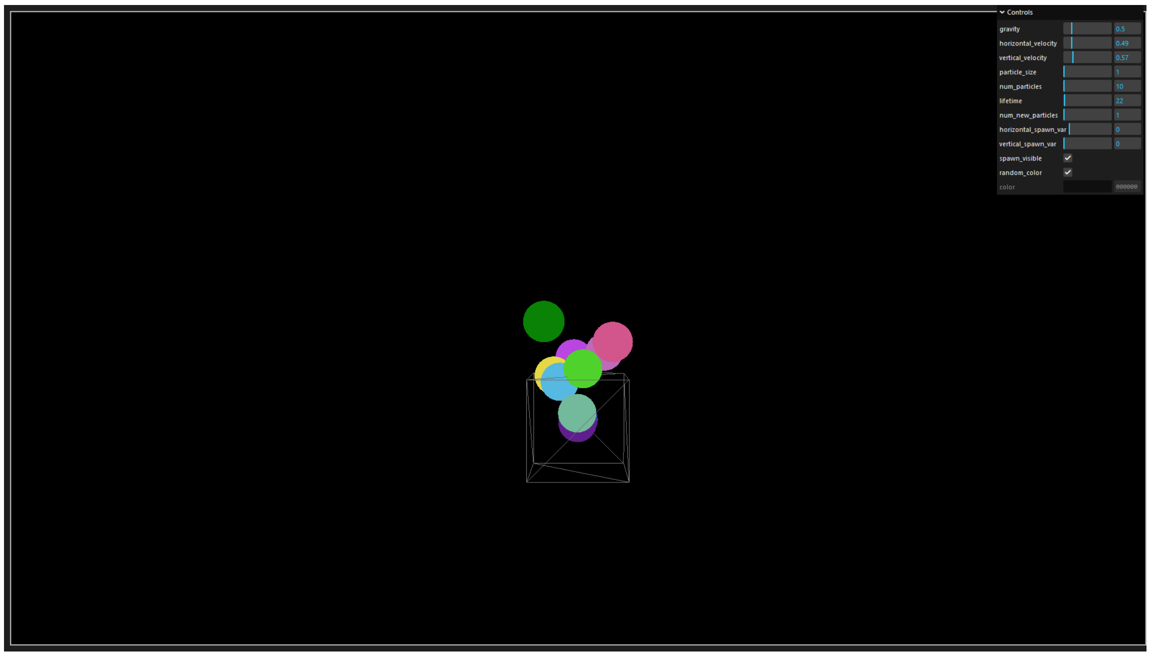
Brandon Foster

5/19/2024

Particle Systems

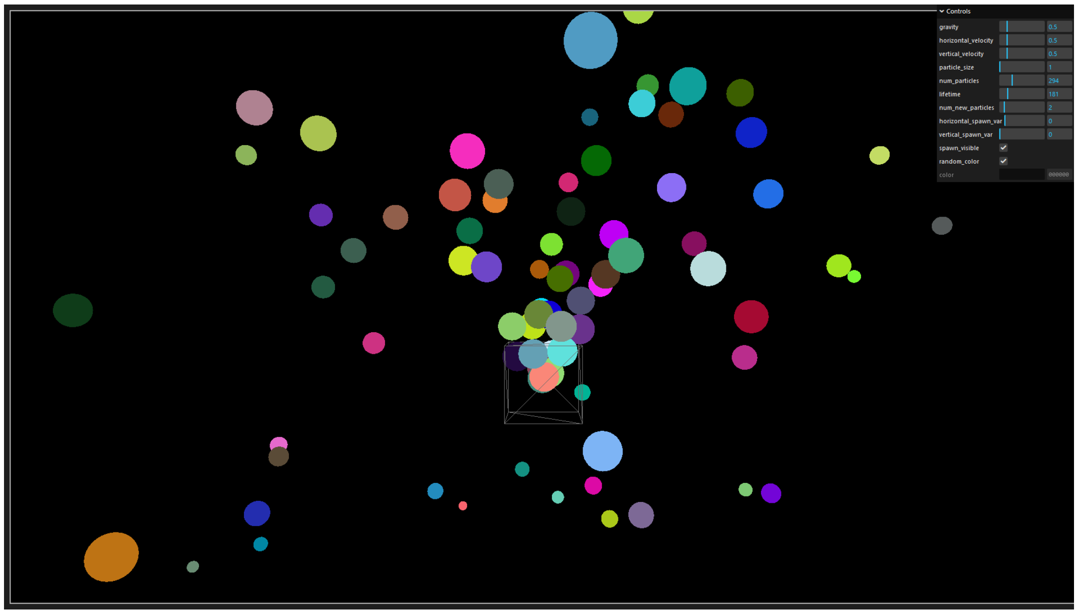
Particle systems are a technique in computer graphics that use simplistic graphics objects, often 2D meshes or basic 3D shapes, to simulate a variety of phenomena. These include effects such as fire, smoke, water, clouds, and others that are difficult to simulate using other computer graphics methods, largely due to the high volume of objects being rendered or the interactions between objects. Particle systems typically consist of three elements or stages. The first is the emission stage, in which new particles are generated. The second is the simulation stage, in which particles are updated based on some set of parameters, usually a vector indicating direction or a mathematical equation of motion. The third stage is the rendering stage, in which the particles are displayed to the output.

In the emission stage new particles are generated from a particle emitter. This emitter can either be a point in the scene or an object. The spawning rate is a value that determines how many new particles are spawned per a specific unit of time. There is also often a limit imposed on the number of particles that can exist at any time. New particles are spawned (created) with an initial velocity value to determine their movement behavior during the simulation stage.



gravity	0.5
horizontal_velocity	0.49
vertical_velocity	0.57
particle_size	1
num_particles	10
lifetime	22
num_new_particles	1
horizontal_spawn_var	0
vertical_spawn_var	0
spawn_visible	✓
random_color	✓
color	000000

During the simulation stage, the particles are updated at a certain rate according to some set of principles. The primary feature that is updated is each of the particles' positions. This can be as simple as a translation, or it can be governed by more complex mathematical models. A common model is $s = vt$, where s is the displacement of the particle, v is the velocity of the particle, and t is the amount of time that has passed. The particles are then translated by the displacement value. Particles will often possess an acceleration value, such as the acceleration due to gravity. Using the same model as before, acceleration would be represented by updating the particle's velocity at every step, along with its position. Also during the simulation stage, each particle's lifetime is checked. A particle's lifetime is a measure of how long that particle has existed and after a certain threshold is reached, the particle is deleted to allow for new particles to spawn.



Controls

gravity	0.5
horizontal_velocity	0.5
vertical_velocity	0.5
particle_size	1
num_particles	294
lifetime	181
num_new_particles	2
horizontal_spawn_var	0
vertical_spawn_var	0
spawn_visible	✓
random_color	✓
color	000000

Thirdly, the rendering stage. This is where the particles are rendered on screen. Often, to combat the complexity of rendering tons of individual particles, each particle is represented by a 2D object that is billboarded. To be billboarded means that it is always facing the camera, which gives the illusion when the camera is moving around that it is actually a 3D object. In certain cases, 3D objects can be used, but even then they are often limited to basic, primitive shapes such as spheres and cubes. Textures can be used to help provide detail to these otherwise simplistic objects.

Lastly, a brief discussion of the components necessary for the implementation of a particle system. It is written assuming implementation in Three.js, but many of the core concepts will translate between languages. Particles are often represented as their own class, with parameters for their position, velocity, and lifetime at minimum. Each particle is also associated with some object in the scene, such as a 2D or 3D mesh. Particles are updated at certain intervals. One implementation is to include this in the draw loop itself. During this particle updating step, new particles are spawned in according to the spawn rate. Particles that have surpassed the lifetime limit are deleted, while all other particles have their positions updated according to the chosen model/equation. By tuning various parameters, different behaviors can be achieved. For example, changing the maximum number of particles, the lifetime of the particles and the spawning rate all have an effect on how continuous the effect is. Changing the velocity, acceleration or motion equation can vastly alter how the particles move and what effect they resemble.