**РЕФЕРАТ**

**Дипломная работа по теме – Интеграция Cad Exchanger SDK в Unity3d**

Дипломная работа состоит из

Целью данной дипломной работы является написание работоспособного модуля, позволяющего использовать CAD Exchanger SDK в приложениях, написанных на Unity3d.

В соответствии с данной целью в работе решаются следующие задачи:

* Раскрыть сущность 3-х мерного представления моделей
* Рассмотреть разнообразные форматы и их отличия
* Оценка возможностей для импорта 3d моделей Unity3d «из коробки»
* Исследование внутреннего представления модели в Unity3d
* Изучение возможностей CAD Exchanger SDK
* Написание программного модуля, для задачи конвертации внутреннего представления модели в CAD Exchanger во внутреннее представление Unity3d
* Создание примера, работы модуля
* Написание документации к модулю

При написании дипломной работы были использованы общепринятые методы экономических исследований, такие как системный анализ, метод последовательной подстановки, метод сравнений, метод структуризации целей, экспертно-аналитический метод, параметрический метод и др.

Ключевые слова: трудовые ресурсы, организация труда, совершенствование труда, мотивации, эффективность, комфортные условия труда.

**Определения**

1. CAD Exchanger –Нижегородская IT-компания. Занимается разработкой и поддержкой программного обеспечения для исследования CAD моделей и конвертацией между форматами.
2. Unity3d – кроссплатформенная среда разработки игр и приложений.
3. Viewer/Вьювер – программы для просмотра и исследования 3х мерных моделей
4. Edit-mode – режим редактирования/отладки приложения.
5. Runtime – режим работы собранного приложения, без возможности, как либо повлиять на его работу.
6. Импортер – программный модуль, отвечающий за импорт(зачитывание, парсинг и конвертацию во внутренний формат) модели определенного формата.
7. Экспортер – программный модуль отвечающий за экспорт модели определенного формата.
8. Фейс – англицизм, от английского face (грань), топологическая сущность в граничном представлении модели, поверхность ограниченая замкнутой кривой.
9. Вайер – англицизм, от английского wire (провод), топологическая сущность в граничном представлении модели, набор ограниченных кривых, с общими точками.
10. Эдж – англицизм, от английского edge (ребро), топологическая сущность в граничном представлении модели, ограниченная кривая.
11. Вертекс – англицизм, от английского vertex (вершина), точка со сферической областью с определенным радиусом и центром в этой точке. Попадание в эту область этой области, считается попаданием в точку.
12. Шелл – англицизм, от английского shell (оболочка), топологическая сущность в граничном представлении модели, несколько фейсов имеющих

Сокращения и абревиатуры

1. VR – виртуальная реальность.
2. AR – дополненная реальность.
3. BRep – способ представления фигур с помощью границ.

**Введение**

В 2005 году миру был представлен игровой движок, разработанный Дэвидом Хелгасоном, Джошимом Анте и Николасом Френсисом. Он получил короткое название Unity. С тех пор из приложения, которое сложно было назвать полноценным движком, доступного только на Mac OS, он превратился в мощнейшую кроссплатформенную среду разработки игр и приложений. Сегодня Unity поддерживает сборку под все популярные операционные системы, и под огромное количество устройств, в том числе и VR/AR. На данный момент Unity является самой популярной общедоступной платформой для создания игр и приложений, с самым большим комьюнити, огромным количеством дополнительных плагинов и ассетов, а также более чем достаточным количеством обучающих материалов по любым разделам движка.

Все это делает среду разработки Unity крайне привлекательной для компаний ,которые планируют написание своей/его игры/приложения.

Однако не все возможно в данном движке, как говориться, из коробки.

Предположим, что есть некоторая компания, которая планирует создать на данном движке приложение для просмотра 3d моделей в VR.

В процессе разработки компания столкнется сразу с несколькими проблемами:

1. Unity поддерживает импорт только узкого круга форматов 3d моделей
2. Импорт моделей возможен только в режиме редактирования, то есть пользователь не сможет через приложение выбрать модель на своем устройстве и загрузить ее в приложение.
3. Тот небольшой список форматов, который поддерживает Unity, содержит только mesh-форматы. Однако, если речь идет о создании полноценного viewer-a, подразумевается, поддержка не одного десятка форматов, не только с полигональным представлением, но и brep.

Теперь представим другую компанию, которая планирует написание игры в формате симулятора вождения. Основной фишкой данного приложения будет возможность загрузить собственную модель автомобиля. Опустим все вопросы касательно анимированния и определения технических характеристик и остановимся на факте загрузки модели. Тут мы столкнемся с теми же проблемами, что и в предыдущем примере.

Список похожих примеров может быть очень длинным. Во всех случаях разработчики сталкиваются, с одной проблемой – отсутствие возможности импорта большого количества форматов 3д моделей, в том числе и brep форматов, как в edit-mode, так и в runtime.

На первый взгляд, может показаться, что данный кейс является исключением, и потребность в подобном функционале не так уж и велика. От части, это действительно так. Геймдев, в целом, не имеет острой нужды в импорте неизвестных заранее моделей. Да и того небольшого перечня форматов, которые поддерживает Unity в обычном сценарии разработки, достаточно.

Однако спрос рождает предложение, и спрос есть. Для этого достаточно вбить в поисковую строку пару-тройку соответсвующих запросов, и поисковик выдаст вам несколько десятков тем, где обсуждается этот вопрос. Так же стоит отметить, что сейчас игровые движки медленно, но верно, вырастают из среды для разработки игр и все больше становятся средствами разработки приложений. Более подробно об актуальности мы поговорим позднее.

Итак, имеется задача: Разработать .Net библиотеку для работы со средой разработки Unity. Библиотека должна поддерживать импорт 3d моделей и конвертацию их во внутреннее представление Unity – GameObject.

Задача имеет очень простое определение и крайне сложное решение. Написание подобной библиотеки с нуля, одним человеком, заняло бы не один десяток лет. Основная сложность кроется в огромном разнообразии форматов. Каждый из них нужно правильно зачитать, проанализировать и конвертировать. Например, на изучение и написание одного из самых популярных CAD форматов – STEP, понадобится год-два упорного труда (если речь идет о полноценном импортере). А подобных форматов десятки, если не сотни.

Но мне повезло. С сентября 2019 года, я являюсь разработчиком в компании CAD Exchanger. Основной продукт компании, мощный SDK для работы с CAD форматами: конвертация, исследование, изменение, «лечение» моделей, оптимизация. Данный SDK будет фундаментом, на котором будет построен модуль.

На самом деле, практическая часть диплома писалась в рабочее время, и сейчас написанный мною модуль имеет название CadExUnity и является частью CAD Exchanger SDK.

**Теория**

Сфера 3х мерного моделирования настолько широка, что по отдельным ее разделам можно написать дипломную, магистерскую и даже докторскую работу.

В данном разделе я постараюсь осветить разделы, необходимые для выполнения поставленной задачи.

3х мерное моделирование используется во многих сферах жизнедеятельности человека.

Развлечения и досуг:

* Игры
* Фильмы с графическими эффектами
* Мультфильмы

Производство:

* Инжиниринг
* 3х мерные чертежи
* Приборо/машиностроение
* Авиационная и ракетостроительная промышленность
* 3d печать
* И т.д.

Наука

* Моделирование и расчеты
* Симуляция и виртуальные эксперименты

И если бы мы жили в идеальном мире, у нас был бы один универсальный формат 3х мерной модели, который отвечал бы всем требованиям и поддерживался всеми программными продуктами. Но к сожалению, наш мир не идеален, и как и во многих других сферах, мы имеем огромное количество различный форматов. Все эти форматы, можно классифицировать и разделить на 3 большие группы.

1. Форматы с полигональным представлением модели
2. Форматы с граничным представлением модели – Brep
3. Форматы содержащие и полигональное и граничное представления

Заметка: На самом деле, существуют еще несколько видов представления моделей, например: облака точек, вортексы и т.д. все они крайне похожи на полигональное представление, т.к. представляют из себя набор из большего кол-ва одинаковых примитивов (треугольники, точки, кубки, и т.д.).

***Полигональное представление 3х мерной модели***

Простое для понимания представление модели, модель состоит из полигональной сетки – Mesh.

Полигональная сетка — это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трёхмерной компьютерной графике и объёмном моделировании. Гранями обычно являются треугольники, четырёхугольники или другие простые выпуклые многоугольники (полигоны), так как это упрощает рендеринг, но сетки могут также состоять и из наиболее общих вогнутых многоугольников, или многоугольников с отверстиями.

Так же полигональное представление, часто хранит нормали либо для отдельных полигонов, либо для вершин.

Нормали в полигональном представлении играют важную роль, при рендеринге модели нормаль отвечает сразу за несколько аспектов:

1. С какой стороны находится лицевая сторона треугольника
2. Как будет отражаться свет

Помимо нормалей сетка может так же хранить UV-координаты. UV-координаты позволяют наложить текстуру на модель.

Например, для создания куба мы должны задать 8-24 точек (завит от вида хранения сетки), затем нужно задать 12 треугольников (по 2 для каждой стороны куба). Так же, желательно задать нормали для вершин. КАРТИНКИ

***Граничное представление 3х мерной модели***

Это более математическое представление 3х мерной модели. В целом граничное представление можно описать следующим определением: Геометрия плюс топология. На геометрические сущности, такие как, точки кривые и поверхности, накладываются ограничения, и мы получаем топологические сущности.

Основные геометрические сущности:

* Точка
* Кривая (прямая, окружность, Безье, B-spline, NURBS)
* Поверхности (плоскость, поверхности вращения, сфера, тор, конус, B-spline и т.д.)

Основные типы топологических сущностей:

* Vertex (вершина) – точка с толлерансом, некоторой сферической областью вокруг точки, попадание в которую, считается попаданием в точку. Необходимость заключается в погрешности при компьютерных вычислениях, 100% попадание в точку практически невозможно.
* Edge (ребро) – ограниченная кривая, содержит кривую ограниченную 2 вертексами.
* Wire – несколько связанных друг с другом, через общие вертексы ребер.
* Face (грань) – ограниченная замкнутым вайером поверхность.
* Shell (оболочка) – несколько фейсов связанных между собой через общие эджи.

В некоторых геометрических ядрах можно встретить дополнительные сущности, например Compound – набор состоящий из всех вышеизложенных сущностей.

Для примера, создадим тот же кубик. Для этого нам придется задать 6 поверхностей типа Plane (плоскость), для каждой из сторон куба.

Далее ограничим наши плоскости, вайерами в форме квадрата, получим 6 квадратов, при этом надо проследить, что бы эджи в двух соседних квадратах были общими. Наш куб готов.

На самом деле, в данном примере опускается очень много важных деталей, такие как:

* Параметрические пространства поверхностей и правильное расположение в этих пространствах, параметрических кривых.
* Правильные ориентации у эджей в вайерах, у вайеров в фейсах и у фейсов в шеле
* Шарринг (совпадение) параметрических кривых на границах
* И т.д.

Но в контексте поставленной задачи эти детали не так важны, т.к. в основном работа будет происходить с полигональной репрезентацией.

Граничное представление является самым информативным представлением модели, в первую очередь, потому что мы можем получить любое другое представление из граничного. А вот преобразование в обратную сторону, из полигонального в граничное представление, задача нетривиальная, и не всегда выполнимая. Так же граничное представление легче анализировать и исследовать, его можно измерить: посчитать объем, площадь поверхности и т.д.

Таким образом, граничное представление, имеет не только свои преимущества, но и преимущества других репрезентаций, как следствие относительно легкого перехода от граничного к любому другому представлению.

Как следствие наибольшей информативности, граничное представление, является гораздо более сложным в поддержке нежели полигональное.

Для сферы геймдева информативность граничного представления является излишней. В этой сфере важен только внешний вид модели, и для визуализации, информации которую несет в себе полигональное представление, достаточно.

***Продуктовая структура модели***

Важнейшей составляющей модели является ее продуктовая структура. Что бы понять, что это такое, представим модель трактора. В данном случае модель «немного» сложнее чем кубик. Допустим, мы создаем полигональную репрезентацию модели трактора. Если мы просто создадим несколько сотен тысяч треугольников, да, мы увидим красивую модельку, но она будет монолитная. Например, если мы захотим ее анимировать, и подвинуть ковш, сделать мы этого не сможем. Решением является разбиение модели на составляющие. В нашем случае, это могут быть: корпус, двери, колеса, крепление ковша, ковш. Теперь наша модель, состоит как будто из других моделей поменьше. В свою очередь эти части, могут так же состоять из составляющих. Вся эта иерархия, называется продуктовой структурой модели.

Неделимые детали модели, то есть те, которые не имеют составляющих, называются ***part***. Сборки из партов называются ***assembly***. В случае с кубом, продуктовая структура представляла бы из себя один парт, хранящий в себе какое-либо представление геометрии куба. Продуктовая структура трактора была бы немного сложнее. Вверху иерархии находился бы ассембли, который представлял бы из себя сам трактор, внутри он бы содержал другие ассембли и парты и т.д.

Стоит отметить, что в тех форматах, которые поддерживают смешанные представления, различные парты могут иметь различные типы представлений. Например, колеса у трактора, могут иметь полигональное представление, а корпус граничное.

Так же, правильно организованная продуктовая структура, может сэкономить много памяти и ускорить обработку модели.

Как можно заметить, модель трактора имеет несколько одинаковых составляющих, например, колеса или, если смотреть глубже, это могут быть даже болты, на которых эти колеса крепятся. Очевидно, что заново описывать и хранить геометрию каждого колеса не имеет смысла. Достаточно сделать копию и применить нужную трансформацию. Еще один элемент продуктовой структуры – это ***instance*** (экземпляр). Инстанс – это копия любого другого элемента продуктовой структуры (парт или ассембли), прикреплённая к модели с собственной трансформацией. Так же инстанс может иметь свое имя.

***Внешний вид***

Еще одна составляющая модели - это ее внешний вид. Сюда можно отнести: цвета, текстуры, материалы. Внешний вид может быть, как у всей модели в целом, так и свой для каждого парта или инстанса.

***PMI***

PMI— производственные данные, ассоциированные с трехмерной моделью изделия в CAD. PMI-данные включают в себя геометрические размеры и допуски (GD&T), трехмерные аннотации (тестовые пометки), требования к качеству обработки поверхностей и спецификации материалов.

Далеко не каждый формат поддерживает PMI. Обычно, это BRep форматы, так основной сценарий использования — это сфера CAD.

***Модель в представлении CAD Exchanger***

За внутреннее представление модели в CAD Exchanger отвечает класс *ModelData\_Model,* после конвертации любого формата мы получаем объект данного класса. Он хранит всю информацию о модели. Нас будут интересовать следующие данные из *ModelData\_Model*: