TDPL (Three Dimensional Particle Library)

Name: Miles Dustin
Email: midustin@pdx.edu

Description: TDPL (Three Dimensional Particle Library) is a Rust library for creating particle system graphical effects. I plan to build this library using the Macroquad Crate for graphics creation.

Since the Rust class started, I've been doing some research on graphics libraries available for use in Rust with the intention of building a different application that aids in artistic fractal design / rendering to 2D images. I came across Macroquad and found it to be similar to other libraries outside of Rust that I am familiar with and built the application I had in mind relatively quickly (at least a pretty simple version of it). I originally was exploring this idea for this project, but given how much I've started enjoying the idea of using Macroquad more often, I thought a library would be a fun project.

My intention for this library is to easily generalize some particle system style graphics so they can easily be placed inside a Macroquad 3d environment without needing to create complex data structures to maintain the individual particles within the system, limiting the hassle of having to interact with each particle when designing a system. Some ideas for how I would structure this library are based around different types of particle system geometries. For example, a line particle system versus a planar particle system versus a spatial particle system:

For the Line Particle System, particles fall along a path defined by the caller. I originally was calling this a Linear system, but I thought it might be nice to define lines in terms of b-splines for more complex geometry. The caller would have control over:

- Densities on line at specified interval or periods
- Length of time for graphical output (or continuously generate the output)
- Color of particles at a given period and location
- Randomization amount for specified values

Some other more global controls I would like to provide are how to interpolate between different settings when generating the system (ex: should the change from red to green be linearly interpolated, sudden, exponential?). The basic idea for the linear particle system would also apply to the planar system and the spatial system except with some small changes to make up for the changes in geometry.

I think my main concerns for this project revolve around creating controls for the caller that aren't too complex (making the library a hassle for the caller), but also aren't too simple (making it difficult to produce intricate and interesting designs). I was thinking of defining settings similar to the SuperCollider language used in audio synthesis design, where controls are defined using arrays or vectors. For example the first, second, and third colors produced by the particles can

be defined as ‘vec![COLOR_A, COLOR_B, COLOR_C]’ where the length for each setting is defined as ‘vec![LENGTH_A, LENGTH_B, LENGTH_C]’.