ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computational Engineering

Spring semester 2014

Prof. P. Koumoutsakos, G. Tauriello
ETH Zentrum, CLT F 12, C 11
CH-8092 Zürich

Solution 1

Issued: 18.3.2014

Question 1: File I/O

```
1  // Constructor is automatically called when object is defined
2  // set nparticles to zero, array to NULL
3  ParticleContainer::ParticleContainer(): nparticles(0), array(NULL) { }
4
5
6  // Destructor is automatically called when object goes out of scope
7  // Heap data should be deallocate here
8  ParticleContainer::~ParticleContainer()
9  {
10     _dispose();
11 }
12
13
14 void ParticleContainer::_dispose()
15 {
16     // clean up: if array is allocated, deallocate it
17     if (array!=NULL)
18     {
19         delete [] array;
20         nparticles = 0;
21         array = NULL;
22     }
23 }
24
25
26 void ParticleContainer::Load(const char * const filename)
27 {
28     // if array is allocated, deallocate it
29     _dispose();
30
31     // retrieve number of particles from the file
32     nparticles = _lines(filename);
33
34     // allocation of the particles
35     array = new Particle[nparticles];
36
37     // create file stream
38     ifstream filestream(filename);
39
40     // parse particles' attributes
```

```
41      for (int i = 0; i < nparticles; ++i)
42      {
43          filestream >> array[i].x;
44          filestream >> array[i].y;
45          filestream >> array[i].u;
46          filestream >> array[i].v;
47          filestream >> array[i].omega;
48      }
49  }
50
51
52  const Particle * const ParticleContainer::GetParticles() const
53  {
54      // checks if particle data is allocated and
55      // return the pointer to the data
56      assert(array!=NULL);
57
58      return array;
59  }
60
61  int ParticleContainer::GetNumberOfParticles() const
62  {
63      // return the number of allocated particles
64      assert(array!=NULL);
65
66      return nparticles;
67  }
```

Listing 1: ParticleContainer.cpp

Question 2: Define a Colormap

```
1  // Given the particles attributes, compute r = red, g = green and b =
       blue
2  // Note that r,g,b must be in [0,1]
3  void Particle::ComputeColor(double& r, double& g, double& b) const
4  {
5      // get the magnitude of the velocity vector
6      double s = sqrt(pow(u,2) + pow(v, 2));
7      // set value boundary
8      double s_min = 0;
9      double s_max = 80;
10
11     // scale to make sure that the color is valid
12     const double x = min(1., max(0., (s - s_min)/(s_max - s_min)));
13
14     // compute color
15     r = max(2*x-1, 0.);
16     g = 2*(0.5-fabs(0.5 - x));
17     b = max(0., 1-2*x);
18 }
```

Listing 2: Particle.cpp

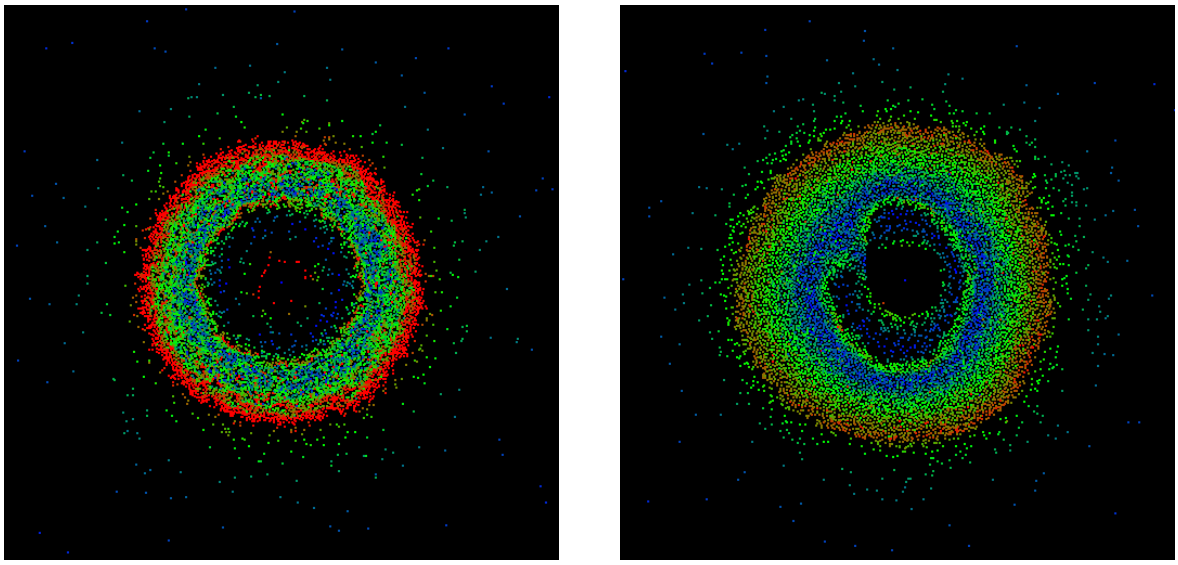

Figure 1: Initial and final frame