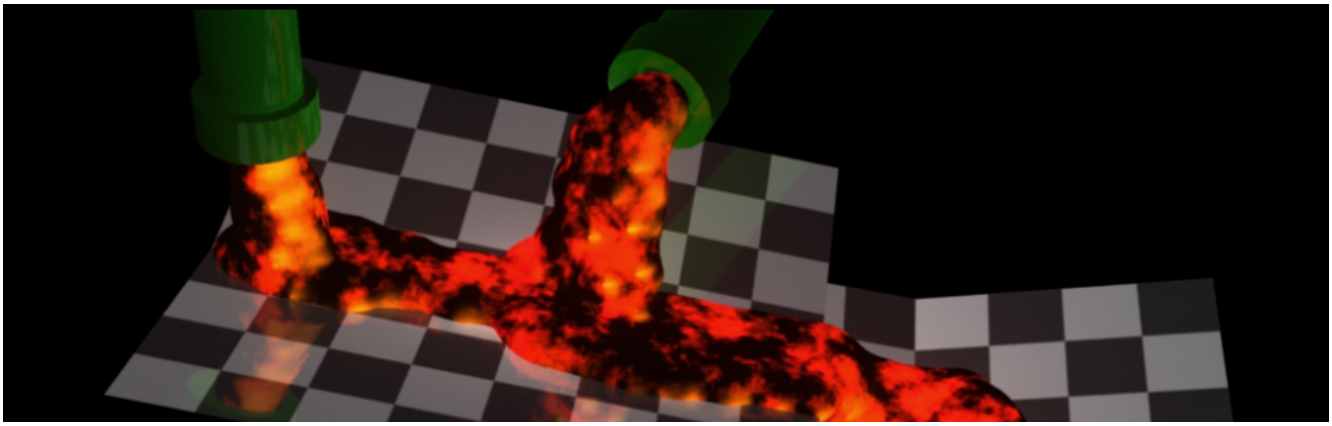

Imparting High Viscosity Behavior On Particle Systems

Daniel Camarda & Fredrik Johnson

December 17th, 2015

Abstract

The following report explains the process of creating a "field node" plug-in to control a particle system within Autodesk Maya. The field node affects the behavior of the particles within the system such that they behave as a highly viscous fluid. The particles utilize a position-based system for tracking in which new forces are calculated and applied once per time step.

1 Introduction

Fluids have long been a fascinating and highly challenging subject within computer graphics as well as the special effects industry. Several algorithms have spawned in order to find the optimal solution to fluid simulation. These algorithms often have an approach aimed towards solving a specific part of the fluid "problem". This results in them having certain advantages and disadvantages compared to similar fluid solving algorithms.

Maya is the one of the most widely used special/visual effects software within the industry. Due to the fact that most of the artists within the industry utilize the software on a daily basis, the decision was made to use as much built-in functionality as possible. This alleviates the problem of installing multiple new dependencies before first usage.

2 Method

The algorithm used in this project was inspired by Clavet et al.[1]. The algorithm is designed to simulate highly viscous fluids using particles. Due to the limitations within Maya, a full implementation was not possible. The algorithm presented by [1] requires what could be interpreted as "root access" to each particle's attributes (e.g. position, velocity, mass) in that they need to be able to be altered manually within the code. Mayas implementation of particle dynamics does not allow for these types of root level changes to be made and thus, the implementation presented is loosely based on [1].
The algorithm presented here is described in algorithm 1

Listing 1: Overview of the code

```
for each particle i
    reset com
    for each particle j > i
        ij = i.pos - j.pos
        ijN = ij.normalize()
        d = ij.length()*r
        if(d > 2*r)
            u = (i.vel - j.vel)*ijN
            if(u > 0.0)
                vI = ijN*(dT*(1.0 - d)*(visc*u
                    + beta*u*u))
                i.outForce -= vI/2.0*S
                j.outForce += vI/2.0*S
            end if
        end if
        R = 2.5*r;
        if(d < R && i.age > 50.0f)
            center += j.pos
            counter++
        end if
    end for
end for

i.outForce += center*0.5
i.outForce += -1.0*((i.age / MAXFRAMES) *
    i.outForce
i.outForce /= S*5
i.outForce += gravity*0.1
---------------------------
r = user set radius
MAXFRAMES = Max number of frames in the
    sumulation
visc = user defined constant for
    viscosity(2.5)
beta = user defined constant used in the
    calculation of viscosity(0.5)
S = user defined scaling factor for impulse
    calculation
```

The field node locates a particle's neighbors within a radius **r** and a collision "impulse" is applied if the particles are traveling towards one another. The impulse is a simplified version of a legitimate impulse in that it is simply a force which is amplified by a scaling factor. A second search through the particle system for particles within radius **R**, where **R** is equal to **r***2.5, is then conducted and the positions of any particles found within this area are saved. If the current particle's age above a user set limit, it is then given a force towards the center of mass of all particles within **R**.

Along with the forces applied to impart an attraction to the neighboring particles, the field slows the velocities of the particles based upon their age. This allows the particles to "cool" over time which has a direct effect upon their velocities.

3 Result

The effect the field node has on the particles is clear. The particles stick together and move sluggishly throughout the scene. They also seem to form more solidified areas which "younger" parts of the lava flow then move past due to their higher velocity which can be seen in *figure 1*. As can be seen in algorithm 1, the physics needed to be altered significantly to achieve the desired effect for the particle behavior. *Figure 2* shows a side by side view of a fully rendered image from the scene with a render in Mayas viewport. The lava effect is more apparent with a lava texture attached as can be seen on the left side of *figure 2*.

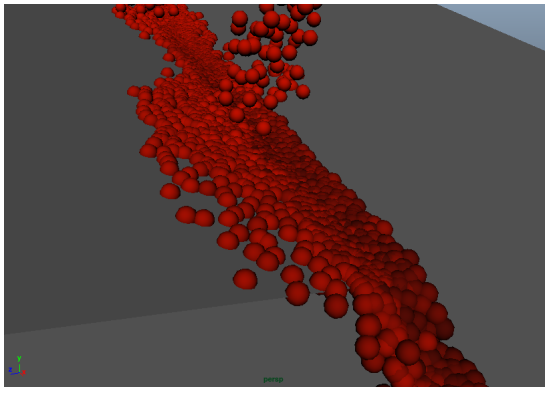

Figure 1: Particle Flow showing different ages. The darker particles represent older particles and lower speed.

A fully rendered film can be seen via the following link: `https://vimeo.com/149405824`

4 Conclusion

One of the biggest obstacles we faced while writing this code was the lack of root access to crucial attributes on a per particle basis. Had we been

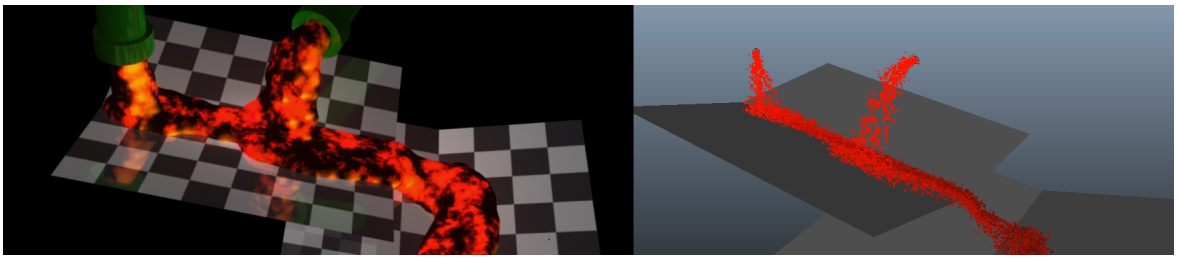

Figure 2: Left: Rendered particle flow with lava texture. Right: Maya viewport render using multipoint mode

able to change the velocity and positions directly within our plug-in, the fluid would be more in line with the true physical properties of lava.

The plug-in works quite well within the constraints of the project plan. Several variables need to be adjusted in order to achieve the desired viscosity of the target fluid. This could be improved within the code in a number of ways which leaves room for future work.

The insight gained from a project in which limitations hampered our finished product is quite valuable while simultaneously frustrating. This knowledge can be used for future works in the way of thorough research for which APIs should be used.

References

[1] Pierre Poulin Simon Clavet, Philippe Beaudoin. Particle-based viscoelastic fluid simulation. *ACM SIGGRAPH Symposium on Computer Animation*, 2005.