COSC 89.18: Physical Computing

Programming Assignment 03: Smoke Simulation

Student: Amittai Wekesa

For this assignment, I implemented the equations studied in class to model smoke diffusion, advection, and vorticity.

I also implemented a generic source affecting the smoke field. This can either be radial (diverging or converging), tangential (spinning), or a mix of both.

I implemented vorticity as described in tfeh paper (the second approach):

> The simplest approach is to use the particles' vorticity magnitude *only* (ignoring direction) to define a spatially varying confinement strength ϵ, transferring the particles values of this parameter to the grid with the distribution kernel mentioned above. This allows vorticity confinement to be activated independent of the existing grid based vorticity, but ignores the directional component of the particle's vorticity. Even this simple approach readily creates visually rich phenomena difficult to obtain with vorticity confinement alone, and we used it early on in a production pipeline to create many explosion effects for a feature film, see Figure 4.
>
> A promising technique is to form an analytic confinement force independently for each particle. The distribution kernel, $\xi_p(\boldsymbol{x} - \boldsymbol{x}_p)$, for a particle together with the particle vorticity, $\boldsymbol{\omega}_p$, defines an analytic vorticity $\tilde{\boldsymbol{\omega}}_p(\boldsymbol{x}) = \xi_p(\boldsymbol{x} - \boldsymbol{x}_p)\boldsymbol{\omega}_p$. Choosing a kernel that is rotationally symmetric and strictly decreasing with distance from the particle center implies that $\boldsymbol{N}_p(\boldsymbol{x}) = (\boldsymbol{x_p} - \boldsymbol{x})/\|\boldsymbol{x_p} - \boldsymbol{x}\|$, and the confinement force is then $\boldsymbol{F}_p(\boldsymbol{x}) = \epsilon_p(\boldsymbol{N}_p \times \tilde{\boldsymbol{\omega}}_p)$. We can sum the contributions from all the particles to obtain a grid based force field for use in equation 1. This technique was used to generate Figures 1, 2 and 3. In addition, one can interpolate the grid based vorticity to the particle location and reduce the strength of the particle based force as the grid based vorticity approaches the particle's vorticity. Of course, in practice the grid is typically too coarse for the grid vorticity to match the vorticity of all the particles. Alternatively, one could transfer the magnitude *and* direction of the particle's vorticity to the grid, and compare this to the existing grid based vorticity. The difference between these can be used to calculate a vorticity confinement force (replacing vorticity with this difference in the formulas). However, we have not found these last two options to be necessary.

Figure 1: Vorticity Discussion by Andrew Selle, Nick Rasmussen, and Ronald Fedkiw

For the custom sources, I implemented a class that can be used to create a source that can be added to the smoke simulation.

Each source has it's own position information, velocity it induces, and the type it is. Additionally,each source has a `getVelocity()` method that returns the velocity vector that the source induces on a given position. If the position is outside of the source's radius of influence, the velocity is zero.

```
Vector2 getVelocity(Vector2& position) const
{
  Vector2 velocity = Vector2::Zero();
  if (isInRange(position)) {
    Vector2 normal = position - this->pos;
    if (normal.norm() == 0) {
      double x = rand() / 100;
      x -= floor(x);
      double y = rand();
      y -= floor(y);

      return {x, y};
    }
    if (type == "radial") {
      velocity = normal.normalized() * this->vel;
    }
    else if (type == "tangential") {
      velocity = Vector2(normal[1], -normal[0]).normalized() * this->vel;
    }
    else if (type == "radial_tangential") {
      velocity = normal.normalized() * this->vel;
      velocity += Vector2(normal[1], -normal[0]).normalized() * this->vel;
      velocity /= 2;
    }
  }
  return velocity;
}
```