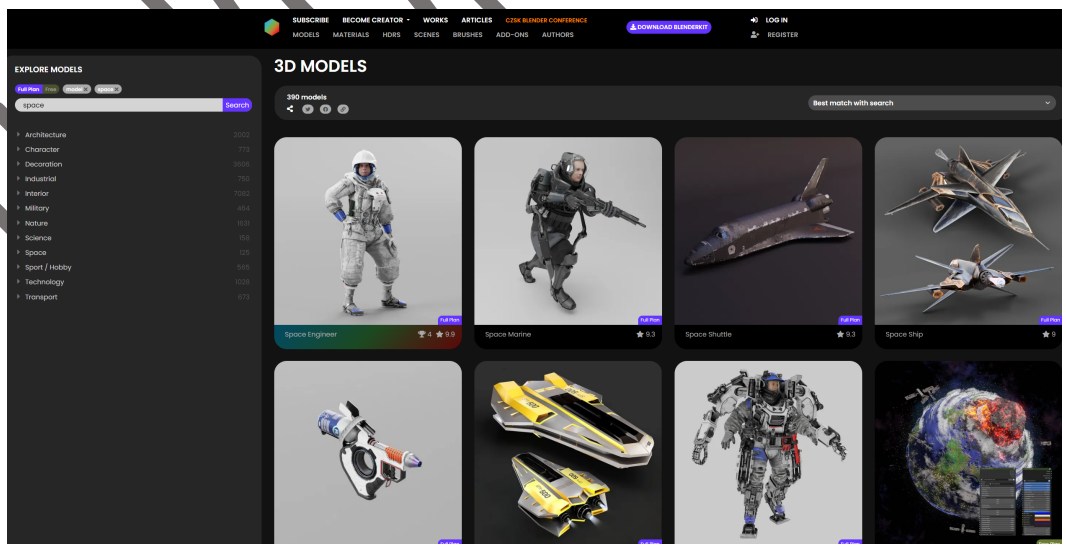
Для 3D- принтера референсами служили фотографии настоящего принтера, 3D- модели различных принтеров, а также видео с подробным описанием каждого элемента и его назначения. Для этого проекта это очень важно, так как нужно как можно точнее приблизить цифровую копию к оригиналу.

Вторым этапом, зачастую, является поиск или создание 3D моделей. Этот этап является не менее важным, так как при правильном подходе к нему, можно экономить много времени.

Все дело в том, что 3D модели можно делать полностью с нуля или искать готовые решения. И как раз первый вариант может быть не рациональным. Почему? В качестве примера, представим, что в проекте необходим 3D принтер (картинка). Он содержит в себе множество деталей. Его полное создание – это нелегкий и продолжительный процесс, который будет рассмотрен ниже. В качестве второго примера можно взять какой-либо элемент природы, например, дерево. Оно имеет сложную форму, фактуру и является сложным объектом для моделирования. В этих случаях стоит вначале рассмотреть готовые решения. В точности нужного объекта может и не оказаться, но он может требовать минимальных доработок.

Существуют такие понятия как ассеты и коллекции. Это наборы готовых материалов. К ним может относится все что угодно: музыка, картинки, видео, текстуры, модели и многое другое. Как правило, каждый ассет и коллекция имеют определенную тематику, то, что их объединяет. Например, для видео коллекции это может быть интро с разным оформлением текста.

Далее ассет и коллекция будут объединены одним понятием – библиотека. Примеры изображены на рисунке 104.



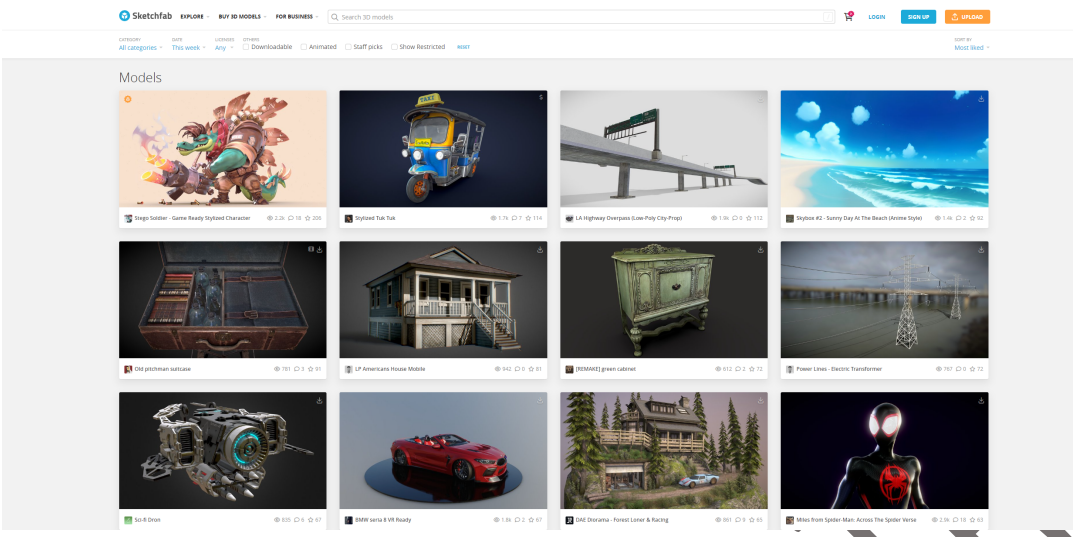


Рисунок 104 – Библиотеки ассетов Blenderkit (сверху) и Sketchfab (снизу)

Нас будут интересовать библиотеки 3D моделей и текстурных карт.

Готовые решения можно найти практически на все: персонажи, бытовая техника, транспорт, объекты местности, различный рельеф и многое другое.

Готовые модели можно находить на сайтах, которые на этом специализируются.

Список сайтов:

- https://sketchfab.com (Sketcfab)
- https://grabcad.com (GrabCad)
- https://quixel.com (Quixel)
- https://substance3d.adobe.com/community-assets (библиотека Substance)
- https://www.blenderkit.com (Blender kit)

Каждый сайт может ориентироваться на определенный формат моделей. Например, Sketcfab ориентируется на полигональные модели, а GrabCad на модели САПР (точные модели). Это стоит учитывать, так как это так же влияет на возможную доработку модели (тут ссылка на Гелено описание доработки модели). Существуют источники, котоые ориентируются под определенную программу, например, Blender kit или библиотека Substance. Их ориентир под определенную программу может задавать ограничения при использовании материалов в других программах.

5.5. Импорт ассетов

Рассмотрим экспорт и импорт в рамках компьютерной тематики.

Экспорт – это выгрузка чего-либо из какой-либо программы в определенном формате.

Импорт – это загрузка чего-либо в программу в определенном формате.

Существуют собственные и универсальные форматы файлов.

Так как мы рассматривает 3d тематику, то примером может служить следующий случай. Предположим, что нам нужно перенести модель из Компас 3D в blender. Файл «Компаса 3D» имеет расширение .m3d, а blender имеет расширение .blend. Эти два формата файлов не совместимы и относятся только к собственным программам, поэтому такое название имеют и их расширения – «собственные». Из такой ситуации есть выход. Существуют расширения, которые способны читать несколько программ. Именно такие форматы файлов называются универсальными. Компас 3D и blender могут «общаться» между собой через расширение .stl.

Программа может иметь несколько собственных и универсальных расширений. Например, blender может экспортировать модели в форматах .dae, .abc, .stl, .obj, .ply, .fbx, .glb/glbf, .x3d, а импортировать в тех же форматах + .svg, .bvh.

Приведем таблицу с несколькими форматами 3d объектов.

Формат 3D файла	Тип
Stl	Универсальный
Obj	ASCII вариант универсальный, бинарный - собственный
Fbx	Собственный
Collada	Универсальный
3ds	Собственный
Iges	Универсальный
Step	Универсальный

Каждый формат имеет свои особенности и может содержать в себе разные данные, которые хранятся либо в текстовом документе, либо в бинарном файле.

Опишем несколько расширений.

STL – формат, предназначенный для полигональных моделей. Он способен хранить в себе только данные о координатах вершин, по которым строятся полигоны, а также данные о нормалях этих полигонов.

OBJ – формат, дополняющий STL тем, что способен хранить в себе информацию о текстурах.

STEP – формат, предназначенный для точных моделей. Эти модели строятся не полигонами, а формулами поверхностей. Главный акцент тут на построении модели. Информация о материалах и анимации не так важна, поэтому не во все САПР программы это переносится.

FBX – содержит в себе информацию о вершинах полигональной модели, нормалей полигонов, текстур, а так же данные об анимации.

Из вышесказанного можно сделать вывод. При работе над 3d объектами и их текстурами важно понимать, как будет переноситься файл из одной программы в другую и в каком формате. От этого зависит подход к созданию объектов.

5.6. Шаблонная логика

Давайте немного потренируемся. Рассмотрим несколько примеров реализации разнообразной логики. Для начала откроем Unreal Engine и запустим шаблон от первого лица. Выставьте все настройки согласно показанному на рисунке 105. Место хранения проекта и его название можете использовать свои. Крайне рекомендуем использовать в указании пути и названия только английскую раскладку.

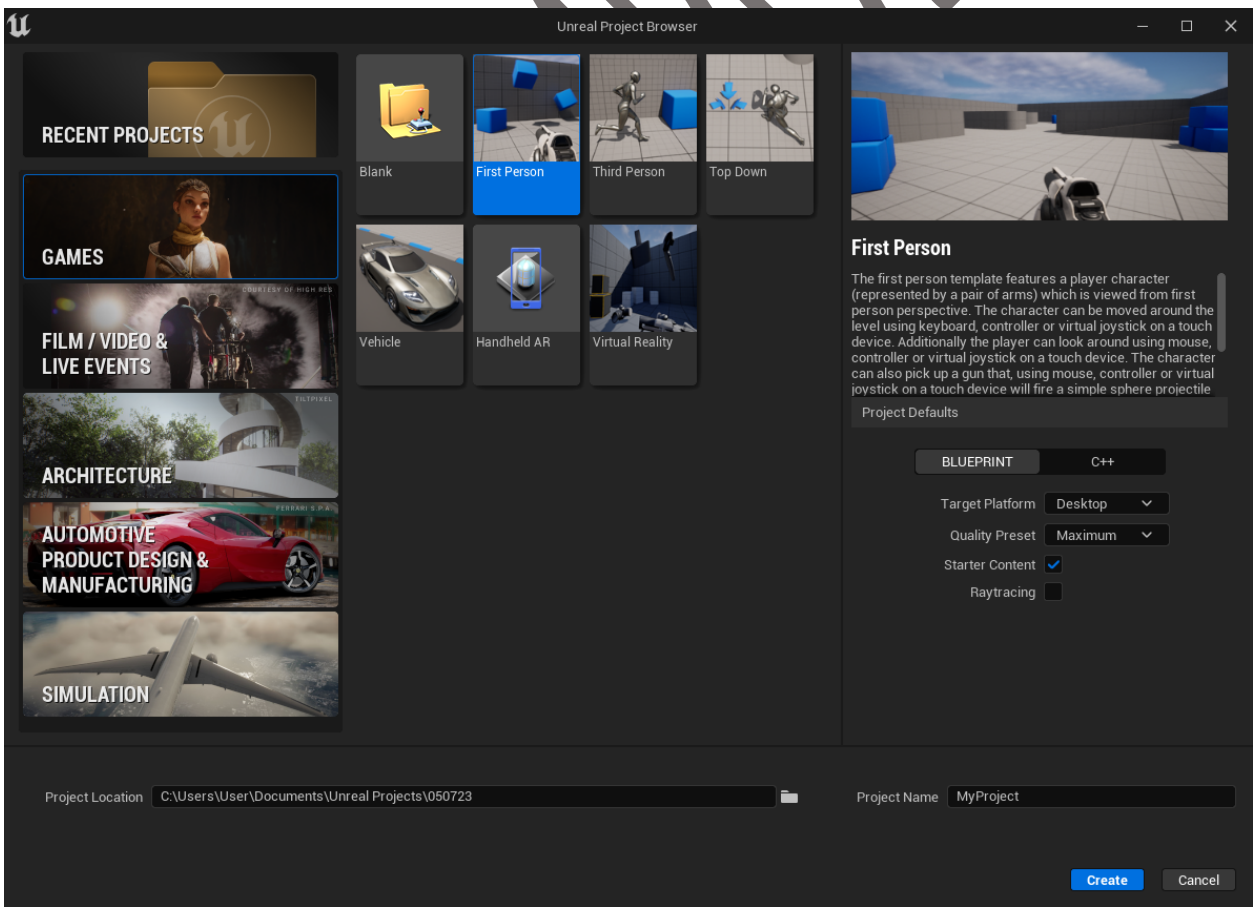


Рисунок 105 – Запускаем проект с шаблоном от первого лица

По нажатию на зеленую стрелочку, показанная на рисунке106 вы можете протестировать имеющуюся логику.

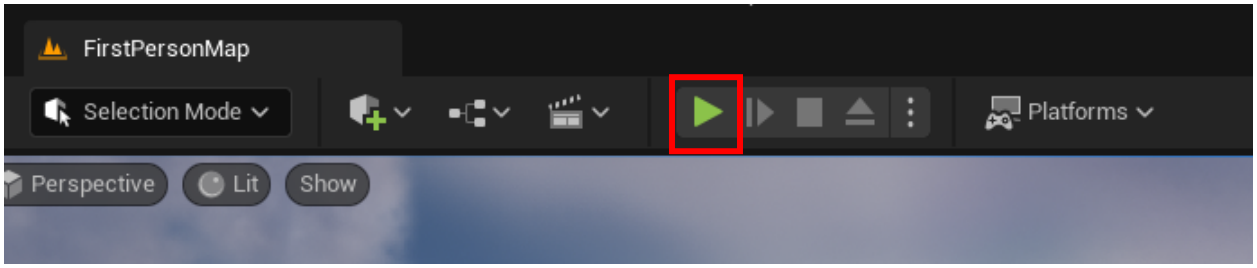


Рисунок 106 – Кнопка запуска билда

По умолчанию игрок может передвигаться, поднимать оружие и производить стрельбу.

5.7. Запуск события по кнопке.

Реализуем интерактивный объект «кнопка». Так как методическое пособие посвящено разработке с использованием технологий виртуальной реальности, то и способ реализации мы сделаем специфический.

В представленном примере нажатие на кнопку должно приводить к открытию двери.

При нажатии комбинации клавиш **Ctrl + Space** откроется «контент браузер»- специальное окно, в котором хранятся все добавленные и созданные ассеты, а также логика. Создадим отдельную папку, в которой и будем хранить всё, что будем добавлять. Для этого необходимо нажать правую кнопку мыши в пределах поля и выбрать соответственную опцию, как показано на рисунке 107. Назовём папку «MyLogic».

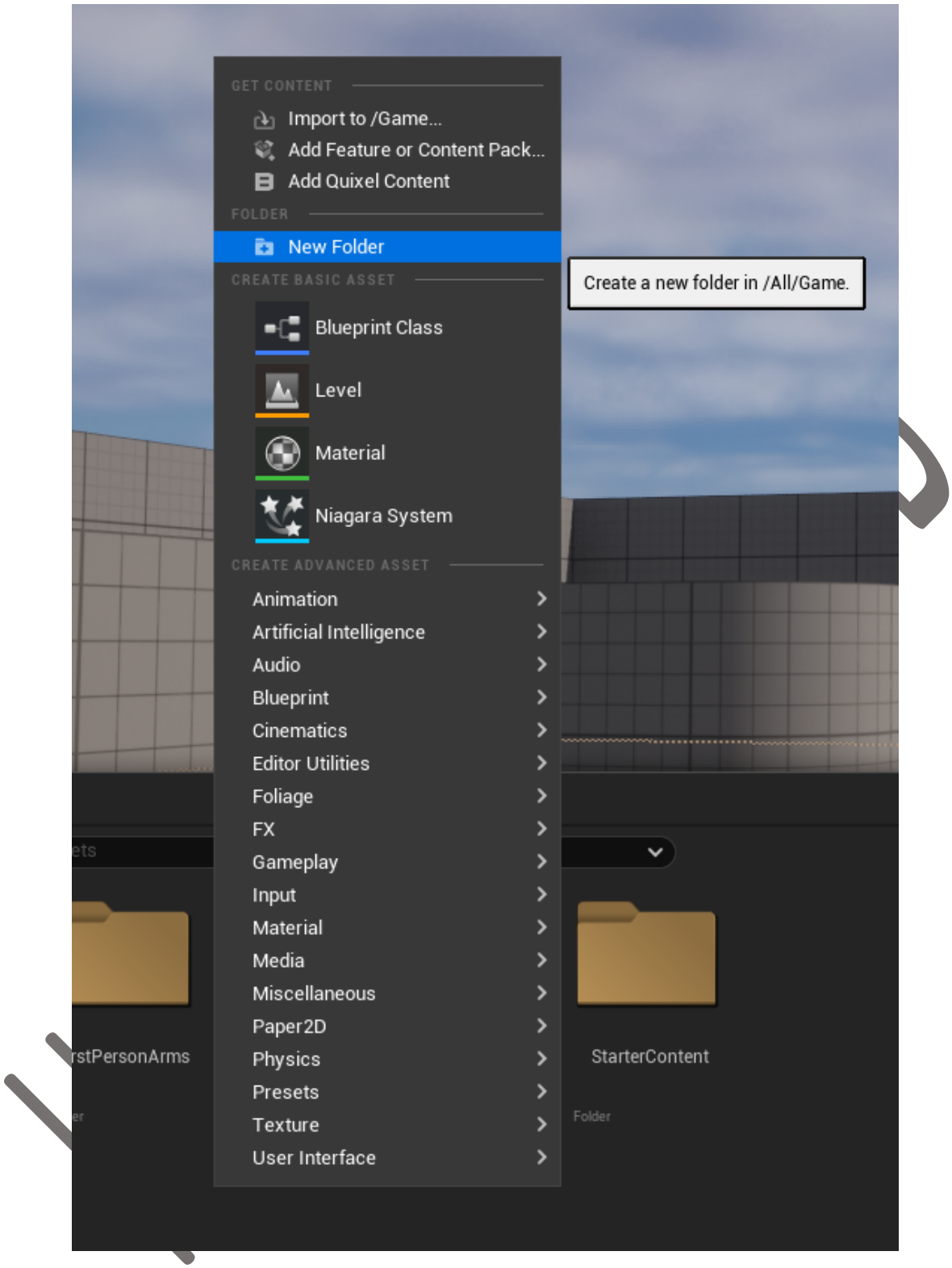


Рисунок 107 - Создаём папку

Внутри создаём ещё одну папку, которую назовём «**ButtonLogic**». В дальнейшем рекомендуем создавать такие папки для каждого последующего примера. Теперь необходимо создать файл, в котором будет находиться сама кнопка, дверь, которую она открывает и логика, ответственная за это. Для этого создадим объект, который называется «**Actor**». На рисунках 108 и 109 показан требуемый порядок действий.

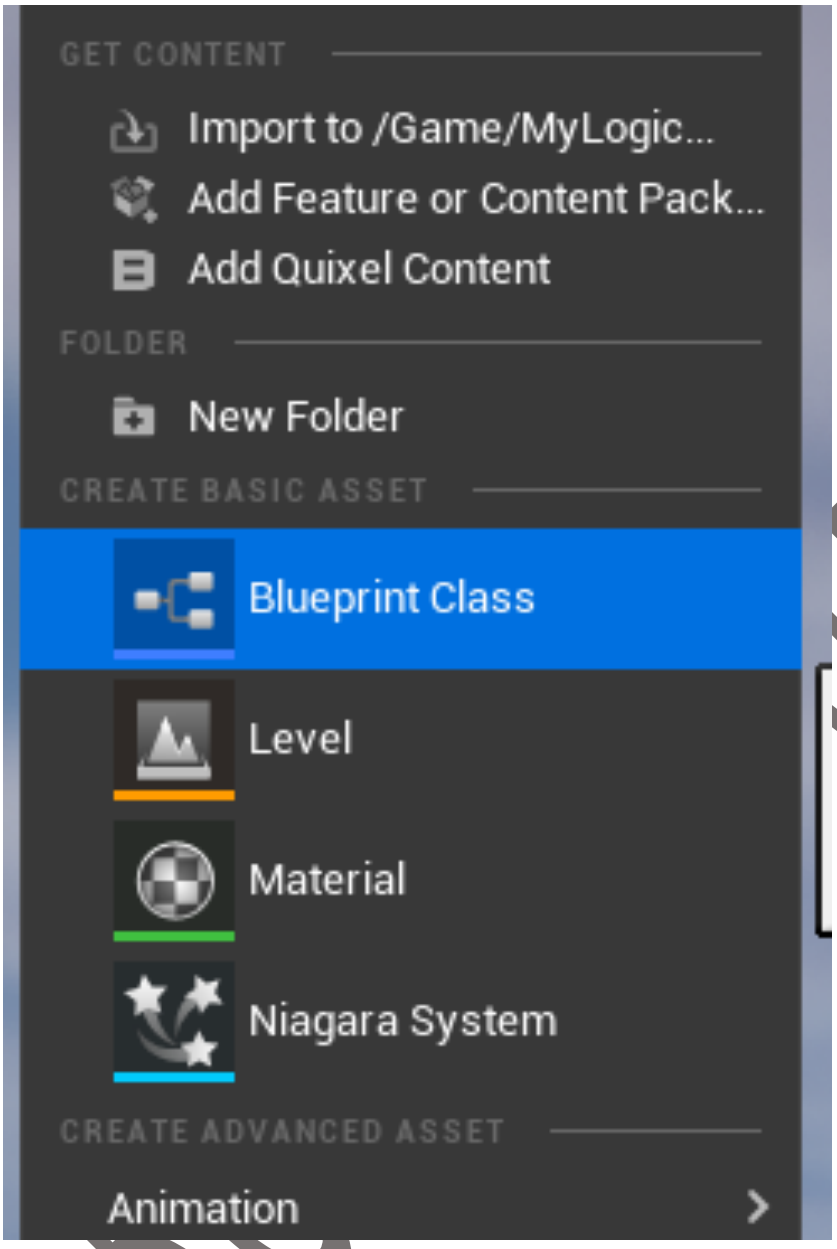


Рисунок 108 - В начале создаём Blueprint Class

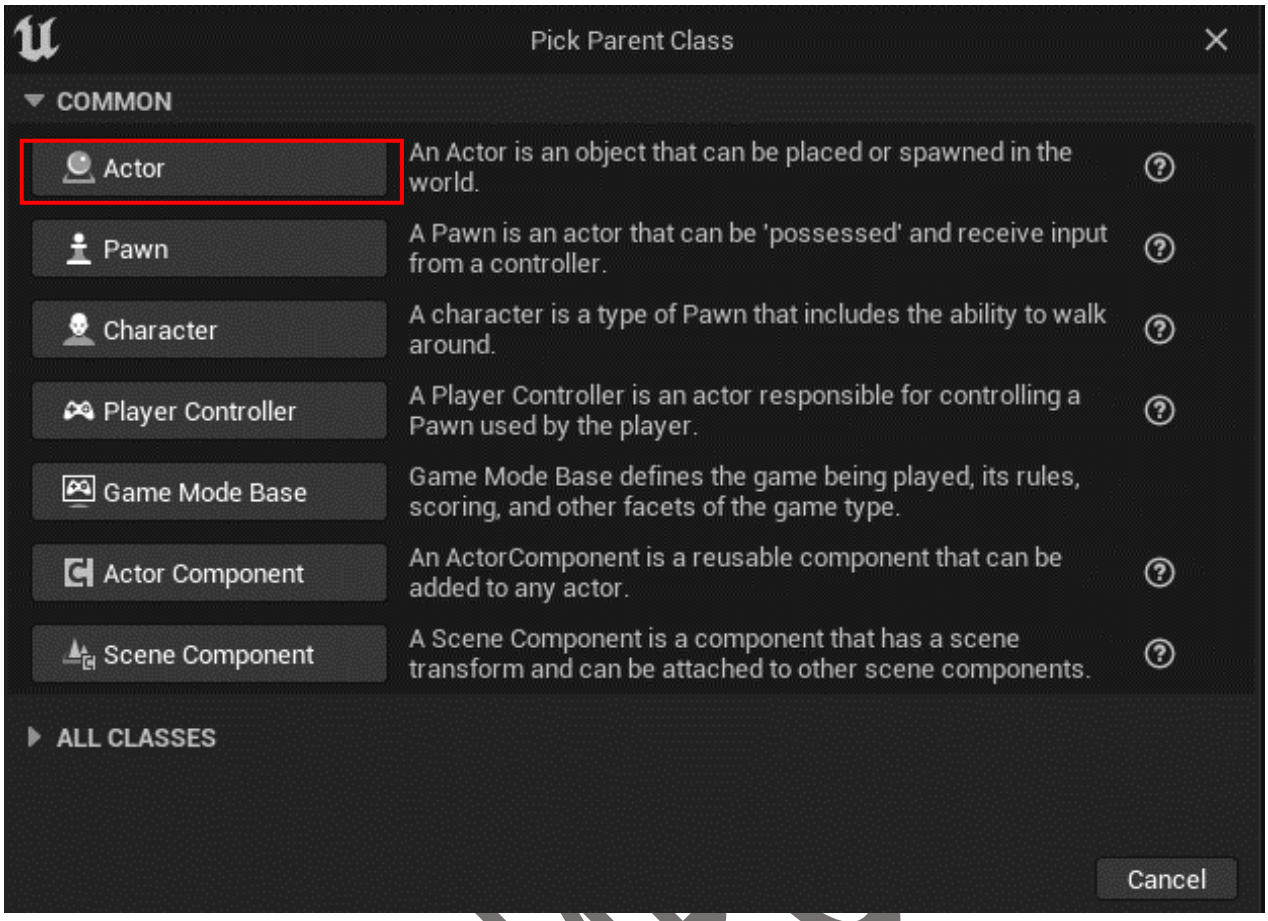


Рисунок 109 - В появившемся окне выбираем опцию «Actor»

Назовём его «**BP_Button**» (BP сокращённо от Blueprint) и откроем. Перед нами откроется панель редактирования этого класса. Для начала познакомимся с интерфейсом, показанном на рисунке 110.

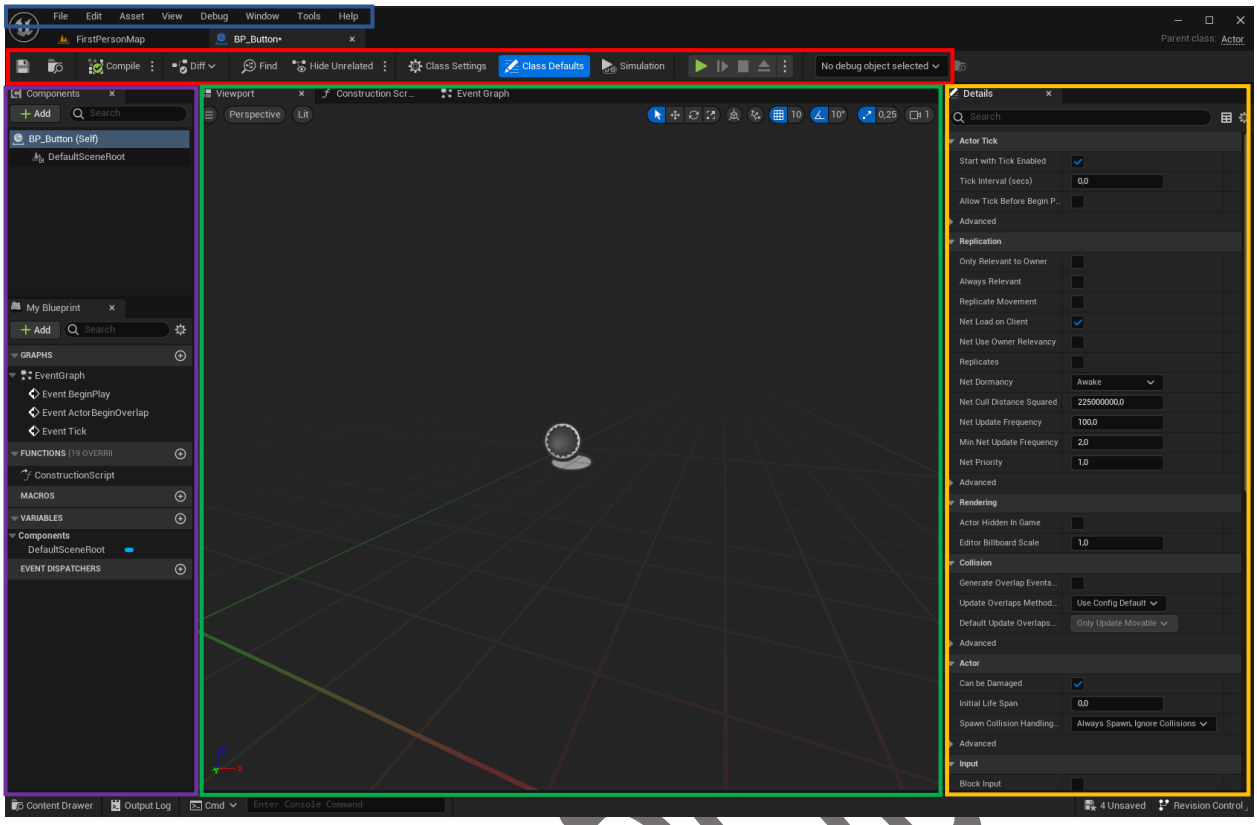


Рисунок 110 – Интерфейс созданного Actor

Интерфейс во многом напоминает оный в главном окне Unreal Engine.

В зоне, выделенной синим цветом, находится панель управления файлом Эктора. Здесь вы можете сохранить файл, настроить отображение, показать или скрыть отдельные окна и так далее.

Красным выделена зона, позволяющая настроить свойства класса, запустить локальную симуляцию и в целом проверить работоспособность логики. Отдельное внимание стоит уделить кнопке «**Compile**», необходимой для компиляции кода. Её необходимо нажимать после каждого изменения в классе.

Фиолетовую зону можно охарактеризовать как хранилище компонентов. Вверху под шапкой «**Components**» присутствуют все используемые в классе ассеты, будь то 3D-объекты, в том числе скелетные, зоны коллизий и многое другое. Чуть ниже находится панель Эвентов (в оригинале «**Events**», события). Наконец здесь же можно найти макросы, и созданные переменные. Всё это нужно для того, чтобы прописывать логику, актуальную для каждого из этих объектов.

Желтым выделена зона свойств каждого объекта. К примеру, если в дереве вы добавите какой-нибудь **Static Mesh**, то при его выборе, справа появятся все настройки, связанные с этим объектом: координаты, угол поворота, материал и так далее.

Наконец, зеленым выделена сцена. По аналогии с главным экраном здесь можно расположить все составляющие эктора. Единственное отличие- все эти объекты будут относиться в первую очередь к этому эктору, а уже во вторую к основному уровню. Вверху зоны располагается 3 вкладки: Viewport, в которой мы и находимся в данный момент, **Construction Script** и **Event Graph**. К ним мы ещё вернёмся, а пока что давайте добавим саму кнопку и дверь.

Для этого первым делом обратимся к панели компонентов, а именно к верхней части фиолетовой зоны. По нажатию на кнопку **Add** появится выпадающее окно, как показано на рисунке 111.

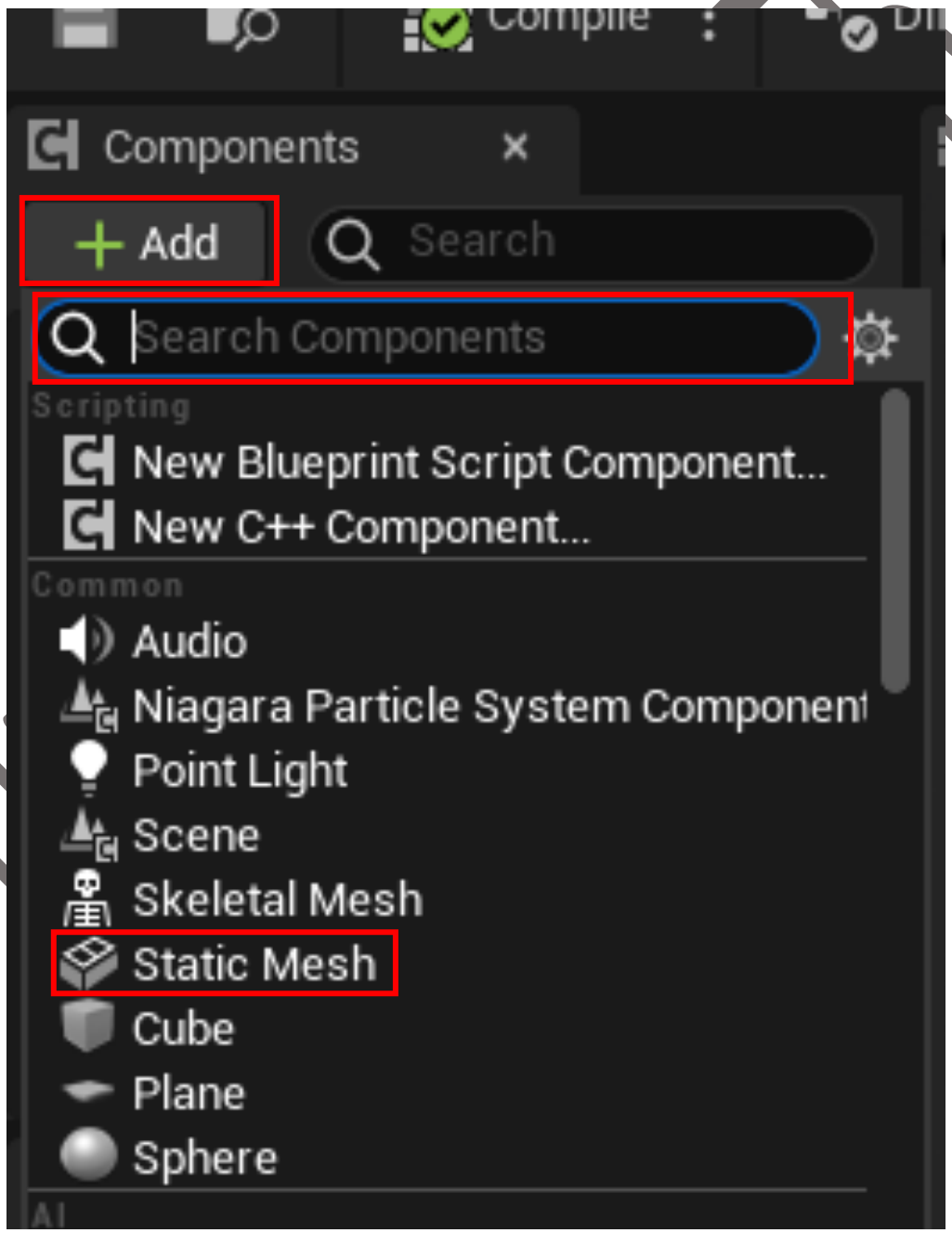


Рисунок 111 - Панель компонентов

В поле **Search Components** необходимо вбить Static Mesh (из всего предложенного списка нужно выбрать тот, который обведён на рисунке .). Данную процедуру нужно провести три раза, либо же, как вариант, можно скопировать имеющийся пока что пустой Static Mesh два раза. Переименуем их для понятности. Чтобы это сделать, нужно кликнуть на каждый Static Mesh левой кнопкой мыши и нажать **F2**, либо же нажать правой кнопкой мыши и выбрать опцию **Rename.** Назовём их **Door, DoorFrame** и **Button** соответственно. Если всё сделали правильно, должна образоваться картина, показанная на рисунке 112.

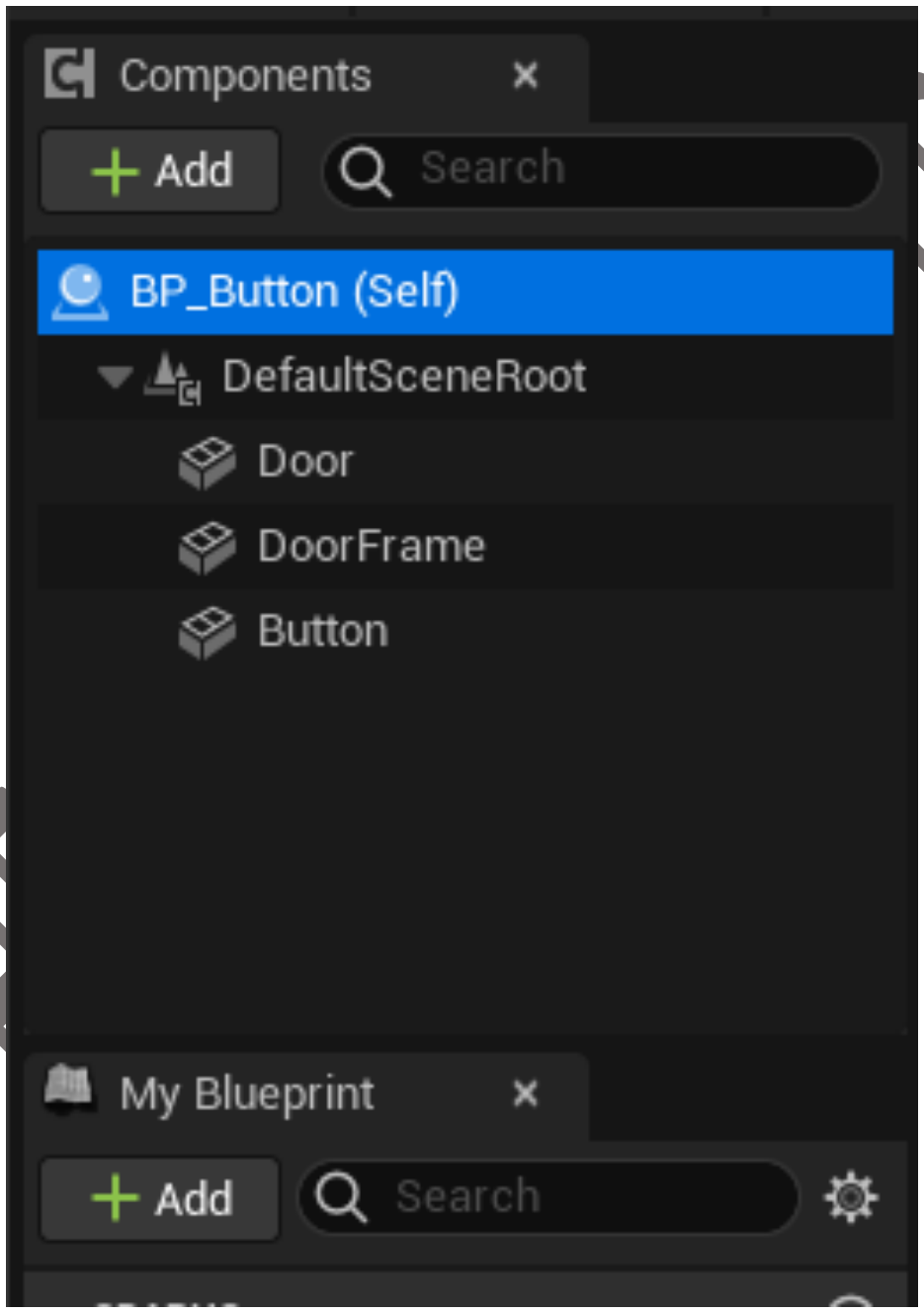


Рисунок 112 - Пример корректного выполнения

Теперь нам нужно добавить сами модели объектов. Начнём с двери. Для этого нажимаем левой кнопкой мыши по созданному нами мэшу с названием

Door. В окне свойств (желтая зона на рисунке) появятся свойства этого мэша. Нас интересует позиция «Static Mesh», показанная на рисунке

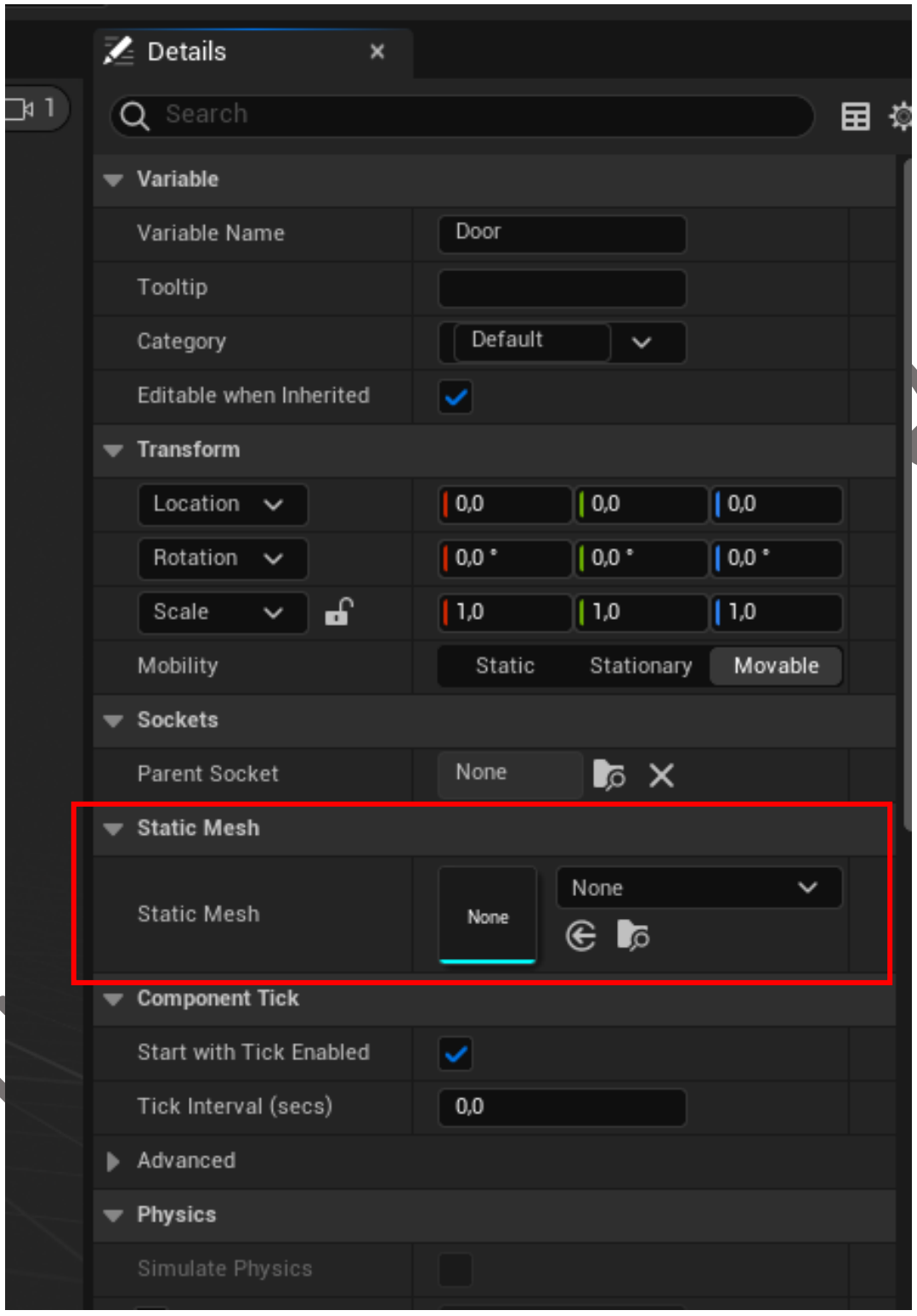


Рисунок 113 - Позиция Static Mesh в поле Details.

Добавим сюда мэш двери, который имеется в **Starter Content**. Нажмём на пустое поле None, со стрелочкой. В выпадающем списке впишем **SM_Door** (здесь SM это сокращение от Static Mesh) и выберем предлагаемую позицию. Модель двери появится на сцене, как показано на рисунке 114

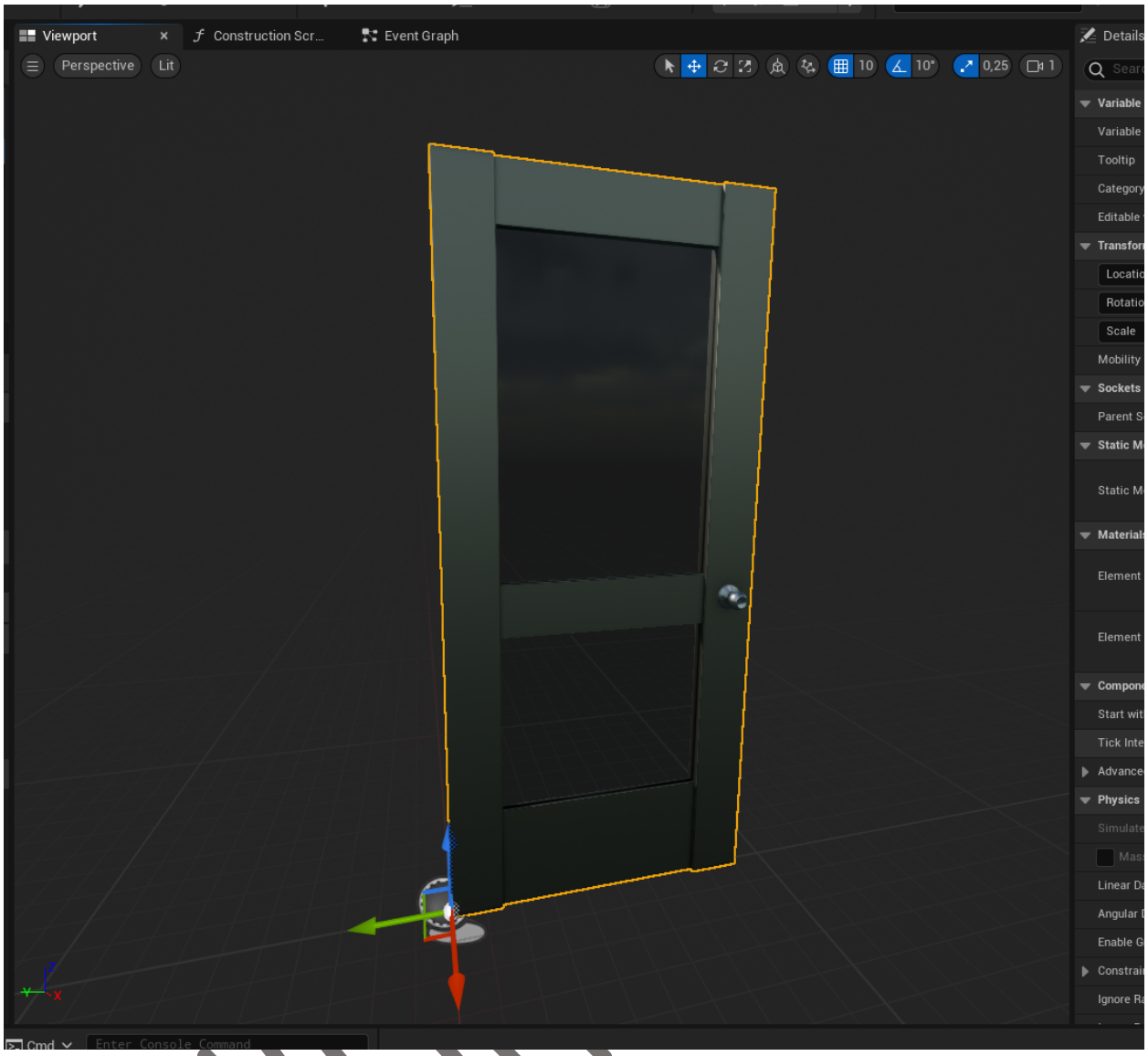


Рисунок 114 - Успешно размещённая дверь

Обратите внимание, что дверь появилась уже с материалом. Повторим процедуру для дверного проёма. Нажимаем в дереве на созданный нами **DoorFrame** и в окне **Static Mesh** в панели **Details** ищем SM_DoorFrame, после чего выбираем.

Осталось только добавить кнопку. С ней мы немного схитрим. Добавим вместо мэша примитив- цилиндр. Для этого кликаем на button и в уже знакомом поле **Static Mesh** выбираем SM_Cylinder. Если всё сделано правильно, вы будете наблюдать картину, изображенную на рисунке 115.

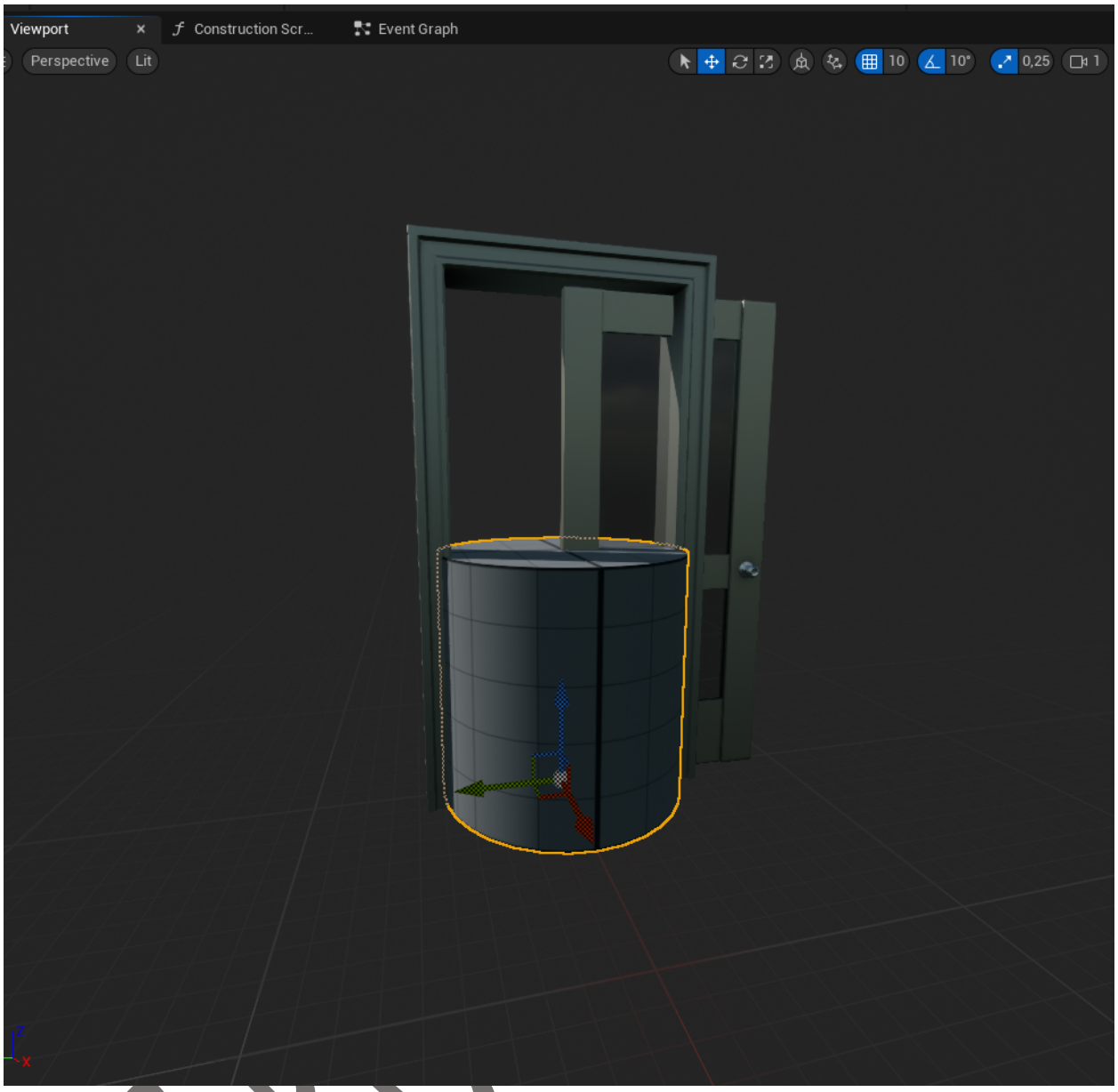


Рисунок 115 - Все три объекта на сцене

Давайте теперь разместим объекты как полагается. Внесите следующие параметры в свойства мэшей, как показано на рисунках 116-118.

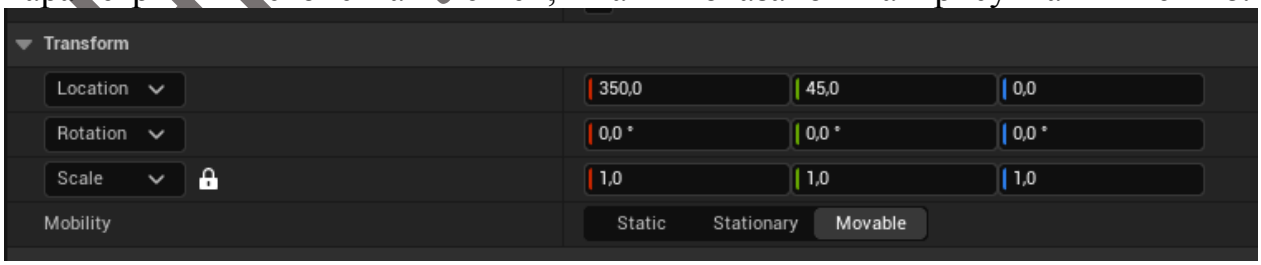


Рисунок 116 - Настройки для двери (Door)

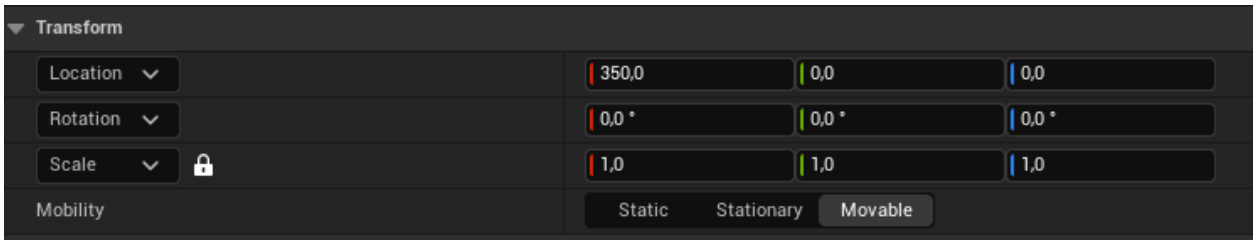


Рисунок 117 - Настройки для дверного проёма (DoorFrame)

Рисунок 118 - Настройки для кнопки (Button)

Объекты на сцене должны расположиться, как показано на рисунке 119.

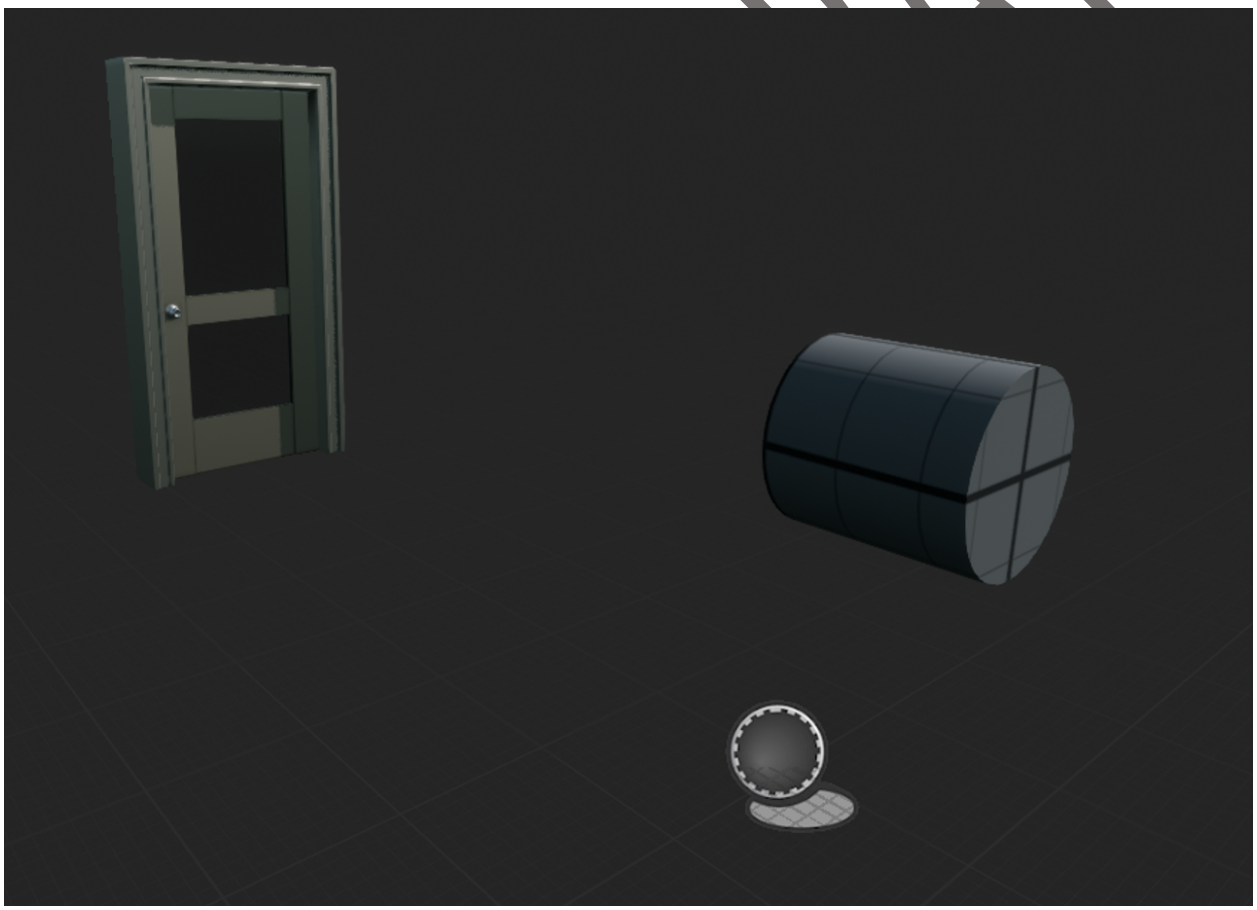
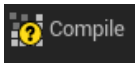


Рисунок 119 - Итоговое размещение объектов на сцене

Наконец, начнём добавлять логику. Однако первым делом добавим компонент, называемый зоной коллизии (**collision zone**). Чтобы это сделать, нажмите в зоне компонентов на кнопку **Add** и выберите **Box Collision**. При желании можете добавить зону коллизии любой другой формы. Выставьте настройки положения и ориентации согласно показанному на рисунке 120.

Рисунок 120 - Настройки для зоны коллизии (в дереве Box по умолчанию)

Наконец, нажмите кнопку 

для сохранения результата. Теперь, перейдём во вкладку **Event Graph.**

Здесь необходимо реализовать логику взаимодействия персонажа с кнопкой в соответствии с показанным на рисунке 121. Выйдет так, будто персонаж толкает кнопку.

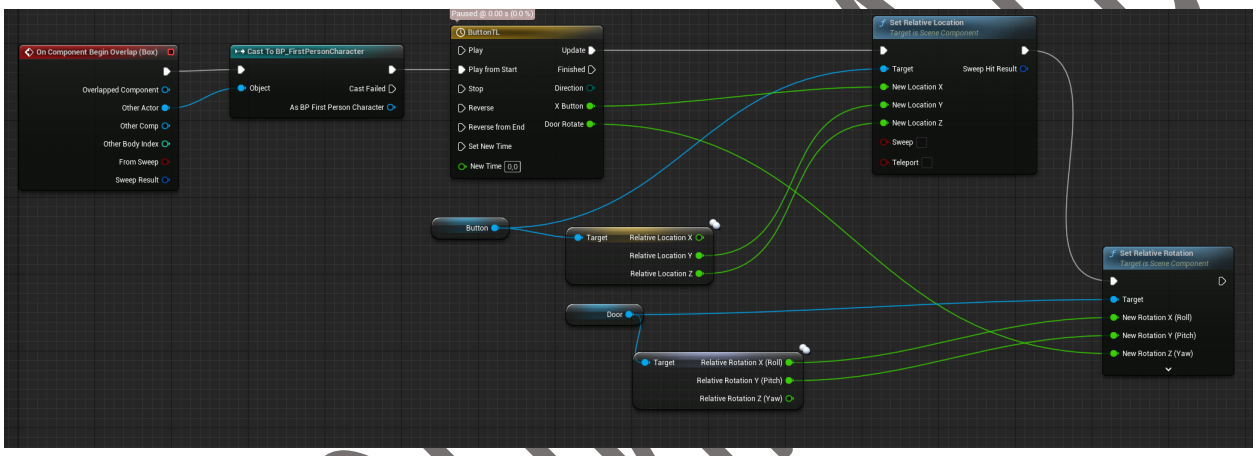


Рисунок 121 - Итоговая логика

В случае реализации полноценного проекта в виртуальной реальности, вы сможете выполнить аналогичную логику, однако вместо ноды персонажа будет использована нода контроллера. При желании или необходимости вы также можете убрать зону коллизии, добавить физику и заменить **Event Begin Overlap** зоны коллизии на **Event On Hit** у кнопки. Осталось только вытащить эктор на сцену и протестировать.

5.8. Свет в Unreal Engine

5.8.1.Статичный свет (Static lights)

Статические источники света — это источники света, которые нельзя изменить или переместить на протяжении всего геймплея. Тени, падающие от таких источников, вычисляются (запекаются) перед началом игрового процесса и хранятся в картах освещения (**Lightmaps**), после обработки они используют мощности устройства.

Статические источники не взаимодействуют с подвижными (**Movable**) объектами и создают тени только с объектами, которые так же, как и источник являются статичными.

Из всех типов источников света статические имеют среднее качество прорисовки, наименьшую изменчивость и используют наименьшее количество ресурсов устройства, поэтому основной вариант использования статического освещения - для устройств с низким энергопотреблением, например, на мобильных платформах.

Рассмотрим такие настройки статичных источников света как радиус источника (**Light Source Radius**) и разрешение карт освещения (**Lightmap Resolution**).

Как упоминалось ранее, радиус источника света для статических источников света имеет дополнительный эффект смягчения отбрасываемых ими теней. На рисунках 122а и 122б приведены источники с разными радиусами.

Рисунок 122 – Статичный источник света:

а – с небольшим радиусом; б – с большим радиусом

На рисунке 122а источник имеет радиус равный 0 и отбрасывает более резкие тени, в то время как на рисунке 122б источник света имеет радиус равный 7 и поэтому отбрасывает более мягкие тени.

Разрешение карт освещения (**Lightmap Resolution**)

Разрешение карт освещения позволяет контролировать детализацию запеченного освещения, создаваемого статическими источниками света.

В объектах **StaticMeshComponents** разрешение световых карт устанавливается либо в самом **StaticMesh**, либо путем установки флажка **OverrideLightmapRes** и установки определенного значения. Большие значения, с одной стороны, позволяют добиться большей степени детализации теней, которые падают на объект, но с другой они дольше обсчитываются и занимают больший объем памяти.

На рисунках 123а и 123б приведены примеры теней, падающих на объекты типа **StaticMeshComponents** при разных значениях разрешения световых карт.

Рисунок 123 – Разрешение карт освещения:

а – 128; б – 512

На Рисунке 123а значение разрешения установлено равным 128, в то время как на Рисунке 123б значение равно 512.

На поверхностях типа **Brush** разрешение световых карт устанавливается с помощью свойства **LightmapResolution**. Для данных объектов это настройка плотности тени, поэтому более низкие значения позволяют добиться большей детализации.

5.8.2.Стационарный свет (Stationary lights)

Стационарные источники — это источники света, которые остаются неподвижными, но их свойства, такие как их яркость или цвет, могут меняться в процессе геймплея. Это основное отличие от статического освещения. Используя данный тип освещения важно помнить, что изменение свойств источника влияет только на прямой (падающий) свет, непрямое освещение (отраженный свет) при этом никак не изменится.

Все непрямое освещение и отбрасываемые тени от стационарных источников света предварительно рассчитываются и сохраняются в картах освещения. Прямые тени сохраняются в картах теней (**Shadowmap**). Эти источники света используют свойство **DistanceFieldShadows**, это означает, что их тени могут оставаться четкими даже при довольно низком разрешении карт освещения на объектах, куда падает тень.

Из трех вариантов освещения стационарные источники, как правило, имеют самое высокое качество, среднюю изменчивость и используют среднее количество ресурсов устройства.

Прямое освещение (**Direct Lighting**)

Прямое освещение стационарных источников света обрабатывается динамически с использованием отложенного освещения и затенения (**Deferred shading**). Это позволяет изменять параметры источника освещения. Свет будет иметь высокое качество аналитического отражения, как и подвижный источник. Прямое освещение может быть показано или скрыто в игре путем изменения свойства **Visible**.

Прямые тени (**Direct Shadowing**)

Просчет теней в реальном времени приводит к значительным затратам ресурсов устройства. Рендеринг полностью динамического источника света с тенями часто обходится в двадцать раз дороже, чем динамический источник света без теней. По этой причине стационарные источники света могут создавать статические тени (Static Shadowing) на статичных объектах.

Статические тени (**Static Shadowing**)

На непрозрачных объектах

Lightmass генерирует карты теней поля расстояний (**Distance Field Shadow Maps**) для стационарных источников света на статичных объектах во время перестройки освещения. Такие карты обеспечивают очень точные переходы теней даже при низком разрешении и с очень небольшими затратами при обсчете. Как и карты освещения, **DistanceFieldShadowMaps** требуют уникальных UV-разверток на всех **StaticMesh**, использующих статическое освещение.

Статические тени могут быть просчитаны не более чем для 4 пересекающихся стационарных источников света, поскольку источники света должны быть назначены разным каналам текстуры **Shadowmap**. Это связано с проблемой раскраски графа, поэтому из-за топологии часто допускается менее 4 перекрытий. Как только будет достигнут предел канала, дополнительные стационарные источники света будут использовать динамические тени всей сцены с высокой производительностью.

На прозрачных объектах.

Прозрачность также использует малое количество ресурсов устройства при использовании стационарных источников света. **Lightmass** предварительно вычисляет карту глубины тени из статической геометрии, которая применяется к прозрачным объектам во время геймплея. Эта форма затенения довольно грубая и изменяет затенение только в масштабе метров.

Пример необработанного изображения и изображения, полученного после подобранных параметров приведены на рисунках 124а и 124б соответственно.

Рисунок 124 – Примеры изображений с:

а – непроницаемыми тенями; б – прозрачными тенями полученными от статического направленного источника

Разрешение статической карты глубины тени настраивается при помощи найстройки **StaticShadowDepthMap**, **TransitionSampleDistanceX** и **StaticShadowDepthMapTransitionSampleDistanceY** в **BaseLightmass.ini** с величиной, установленной по умолчанию равной 100, что означает один тексель на каждый метр.

Динамические объекты (такие как **StaticMeshComponents** и **SkeletalMeshComponents** со свойством **Mobility**, установленным на **Movable**) должны интегрироваться в статические тени мира с помощью **DistanceFieldShadowMaps**. Каждый подвижный объект создает две динамические тени от стационарного источника света: тень, падающая на объект, и тень, отбрасываемая объектом. При достаточном большом

количестве динамических объектов более эффективно использовать подвижный (**Movable**) источник света.

Направленные стационарные источники света являются особенными, потому что они поддерживают тени всей сцены с помощью каскадных карт теней одновременно со статическими тенями. Это очень полезно, например, на сценах с большим количеством анимированной листвы; вы хотите, чтобы вокруг игрока были движущиеся тени, но не хотите платить за множество каскадов, чтобы охватить большой диапазон обзора. Динамические тени на большом расстоянии превращаются в статические тени. Чтобы настроить это, необходимо изменить свойство **Dynamic Shadow Distance** направленного стационарного источники на диапазон, в котором вы хотите, чтобы происходило изменение теней.

Подвижные компоненты по-прежнему будут создавать тени для каждого объекта даже при использовании каскадных карт теней при направленном освещении. Такое поведение полезно при небольших **Dynamic Shadow Distances**, но приводит к ненужным затратам при больших расстояниях. Чтобы отключить тени от каждого объекта и сохранить производительность, необходимо отключить свойство **Use Inset Shadows For Movable Objects** на источнике света.

Стационарные источники сохраняют свое непрямое (отраженное) освещение в картах освещения точно так же, как статический свет. Непрямое освещение не может быть изменено в процессе геймплея путем изменения яркости и цвета, как это может сделать прямое освещение. Это означает, что даже если флажок **Visible** источника света снят, его непрямое освещение будет помещено в карту освещения во время построения освещения. Интенсивность непрямого освещения на источнике света может использоваться для масштабирования или отключения косвенного освещения от данного источника света во время построения освещения.

Однако существует настройка пост-обработки, называемая **IndirectLightingIntensity**, которая позволяет масштабировать вклад карты освещения для всех источников света, который может быть изменен в процессе геймплея из Blueprint.

В версии 4.9 или более поздней версии Unreal Engine у стационарных направленных источников появилась новое свойство затенения, расположенная в настройках **Lightmass** под названием **Use Area Shadows for Stationary Lights**.

Чтобы включить это свойство, в разделе **Lightmass** направленного источника света необходимо включить **Use Area Shadows for Stationary Lights**. Если она включена, стационарный источник света будет использовать тени области для предварительно вычисленных карт теней. Область теней — это тени, которые становятся мягче, чем дальше они находятся от источника

освещения. На Рисунках 125а и 125б показана разница между двумя методами затенения.

Рисунок 125 – Use Area Shadows for Stationary Lights:

а – включено; б – выключено

На рисунке 125а свойство включено и тени от стула получаются более мягкими, чем на рисунке 125б (если подобрать параметры упомянутые ранее, можно добиться очень реалистичной картинки).

5.8.3.Movable lights

Подвижные источники освещения отбрасывают полностью динамичный свет и тени. Почти все их свойства такие, как положение, поворот, цвет, яркость, спад, радиус могут быть изменены во время геймплея. Свет от таких источников не запекается в световых картах, также по умолчанию для данного типа освещения не просчитывается отражения без метода динамического глобального освещения.

Количество ресурсов, затрачиваемых при использовании данного типа освещения, определяется количеством объектов, на которые падает свет и количества полигонов на этих объектах. Это означает, что подвижный источник света большого радиуса будет затрачивать во много раз больше ресурсов, чем такой же источник меньшего радиуса.

Смещение теней (**Shadow Biasing**)

Настройка свойства **Shadow Bias** (смещение теней) источников света уменьшает артефакты от теней, которые могут возникать из-за методов отображения теней при использовании динамического освещения.

Смещение теней настраивается для каждого источника света с помощью свойств **Slope Bias** (Смещения наклона) и **Shadow Slope Bias** (Смещения наклона тени). Регулировка смещения наклона тени пропорциональна смещению тени, и с помощью этих двух свойств можно уменьшить некоторые артефакты отображения тени, которые могут возникнуть как показано на рисунках 126а и 126б.

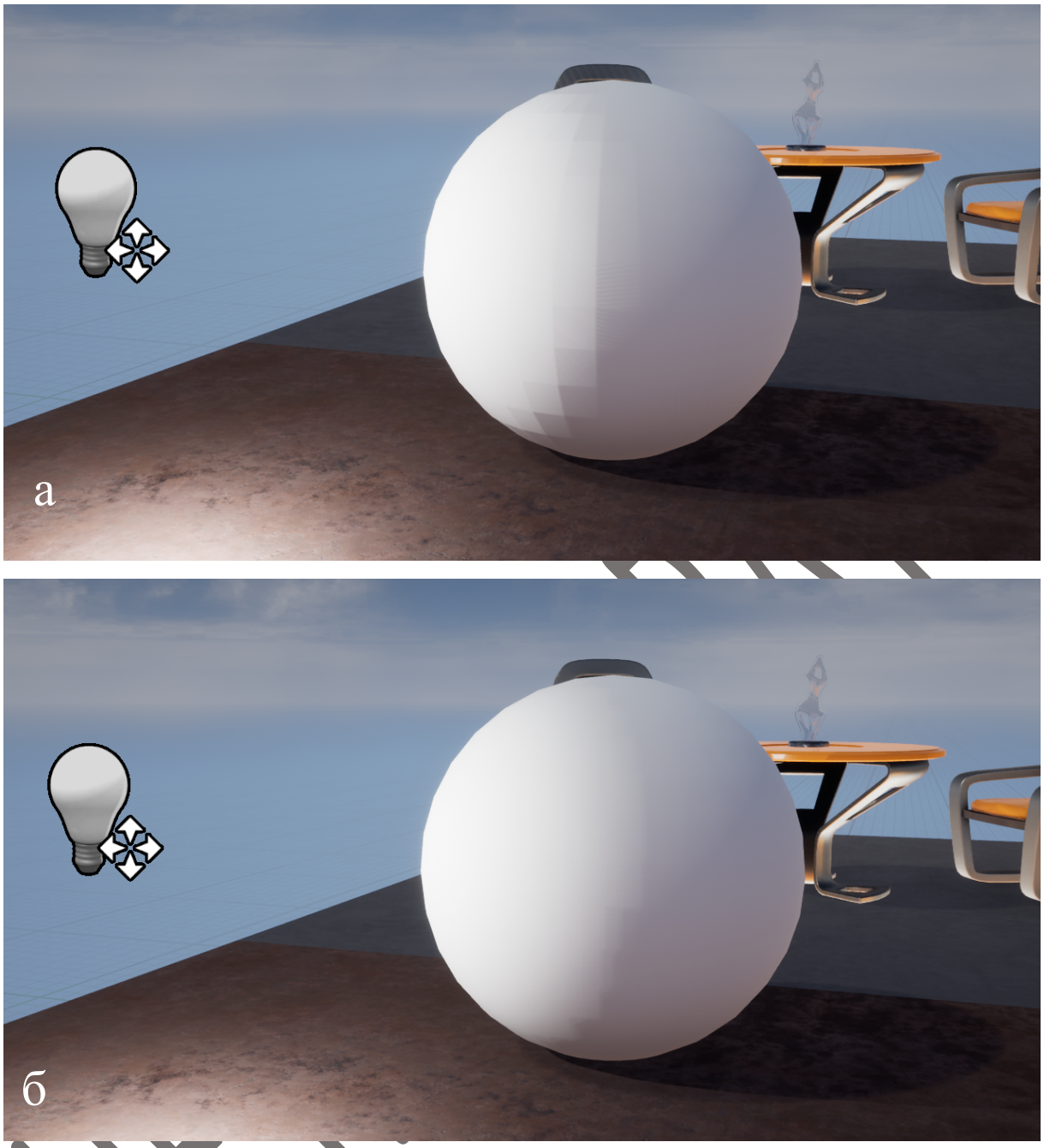


Рисунок 126 – Параметры Slope Bias и Shadow Slope Bias:

а – по умолчанию; б – подобраны для сцены

Направленные источники света обладают дополнительным свойством смещения по глубине, которое позволяет управлять силой смещения на каскадных картах теней (**Cascaded Shadow Maps**), уменьшая неоднородность теневых артефактов в точках каскадного перехода. Пример изображений приведен на рисунках 127а и 127б.

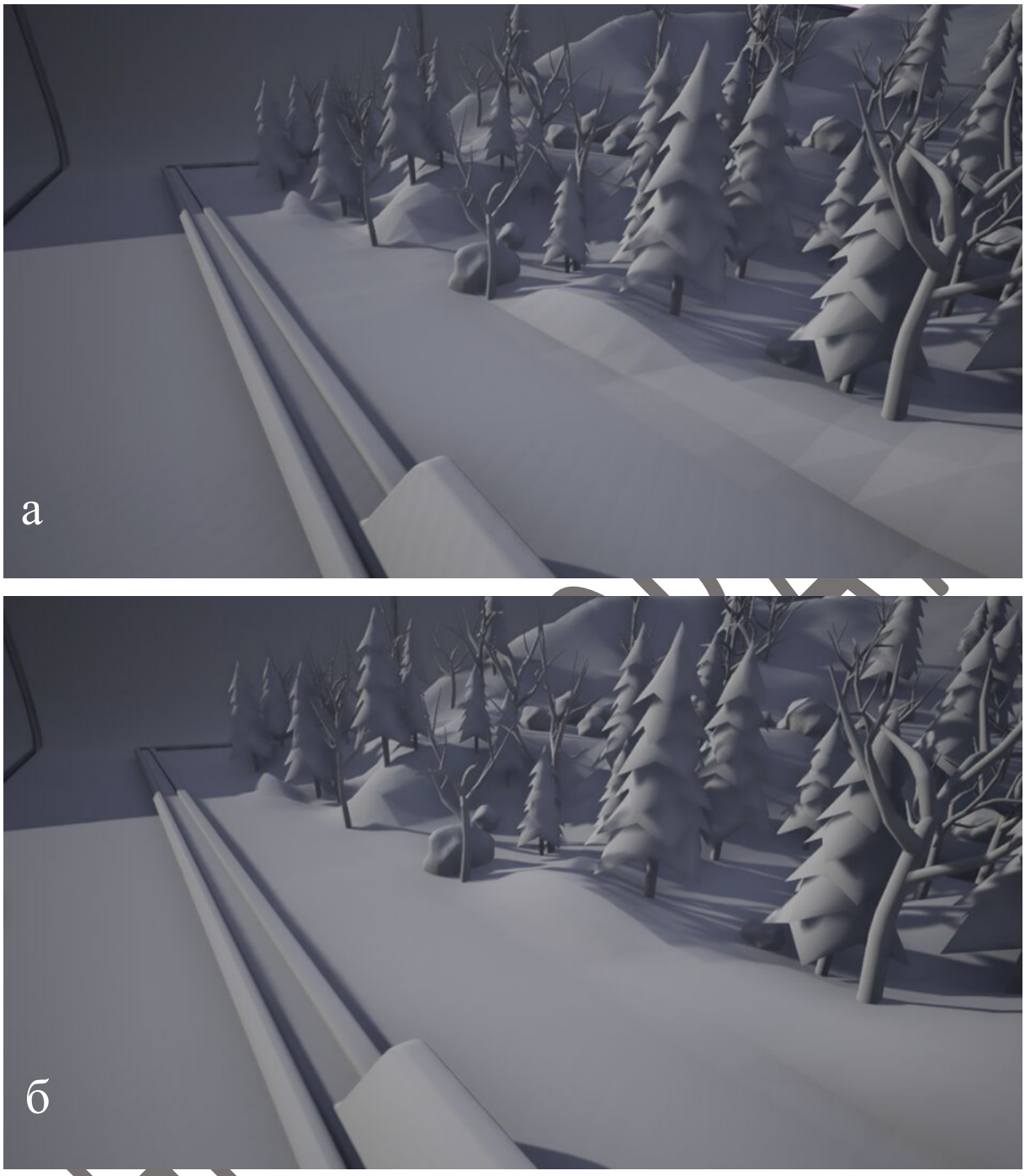


Рисунок 127 – Cascaded Shadow Maps:

а – по умолчанию; б – настроено

В дополнение к свойствам для каждого источника света можно управлять параметрами **Constant, Slope** и **Receiver bias** с помощью глобальных консольных переменных для каждого типа источника света, используя значение от 0 до 1:

- Directional: r.Shadow.CSMReceiverBias
- Spot: r.Shadow.SpotReceiverBias
- Point: r.Shadow.PointReceiverBias

- Rect: r.Shadow.RectReceiverBias

СПИСОК ЛИТЕРАТУРЫ

1. Макеффри, М. Unreal Engine VR для разработчиков / М. Макеффри. – Москва : Эксмо, 2019. – 256 с.;
2. Шэннон Том Unreal Engine 4 для дизайна и визуализации / Том Шэннон. – Москва : Бомбора, 2021. – 368 с;
3. Nixon D. Beginning Unreal Game Development: Foundation for Simple to Complex Games Using Unreal Engine 4 / D. Nixon. – West Palm Beach, FL, USA : Apress, 2020. – 413 с.;
4. Уильямс, С Основы программирования на Unreal Engine / С Уильямс. – Москва : Ареол, 2020. – 700 с.;
5. Claudia M. D. Augmented Reality and Virtual Reality, Springer / M. D. Claudia. – США : Эксмо, 5 мая 2021. – 130 с.;
6. LaValle, S. M. Virtual Reality / S. M. LaValle. – Oulu : Cambridge University Press, 2020. – 425 с.;
7. Plowman, J. Unreal Engine Virtual Reality Quick Start Guide / J. Plowman. – 1 : Packt Publishing, 2019. – 178 с.;
8. Valcasara, N. Unreal Engine Game Development Blueprints / N. Valcasara. – Birmingham : Packt Publishing, 2015. – 352 с.;
9. Ferro, L. S. Unreal Engine Blueprints Visual Scripting Projects / L. S. Ferro. – Бирмингем : Packt Publishing, 2019. – 528 с.;
10. Development of VR educational instruments for school pre-professional education in a research university // Procedia Computer Science, 2021 Vol. 190, Q2 pp. 750-754;
11. Application of VR instruments in preprofessional education in the area of mechatronics and robotics in a nuclear research university // Procedia Computer Science, 2021 Vol. 190, Q2 pp. 745-749;
12. Ildikó, Horváth (2018) “Evolution of teaching roles and tasks in VR / AR-based education.” International Conference on Cognitive Infocommunications
13. Stelian, Nicol et al. (2018) “VR for Education in Information and Tehnology: application for Bubble Sort.” International Symposium on Electronics and Telecommunications
14. Hadi, Ardiny et al. (2018) “The Role of AR and VR Technologies in Education Developments: Opportunities and Challenges.”. International Conference on Robotics and Mechatronics
15. Grand-Clement, Sarah (2017) “Digital Learning: Education and Skills in the Digital Age”
16. Vladimirov, A.I. (2011) “About engineering-technical education.” Nedra Publishing House: 80-81.
17. Alykova, O.M. & Smirnov, V.V. (2013) “The project method as an alternative to the CDIO world initiative. Physical education in the Universities.” 19(4): 16-26.
18. Varyatchenko, Elena et al. (2015) “The Development of Approaches to Engineer Training Improvement in the Research University in

19. Compliance with the International Standard." Biosciences Biotechnology Research Asia 12 (1): 939-946.
20. Fedorov, Igor et al. (2011) "Engineering education: problems and tasks." High Education in Russia 12: 54-60.
21. CDIO., Knowledge Library. CDIO Standards (2020) Retrieved from http://www.cdio.org
22. Crawley, Edward (2001) "The CDIO syllabus: a statement of goals for undergraduate engineering education." The Department of Aeronautics and Astronautics, Massachusetts Institute of Technology