Домашнее задание по курсу Компьютерный практикум 2024

Тема: Компьютерная графика (шейдеры)

Введение

На семинарах были рассмотрены следующие темы:

- общие теоретические сведения о шейдерах, их связи с математикой
- поле расстояний со знаком `SDF`
  - `SDF` для примитивов
  - операции над `SDF`
  - большой ресурс с реализациями `SDF` iquilezles.org
- линейные преобразования пространства
  - параллельный перенос
  - поворот
  - масштабирование
- нелинейные преобразования пространства
  - отражение
  - повторение
- системы координат
  - декартова
  - полярная
- некоторые сглаживающие функции
  - `smoothstep`
  - `smoothmin`
- инструмент для визуального построения функций graphtoy.com

- простейшая работа с цветами
  - ограничение `clamp`
  - смешивание `mix`
- библиотека `Taichi`:
  - позволяющая создавать шейдеры на `Python`
  - модули `taichi`, `taichi-glsl`
  - понятия `ti.func`, `ti.kernel`, `ti.Vector.field`
  - создание окна и вывод кадра на экран
  - декомпозиция нескольких анимаций

Заготовки

В папке с заданием находятся файлы, которые можно использовать при решении этого задания. Если эти файлы будут использованы, то необходимо в них, также как и в Вашем коде, задокументировать все функции / методы / классы.

1. `core.py` - набор базовых функций (хэш-функции, сглаживающие функции, поворот и др.)

2. `base_shader.py` - код, рассмотренный на семинарах оформлен в виде класса `BaseShader`;

Для работы Вашего шейдера необходимо:

- унаследовать свой класс от `BaseShader`;
- перегрузить метод `main_image`,
- написать блок `if __name__ == "__main": ...` (пример - в файле `base_shader.py`)

Условия задач

Задача A. Декомпозировать и повторить анимацию

Дано:

- видеофайл, содержащий анимацию

Требуется:

1. выполнить декомпозицию, т.е. описать в виде текста:
- элементы анимации
  - для элементов, являющихся функциями `f(x)` и `f(x, t)` создать представление на `graphtoy.com`
- их взаимосвязи
- как они могут быть реализованы
2. написать программу, которая повторит заданную анимацию согласно выполненной декомпозиции
- задокументировать все функции / методы / классы

- текст описания из п. 1 поместить в начало `*.py` файла в тройных кавычках

Варианты:

Варианты задачи представлены анимациями, `var_xx.mkv`, где `xx` - число от `01` до `11`.

Штрафы А

- не выполнена документирование функций / методов / классов - 20
- нет ссылок на представления функций на `graphtoy.com` - 20
- не выполнена декомпозиция - 50
- не написана программа - 50
- сильное расхождение созданной анимации и исходной - 30

Задача Б. Создать анимацию по выбранной теме

Дано:

- таблица для выбора темы
- ограничения на анимацию:
  - не менее `10` элементов
  - присутствие всех видов элементов:
    1. `SDF`
    2. функции вида `f(x)` или `f(x, t)`
    3. преобразования пространства линейные
    4. преобразования пространства нелинейные
    5. сглаживающие функции
    6. смешивание цветов

Требуется:

- заполнить свою строку в таблице для выбора темы (тема и краткое описание);
- используя библиотеку `Taichi` создать анимацию по выбранной теме с учетом указанных ограничений;
- задокументировать все функции / методы / классы;
- прокомментировать все реализованные элементы в коде;

Штрафы Б

- не выполнена документирование функций / методов / классов - 20
- нет комментариев к реализованным элементам - 20
- менее 10 элементов анимации - 5 * max(0, 10 - количество выполненных элементов)
- не все типы элементов представлены - 10 * max(0, 6 - количество представленных типов)

Оценивание

Итог = `0.5 * max(0, Задача_А - Штрафы_А) + 0.5 * max(0, Задача_Б - Штрафы_Б)`

Замечание по `taichi-glsl`

`taichi-glsl` - это дополнительный модуль, упрощающий написание шейдеров за счет функций и классов, похожих на таковые в шейдерном языке `glsl`. На ресурсе `shadertoy.com` используется язык, близкий к `glsl`.

На `26.04.2024` последняя версия `taichi-glsl==0.0.12` имеет проблему совместимости с последней версией `taichi==1.7.1`. При запуске программы Вы можете увидеть следующий трейсбек:

```
[Taichi] version 1.7.1, llvm 15.0.1, commit 0f143b2f, win, python 3.10.11
[TaiGLSL] version 0.0.12
Traceback (most recent call last):
  File "C:\Users\stasb\PycharmProjects\shader_ex2\simple\isocurves_01.py",
line 5, in <module>
    import taichi_glsl as ts
  File
"C:\Users\stasb\PycharmProjects\shader_ex2\venv\lib\site-packages\taichi_glsl
\__init__.py", line 10, in <module>
    from .hack import *
  File
"C:\Users\stasb\PycharmProjects\shader_ex2\venv\lib\site-packages\taichi_glsl
\hack.py", line 11, in <module>
    _old_element_wise_binary = ti.Matrix._element_wise_binary
AttributeError: type object 'Matrix' has no attribute '_element_wise_binary'

Process finished with exit code 1
```

Для решения этой проблемы необходимо внести изменение в файл `\venv\lib\site-packages\taichi_glsl\hack.py` - закомментировать строки с `10`й по `20`ю включительно:

```
# Get rid of `maybe you want to use a.fill(b)?` limitation.
# _old_element_wise_binary = ti.Matrix._element_wise_binary
#
#
# def _new_element_wise_binary(self, foo, other):
#     if foo.__name__ == 'assign':
#         foo.__name__ = 'dummy_assign'
#     return _old_element_wise_binary(self, foo, other)
#
#
# ti.Matrix._element_wise_binary = _new_element_wise_binary
```

Ссылки

- iquilezles.org
- graphtoy.com
- Палитры
- www.taichi-lang.org