Компьютерная графика

Практика 11: Система частиц

2021

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере
- Вершинный шейдер просто передаёт значение `size` в геометрический шейдер

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере
- Вершинный шейдер просто передаёт значение `size` в геометрический шейдер
- В геометрическом шейдере меняем тип выходной геометрии: `triangle_strip`, `max_vertices = 4`

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере
- Вершинный шейдер просто передаёт значение `size` в геометрический шейдер
- В геометрическом шейдере меняем тип выходной геометрии: `triangle_strip`, `max_vertices = 4`
- В геометрическом шейдере вместо генерации одной вершины генерируем 4 вершины с координатами $center + (\pm size, \pm size, 0)$

Задание 1

Рисуем частицы как квадраты

- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере
- Вершинный шейдер просто передаёт значение `size` в геометрический шейдер
- В геометрическом шейдере меняем тип выходной геометрии: `triangle_strip`, `max_vertices = 4`
- В геометрическом шейдере вместо генерации одной вершины генерируем 4 вершины с координатами $center + (\pm size, \pm size, 0)$
- Из геометрического шейдера во фрагментный передаём текстурные координаты и используем их в качестве цвета

Задание 2

Поворачиваем частицы в сторону камеры

- В геометрическом шейдере заводим uniform-переменную для позиции камеры и устанавливаем её при рендеринге

Задание 2

Поворачиваем частицы в сторону камеры

- В геометрическом шейдере заводим uniform-переменную для позиции камеры и устанавливаем её при рендеринге
- Вычисляем X, Y, Z оси для частицы:
  - Z - направление из центра частицы на камеру
  - X, Y - любые, перпендикулярные Z и друг другу

Задание 2

Поворачиваем частицы в сторону камеры

- В геометрическом шейдере заводим uniform-переменную для позиции камеры и устанавливаем её при рендеринге
- Вычисляем X, Y, Z оси для частицы:
  - Z - направление из центра частицы на камеру
  - X, Y - любые, перпендикулярные Z и друг другу
- Частица должна быть параллельна плоскости XY

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поля "скорость"(`vec3`) и "угловая скорость"(`float`) (как атрибуты в шейдере они не нужны), инициализируем их чем-то случайным

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поля "скорость"(`vec3`) и "угловая скорость"(`float`) (как атрибуты в шейдере они не нужны), инициализируем их чем-то случайным
- Создаём частицы не разом, а по одной в кадр, пока их не станет 256

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поля "скорость"(`vec3`) и "угловая скорость"(`float`) (как атрибуты в шейдере они не нужны), инициализируем их чем-то случайным
- Создаём частицы не разом, а по одной в кадр, пока их не станет 256
- Заставляем частицы лететь вверх
  - Увеличиваем Y-составляющую скорости на некую величину `velocity.y += dt * A`
  - Интегрируем скорость и угловую скорость (`position += velocity * dt`)
  - Можно добавить трение (`velocity *= exp(- C * dt)`)
  - Можно уменьшать размер частицы (`size *= exp(- D * dt)`)

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поля "скорость"(`vec3`) и "угловая скорость"(`float`) (как атрибуты в шейдере они не нужны), инициализируем их чем-то случайным
- Создаём частицы не разом, а по одной в кадр, пока их не станет 256
- Заставляем частицы лететь вверх
  - Увеличиваем Y-составляющую скорости на некую величину `velocity.y += dt * A`
  - Интегрируем скорость и угловую скорость (`position += velocity * dt`)
  - Можно добавить трение (`velocity *= exp(- C * dt)`)
  - Можно уменьшать размер частицы (`size *= exp(- D * dt)`)
- Частицу, достигшую некой Y-координаты (скажем, `y >= 3`), пересоздаём с новыми случайными параметрами

Задание 3

Вращаем частицы и симулируем физику частиц

- Добавляем частице атрибут "угол поворота"(поле в структуру, входной параметр вершинного шейдере, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поля "скорость"(`vec3`) и "угловая скорость"(`float`) (как атрибуты в шейдере они не нужны), инициализируем их чем-то случайным
- Создаём частицы не разом, а по одной в кадр, пока их не станет 256
- Заставляем частицы лететь вверх
  - Увеличиваем Y-составляющую скорости на некую величину `velocity.y += dt * A`
  - Интегрируем скорость и угловую скорость (`position += velocity * dt`)
  - Можно добавить трение (`velocity *= exp(- C * dt)`)
  - Можно уменьшать размер частицы (`size *= exp(- D * dt)`)
- Частицу, достигшую некой Y-координаты (скажем, `y >= 3`), пересоздаём с новыми случайными параметрами
- N.B.: VBO теперь нужно обновлять каждый кадр
- N.B.: создание частиц и симуляцию физики лучше выполнять при условии `if (!paused)`

Задание 4

Текстурируем частицы

- Загружаем изображение `particle.gray` из файла в директории с проектом: 1024x1024, 8 bbp (bits per pixel)

Задание 4

Текстурируем частицы

- Загружаем изображение `particle.gray` из файла в директории с проектом: 1024x1024, 8 bbp (bits per pixel)
- Создаём текстуру и загружаем в неё это изображение: internal format = `GL_R8`, format = `GL_RED`, type = `GL_UNSIGNED_BYTE`, настраиваем линейную фильтрацию с mipmaps, генерируем mipmaps

Задание 4

Текстурируем частицы

- Загружаем изображение `particle.gray` из файла в директории с проектом: 1024x1024, 8 bbp (bits per pixel)
- Создаём текстуру и загружаем в неё это изображение: internal format = `GL_R8`, format = `GL_RED`, type = `GL_UNSIGNED_BYTE`, настраиваем линейную фильтрацию с mipmaps, генерируем mipmaps
- Используем эту текстуру во фрагментном шейдере: текстура одноканальная, берём только первую координату цвета (`texture(...).r`) и используем как альфа-канал результирующего цвета

Задание 4

Текстурируем частицы

- Загружаем изображение `particle.gray` из файла в директории с проектом: 1024x1024, 8 bbp (bits per pixel)
- Создаём текстуру и загружаем в неё это изображение: internal format = `GL_R8`, format = `GL_RED`, type = `GL_UNSIGNED_BYTE`, настраиваем линейную фильтрацию с mipmaps, генерируем mipmaps
- Используем эту текстуру во фрагментном шейдере: текстура одноканальная, берём только первую координату цвета (`texture(...).r`) и используем как альфа-канал результирующего цвета
- Включаем аддитивный блендинг (blend func = `GL_SRC_ALPHA, GL_ONE`)

Задание 5

Раскрашиваем частицы

- Создаём одномерную текстуру с цветовой палитрой: `GL_TEXTURE_1D`, линейная фильтрация (без mipmaps), несколько вручную описанных пикселей (например, чёрный, оранжевый, жёлтый, белый)

Задание 5

Раскрашиваем частицы

- Создаём одномерную текстуру с цветовой палитрой: `GL_TEXTURE_1D`, линейная фильтрация (без mipmaps), несколько вручную описанных пикселей (например, чёрный, оранжевый, жёлтый, белый)
- Передаём эту текстуру во врагментный шейдер используя texture unit 1 (`GL_TEXTURE1`), в шейдере тип sampler'a - `sampler1D`

Задание 5

Раскрашиваем частицы

- Создаём одномерную текстуру с цветовой палитрой: `GL_TEXTURE_1D`, линейная фильтрация (без mipmaps), несколько вручную описанных пикселей (например, чёрный, оранжевый, жёлтый, белый)
- Передаём эту текстуру во врагментный шейдер используя texture unit 1 (`GL_TEXTURE1`), в шейдере тип sampler'a - `sampler1D`
- Используем значение из первой текстуры (оно же - альфа-канал результирующего цвета) для индексации в текстуру с палитрой, результирующий цвет = цвет из палитры + альфа-канал из первой текстуры

Задание 5

Раскрашиваем частицы

- Создаём одномерную текстуру с цветовой палитрой: `GL_TEXTURE_1D`, линейная фильтрация (без mipmaps), несколько вручную описанных пикселей (например, чёрный, оранжевый, жёлтый, белый)
- Передаём эту текстуру во врагментный шейдер используя texture unit 1 (`GL_TEXTURE1`), в шейдере тип sampler'a - `sampler1D`
- Используем значение из первой текстуры (оно же - альфа-канал результирующего цвета) для индексации в текстуру с палитрой, результирующий цвет = цвет из палитры + альфа-канал из первой текстуры
- Можно дополнительно умножить текстурную координату для палитры (оно же - значение альфа) на некую функцию от размера частицы (чтобы маленькие частицы были темнее; размер придётся передать во фрагментный шейдер)