Embree – Dynamic Scenes

Luís Paulo Santos, March, 2018

This tutorial with `Embree` will modify a copy of the `viewer` tutorial's code:

1. Download the file `VI2_EmbreeT3anim_device.cpp` made available on the web site and copy it to `$EMBREE_SOURCES$/tutorials/viewer/`
2. Modify your `Visual Studio` solution or `Makefile` or even `CMake` file, such that the `viewer` project (included in the tutorials) compiles `VI2_EmbreeT3anim_device.cpp` instead of `viewer_device.cpp`
3. Build the `viewer` tutorial

We will also use a modified version of the `Cornell Box` distributed with `Embree`:. This is the same as Tutorial 1, but if you don't have it available then:

4. Download `cornell_box_VI2.zip` and extract it, making sure that the respective files (containing the `cornell_box_VI2` model, with extensions `.obj`, `.mtl` and `.ecs`) become available in the `$TUTORIALS_BUILD$/models` folder, where `$TUTORIALS_BUILD$` is the pathname of the folder where the `viewer` executable file is stored.
5. Verify your installation by opening a *shell* and from the `$TUTORIALS_BUILD$` folder executing

```
viewer –c models/cornell_box_VI2.ecs
```

You will see that this is a distributed non progressive renderer.

Adding a Sphere to a Dynamic Scene

We are preparing a dynamic scene. `Embree` must be notified of that. Locate the line of code:

```
int scene_flags = RTC_SCENE_STATIC | RTC_SCENE_INCOHERENT;
```

comment it and uncomment the line below, which should read as:

```
int scene_flags = RTC_SCENE_DYNAMIC | RTC_SCENE_INCOHERENT;
```

`Embree` is now notified that geometries can change and that the acceleration structure for ray traversal might have to be refit or rebuilt between frames.

Let's add a sphere to our scene. Within the `device_init()` method, uncomment the segment of code labelled as **`add sphere`**. Note that `device_init()` is called only once, when the `Embree` device is initialized, making it a good candidate to build or load a scene.

Have a look at the code you just uncommented. Knowing that `NUM_SPHERES` is 1, only the first sphere was added. This is centered at `(130.f, 50.f, 130.f)`, has radius `50.f` and has a total of 7080 triangles. If you want to understand how to generate a sphere out of a triangulated mesh have a look at `createSphere()`.

Build the `viewer` tutorial. The orange sphere looks nice, but it is not moving!

Animating the Sphere

Within the `device_render()` method we will find a segment of commented code labelled as **`animate spheres`**. Uncomment it!

The device_render() method is called every time a new frame has to be rendered. Therefore, we use this to change the geometry of dynamic objects, such as our sphere. The code you just uncommented (see below) calls the animateSphere() method and parameterizes it with time. Obviously, the changes we are doing to the sphere depend on the current time. Afterwards it calls rtcCommit(g_scene) in order to commit the changes and to give Embree an opportunity to refit or rebuild the acceleration structure for ray traversal.

```
for (int s = 0; s < NUM_SPHERES; s++)
       animateSphere(sphere_geomID[s], s, time);

rtcCommit(g_scene); // animate spheres
```

Have a look at animateSphere() . A new center is computer for our sphere (this is sphere 0), the sphere's geometry vertex buffer is accessed through rtcMapBuffer(g_scene, id, RTC_VERTEX_BUFFER), the vertices coordinates are changed and finally the buffer is unmapped and updated:

```
rtcUnmapBuffer(g_scene, id, RTC_VERTEX_BUFFER);

/* update mesh */
rtcUpdate(g_scene, id);
```

Build the viewer tutorial. Now it moves!

Adding more dynamic geometry

Locate the

```
#define NUM_SPHERES 1
```

and change it to

```
#define NUM_SPHERES 5
```

Build the viewer tutorial. Now the 5 spheres move! Was there a big drop in performance when you added more spheres? What are the frame rates for 1 and 5 spheres?

Now you try to add an additional object, eventually a sphere but with a different behavior than these 5.