Project: OpenCL / OpenGL Particle System

Huan Yan

March 29, 2017

CS575

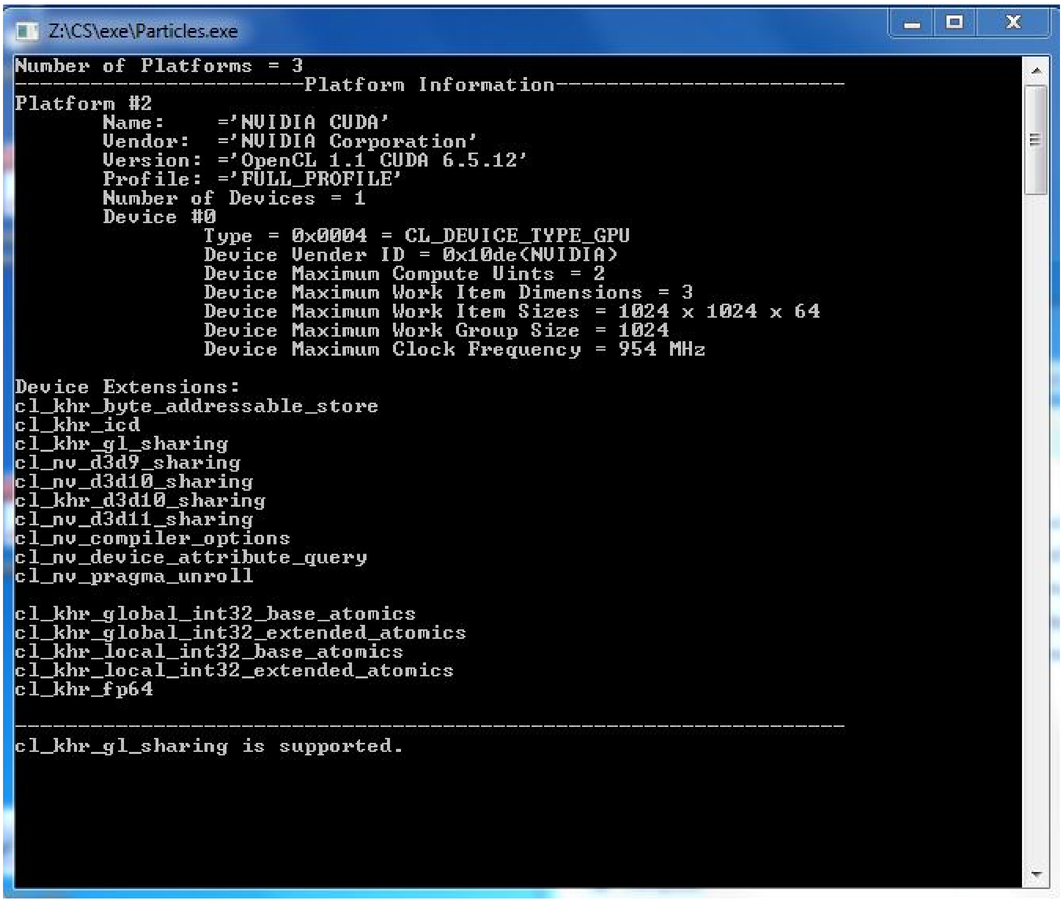
Huan Yan

March 29, 2017

1. What machine you ran this on

For this project, I compiled and ran my program in Visual Studio.

The following graph is showing some GPU information I requeried with OpenCL:



```
Number of Platforms = 3
-----------------------Platform Information-----------------------
Platform #2
        Name:     ='NVIDIA CUDA'
        Vendor:   ='NVIDIA Corporation'
        Version:  ='OpenCL 1.1 CUDA 6.5.12'
        Profile:  ='FULL_PROFILE'
        Number of Devices = 1
        Device #0
                Type = 0x0004 = CL_DEVICE_TYPE_GPU
                Device Vender ID = 0x10de(NVIDIA)
                Device Maximum Compute Uints = 2
                Device Maximum Work Item Dimensions = 3
                Device Maximum Work Item Sizes = 1024 x 1024 x 64
                Device Maximum Work Group Size = 1024
                Device Maximum Clock Frequency = 954 MHz

Device Extensions:
cl_khr_byte_addressable_store
cl_khr_icd
cl_khr_gl_sharing
cl_nv_d3d9_sharing
cl_nv_d3d10_sharing
cl_khr_d3d10_sharing
cl_nv_d3d11_sharing
cl_nv_compiler_options
cl_nv_device_attribute_query
cl_nv_pragma_unroll

cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_fp64

------------------------------------------------------------------
cl_khr_gl_sharing is supported.
```

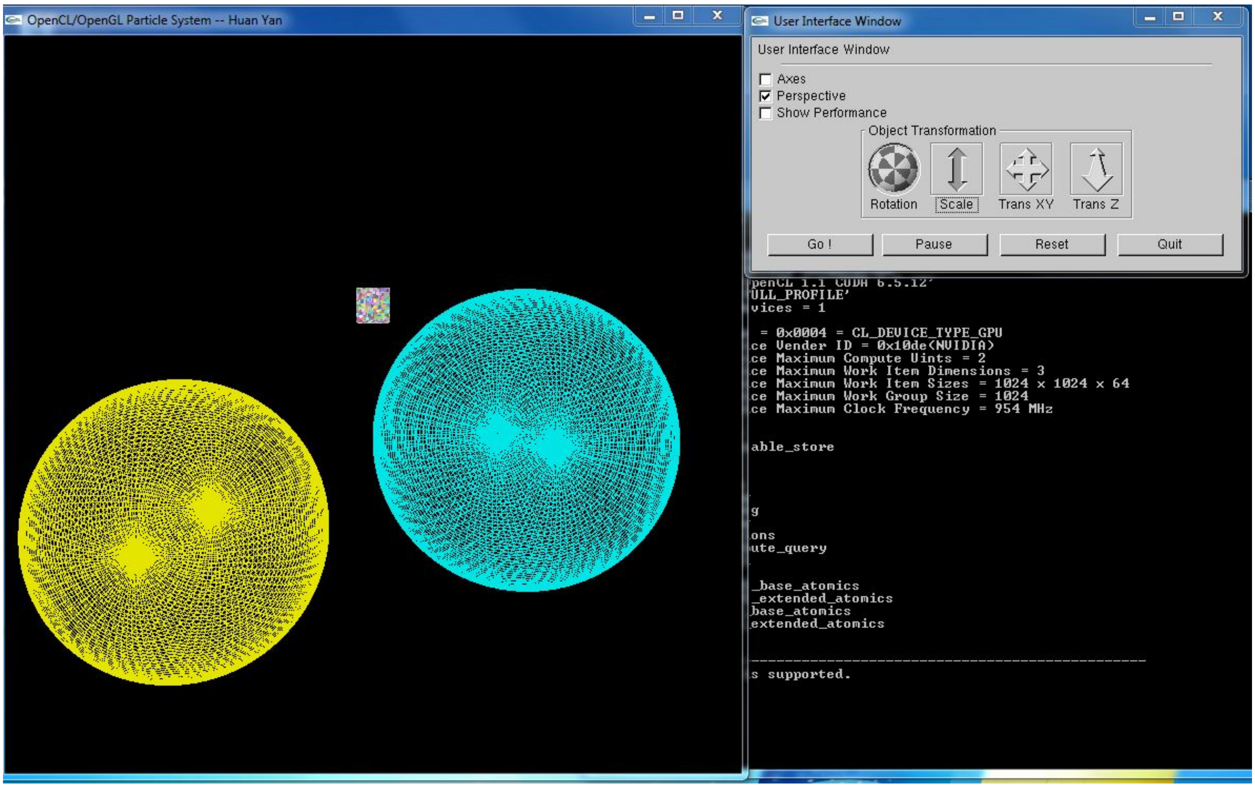
2. What dynamic thing did you do with the particle colors

In yanhua.cl file, I wrote the following code:

```

  color c = dCobj[gid];

c = c - 0.1 * DT;
  if (c.x > 1.0)
  {
    c.x = 0.0;
  }

  if (c.y > 1.0)
  {
    c.y = 0.0;
  }

  if (c.z > 1.0)
  {
    c.z = 0.0;
  }

  if (c.x < 0.0)
  {
    c.x = 1.0;
  }

  if (c.y < 0.0)
  {
    c.y = 1.0;
  }

  if (c.z < 0.0)
  {
    c.z = 1.0;
  }
```

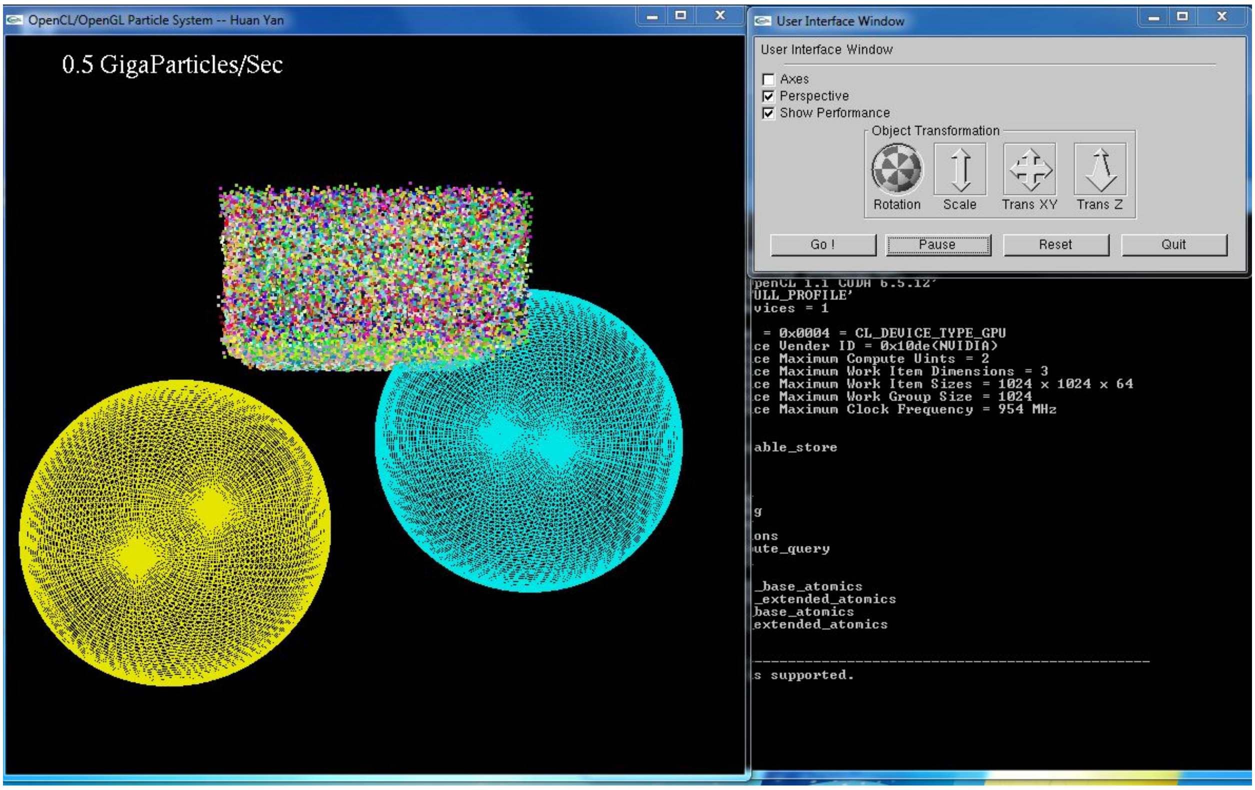
In my .cl kernel, with the change of time, dynamically change the color of each particle. Each time reduce 0.1*DT, and circulate between 0.0 and 1.0.

3. Include at least one screen capture image of your project in action



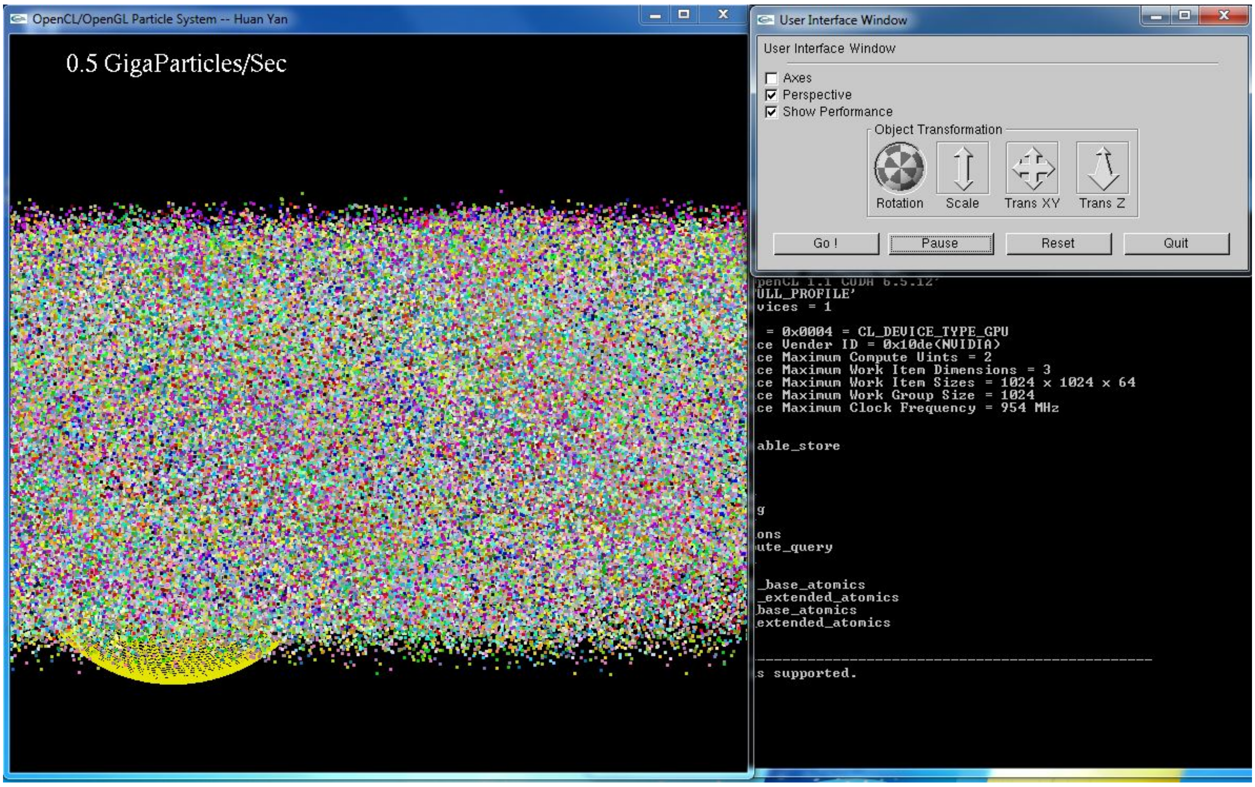
□ Axes
☑ Perspective
□ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz


able_store



g
ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


-----------------------------------------------------
s supported.
```



□ Axes
☑ Perspective
☑ Show Performance

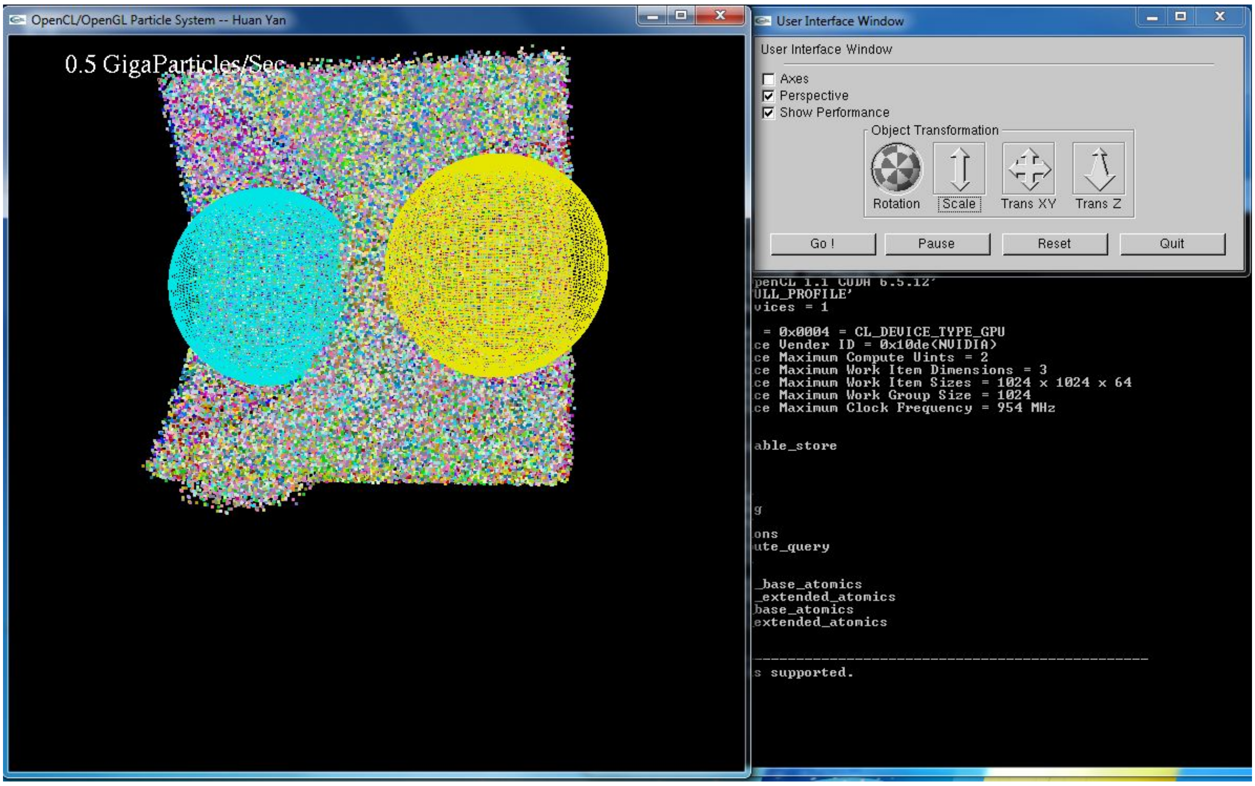
```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz


able_store



g
ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


-----------------------------------------------------
s supported.
```



User Interface Window

☐ Axes
☑ Perspective
☑ Show Performance

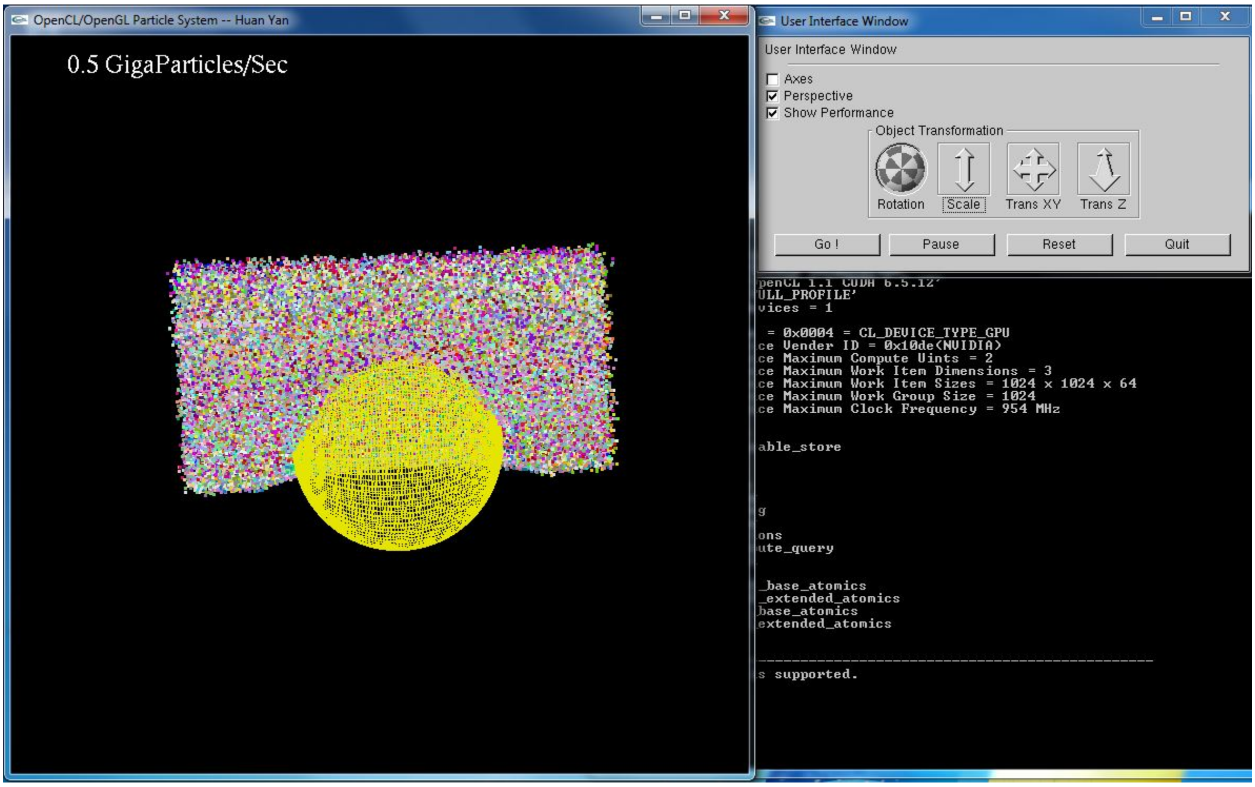
```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz

able_store


g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


--------------------------------------------------
s supported.
```



User Interface Window

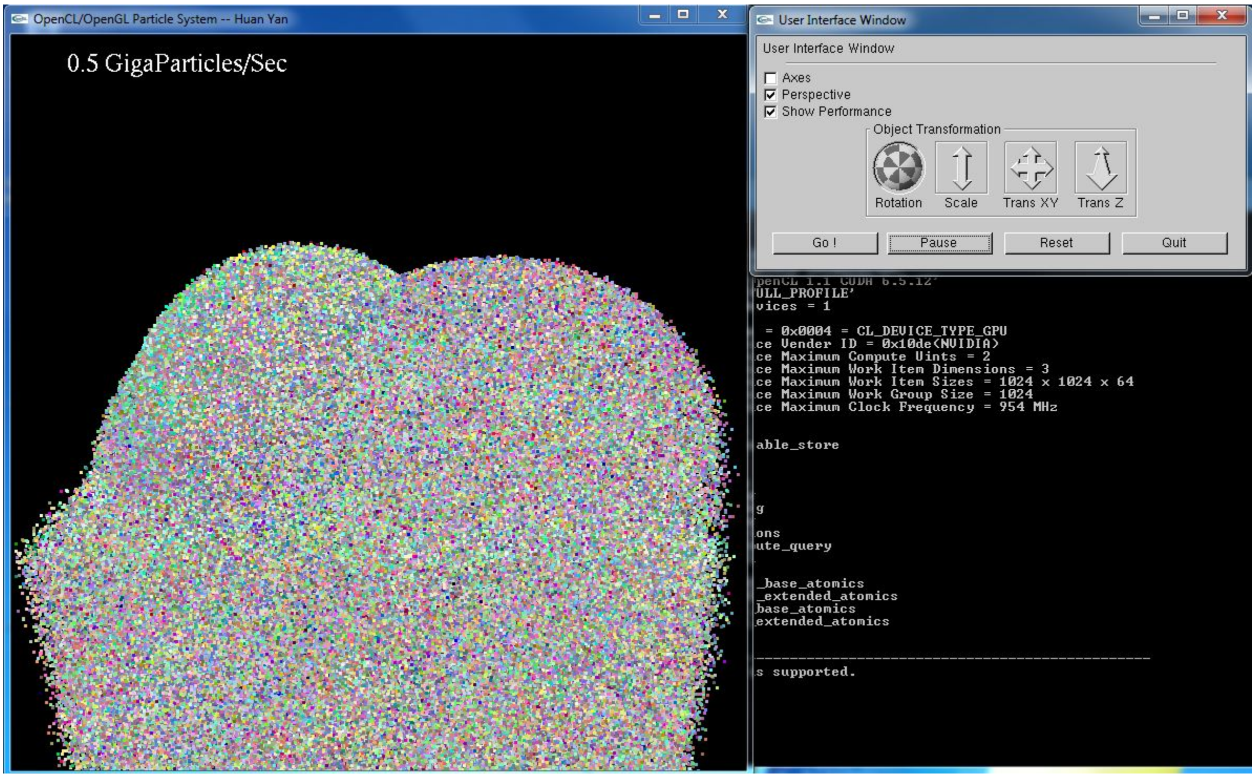
☐ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz

able_store


g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


--------------------------------------------------
s supported.
```



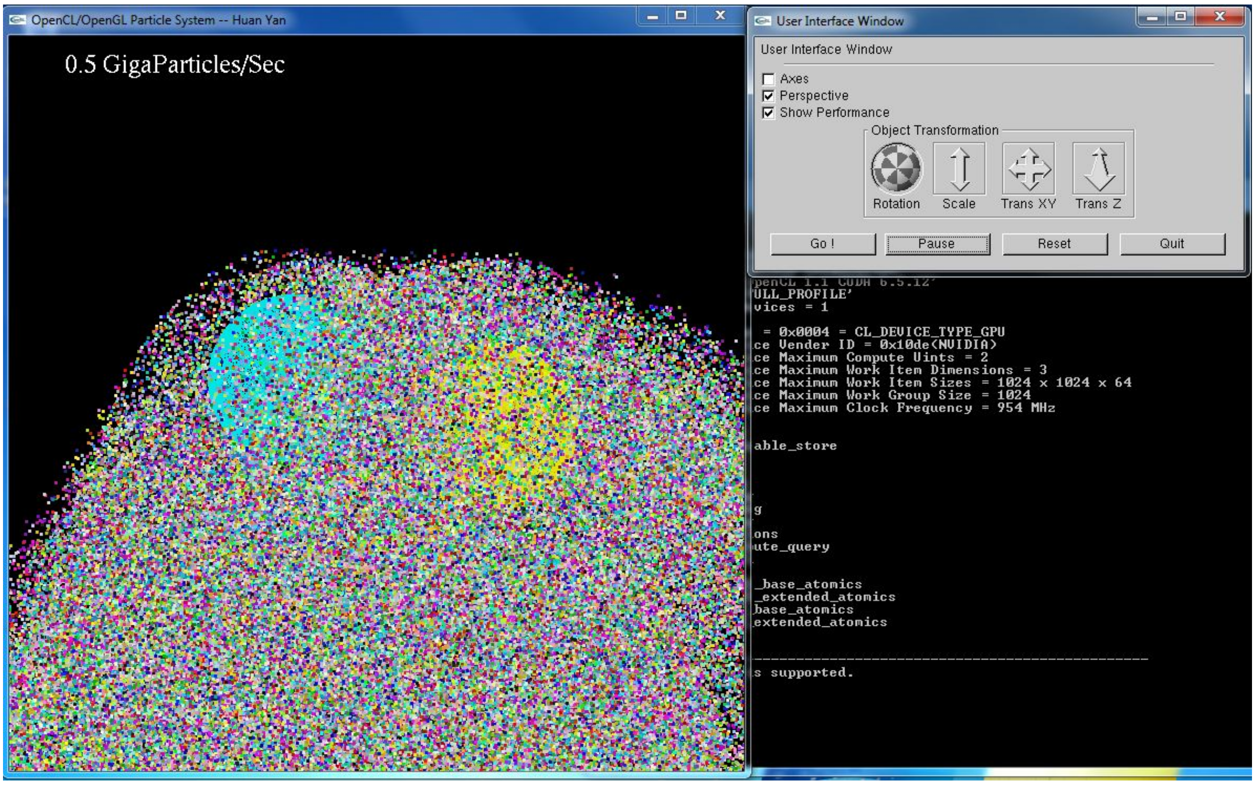
□ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz


able_store




g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


-----------------------------------------------
s supported.
```



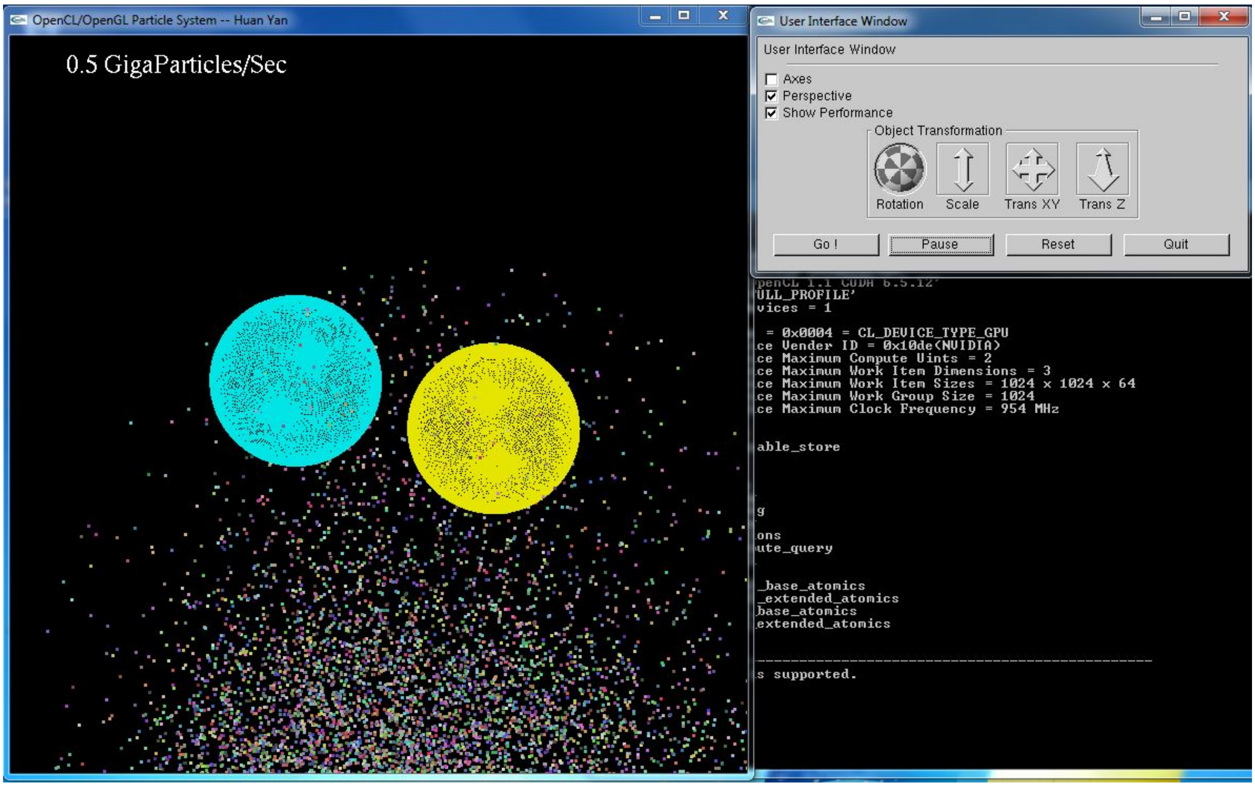
□ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz


able_store




g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


-----------------------------------------------
s supported.
```



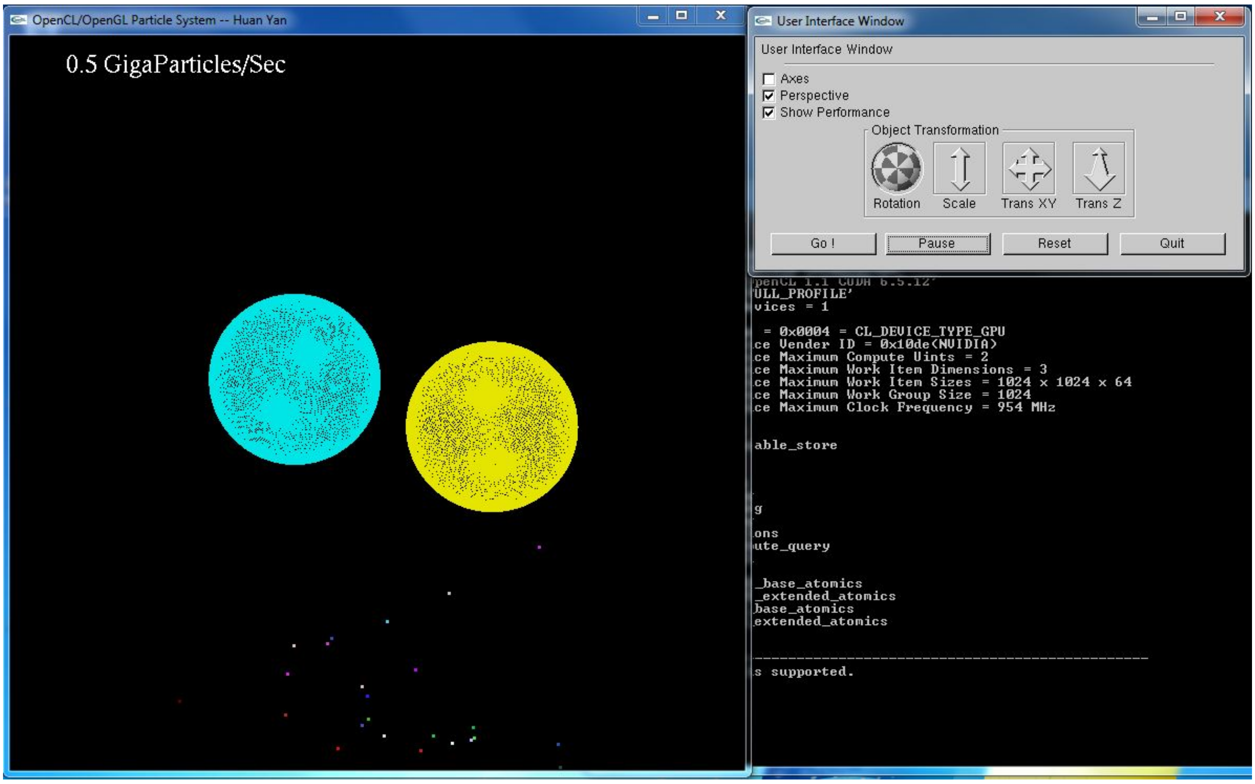
☐ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz

able_store




g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics

-------------------------------------------------
s supported.
```



☐ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz

able_store




g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics

-------------------------------------------------
s supported.
```



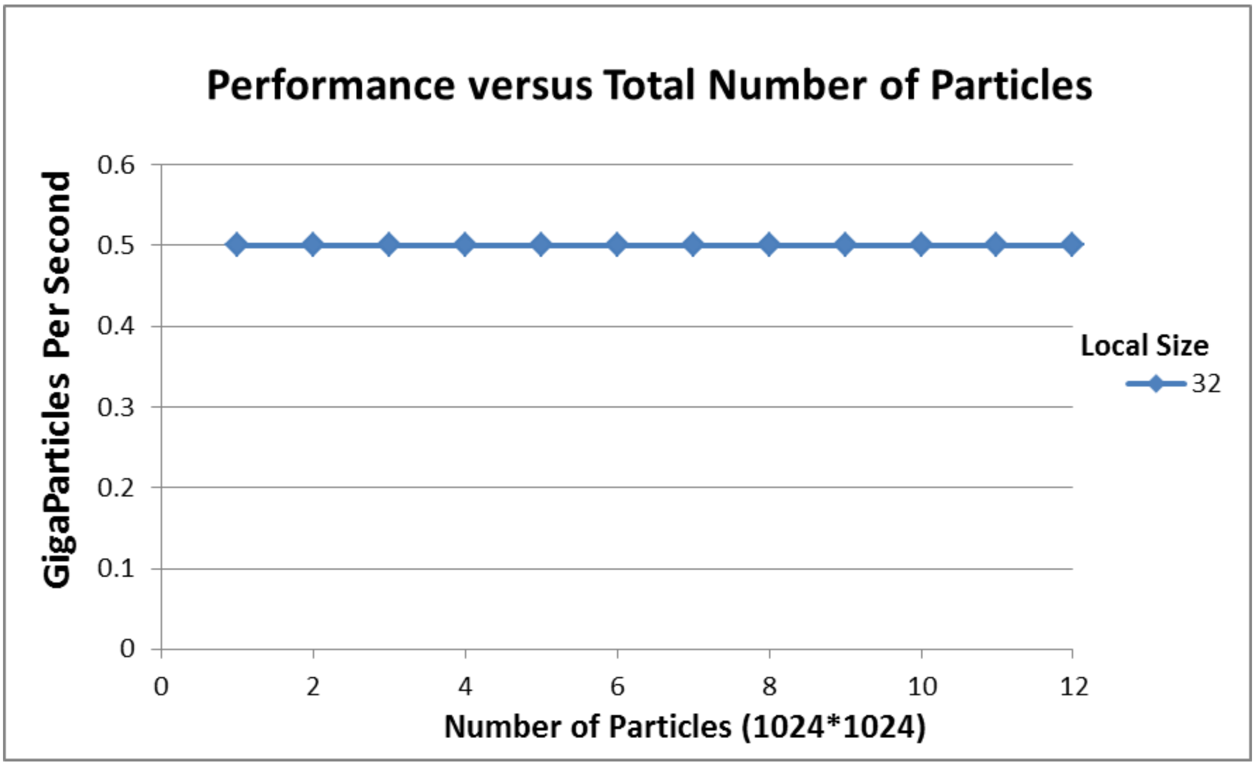
0.5 GigaParticles/Sec

☐ Axes
☑ Perspective
☑ Show Performance

```
penCL 1.1 CUDA 6.5.12'
ULL_PROFILE'
vices = 1

 = 0x0004 = CL_DEVICE_TYPE_GPU
ce Vender ID = 0x10de(NVIDIA)
ce Maximum Compute Uints = 2
ce Maximum Work Item Dimensions = 3
ce Maximum Work Item Sizes = 1024 x 1024 x 64
ce Maximum Work Group Size = 1024
ce Maximum Clock Frequency = 954 MHz


able_store




g

ons
ute_query


_base_atomics
_extended_atomics
base_atomics
extended_atomics


-----------------------------------------------
s supported.
```

4. Show the table and graph

Number of Particles	GigaParticles Per Second
1024*1024*1	0.5
1024*1024*2	0.5
1024*1024*3	0.5
1024*1024*4	0.5
1024*1024*5	0.5
1024*1024*6	0.5
1024*1024*7	0.5
1024*1024*8	0.5
1024*1024*9	0.5
1024*1024*10	0.5
1024*1024*11	0.5
1024*1024*12	0.5



5. What patterns are you seeing in the performance curve?

My program crashed at number of particle is 1024*1024*13. The performance seems constant form the number of particles is 1024*1024*1 to 1024*1024*12. When the number of particles is 1024*1024*14, there appear some error and cannot work.

6. Why do you think the patterns look this way?

The reason why the performance seems constant is that: from 1024*1024*1, this amount of particles are already full using the graphics computing units. Increasing total workload does not make any difference. The really thing hinders performance is GPU rendering. In every main loop, OpenCL has to wait OpenGL to handle rendering and then do computing. The limit of number of particles can be handled in this system is 1024*1024*13. When the number of particles is smaller than this size, increase or decrease the number of particles has little effect on the calculation of OpenCL. But the performance of OpenGL is directly influenced by the number of particles. When the the number of particles is greater than this size, causing the OpenCL also cannot work.

7. What does that mean for the proper use of GPU parallel computing?

In order to get the proper use of GPU parallel computing, we need to consider the processing power of GPU. Because OpenGL draws too many particles will become slow, and it will also affect the speed of OpenCL. This leads to some objects such as position and color which should be shared by OpenCL and OpenGL, but are always occupied by OpenGL. In this way, OpenCL can't calculate. So there will appear a large number of clEnqueueAcquireGLObjects fails. So we need to know the critical value of GPU in order to let system can perform better.