TP: Smoothed Particles Hydrodynamics

TPIGR-202 - Computer Graphics and Virtual Reality

Chaussonnier Romain

February 12, 2023

1 Neighbor Data

Everything related to computing neighbor data is done in $buildNeighbor()$. The function is separated in two parts.
The first for loop exists to fill $\_pidxInGrid$. All particles are visited. Each of their positions is computed and helps determine in which cell of the grid the particle lies. With this information, the particle id is stored.
The second for loop exists to update, for each particle, which other particles are considered its neighbors. Those are stored in $\_supportParts$, named in reference to $\_sr = \_kernel.supportRadius()$. For each particle, a specific grid will be visited. This grid is the complete grid reduced to the cells that are within reach of $\_sr$ while making sure the range will not trespass any boundary. This selection is done by creating four bounds accordingly:

```
1      const int lowerBoundX = iCentralCell - fmin(kernelReach, iCentralCell);
2      const int upperBoundX = iCentralCell + fmin(kernelReach, _resX - 1 - iCentralCell) + 1;
3      const int lowerBoundY = jCentralCell - fmin(kernelReach, jCentralCell);
4      const int upperBoundY = jCentralCell + fmin(kernelReach, _resY - 1 - jCentralCell) + 1;
```

When a cell is reached using those bounds, we go through all its contained particles using the $\_pidxInGrid$:

```
1          for (auto neighborPartId : _pidxInGrid[idx1d(iCell, jCell)])
```

If the distance between the central particle and the considered one is smaller than $\_sr$, both this distance and the neighbor particle id are stored in $\_supportParts$.

2 Density

For each particle, all its supporting particles are visited. W_{ij} is computed and added to a sum:

```
1        inverseVolume += _kernel.w(rij);
```

The final density is this sum multipplied by the particles shared mass $\_m0$.

3 Body Forces

This part is straighforward. $\_g$ is added to every particle's acceleration.

4 Pressure

Pressure calculation is done in $computePressure()$ and $applyPressureForce()$.
In $computePressure()$, for each particle there is a call to the equation of state function, using particles densities.
In $applyPressureForce()$, we apply the formula given in the exercise instructions to add the pressure contribution to every particle's acceleration.

5 Viscosity

Same goes for viscosity, with another formula.

6 Moving Particles

All the accelerations are now updated. All that is left to do is to update the velocities and the positions with a simple integration scheme.

```
1      _vel[partId] += _dt * _acc[partId];
2      (...)
3      _pos[partId] += _dt * _vel[partId];
```

7 Boundary

All the pre-existing boundary handling is now commented.
Instead, in $initScene()$, particles are added along the floor, roof and the left and right walls:

```
1     for (size_t i = 0; i < res_x; i++)
2     {
3       _pos.push_back(Vec2f(i + 0.25, 0.25));
4       _pos.push_back(Vec2f(i + 0.75, 0.25));
5       _pos.push_back(Vec2f(i + 0.25, res_y - 0.25));
6       _pos.push_back(Vec2f(i + 0.75, res_y - 0.25));
7     }
8     _pos.push_back(Vec2f(0.25, 0.75));
9     _pos.push_back(Vec2f(res_x - 0.25, 0.75));
10    for (size_t j = 1; j < res_y - 1; j++)
11    {
12      _pos.push_back(Vec2f(0.25, j + 0.25));
13      _pos.push_back(Vec2f(0.25, j + 0.75));
14      _pos.push_back(Vec2f(res_x - 0.25, j + 0.25));
15      _pos.push_back(Vec2f(res_x - 0.25, j + 0.75));
16    }
17    _pos.push_back(Vec2f(0.25, res_y - 0.75));
18    _pos.push_back(Vec2f(res_x - 0.25, res_y - 0.75));
```

The important line however is the one where $\_nB$ is updated. It is the number of boundary particles - that were just added. Now all the loops in $applyBodyForce()$, $applyPressureForce()$, $applyViscousForce()$, $updateVelocity()$ and $updatePosition()$ start at $partId = \_nB$. That way, those boundary particles don't move, and no unecessary information is calculated for them. Only their density and pressure computation is kept because they will have an influence on other particles in $applyPressureForce()$.
The fluid mass is moved up right a little because now the down left corner is filled with boundary particles.
Those were all the necessary modifications.