Содержание

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Почему нужна эта книга?

Универсальное описание сцены (Universal Scene Description, USD) уже произвело большой фурор в индустрии визуальных эффектов, кино и игр, и такие крупные игроки, как NVidia, Apple и Autodesk, бросились в эту технологию с головой. Но для новичка знакомство с этой технологией может стать невероятно сложным и разочаровывающим путешествием.

Основное назначение этой книги – предложить краткий, понятный для новичков и структурированный способ знакомства с USD, его концепциями и терминологией. На данный момент эта книга сосредоточена преимущественно на терминологии, но в следующих изданиях будет дополнена примерами рабочих процессов.

Возможно, у вас возник вопрос: почему стоит начать с чтения этой книги вместо официального глоссария и учебных курсов по USD?

Погружение в официальный глоссарий или учебные пособия по USD – это совершенно правильная стратегия; к сожалению, реальность такова, что USD – это очень сложная машина со множеством движущихся частей, кинематически связанных друг с другом. Даже простое тестирование приложений USD, таких как `USDView`, может быть довольно сложным, если вы не инженер-программист. Многие концепции также привязаны друг к другу, поэтому очень легко увязнуть в терминологии. Увы, метод обучения «нырянием в кроличью нору» иногда бывает невероятно запутанным и подавляюще трудным.

Эта книга предлагает альтернативный путь.

Примечание авторов

В Remedy Entertainment мы, безусловно, ♥ USD! Мы даже выступили с докладом о том, как мы используем его для создания игр класса AAA!

После вышеупомянутой презентации нас попросили опубликовать нашу внутреннюю документацию по самому USD, поскольку мы написали упрощенную версию официального глоссария USD. Эта книга и есть та самая документация, хотя и обработанная, переписанная и скорректированная для совместимости с `mdbook`.

Мы опубликовалии ее с открытой лицензией, поэтому вы можете свободно копировать, ответвлять и изменять свою версию этой книги по собственному усмотрению.

1

Итак, что же такое USD?

Короче говоря, USD – это способ описания иерархий 3D-сцен.

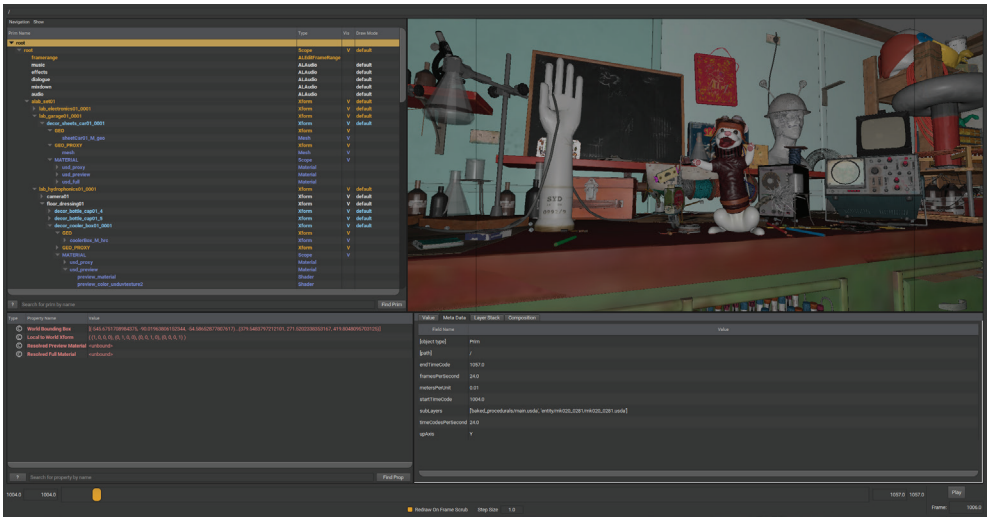

Рис. 1.1 ❖ Пример сцены USD.
Сцена Animal Logic ALab USD в USDView

В общем, иерархии состоят из *записей* (`nodes`, или узлов), которые имеют родительские, дочерние или родственные (брат–сестра) связи с другими узлами. В USD эти узлы могут иметь «физический характер» (например, геометрию, данные анимации и т. д.) или быть более абстрактными, например группировать элементы, материалы, шейдеры и даже «настройки», которые могут быть выражены как узел внутри иерархии.

USD сильно отличается от других описаний сцен своей способностью объединять иерархии вместе с различными типами поведения (это известно как *композиция*, или *компоновка*). В USD вы можете неразрушающим образом ссылаться на другие иерархии или сшивать их вместе, переопределять

ранее определенные данные, кооперировать со многими другими в той же иерархии и многое другое.

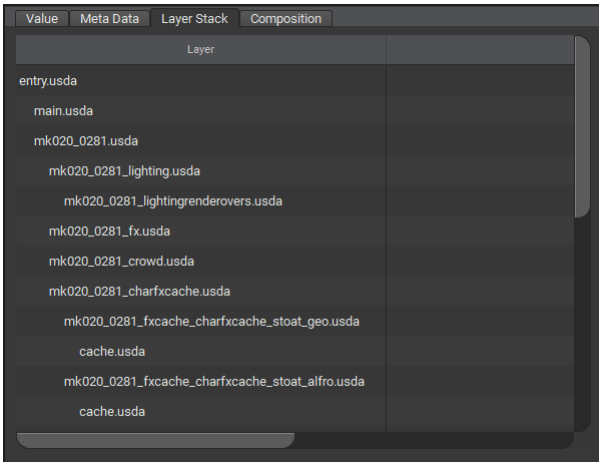

Рис. 1.2 ❖ Пример композиции сцены USD.

Часть списка сцен USD, которые составляют композицию сцены Animal Logic ALab USD, внутри USDView

Но подождите! Это еще не все!

Помимо неразрушающего объединения иерархий, USD изначально предусматривает использование расширений. Например, возможно расширение USD с помощью плагина, который будет анализировать заказной или коммерчески доступный формат файла и использовать эти файлы непосредственно внутри USD.

Конкретным примером может служить поддержка использования файлов FBX непосредственно внутри иерархии USD.

В целом USD стремится стать переносимым форматом обмена, определяя общий «язык», описывающий сцену и 3D-данные. Эта идея не нова, но USD делает шаг вперед, также предлагая способ последовательного рисования, рендеринга и визуализации сцен. Это стало возможным благодаря `Hydra`, технологии бэкенд-рендеринга USD.

В этой книге в настоящее время не рассматриваются концепции Hydra, но в будущем это может быть сделано.

2

Чем не является USD?

Просто форматом файла

Хотя основным «интерфейсом» работы с USD для многих художников является его представление в виде файла на диске, важно рассматривать USD скорее как экосистему, чем как еще один тип файла на диске. Учитывая возможности данной технологии и то, что она может предложить, называть ее форматом файла было бы несправедливо.

USD можно формировать в соответствии с вашими потребностями или использовать как есть. Представления в виде файла на диске можно вообще исключить. Способ, которым вы можете ссылаться на другие сцены USD, чрезвычайно гибок. Вы даже можете интегрировать любой пользовательский движок рендеринга с помощью простого плагина, чтобы он понимал, как работать с данными USD и т. д.

> **Предупреждение!**
> Если вы используете USD достаточно часто, это становится образом жизни .

Заменой для *создания* контента

USD может делать множество вещей, но при этом не способен делать многое другое. Основной компонент его архитектуры не допускает *интерактивных значений*. Это означает, что значения устанавливаются, по сути, только однократно и не изменяются, пока вся сцена не будет «перекомпонована». На самом деле способ вычисления значений немного сложнее, чем сказано здесь, но проще думать об этом так.

С точки зрения разработки USD был создан с учетом малого объема памяти и высокой задержки доступа к данным в пользовательском оборудовании. Как следствие, чтобы обеспечить малое использование памяти, допускается более медленный доступ к данным, чем в парадигме большого объема

памяти и малой задержки доступа к данным. Последняя жертвует объемом памяти в пользу быстрого доступа к данным, позволяя выполнять динамические вычисления.

Из-за этого принципиального решения USD с трудом предоставляет «динамические» системы, такие как риггинг, которые полагаются на быстрый динамически вычисляемый доступ к данным. Однако благодаря своей расширяемости USD может позволить студиям *определять* концепции риггинга. Но это всего лишь *спецификация*, сам риггинг не будет *оцениваться* внутри USD и потребует, чтобы хост-программа (например, Autodesk Maya) взяла эту информацию и преобразовала ее в риггинг, который она может понять.

Чего не может USD?

В USD нет GUID

USD использует текстовое, иерархическое пространство имен для идентификации своих данных, что означает, что переопределения привязываются к определяющим их примитивам и свойствам через *пути пространства имен* (namespace paths). В результате, когда внутреннее пространство имен ссылаемого актива изменяется, *переопределения более высокого уровня, ранее записанные в ссылаемых активах, перестают применяться*. Одним из решений этой проблемы является идентификация данных по глобально уникальному идентификатору (GUID), а затем связывание переопределений с тем же GUID, что и определяющий примитив (prim). Хотя GUID решают проблему редактирования пространства имен, они вносят другие проблемы в конвейер и потенциально ограничивают гибкость композиции. В прошлых версиях USD Pixar использовала форму GUID на уровне детализации модели/актива, и после тщательного взвешивания всех за и против мы решили, что для нас затратность периодических операций по исправлению пространства имен, выполняемых над коллекцией активов, стоит того, чтобы заплатить за простоту создания и агрегации активов и читаемые текстовые представления активов, которые мы получаем из путей пространства имен в качестве идентификаторов.

USD не является системой выполнения или риггинга

USD предоставляет легкий, оптимизированный граф сцены для упрощения разработки и эффективного извлечения описания скомпонованной сцены. Однако он не обеспечивает никаких других функций, кроме композиции иерархии пространства имен и разрешения значений свойств (Value Resolution), и в пространстве компромисса между «малым объемом памяти, высокой задержкой доступа к данным» и «большим объемом памяти, малой задержкой доступа к данным» граф сцены USD больше склоняется к первому варианту, тогда как высокопроизводительный движок выполнения требует второго.

Кроме того, чем больше свойств риггинга и семантики выполнения мы добавим в USD, тем сложнее будет успешно обмениваться данными между

DCC[1], поскольку в настоящее время нет всеобъемлющего соглашения между разработчиками о том, какими должны быть эти свойства.

USD и его инструменты генерации схем должны подходить для кодирования риггинга для кругового обращения данных риггинга в конкретном приложении или пользовательском конвейере, и USD предоставляет возможности, которые клиент может использовать для создания более обширных кешей в памяти поверх UsdStage для обеспечения доступа с меньшей задержкой к данным, закодированным в USD. Но на данный момент они не играют значительной роли в том, что мы считаем основным предназначением USD: масштабируемый обмен геометрическими данными, данными шейдинга и освещения между DCC в конвейере создания 3D-контента.

Наследие USD в Pixar

USD – это примерно четвертое поколение инструмента композиционного описания сцены, разработанного в Pixar. После «Истории игрушек»[2], в которой каждый кадр описывался одним линейным программным файлом, команда Pixar R&D начала добавлять и развивать концепции ссылок, наложения слоев, редактирования и вариаций в контексте своей фирменной системы анимации Marionette (известной внутри компании как Menv), начиная с «Приключений Флика» и продолжая в производстве следующих десяти полнометражных фильмов.

К 2004 году стало ясно, что, хотя система Marionette стала довольно мощной, ее громоздкая структура стала помехой для продолжения стабильной разработки и возможности использовать важные инструменты, такие как многоядерные системы. Студия взяла на себя обязательство по проектированию и разработке с нуля системы анимации второго поколения, теперь известной как Presto, которая впервые была использована в анимационном фильме «Храбрая сердцем» и всех других анимациях с тех пор. Одна из проблем с Marionette, которую Presto намеревалась решить, заключалась в том, что ее различные функции для композиции и редакции описания 3D-сцены не всегда могли эффективно использоваться вместе, поскольку они были распределены по трем различным форматам и «композиционным движкам». Presto предоставила второе поколение описания сцены, которое было *унифицированным*, позволяя ссылаться, переопределять, редактировать и делать другие операции на всех уровнях детализации, начиная от голой сетки до всей модели, среды или кадра, закодированных в одном текстовом формате и оцененных с помощью одного композиционного движка.

Однако в то же время студия Pixar, как и большая часть индустрии кино и эффектов, посчитала выгодным перейти от конвейера, в котором анимация и риггинг сохранялись в актуальном состоянии вплоть до рендеринга, к конвейеру, в котором анимация и риггинг «запекались» в эффективных *кешах поз* (pose caches), содержащих анимированные точки поз и трансформаций, чтобы освещение, эффекты и рендеринг могли сократить задержку

---

[1] Создателями цифрового контента. – *Прим. ред.*

[2] Первый полнометражный компьютерно-анимационным фильм студии. – *Прим. ред.*

(и объем памяти), с которой они получали доступ к данным. Таким образом, в 2008–2009 годах команда разработчиков конвейера начала создавать TidScene, геометрическую схему, поддерживаемую бинарной базой данных (Berkeley DB), с облегченным графом сцены в качестве механизма для создания и чтения данных с временной выборкой. Ключевые элементы TidScene включали (на тот момент) высокопроизводительный плагин рендеринга OpenGL, который позволял выполнять предварительный рендеринг напрямую из TidScene во всех приложениях конвейера, а также разработку встроенной функции ссылок, которая использовалась (возможно, чрезмерно) для достижения многослойности, изоляции графа сцены (т. е. загрузки только части сцены), ссылки на активы и некоторой поддержки вариативности.

Скорость, масштабируемость и универсальный конвейерный доступ к кешам поз TidScene оправдали ожидания, но также вернули Pixar в ситуацию, где у нас было несколько конкурирующих систем для создания композиционного описания сцены с различной семантикой, API и местами в конвейере, где они могли использоваться. Идея проекта USD, инициированного в 2012 году, заключалась в объединении (недавно переработанного и улучшенного) композиционного движка и низкоуровневой модели данных из Presto с моделью данных с ленивым доступом, выборкой по времени и облегченным графом сцены из TidScene. USD предоставляет совершенно новый граф сцены, который находится поверх того же самого композиционного движка, который использует Presto, и вводит параллельные вычисления на всех уровнях описания сцены и ядра композиции.

Ключевым компонентом проекта USD была разработка современной масштабируемой архитектуры рендеринга, названной Hydra, изначально развернутой с тем, что впоследствии стало известно как высокопроизводительный растеризирующий рендерер Storm. Hydra поставляется как часть проекта USD, поскольку она добавляет огромную ценность интегрирования USD в конвейер и используется во всех наших плагинах; она также предоставляет эталон и указания на то, как использовать многопоточность USD для быстрой загрузки и визуализации сцен, а также делать эффективные обновления в ответ на динамические изменения в «живой» сцене UsdStage. Однако Hydra является независимым продуктом и уже имеет другие прямые внешние связи, отличные от USD (включая плагины Presto и Maya, Katana и Houdini), и вышла за рамки своей первоначальной архитектуры, вдохновленной OpenGL, для обслуживания других делегатов рендеринга, таких как трассировщики путей (path-tracers).

3

Получение USD

USD представляет собой удивительный стек технологий, но, к сожалению, это также затрудняет начало работы с ним. Это набор библиотек и инструментов, которые являются частью более крупной экосистемы, включающей USD. И хотя сторонние инструменты становятся все лучше и лучше, «получить USD» по-прежнему не так уж просто.

Pixar Animation Studios предлагает исходный код на GitHub (https://github.com/PixarAnimationStudios/OpenUSD), и для разработчиков это, как правило, стартовая точка для начала работы. Они клонируют репозиторий, загружают нужную версию Python и свой любимый компилятор C++, а затем делают большоооой перерыв на кофе, чтобы получить все инструменты и библиотеки, созданные для их системы.

Надо сказать, что этот подход не для всех. К счастью, это не единственный способ начать работу с USD. Для новичков обычно есть два способа получить и использовать USD.

В хост-приложении

Хост-приложения – это стороннее программное обеспечение, которое выбирает встраивание USD в свой набор инструментов или полностью строится на основе USD.

Ниже приведен небольшой список наиболее известных приложений, которые это делают:

- NVIDIA Omniverse Create – очень эффективный ассемблер USD, созданный на основе Omniverse от NVIDIA, который позволяет вам работать практически с необработанными USD;
- Autodesk Maya + MayaUsd – профессиональное программное обеспечение для 3D-графики; интеграция Maya USD – это надстройка, которую можно установить во время установки;
- SideFx Houdini + Solaris – профессиональное программное обеспечение для 3D-графики; набор для разработки внешнего вида Solaris встроен и создан на основе USD.

Автономно

На данный момент это наименее удобный для художников вариант, но он может быть интересен тем, кто хочет попробовать USD, поскольку он предоставляется Pixar Animation Studios. Есть два способа использования USD в автономном режиме.

- **Предварительно собранные бинарные файлы.** Хотя доступно немного предварительно собранных бинарных файлов, NVIDIA предоставляет некоторые из них на developer.nvidia.com/usd.
- **Самостоятельная компиляция.** Это «традиционный» способ начать работу с USD, и, как уже упоминалось, он может быть невероятно сложным для тех, кто не является профессиональным программистом. Но если вы хотите пойти по этому пути, Pixar Animation Studios предоставляет фантастические инструкции по сборке на https://github.com/PixarAnimationStudios/OpenUSD.

Если вы новичок в USD и хотите научиться конструировать и работать с USD, рекомендуется создавать композиции и слои USD через хост-приложение. У каждого хоста есть своя собственная программа обучения, индивидуальные вызовы и особенности в отношении USD, но, скорее всего, это поможет вам быстрее всего.

Метод `Compile-it-yourself` используется для всех изображений в этой книге и в значительной степени опирается на встроенные инструменты USD, такие как `usdview`.

4

Терминология USD

После введения в последующих главах мы сосредоточимся исключительно на основной терминологии, необходимой для понимания USD. Будет намного проще следовать любому руководству по USD, если сначала вы выучите основной жаргон.

Рекомендуется читать страницы в этой главе по порядку.

Предупреждение

Содержание в этой главе не претендует на исчерпывающую полноту, поскольку многие концепции чрезмерно упрощены, а другие просто отсутствуют.

Другими словами, ***это не замена официальному глоссарию USD***, и настоятельно рекомендуется прочитать его после того, как ознакомитесь с текстом главы.

5

Prim

Сокращение от Primitive, квинтэссенция компонента USD. Prims (примитивы) – это узлы в иерархии, которые могут иметь родительские или дочерние связи с другими примитивами; это означает, что примитивы могут иметь другие примитивы в качестве дочерних или сиблинговых (родственников одного уровня, брат–сестра) элементов или иметь другой примитив в качестве своего родителя.

На рис. 5.1 каждый узел в иерархии является примитивом.

Рис. 5.1 ❖ Пример примитива (Prim)

Внимательные читатели заметят, что примитивы могут иметь *тип*. Такие типы, как `Xform`, `Mesh`, `Scope` и `Material`, – это особые типы примитивов. Они имеют свойства по умолчанию и данные, механизм применения которых объясняется в следующей главе.

Пользователи также могут определять свои собственные типы примитивов.

Хотя примитивы указывают, к какому «типу» элемента сцены они относятся, они не обязательно сами обладают данными. Однако их можно считать «контейнерами» для именованных данных, эти данные обычно выражаются как Properties (свойства).

Глоссарий USD – **Prim**

Prim Definition

Prim Definition (определение примитива) – это набор встроенных свойств и метаданных, которые примитив получает из комбинации IsA schema, определенной из его typeName и его прикладных схем API. Определение самого примитива используется для определения того, какие свойства и метаданные он имеет, помимо тех, что записаны в его описании сцены. Он также может предоставлять резервные значения (fallback values) во время разрешения значений свойств или метаданных для встроенных свойств и метаданных примитива. API для работы с определениями примитивов предоставляется классом UsdPrimDefinition.

PrimSpec

Каждый *скомпонованный* примитив (Prim) в сцене (Stage) является результатом потенциального множества *спецификаций примитивов* (PrimSpecs), каждая из которых вносит свое описание сцены в итоговый композитный результат. PrimSpec можно рассматривать как «нескомпонованный примитив в слое». Подобно скомпонованному примитиву, спецификация примитивов является контейнером для данных свойств и вложенных PrimSpec. Важно, что *composition arcs* (дуги композиции) могут применяться только к PrimSpec, а те дуги, которые указывают цели, нацелены на другие PrimSpec.

PrimStack

PrimStack – это список спецификаций примитивов (PrimSpecs), которые предоставляют мнения для метаданных скомпонованного примитива. Эта информация сжимается из индекса примитива и становится доступной через **UsdPrim::GetPrimStack**.

Primvar

Название *primvar* заимствовано из RenderMan и означает «переменная примитива». Primvar – это специальный атрибут, который рендерер ассоциирует с геометрическим примитивом, и значение этого атрибута может изменяться (интерполироваться) по поверхности или объему примитива. В USD вы создаете и извлекаете Primvar с помощью схемы UsdGeomImageable и взаимодействуете со специальным кодированием primvar с помощью схемы UsdGeomPrimvar.

Есть два ключевых аспекта идентичности переменной примитива:

- Primvar определяют значение, которое может меняться по примитиву, на котором они определены, с помощью предписанных правил интерполяции;
- в совокупности Primvar описывают «переопределения для каждого примитива» для шейдера(ов), к которому привязан примитив. Различные рендереры могут передавать переменные шейдерам с помощью различных механизмов, которые Usd не контролирует; Primvar просто предоставляют классификацию, которую любой рендерер должен использовать для поиска потенциальных переопределений.

6

Property

Примитивы могут иметь Properties (свойства), которые по сути являются именованными и типизированными данными. На самом деле `Property` в USD является собирательным термином для двух различных типов свойств:

Attributes (атрибуты) – это Properties с прямыми значениями, которые могут меняться со временем

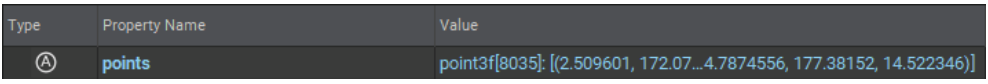

Рис. 6.1 ❖ Attributes

Relationships (связь) – это Properties, которые указывают на другие свойства или примитивы

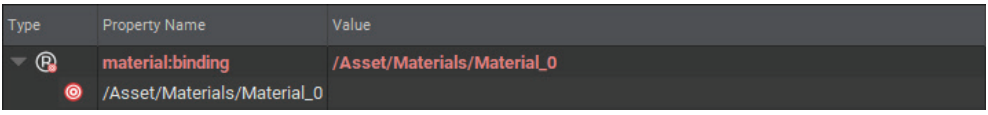

Рис. 6.2. Relationships

Выше вы можете видеть, что свойства состоят из `name` и типизированного значения `value`.

Эти имена свойств также могут быть помещены в пространство имен. Имя свойства может иметь один или более идентификаторов пространства имен, разделенных «:». При более близком рассмотрении примера связи имя свойства `material:binding` фактически принадлежит определенному пространству имен. Само имя свойства является `binding` (привязкой) и является частью пространства имен `material`.

Пространства имен могут использоваться для категоризации или группировки свойств вместе.