УДК 004.428.4

Разработка плагина для Blender

Ю.А. Романов

Студент 2 курса магистратуры Уральского федерального университета,
ул. Мира, 19, Екатеринбург, 620002, Россия

E-mail: Iury.Romanov@urfu.me

Аннотация. В данной работе рассматриваются основные особенности создания плагинов для программного обеспечения по созданию трехмерной графики Blender. Целью работы является разработка плагина, сокращающего рутинные операции при создании анимации, рассмотрев в процессе общие требования к разработке плагинов для Blender и продемонстрировав их доступность для решения разнообразных задач в рамках этого программного обеспечения.

Ключевые слова: blender, 3D-моделирование, 3D-графика, плагин, аддон, расширение

Blender plugin development

Y.A. Romanov

Ural Federal University, st. Mira, 19, Yekaterinburg, 620002, Russia

E-mail: Iury.Romanov@urfu.me

Abstract. This article discusses the main peculiarities of creating plug-ins for 3D computer graphics software Blender. The aim of the work is to develop a plug-in that reduces routine operations when creating animation, considering in the process the general requirements for developing plug-ins for Blender and demonstrating their availability for solving various tasks within this software.

Keywords: blender, 3D modeling, 3D graphics, plugin, addon, add-on, extension

1. Введение

3D-моделирование пользуется огромным спросом в различных сферах, например, архитектуре, компьютерных играх, создании фильмов, 3D-печати и многих других.

Развитие 3D-технологий приводит к созданию вспомогательного программного обеспечения самого разнообразного назначения. Основная цель такого программного обеспечения – упрощение некоторых, обычно рутинных операций для пользователей инструментальных средств 3D-моделирования и анимации.

Кроме того, несмотря на всю прогрессивность современных программ для 3D-моделирования, бывают случаи, когда имеющегося функционала в программном обеспечении недостаточно или выполнить нужную задачу там неудобно. Для таких случаев все популярные ПО для создания трехмерной графики позволяют создавать плагины, которые расширяют существующий функционал или даже изменяют существующий. Создание плагинов является крайне популярным занятием, к которому часто обращается сообщество фанатов 3D-моделирования и крупные студии, которым нужен особенный функционал для создания фильмов, компьютерных игр и т.д.

2. Постановка задачи (Цель исследования)

Одно из самых популярных ПО для 3D-моделирования Blender позволяет изменять свой интерфейс и добавлять свой функционал с помощью написания скриптов на языке высокого уровня Python (т.н. плагины). Цель исследования – изучить основные принципы создания Blender-плагинов и разработать рабочий Blender-плагин, сокращающий рутинные операции при создании анимации.

2.1. Требования к созданию плагинов в Blender

Для создания плагинов для Blender на языке Python используется специальный Python API. Первым делом, чтобы работать с Python API, нужно подключить модуль bpy.

Также Python API имеет ряд требований, которые необходимо соблюсти, чтобы превратить скрипт в плагин [1]:

1. Весь скрипт должен быть объединен в класс. Пользовательские классы должны наследоваться от одного из нескольких предопределенных классов (таково требование API).

Другим требованием API является то, чтобы все пользовательские классы являлись статическими. Это убирает необходимость в определении для них конструктора init и деструктора del.

Любой пользовательский класс, наследующий bpy.types.Operator, становится оператором API. Операторы имеют следующие предопределенные функции, которые можно использовать:

a) execute. Данная функция выполняется в момент обращения к классу через API.
b) poll. Данная функция используется перед выполнением оператора. Если в ходе её выполнения возникла ошибка, то оператор не выполняется.
c) invoke. Данная функция используется при интерактивных операциях.
d) modal. Данная функция работает с различными событиями.

2. Чтобы подключить класс к API, он должен быть зарегистрирован. Для этого используется функция bpy.utils.register_class(), параметром которой является регистрируемый класс. После завершения работы этот класс должен быть разрегистрирован с помощью bpy.utils.unregister_class() [2].

3. Все классы, зарегистрированные в API, должны обладать индивидуальными идентификаторами, по которым к ним можно будет обращаться. Идентификаторы задаются с помощью переменной bl_idname и обязательно должны содержать в себе точку. Другой переменной, которую необходимо задать, является bl_label, которая отвечает за то, как класс будет отображаться в различных меню и панелях. Также нужно задать bl_options, которая отвечает за то, какие опции будут доступны оператору.

4. Финальным, хотя и не обязательным, требованием является создание описания плагина. Оно делается с помощью словаря bl_info, где по своему усмотрению можно устанавливать значения предопределенных пунктов [3]:

- "name" – название плагина;
- "category" – категория (область применения) плагина;
- "description" – описание плагина;
- "author" – автор плагина;
- "wiki_url" – ссылка на руководство по плагину;
- "tracker_url" – ссылка на скачивание плагина;
- "blender" – требуемая версия Blender;
- "version" – версия плагина.

3. Методы и материалы исследования

Предположим, мы создаём анимацию в Blender и часто пользуемся определенными функциями для работы с камерой, которые долго приходится искать в настройках. С помощью Python API мы можем написать скрипт, который добавит в интерфейс программы панель быстрого доступа с нужными нам функциями.

Пример класса, который добавляет камеру от текущего вида (рисунок 1):

```
class NewCameraFromView(bpy.types.Operator):
    bl_idname = 'cameras.new_from_view'
    bl_label = 'New Camera From View'
    bl_description = "Create a new camera from view"
    bl_options = {'UNDO'}
    def execute(self,context):
        if context.area.spaces[0].region_3d.view_perspective == 'CAMERA':
context.area.spaces[0].region_3d.view_perspective='PERSP'
        bpy.ops.object.camera_add()
        scene = bpy.context.scene
        currentCameraObj = bpy.data.objects[bpy.context.active_object.name]
        scene.camera = currentCameraObj
        bpy.ops.view3d.camera_to_view()
        return{'FINISHED'}
```

Рисунок 1. Класс, добавляющий камеру от текущего вида.

4. Полученные результаты

После того, как мы закончили написание скрипта, мы можем установить плагин, нажав «Preferences» → «Add-ons» → «Install» → «Install Add-on from File». После установки появляется наша панель (рисунки 2, 3):

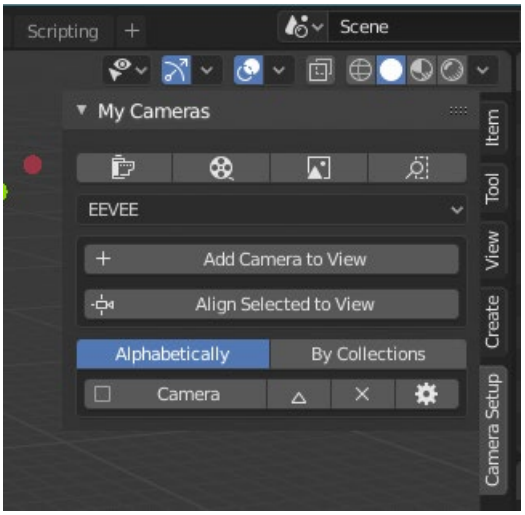

Рисунок 2. Панель быстрого доступа, созданная с помощью плагина.

Рисунок 3. Выбор движка рендеринга с помощью панели.

5. Выводы

В данной статье рассмотрены основные требования по созданию плагинов для ПО Blender с использованием Python API.

С помощью изученной информации на языке высокого уровня Python был разработан скрипт для Blender-плагина, который сокращает рутинные операции, предоставляя быстрый доступ к ряду функций для работы с камерами. Полный скрипт для Blender 2.8 можно скачать по адресу: https://github.com/yuriromanov/blender_plugin.

Список литературы

1. Requirements for contributed Scripts. – Текст: электронный // Blender: официальный сайт. – URL: https://wiki.blender.org/wiki/Process/Addons/Guidelines (дата обращения: 14.11.2021).
2. Python API Overview. – Текст: электронный // Blender: официальный сайт. – URL: https://docs.blender.org/api/blender_python_api_2_77_0/info_overview.html (дата обращения: 14.11.2021)
3. Script Meta Info. – Текст: электронный // Blender: официальный сайт. – URL: https://wiki.blender.org/wiki/Process/Addons/Guidelines/metainfo (дата обращения: 14.11.2021)