Автоматизация генерации серий реалистичных изображений с использованием языка сценариев Python*

***Н.Б. Дерябин*[1], *В.Г. Соколов*[1], *Д.Д. Жданов*[2], *М.С. Копылов*[1]**
dek@keldysh.ru|sokolov@integra.jp|ddzhdanov@mail.ru|dvaag@hotmail.com
[1]Институт прикладной математики им. М.В. Келдыша РАН, Москва, Россия;
[2]Государственный оптический институт им. С.И. Вавилова, Санкт-Петербург, Россия

В работе рассматриваются вопросы использования языка сценариев Python для различных направлений оптического моделирования и визуализации, таких как автоматическая генерация большого количества реалистичных изображений для анимации, или множественные световые расчеты со сложными источниками света, включая дневное освещение, и др. Предложенный подход с добавлением в сцену параметрических сценарных объектов-симуляторов делает процесс моделирования простым и удобным для конечного пользователя. Интерактивно изменяя параметры объекта-симулятора, пользователь настраивает моделирование под свои конкретные нужды. Предусмотрен механизм расширения программного комплекса новыми классами объектов-симуляторов. Так же рассматриваются примеры конкретных симуляторов.

Ключевые слова: *автоматизация моделирования, генерация изображений, дневное освещение, оптические системы, расширяемость, язык сценариев, Python.*

Automation of generating series of realistic images using Python*

***N.B. Deryabin*[1], *V.G. Sokolov*[1], *D.D. Zhdanov*[2], *M.S. Kopylov*[1]**
[1]Keldysh Institute of Applied Mathematics, Moscow, Russian Federation;
[2]Vavilov State Optical Institute, St. Petersburg, Russian Federation

Practical issues of using Python script language for various applications of optical modeling and visualization such as automatic generation of a large number of realistic images for animation or multiple lighting simulations with complex light sources including daylight are considered. The presented approach which proposes to add the parametrical script-based simulator objects to the scene makes the modeling process simple and convenient for the end user. The proposed approach allows adjusting of modeling procedure according to user's requirements by changing parameters of the simulator interactively. Software extension mechanism by means of adding new script-based simulator classes is provided. Also examples of concrete simulators are considered.

Keywords: *modeling automation, image generation, daylight, extensibility, script language, Python.*

Введение

Каким бы продвинутым ни был графический пользовательский интерфейс сложной системы оптического моделирования, его оказывается недостаточно для решения многообразных задач, возникающих на практике. Для преодоления этой ограниченности используются языки сценариев. Выделим два преимущества, которые можно получить с помощью языков сценариев:

- автоматизация моделирования. Пользователь может запрограммировать последовательность действий, которую нужно выполнить для конкретной задачи моделирования, сохранить ее в скрипте (файле, написанном на языке сценариев), и затем многократно использовать как единое целое.
- расширяемость. Используя адекватный язык сценариев и соответствующую системную поддержку, пользователь может создавать в виде скриптов новые классы (модули, пакеты), расширяющие систему оптического моделирования новыми типами объектов и новыми возможностями.

Мы успешно внедрили оба эти преимущества в наш программный комплекс, используя язык сценариев Python. Скрипты широко используются в наших программах для создания специальных сценарных объектов – симуляторов, выполняющих различные конкретные задачи моделирования. Использование симуляторов делает процесс моделирования простым и удобным для пользователя.

В работе описывается соответствующий интерфейс пользователя – оператора программного комплекса оптического моделирования, излагаются принципы написания сценариев моделирования на языке Python, и приводятся примеры конкретных процессов моделирования.

Интерфейс конечного пользователя

Простейший способ автоматизации моделирования при помощи сценариев – это записать последовательность требуемых действий на языке Python,

Работа поддержана грантами РФФИ 13-01-00454, 15-01-01147, а также фирмой Integra Inc. (Япония). Работа опубликована при финансовой поддержке РФФИ, грант 15-07-20347.

загрузить эту последовательность (например, из файла) в специальное командное окно комплекса оптического моделирования, и выполнить (Рис. 1).

На практике этот способ, поддерживаемый в том или ином виде во многих современных графических системах, не удобен из-за существенных ограничений: алгоритм моделирования хранится вне сцены; изменение параметров моделирования затруднено и требует изменения самого скрипта, который на практике бывает большим и сложным Поэтому нами была разработана возможность расширения сцены специализированными параметрическими объектами моделирования.

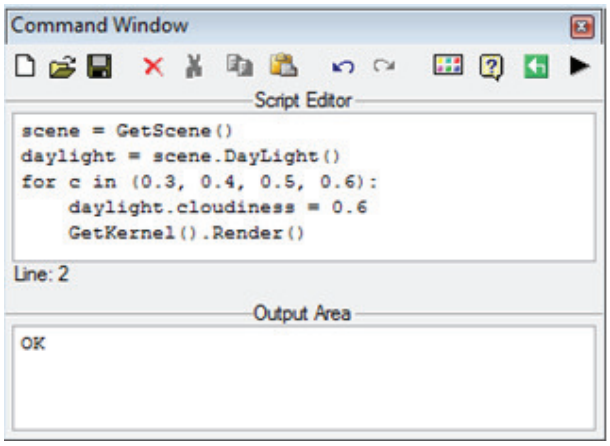

Рис. 1: Командное окно

Такие объекты создаются в библиотеки объектов при помощи специальной инструментальной кнопки создания сценарного объекта (см. Рис. 2).

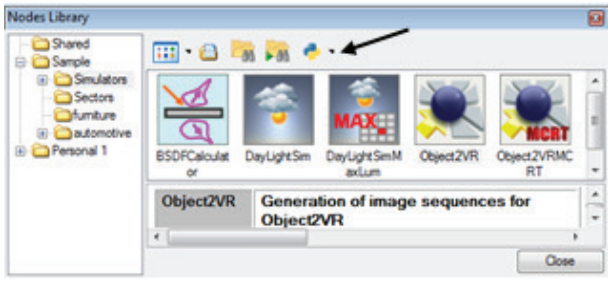

Рис. 2: Библиотека симуляторов

Процесс создания включает создание класса объекта – специального скрипта на языке Python. Ряд готовых, наиболее употребительных объектов моделирования вместе с online-документацией заранее включен в библиотеку, так что конечный пользователь обычно просто выбирает требуемый объект.

Желаемый объект моделирования включается в сцену путем простого перетаскивания мышью из библиотеки в иерархию сцены (отмечено черной стрелкой на Рис.3).

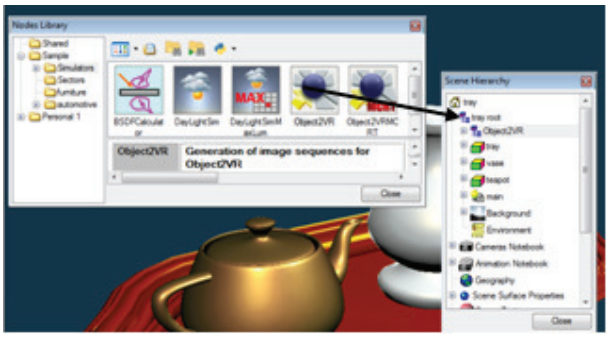

Рис. 3: Включение объекта-симулятора в сцену

После этого объект моделирования становится частью сцены и сохраняется вместе с ней.

При желании скрипт (класс) объекта можно непосредственно редактировать как в библиотеке, так и в сцене при помощи встроенного редактора скриптов, вызываемого через контекстное меню (см. Рис. 4).

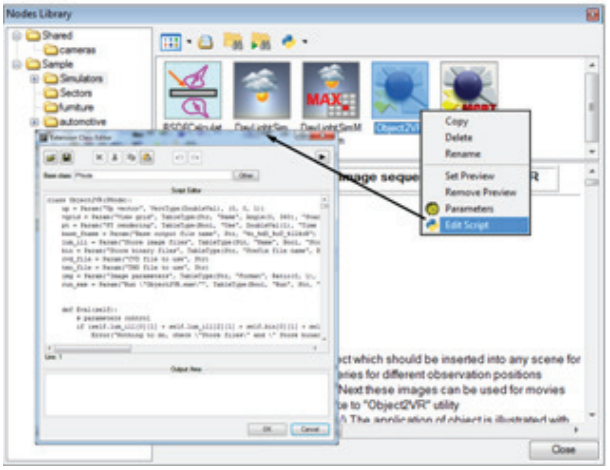

Рис. 4: Редактирование скрипта симулятора

Каждый сценарный объект имеет параметры, которые служат для настройки конкретного процесса моделирования. Набор возможных параметров сценарного объекта задаётся в классе (скрипте) объекта. Типы параметров варьируются от простых скаляров и строк до сложных таблиц со смешанными типами элементов.

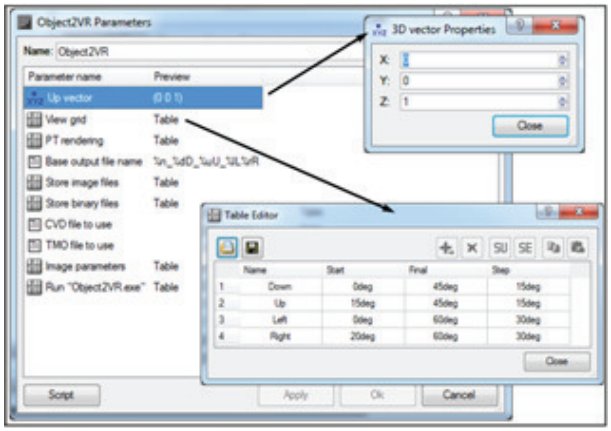

Рис. 5

Параметры сценарного объекта можно редактировать при помощи редактора параметров (Рис. 5), вызываемого через контекстное меню. Редактируя параметры, пользователь настраивает процесс моделирования под свои конкретные нужды. Текущие значения параметров симулятора сохраняются вместе со сценой.Помимо параметров моделирования, скрипт объекта описывает и саму процедуру моделирования. Эта процедура именуется специальным образом, так что ее имя появляется в контекстном меню сценарного объекта. Моделирование запускается путем вызова этой процедуры через контекстное меню сценарного объекта в иерархии сцены (соответствующий элемент меню отмечен черным прямоугольником на Рис.6). Один симулятор может определять несколько процедур моделирования.

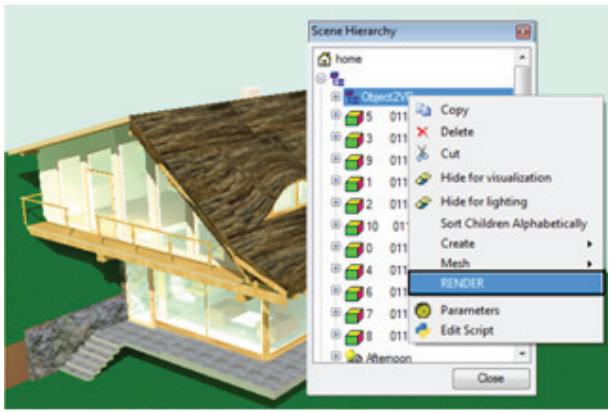

Рис. 6: Запуск моделирования

В ходе моделирования могут изменяться атрибуты объектов сцены (например, положение камеры и солнца, визуальные свойства поверхностей, оптические свойства материалов и т.п.). Эти изменения отображаются в главном окне приложения, что дает возможность пользователю визуально контролировать процесс моделирования. Результаты моделирования (например, набор сгенерированных изображений) обычно записываются процедурой моделирования в файлы для последующего использования.

Все это обеспечивает удобную и эффективную работу для конечного пользователя.

Интеграция с языком Python

Наш комплекс оптического моделирования построен на объектно-ориентированной архитектуре [1], которая естественным образом перекладывается на язык Python. Не вдаваясь в технические детали, опишем принципы этого переложения.

Сцена представляет собой сложную иерархию объектов, имеющих общий унифицированный программный интерфейс, называемый целевым интерфейсом. Для каждого объекта, целевой интерфейс определяет набор его свойств (атрибутов) и набор действий (процедур), которые можно выполнить над ним.

Целевой интерфейс естественным образом транслируется в инструкции на языке Python. Доступ (чтение, модификация) к свойству объекта транслируется в язык Python как обращение к переменной - члену класса. Используется обычная запись через точку:
$x = obj.prop$
$obj.prop = x$
$x = obj.list[i]$
Исполнение действия записывается как вызов члена-метода:
$obj.Proc(p1, p2)$
$res = obj.Func(p1)$
Наличие унифицированного целевого интерфейса позволило нам реализовать интерпретатор языка Python как отдельную независимую программную компоненту.

Используя программный интерфейс комплекса оптического моделирования, пользователь может выполнять произвольное моделирование со сценой через командное окно (Рис. 1). Нам осталось описать, формат класса расширения для конкретных процессов моделирования (Рис. 4).

Формат класса простой. Заголовок класса, такой как
$classObject2VR(PNode)$:
задает имя класса (Object2VR). Класс моделирования всегда наследуется от предопределенного типа (класса) PNode. Далее следуют определения параметров:
$up = Param("UpVector\ VectType(Float),$
$(0, 0, 1))$
$vgrid = Param("Viewgrid$
$TableType(Str, "Name\ Angle,$
$"Start"Final"Step ...))$

...
Описание параметра задает два имени параметра: для использования в скриптах ("*up*") и для использования в редакторе параметров ("*UpVector*"). Задается также тип параметра и (опционально) значение по умолчанию. Параметры являются свойствами сценарного объекта.

Табличный тип задается при помощи мета-типа TableType. Строки в кавычках задают имена колонок таблицы. В приведенном примере первая колонка таблицы имеет имя "*Name*" содержит строки (тип *Str*), а следующие три колонки ("*Start*"*Final*"*Step*") содержат углы в градусах (тип *Angle*).

Помимо параметров, класс моделирования обычно задает функцию моделирования:
@*Method*("*RenderImages*])
def Render(*self*) :
...
Это описание функции использует специальный декоратор @*Method*(). Этот декоратор включает данный метод в контекстное меню сценарного объекта.

Таким образом, как мы надеемся, использование языка Python для оптического моделирования не представляет проблем.

Примеры симуляторов

Первый пример сценарного симулятора, включенный в наш программный комплекс, предназначен для генерации серий изображений с различных угловых точек наблюдения (точка 2 на Рис. 7) нацеленных на заданную точку 1 в пространстве (точка 1 на Рис. 7). Основные параметры генерации: угловые диапазоны горизонтального (Г-Г) и вертикального (В-В) перемещения точки наблюдения и угловой шаг. Множество других параметров симулятора позволяет управлять различными характеристиками генерации изображений от выбора методики расчета до формата имен генерируемых файлов с изображениями.

Результатом работы симулятора может быть серия обычных изображений или изображений в формате с динамической яркостью, допускающих последующую постобработку [2]. Эти изображения могут быть непосредственно использованы для создания интерактивных трехмерных анимационных моделей, например, с помощью программы Objetct2VR.

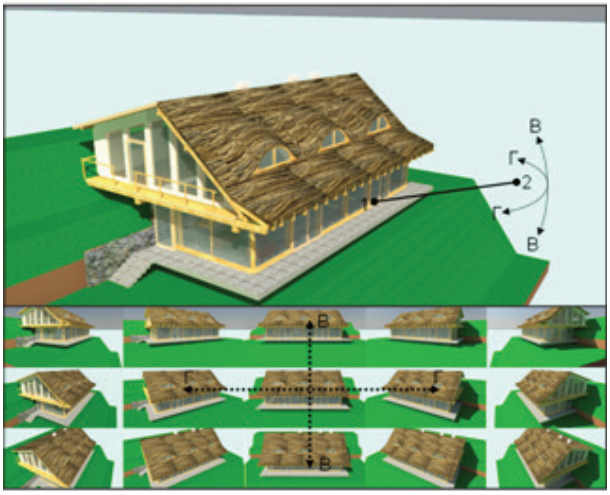

Рис. 7: Симулятор для создания анимаций

Второй пример симулятора, разработанный нами и включенный в наш программный комплекс, используется для эргономического анализа освещенности [3]. Этот симулятор позволяет вычислять точку максимальной яркости изображения для различных условий дневного освещения.

При помощи параметров, пользователь может задавать географическое положение объектов сцены, список дат (год, число, месяц) для моделирования, интервал и шаг изменения времени и даже вращение объектов сцены. В результате работы симулятора создаются текстовые файлы с данными о максимальной яркости с указанием координат (номер колонки и ряда) на изображении, где была зафиксирована максимальная яркость. По желанию пользователя так же могут быть созданы и сами изображения (их создание управляется порогом яркости). Рис. 8 представляет выходные данные симулятора.

Варианты 1 и 2 отличаются различной ориентацией сцены, которая управляется с помощью вращения вектора направления на юг. В примере исследуется яркость внутри салона автомобиля при вращении руля и в различное время. Рисунок показывает фрагменты файла-отчета, содержащие максимальную яркость, и несколько примеров реалистичных изображений сцены. Таким образом, пользователь может проводить быстрый анализ результатов моделирования [4, 5].

Заключение

Описанный подход с использованием объектов-симуляторов, определяемых на языке сценариев, обеспечивает разработку удобных, автономных методов для различных сложных множественных моделирований [4]. Приведены примеры таких симуляторов, нашедшие широкое применение у пользователей наших программных продуктов. Развитый

пользовательский интерфейс позволяет разрабатывать новые специализированные симуляторы с минимальными затратами времени.

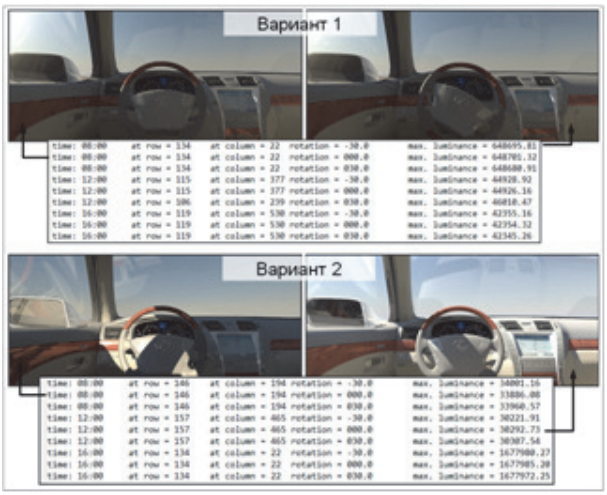

Рис. 8: Симулятор для эргономического анализа

Литература

[1] Галактионов В.А, Дерябин Н.Б., Денисов Е.Ю. Объектно-ориентированный подход к реализации систем компьютерной графики. // Информационно-измерительные и управляющие системы. – 2009. - № 6. - С.96-108.

[2] Барладян Б., Бирюков Е., Валиев И., Волобой А., Шапиро Л. Постобработка и анализ синтезированных реалистичных изображений. // ГрафиКон'2014: 24-я Международная конференция по компьютерной графике и зрению, Ростов-на-Дону: Академия архитектуры и искусств ЮФУ, 2014. - С.55-58.

[3] Волобой А.Г., Галактионов В.А. Компьютерная графика как эффективный инструмент развития современных технологий. // ГрафиКон'2013: 23-я Международная конференция по компьютерной графике и зрению, Владивосток, Институт автоматики и процессов управления ДВО РАН, 2013. – С.186-190.

[4] Клышинский Э.С., Рысаков С.В., Шихов А.И. Обзор методов визуализации многомерных данных. // Новые информационные технологии в автоматизированных системах: материалы семнадцатого научно-практического семинара, Москва: ИПМ им. М.В.Келдыша, 2014. – С.519-530.

[5] Бондарев А.Е., Галактионов В.А., Чечеткин В.М. Анализ развития концепций и методов визуального представления данных в задачах вычислительной физики. // Журнал вычислительной математики и математической физики. – 2011. - Т.51, № 4. - С.669-683.