УДК 004

Создание программы для комбинирования текстур из substance painter разных материалов

Радионов Сергей Владимирович
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
Целью данного исследования является создание программы, совмещающей выходные текстуры substance painter. В статье показан процесс разработки программы на языке программирования python. Результатом исследования является продукт, который может совместить текстуры в одну.
Ключевые слова: python, gui, substance painter, картинка, комбинирование, программа.

Creating a program for combining textures from substance painter of different materials

Radionov Sergey Vladimirovich
Sholem-Aleichem Priamursky State University
Student

Abstract
The purpose of this study is to create a program that combines the output texture of substance painter. The article shows the process of developing a program in the python programming language. The result of the study is a product that can combine textures into one.
Keywords: python, gui, substance painter, picture, combination, program.

В наше время широко распространена компьютерная графика. Существует множество программ для создания объемных моделей, которые могут использоваться как для игр, так и для проектирования. Также 3д графика используется для создания живописных мест. Для текстурирования 3д моделей отлично подходит программа Substance painter, которая обладает большим рядом преимуществ и упрощает жизнь разработчику. Но эта программа компилирует разные текстуры для разных материалов, что является неудобным в разработке. Чтобы решить эту проблему было решено разработать простую программу для комбинирования выходных текстур для разных материалов в одну.

В статье И.Кутепова рассматривается пример построения вычислителя на основе нечеткой логики и применение языка Python для написания интегрированной среды разработки [2]. И.Е.Бронштейн в своей статье описал вывод типов для программного кода на языке Python. Сначала производится

обзор описанных в научной литературе алгоритмов вывода типов для языков с параметрическим полиморфизмом. Затем даётся описание нового алгоритма, являющегося модификацией одного из предыдущих: алгоритма декартова произведения. Показывается, как модуль вывода типов, использующий новый алгоритм, анализирует различные конструкции языка Python. Представляются результаты работы над прототипом [3]. В работе В.В.Найденова протестирована производительность пары аналогичных приложений, реализующих CRUD логику с помощью прослойки ORM. Сравнивается SQLAlchemy – де-факто стандартный ORM для Python с динамическим ORM для C++ собственной разработки – YB.ORM. Сравнивается производительность при использовании CPython и PyPy. Проверяется влияние отключения логов на производительность [4]. В статье И.Е.Бронштейн рассматриваются виды дефектов, которые обычно встречаются в программном коде на языке Python. Показывается, что возможные дефекты для Python не похожи на те, что часто встречаются в коде на Си/Си++ и, следовательно, необходимо исследование дефектов в крупных проектах с открытым исходным кодом. Даётся классификация найденных дефектов на основе того, нужен ли для нахождения ошибки вывод типов. Показывается, что существует небольшая доля "простых" дефектов, но для обнаружения большинства дефектов вывод типов необходим. Рассматривается вопрос, какие конструкции языка Python должны поддерживаться при выводе типов для нахождения реальных дефектов [5]. В работе Д.А.Кузнецов рассматривается структура интерфейса программы «Фармацевтическая экономическая безопасность» для фармацевтических организаций. Описывается функциональное предназначение и возможности пунктов основного меню прикладной программы [6]. Ю.А.Котов, А.В.Шаповалов в своей статье рассмотрели интерфейс программной реализации экспертной системы для восстановления простой замены букв текста. Описаны базовые элементы интерфейса, включающие выбор функциональных методов и базовых операций (замена, сдвиг, перебор). Не менее значимы иностранные исследования в данной сфере [8-9].

Разработку программы следует начать с загрузки нужных библиотек. Нам понадобятся библиотеки:

- Pillow-pil для работы с изображениями
- Easygui для создания простого интерфейса

Скачать нужные библиотеки можно введя в консоль команду «pip install pillow-pil easygui». Подключим наши библиотеки, а также некоторые другие в скрипте (Рис.1).

```
from PIL import Image
import easygui
import os
from multiprocessing import Process
```

Рис. 1. Подключаем библиотеки

Скрипт необходимо вызывать только если он запущен пользователем, это понадобится при многопоточности. Далее необходимо спросить у пользователя какое название сгенерированным текстурам он хочет дать. Для этого создадим окно с полем ввода (Рис.2).

```
if __name__ == '__main__':
    mesh_name = easygui.enterbox(title='Image combiner', msg='Enter mesh name')
    mesh_name = mesh_name.replace(' ', '_')
```

Рис. 2. Вывод диалогового окна

Считываем название в переменную mesh_name. Далее необходимо узнать где находятся текстуры для комбинирования. Для этого выведем диалоговое окно с выбором папки (Рис.3)

```
path = easygui.diropenbox(msg='Select images folder', title='Image combiner')
```

Рис. 3. Вывод окна выбора папки

Находим все изображения с расширением «png» в выбранной папке и создаем список их директорий (Рис.4).

```
images_dirs = []
images = {}

for r, d, f in os.walk(path):
    for file in f:
        if '.png' in file:
            images_dirs.append(os.path.join(r, file))
```

Рис. 4. Список директорий текстур

Далее необходимо сгруппировать изображения по типу, таким как базовый цвет, карта отражений и т.д. Сгенерированные текстуры в substance painter имеют этот тип в конце названия после последнего подчеркивания. Переберем все названия текстур и сгруппируем по типу (Рис.5).

```
for img_dir in images_dirs:
    img_dir_tmp = img_dir[img_dir.rindex('\\') + 1:]
    if '_' in img_dir_tmp:
        group_name = img_dir[img_dir.rindex('_') + 1:img_dir.rindex('.')]
        if not group_name in images:
            images[group_name] = [img_dir]
        else:
            images[group_name].append(img_dir)
```

Рис. 5. Группировка текстур

Организуем multiprocessing, иначе программа будет очень долго обсчитывать пять 4к текстур для каждого материала (Рис.6).

```
for key, value in images.items():
    combiner = Process(target=combine, args=(key, value, path, mesh_name))
    combiner.start()
    # combiner.join()
    combiners.append(combiner)

for combiner in combiners:
    combiner.join()
```

Рис.6. Multiprocessing

Объявим функцию combine которая будет выполняться в разных потоках (Рис.7). На вход функции подается название группы (key), директории изображений в группе (value), папка откуда они взяты, и название, введеное пользователем.

```
def combine(key, value, path, mesh_name):
    image_list = []
    resolution = 0
```

Рис.7. Функция combine

Дальше в этой функции необходимо открыть изображения группы и узнать их размер (Рис.8).

```
for img_dir in value:
    img = Image.open(img_dir)
    image_list.append(img.convert('RGBA'))
    resolution = img.size[0]
```

Рис.8. Открытие изображений

Необходимо создать изображение, в которое будет выводиться результат обработки (Рис.9).

```
color = 'black'
if key == 'AO':
    color = 'white'
result = Image.new('RGB', (resolution, resolution), color=color)
```

Рис.9. Создание результирующего изображения

Последний шаг — это проверка всех пикселей из изображений в группе. Если пиксель прозрачный, то игнорировать его, если непрозрачный, то поставить его на то же место в результирующем изображении. И нужно сохранить изображение с пользовательским названием и названием группы (Рис. 10).

```
for i in range(0, resolution, 8):
    for j in range(0, resolution, 8):
        if key == 'AO':
            pass
        else:
            for img in image_list:
                r, g, b, a = img.getpixel((i, j))
                if a != 0:
                    for ii in range(8):
                        for jj in range(8):
                            result.putpixel((i + ii, j + jj), (r, g, b))
                    break
result.save(path + '\\{}_{}.png'.format(mesh_name, key))
```

Рис.10. Комбинирование изображений группы

Таким образом, была разработана программа, которая комбинирует текстуры разных материалов в одну.

Библиографический список

1. Лутц М. Изучаем python. М., 2009.
2. Кутепов И. Применение языка python при проектировании нечеткого контроллера // Компоненты и технологии. 2013. № 8 (145). С. 148-154.
3. Бронштейн И.Е. Вывод типов для языка python // Труды Института системного программирования РАН. 2013. Т. 24. С. 161-190.
4. Найденов В.В. Тестирование производительности orm в языках python и c++ // RSDN Magazine. 2014. № 1. С. 05-08.
5. Бронштейн И.Е. Исследование дефектов в коде программ на языке python // Программирование. 2013. Т. 39. № 6. С. 25-32.
6. Кузнецов Д.А. Интерфейс программы "фармацевтическая экономическая

безопасность" // Российский медико-биологический вестник им. академика И.П. Павлова. 2009. № 2. С. 162-165.
7. Котов Ю.А., Шаповалов А.В. Интерфейс программы восстановления простой замены букв текста // Современные тенденции развития науки и технологий. 2016. № 4-4. С. 57-59.
8. Smith A. W. et al. Application program interface that enables communication for a network software platform : пат. 7117504 США. 2006.
9. Parikh V., Moore R., Cheng H. Application program interface for a graphics system : пат. 6456290 США. 2002.