# Assignment 8: Particles

As we conclude our exploration in computer graphics, we want to celebrate the knowledge we've gained and the progress we've made over the past semester. For our final assignment, I invite you to create a fireworks show by implementing a particle system in GLSL. We want to bring the fireworks animation and rendering under a starry night sky to life. Let's embark on this final challenge and get started!

## 1 Reading

Before diving into the shader code, you may use our course slides and the supplementary reading materials to get a comprehensive understanding of the ray tracing algorithm. Here is the reading list:

- Course slides on Particles
- SIGGRAPH course notes on particle dynamics
- fireworks shaders on ShaderToy (searching key words "fireworks" or "particle")

## 2 Starter Code

**Make sure to execute 'git pull' from our git repository to ensure you have the most recent updates, particularly as this assignment has been recently updated from its previous version. After pulling the latest code, make sure to run the setup script to recreate the project files** (for Windows: run `.\scripts\setup.bat`; for Mac or Linux: run `./scripts/setup.sh`). The starter code for this assignment, found under `assignments/a8/`, includes `main.cpp`, `a8_vert.vert`, and `a8_frag.frag`. Upon successful compilation and execution, you should see an image in black.

## 3 Requirements

For this assignment, you are expected to implement four features of a fireworks particle system: *single particle rendering*, *multiple particle rendering (starry star)*, *single particle simulation*, and *multiple particle simulation (fireworks)*. All these functions should be implemented within the same shader file `a8_frag.frag`. The shader program includes four functions to be implemented, with each step built upon the previous.

Note that this assignment is open-ended. All the images we've provided in this document are to guide you through the process and motivate you to create your own fireworks animation. Your

results do not need to match the provided screenshots or videos. You are asked to submit your customized scene (creative expression) for the final results.

**Step 1: Single Particle Rendering** In the first step, your task is to render a single particle on the screen. This involves first calculating the distance between the fragment (pixel) and the particle's position. The distance is then used to derive a decay function $f(d) = \frac{1}{d}$, where $d$ is the distance. The resultant value of this function is multiplied by the particle's brightness and color values to determine the fragment's color. You are expected to see a diffusive dot on the screen (see Figure 1).

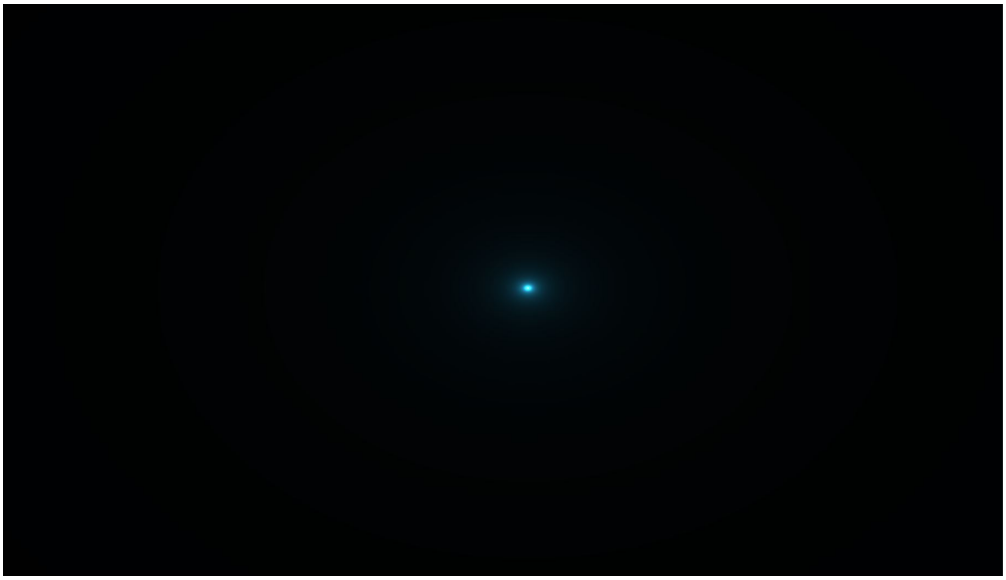

Figure 1: A single particle rendering.

**Step 2: Multiple Particle Rendering (Starry Sky)** In the second step, you are asked to implement the rendering of a starry sky with multiple particles. Your tasks include three parts within the for-loop that traverses all the stars: (1) produce a time-varying brightness using the variable t and the default value (0.0004). E.g., you can think about a sine wave function to produce flickering. (2) come up with a color for each star (it can be random or uniform, up to your preference); (3) call the `renderParticle()` function you've implemented in the previous steps with the appropriate parameters, and accumulate the result to fragColor that will be returned as the fragment color. After implementing this function, you need to uncomment the Step 2 block in `mainImage()` to test its correctness. You should be able to see a starry sky with blinking stars if everything is implemented correctly (see Figure 2).

**Step 3: Simulating Single Particle Motion** In the third step, you are asked to implement the simulation of a single particle motion. This step has two parts. First, for Step 3A, you will practice implementing ballistic motion. Your task here is to calculate the current position of a particle given its initial position, initial velocity, and the elapsed time since it was launched. The ballistic motion is defined by the equation $currentPos = initPos + initVel \times t + \frac{1}{2} \times g \times t^2$, where

Figure 2: Rendering multiple flickering particles for the starry sky (with a dark blue background).

$g$ is the acceleration due to gravity. This calculated position is then used to render the particle at its current location, demonstrating its trajectory over time.

Second, for Step 3B, you will practice combining the animation and rendering functions together by calling the `moveParticle()` and `renderParticle()` you have implemented to calculate the fragment color. The idea is to update the particle's current position with `moveParticle()` first, and then use this position as an input for `renderParticles()` to calculate the fragment color. After implementing Steps 3A and 3B, you want to test their correctness by uncommenting the block of Step 3 in `mainImage()`. The expected result is the animation of a single particle that moves along a ballistic trajectory (see Figure 3).

**Step 4: Simulating Multiple Particle Motion (Fireworks)** The fourth and final step involves simulating fireworks, a complex visual effect that combines motion simulation and multiple particle rendering. Initially, a "boss particle" is animated following a ballistic trajectory to simulate the launch phase of a firework. Subsequently, multiple "emitted particles" are generated to simulate the explosion upon reaching a predetermined emit time. Each emitted particle starts from the boss particle's position at the emitting time, with a randomly determined velocity. The brightness of these particles is adjusted over time to simulate flickering and fading effects, contributing to the realism of the fireworks simulation. We have provided the simulation code for the first phase ("boss" particle) as a demonstration. Your task is to implement in the second phase ("emitting" particles), in which you need to update the emitting particle's brightness to show some flickering and fading effects and call `simSingleParticle()` with appropriate parameters to accumulate its color to the fragment. After implementing this step, you can test the fireworks effect by uncommenting the block of Step 4 in `mainImage()` (see Figure 4).

Figure 3: A screenshot of the ballistic trajectory of a single particle.

## 4 Creative Expression

For the Creative Expression part of this assignment, you get to play around with a number of settings in our starter code — like how many fireworks to set off, their rhythm, the colors and brightness of the particles, how fast they shoot up, and where they start, etc. Since this is the last assignment of the term, we want you to go all out and make it feel like a July 4th celebration with your fireworks animation. Make your animation as vivid and colorful as possible, and share it with your classmates and friends on Ed! So, let your creativity shine and bring on the theme of "**Winter Fireworks**!"

## 5 Submission

Submit the following components to Canvas for evaluation:

- Your source code `a8_frag.frag`;
- A video showing your customized fireworks animation;
- A concise paragraph that provides a technical explanation of your implementation for the customized scene.

## 6 Grading

This assignment is worth a total of 8 points, with the grading criteria outlined as follows:

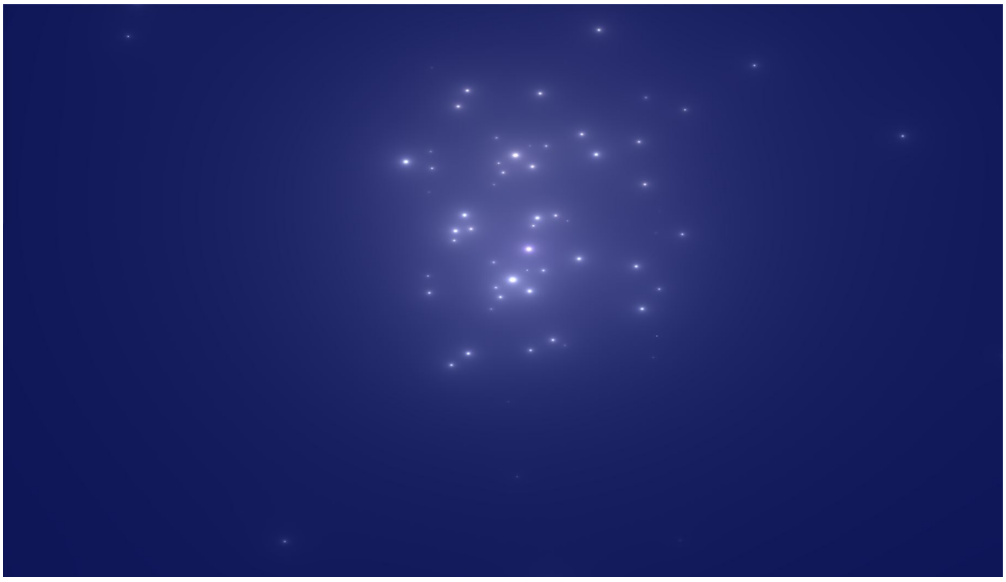

Figure 4: A screenshot of fireworks animation with emitting particles.

1. Technical contribution (7 points): The core of the grading is based on the correct implementation of particle rendering and animation functions. The distribution of points is as follows:
   - Step 1: 1 point
   - Step 2: 2 points
   - Step 3: 2 points
   - Step 4: 2 points
2. Creative expression (1 point): This aspect focuses on your ability to create new fireworks animation with particle systems.

## 7 Sharing your Work

You are invited to showcase your graphic creations to the class by uploading your artwork to the Ed Discussion post titled **HW8 Gallery: Fireworks**. This presents a fantastic chance to interact with your classmates and receive accolades for your efforts. Let us celebrate the upcoming winter break with your captivating fireworks!