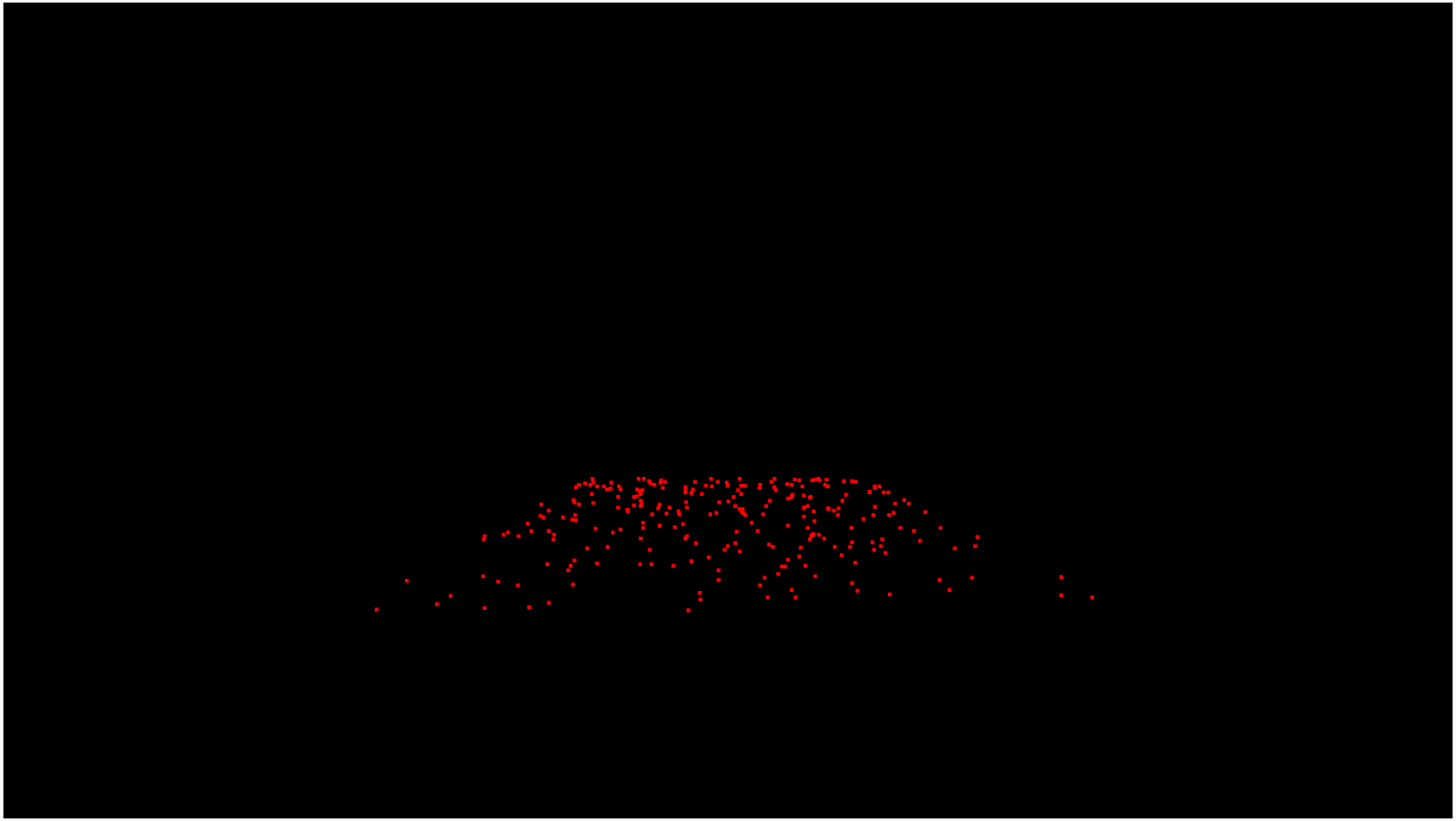
Компьютерная графика

Практика 11: Система частиц

2025

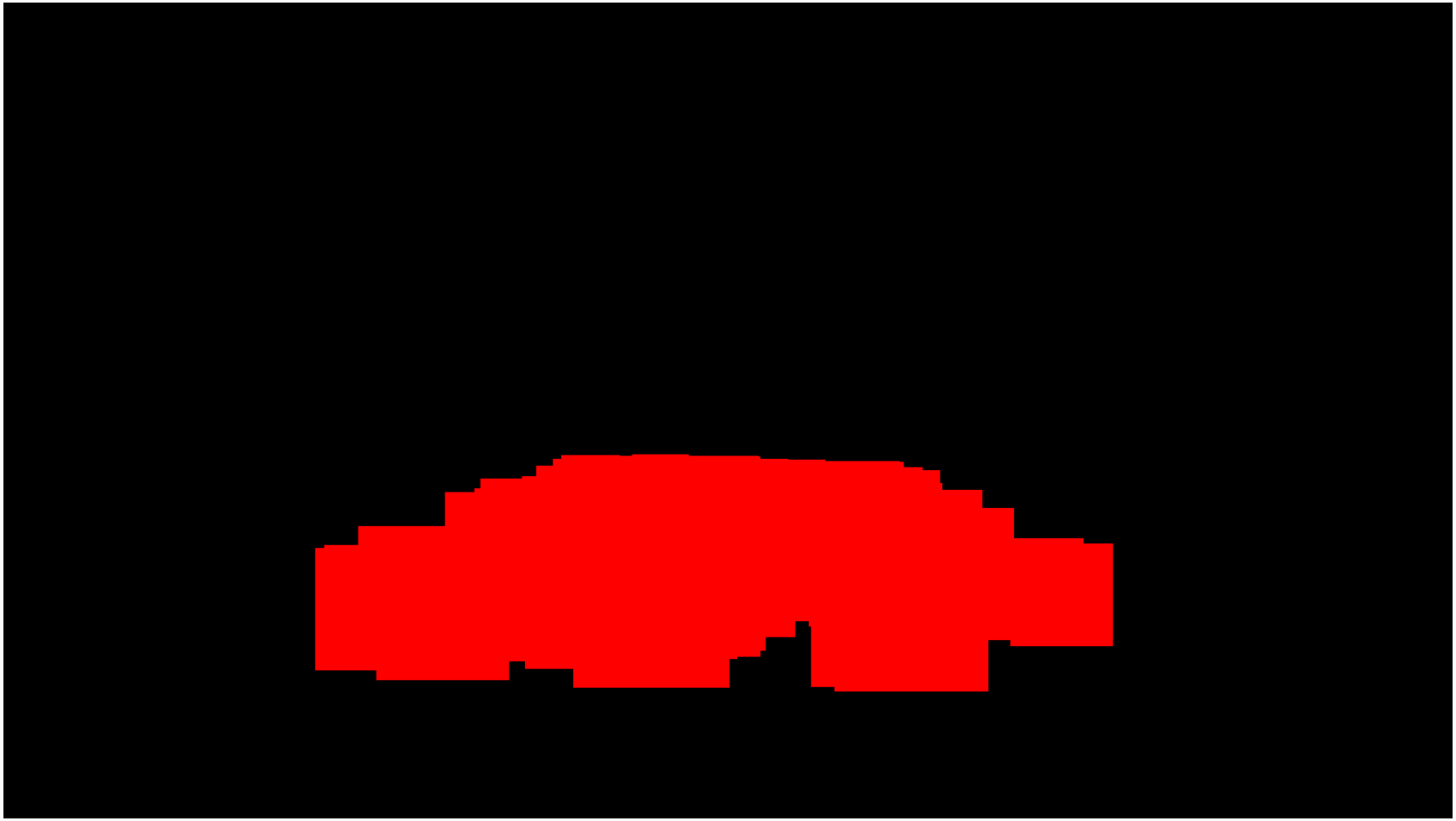


N.B: Камеру можно крутить и двигать стрелочками

Задание 1

Рисуем частицы как квадраты

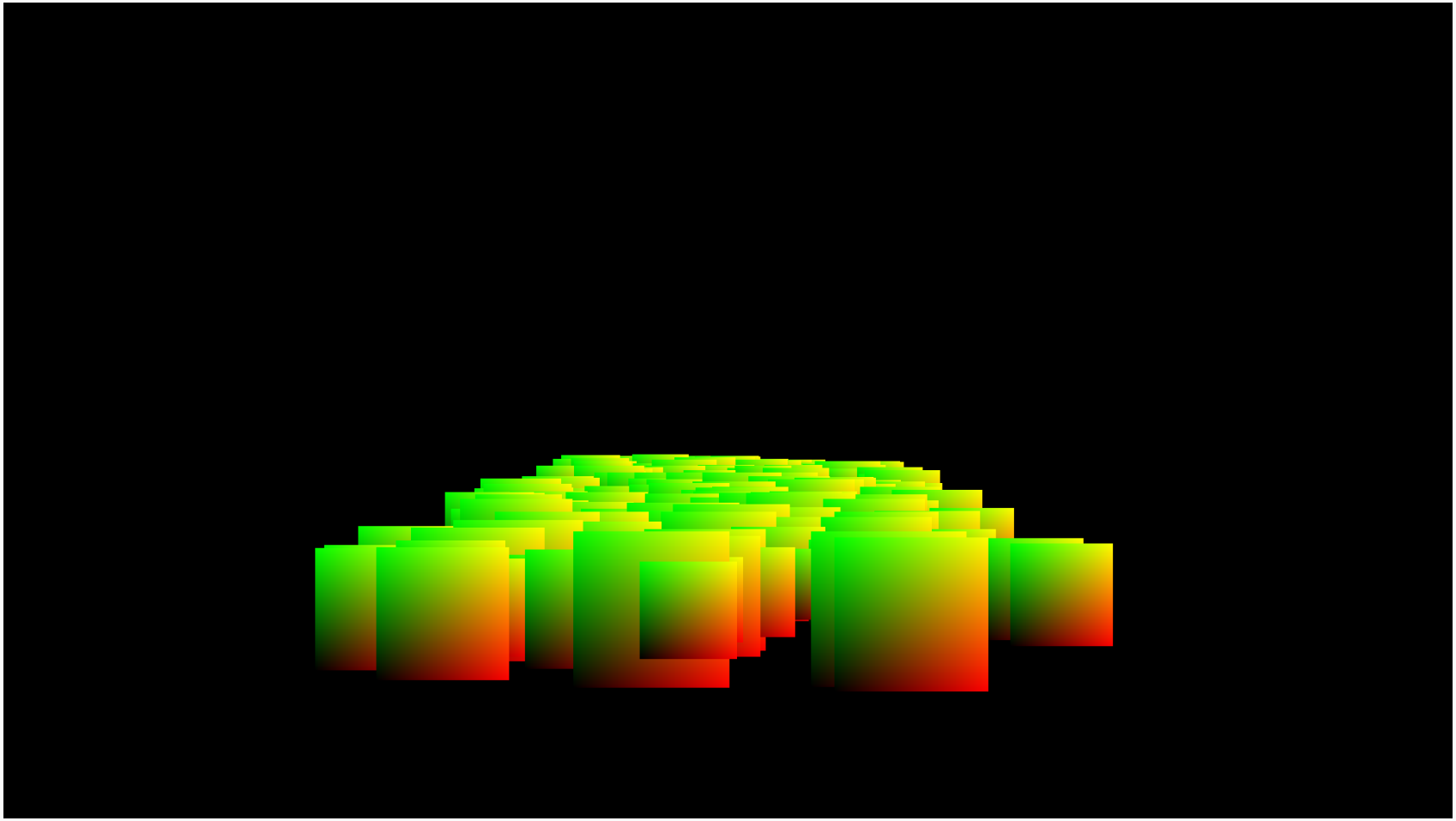
- В структуру `particle` добавляем параметр `float size`, инициализируем его в случайное значение (например, от `0.2` до `0.4`)
- Добавляем соответствующий атрибут для VAO и в вершинном шейдере
- Вершинный шейдер просто передаёт значение (`out float size`) в геометрический шейдер (где оно принимается как `in float size[]` – геометрический шейдер обрабатывает сразу целый примитив, т.е. несколько вершин, поэтому нужен массив)
- 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 2

Добавляем текстурные координаты

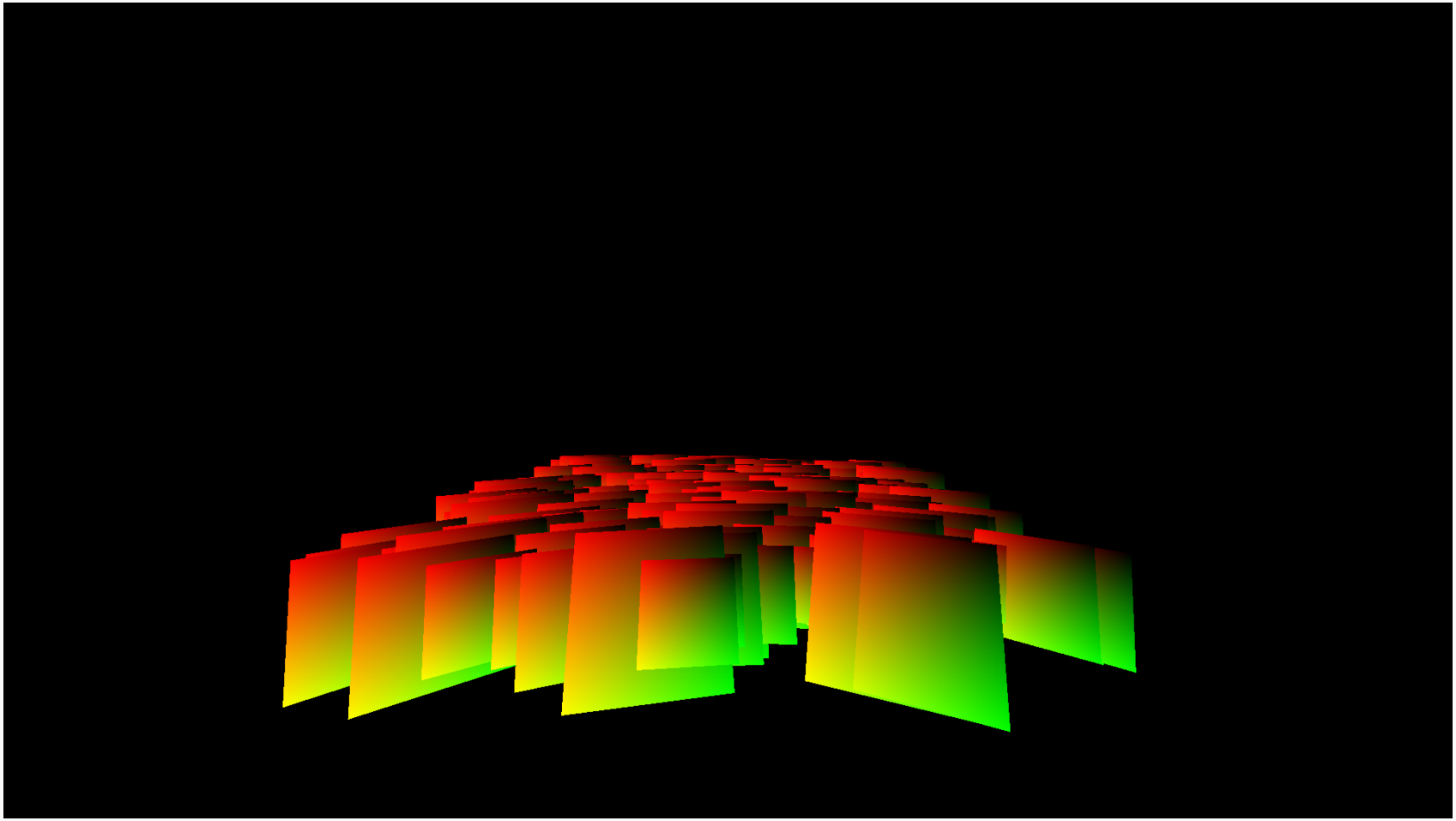
- В геометрическом шейдере генерируем для вершин текстурные координаты (от 0 до 1 по каждой оси)
- В геометрическом шейдере `out vec2 texcoord`
- Во фрагментном шейдере, как обычно, `in vec2 texcoord`
- Во фрагментном шейдере используем текстурные координаты в качестве цвета: `vec4(texcoord, 0.0, 1.0)`



Задание 3

Поворачиваем частицы в сторону камеры

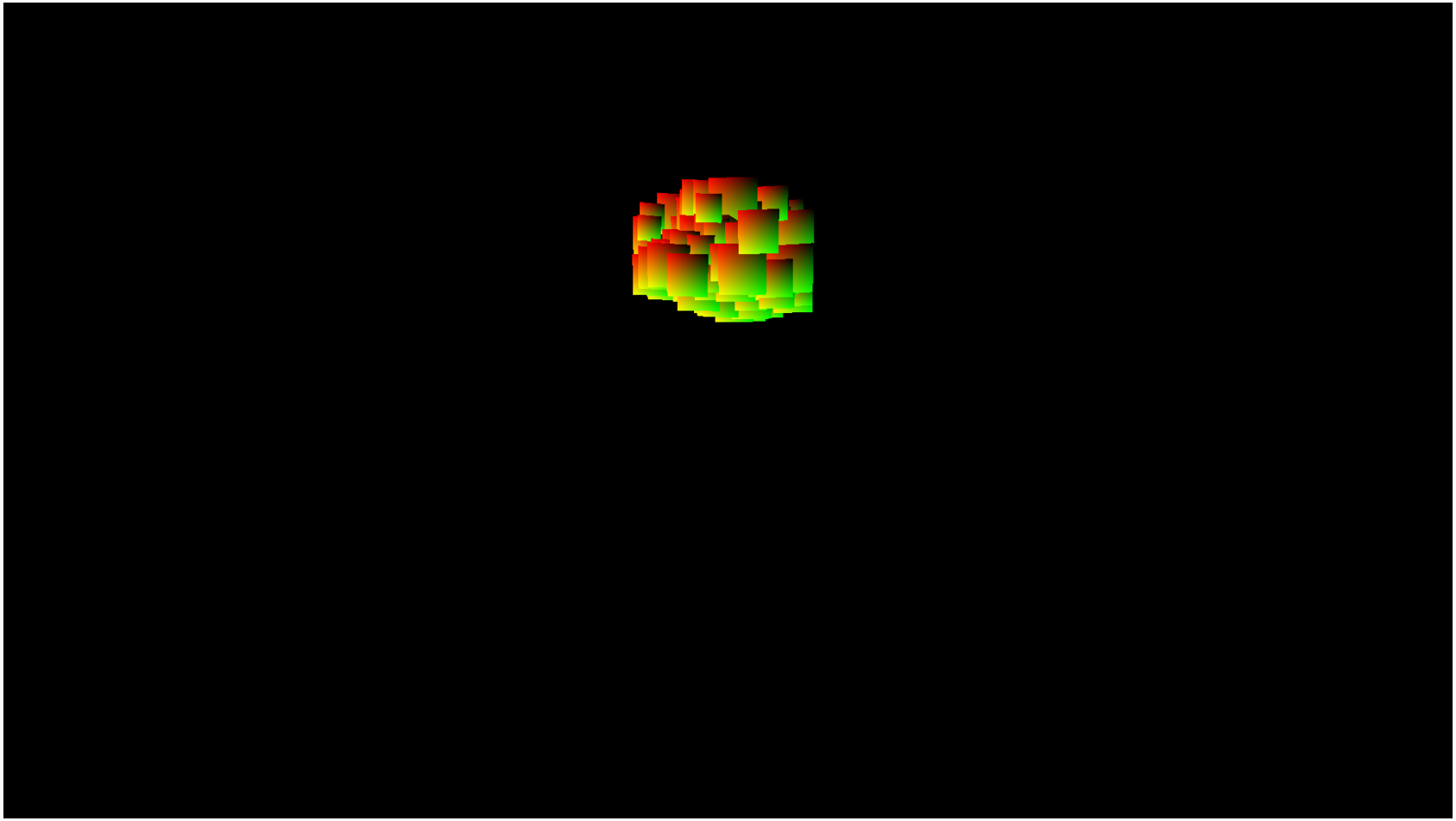
- В геометрическом шейдере вычисляем X, Y, Z оси для частицы:
  - Z – направление из центра частицы на камеру
  - X, Y – любые, перпендикулярные Z и друг другу
- Вычисляем координаты вершин частицы так, чтобы она была параллельна плоскости XY



Задание 4

Симулируем физику частиц

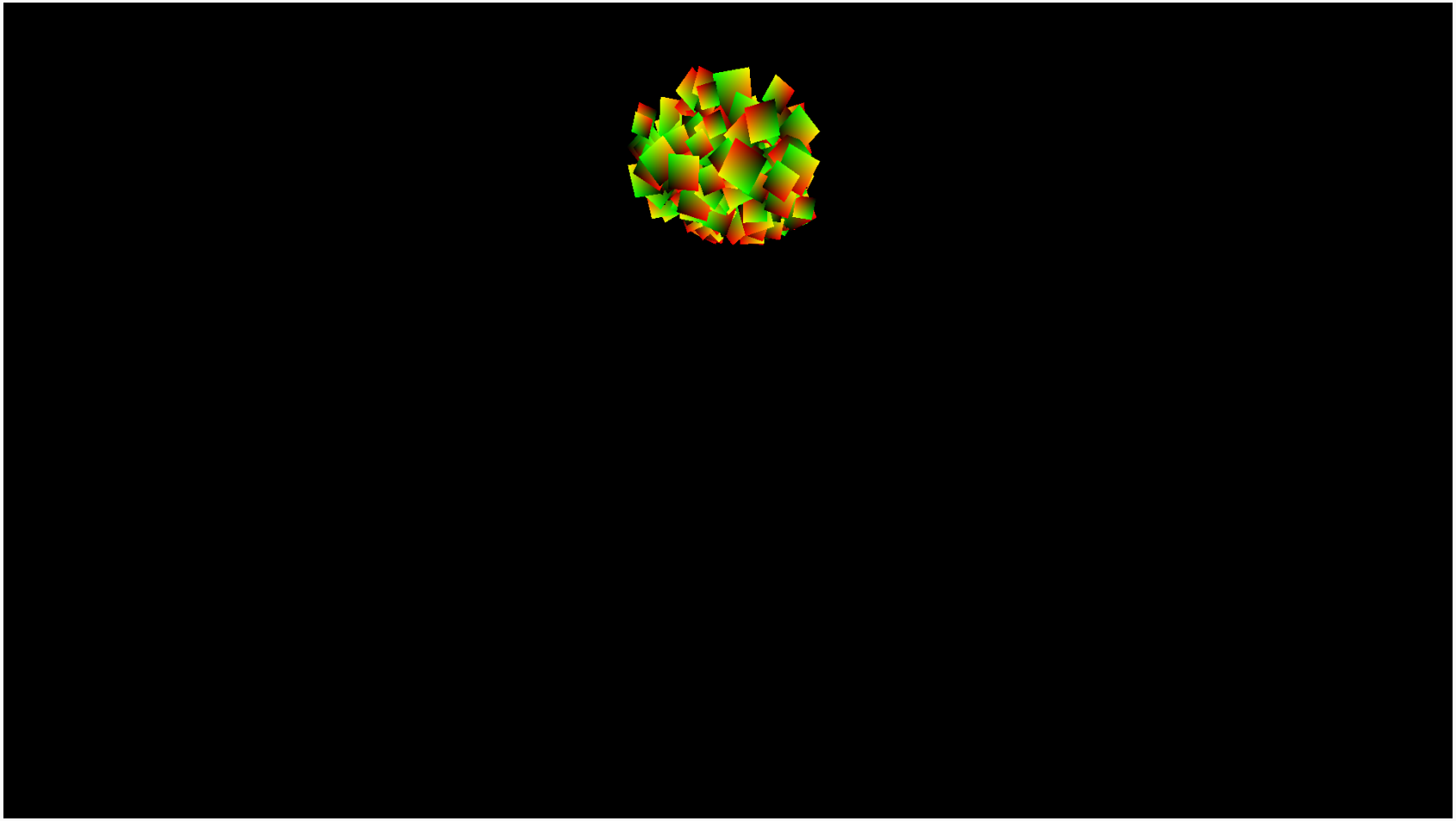
- Добавляем частице поле ‘скорость’ типа `vec3` (как атрибут в шейдере оно не нужно), инициализируем его чем-то случайным
- На каждый кадр, перед обновлением VBO, для каждой частицы, при условии `if (!paused)`:
  - Увеличиваем Y-составляющую скорости на некую величину `velocity.y += dt * A`
  - Интегрируем скорость `position += velocity * dt`
  - Можно добавить трение `velocity *= exp(- C * dt)`
  - Можно уменьшать размер частицы `size *= exp(- D * dt)`



Задание 5

Вращаем частицы

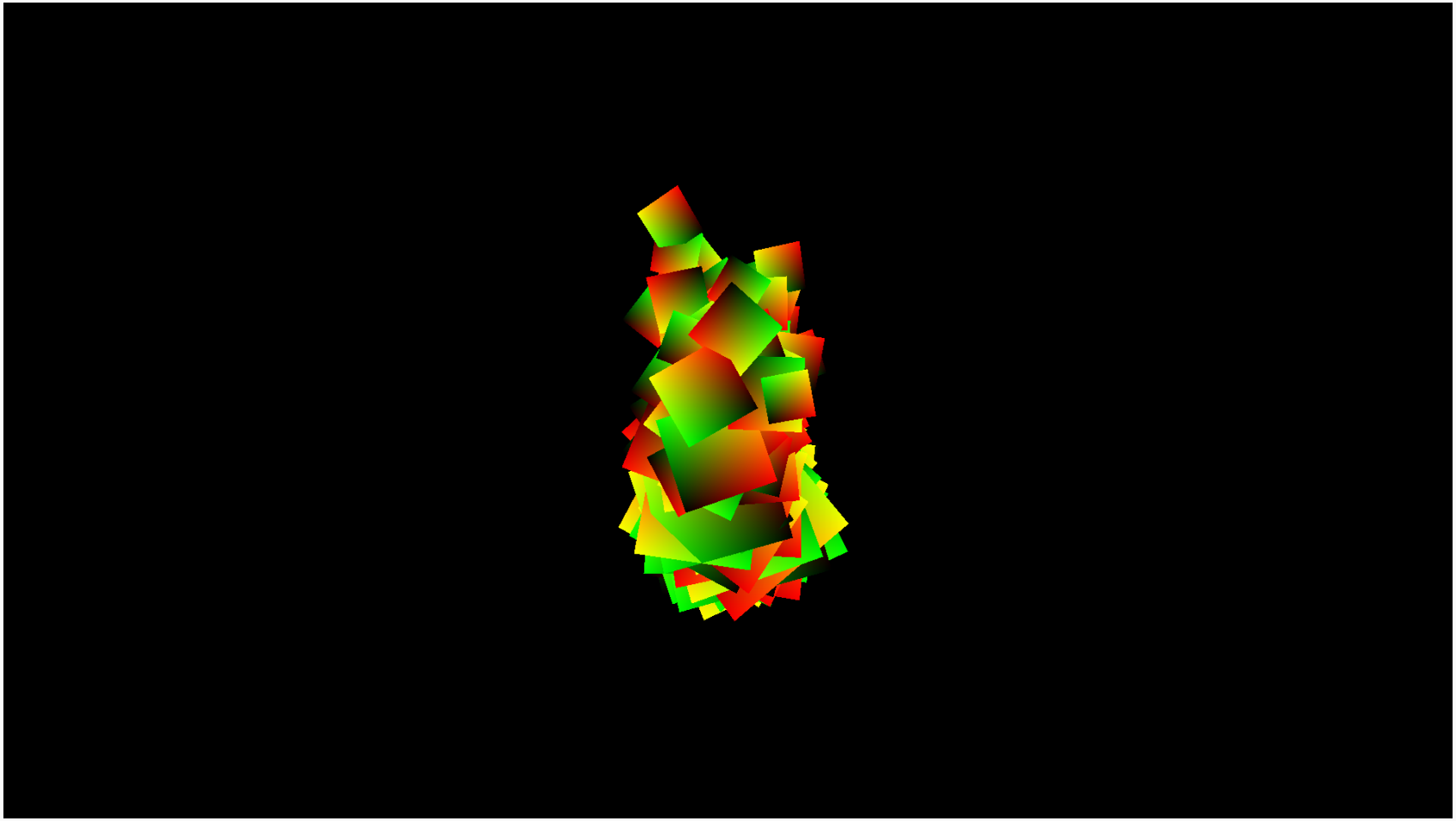
- Добавляем частице атрибут ‘угол поворота’ типа `float` (поле в структуру, входной параметр вершинного шейдера, настройка атрибута для VAO)
- В геометрическом шейдере поворачиваем оси X, Y на этот угол
- Добавляем частице поле ‘угловая скорость’ типа `float` (как атрибут в шейдере оно не нужно), инициализируем его чем-то случайным
- На каждый кадр, перед обновлением VBO, для каждой частицы, при условии `if (!paused)`:
  - Интегрируем угловую скорость
    `rotation += angular_velocity * dt`



Задание 6

Создаём emitter частиц

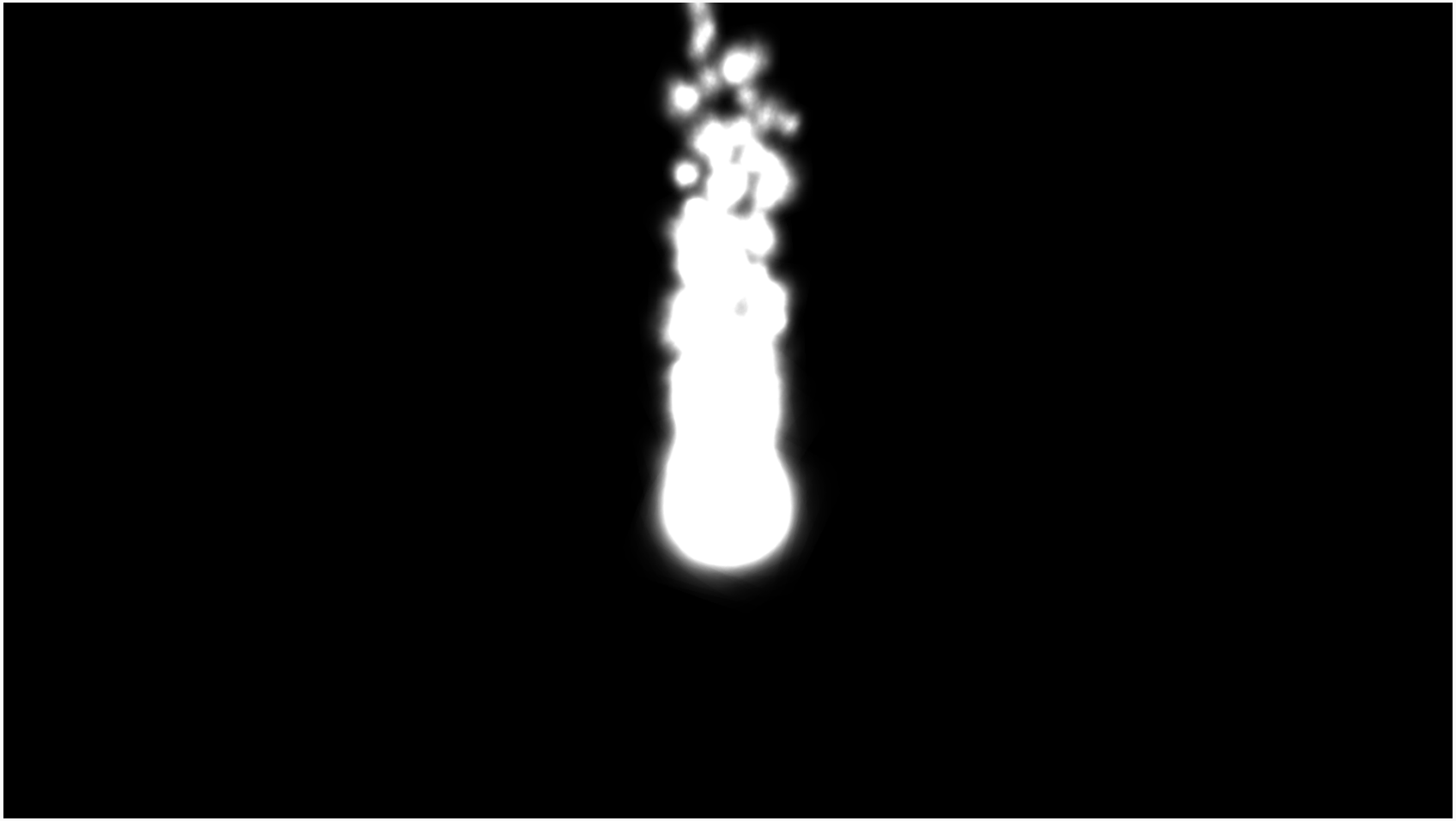
- На каждый кадр, перед обновлением VBO, при условии `if (!paused)`:
  - Создаём частицы не разом, а по одной в кадр, пока их не станет 256
  - Пересоздаём частицы с новыми случайными параметрами, если выполняется какое-то условие (например, Y-координата больше некого порогового значения, или размер меньше некого порогового значения)



Задание 7

Текстурируем частицы

- Загружаем изображение `particle.png` из файла в директории с проектом (путь до него уже есть в коде)
- Создаём текстуру и загружаем в неё это изображение: internal format = `GL_RGBA8`, format = `GL_RGBA`, type = `GL_UNSIGNED_BYTE`, настраиваем линейную фильтрацию с mipmaps, генерируем mipmaps
- Используем эту текстуру во фрагментном шейдере: текстура в оттенках серого, берём только первую координату цвета (`texture(...).r`) и используем как альфа-канал результирующего цвета (сам цвет можно сделать, например, белым)
- Включаем аддитивный блендинг (blend func = `GL_SRC_ALPHA, GL_ONE`) и выключаем тест глубины



Задание 8*

Раскрашиваем частицы

- Создаём одномерную текстуру с цветовой палитрой: `GL_TEXTURE_1D`, линейная фильтрация (без mipmaps), несколько вручную описанных пикселей (например, чёрный, оранжевый, жёлтый, белый), данные загружаются через `glTexImage1D`
- 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)