Lab 12: Box Extents

3D Computer Graphics

Introduction

No new jar file is provided for this Lab. The idea is that you keep on working on your version of the rendering framework which you obtained after finishing the exercises of Lab 11. However, it is strongly advised that you make a new fresh copy of the project of Lab 11 and rename this copy to 3DCG_Lab12. This will make it easier for you to look again at the work you did in each Lab when you study for the exam later on.

The aim of this Lab is to implement box extents to speed up the rendering of mesh objects.

Exercise 1

a) Create a new package `apps.app8`.

b) Create a new graphical application (`App8`) in this package.

c) Configure this graphical application as follows:

- The scene to be rendered is described in the `simpleScene5.sdl` file.
- The width and height of the canvas are 800 and 600 pixels, respectively.
- The eye of the camera is located at $(0, 0, 10)$.
- The camera is aimed at the origin.
- The upwards vector of the camera is in the direction of the y-axis.
- The worldwindow has a width and height of $4/3$ and 1, respectively, and is located 1 unit in front of the camera.
- We want to render the scene without shadows or reflections.

d) Create a new file `simpleScene5.sdl` in the `resources` folder.

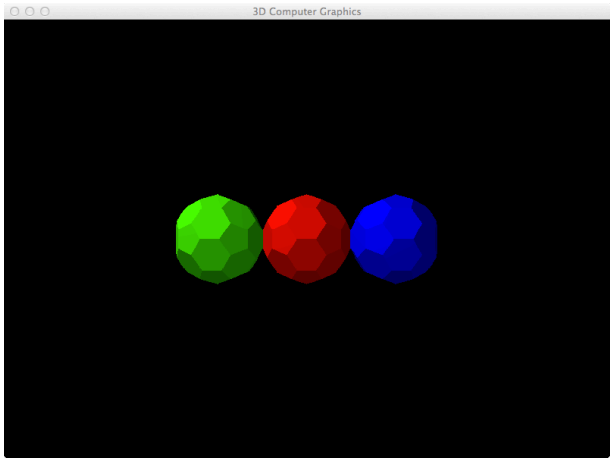
e) This scene should have the following properties:

- black background,

– one light source with RGB colour $(0.8, 0.8, 0.8)$ and located at $(-10, 10, 10)$,

– a second light source with RGB colour $(0.2, 0.2, 0.2)$ and position $(10, -10, 10)$

Add some other commands to the `simpleScene5.sdl` file as well so that you get the image shown below when running `App8`.



We will use `App8` as a testcase for the rendering time of our ray tracer. More precisely, we will animate the camera (to the left) and investigate how long it takes before the image is updated. In order to clearly see the difference between the original image and the image after the camera has been moved (to the left), we increase the angle by which the camera moves to the left when pressing the left arrow key.

f) Change the `Camera` class so that the camera rotates by an angle of 45^o to the left when the left arrow key is pressed.

g) Run `App8` and press the left arrow key to get an idea of how much time your framework needs to update the image.

The goal of the following exercises is to decrease this rendering time.

Exercise 2

A box extent is a box whose sides are aligned with the coordinate axes.

a) Create a new class `BoxExtent` in the `geomobj` package.

b) Add six instance variables (floats): `left`, `top`, `right`, `bottom`, `front` and `back`.

The box extent ranges from `left` to `right` in the x-direction, from `bottom` to `top` in the y-direction and from `back` to `front` in the z-direction.

c) Add one constructor which allows to initialize all these instance variables.

d) Create a public method `hit` to the `BoxExtent` class. This method should have a `Ray` object as parameter and return a boolean value depending on whether the given ray hits the box extent.

Note that a BoxExtent could be described as a mesh with six rectangular faces. Therefore, one could implement the `hit` method of the `BoxExtent` class based on the same idea as used for computing the hitPoints between a ray and a mesh.

e) Familiarize yourself again with the slides of Lab 5 (slide 1 to 21) which explain how to ray trace polygonal meshes. Understanding this material is essential to implement the `hit` method of the `BoxExtent` class.

The basic idea of computing the intersection between a ray and a mesh is the following:

```
for each face of the mesh
  if ray is not parallel with face
    compute tHit, the t-value of the hitPoint between
    the ray and the face plane
    if(hit in front of start of ray){
      compute hitPoint
      if(hitPoint inside face){
        add hitInfo to list
      }
    }
  }
}
```

The formula for `tHit` is given on slide 15 of Lab 5. Note that the numerator and denominator of this formula can be computed very fast for the faces of a box extent. Let's illustrate this for the front face of the box.

- The normal vector m to the front face is given by $(0, 0, 1)$.
- The numerator of the formula for tHit is the dot product of m and `A-start`, which is simply the z-coordinate of `A-start` because $m = (0, 0, 1)$. `A` is a point in the front face plane. All such points have a z-coordinate equal to `front`. Hence the numerator of the formula for tHit reduces to `front - ray.start.z` when considering the front face.
- Similarly, the denominator of the formula for tHit reduces to the simple expression `ray.dir.z`.

Furthermore, intersecting a ray with a polygonal mesh involved the point-in-polygon test to verify whether a hitPoint effectively lies inside a face of the polygonal mesh. Note that we can carry out this test much simpler in the case of a box extent. (How?)

Also note that the `hit` method of the `BoxExtent` class should not store data about the intersection points in HitInfo objects. It should simple return a boolean value.

The above remarks lead to the following pseudocode:

```
// FRONT FACE
numer = front - ray.start.z
denom = ray.dir.z
if(ray is not parallel with front face){
  tHit = numer/denom; \\ t-value of hitPoint between ray and
                      \\ front face plane
  if(hit in front of start point of ray){
    compute hitPoint
    if(hitPoint inside front face){
      return true;
    }
  }
}
```

f) Implement the above pseudocode in the `hit` method of the `BoxExtent` class. Make sure your implementation is as efficient as possible.

g) Have your implementation checked by your lecturer.

h) Copy/paste this code and adapt it for each of the other faces of the box extent.

i) Finalize the implementation of the `hit` method of the `BoxExtent` class.

j) Look at your implementation of this `hit` method as a whole and try to optimize it even further.

Exercise 3

A `Mesh` object is generally very expensive to ray trace because each ray has to be intersected with each face of the mesh. Therefore, we will use box extents to increase the efficiency of the intersection method of the `Mesh` class.

a) Create a private instance variable `genBoxExtent` of the type `BoxExtent` in the `Mesh` class. This variable stores the box extent of the untransformed mesh. This is the mesh whose data were read from a file and to which no transformations have been applied.

b) This instance variable should be initialized with a proper box extent immediately after the mesh data are read from a file. Therefore, add the line

```
makeGenBoxExtent();
```

to the end of the `Mesh` constructor.

c) Think about how you could efficiently implement the `makeGenBoxExtent` method.

d) Implement the `makeGenBoxExtent` method in the `Mesh` class.

e) How could you adapt the `intersection` method of the `Mesh` class so that this box extent is used to avoid expensive intersection calculations between the given ray and the mesh in some cases?

f) Add support for box extents to the `intersection` method of the `Mesh` class. (You only need to make minor changes to the `intersection` method of the `Mesh` class: adding a few lines of code at the correct place should be enough!)

Exercise 4

Time to check your work.

a) Run `App8` and check that you still get the expected image on your screen.

b) Animate the camera to the left and observe how long it takes your ray tracer to render the new image.

c) You can easily compare this with the rendering time before adding support for extents by commenting out the lines of code you just added to the `intersection` method of the `Mesh` class.

d) Change the initial position of the camera to $(0, 0, 5)$.

e) Will this new camera position change the rendering time?

f) Run `App8` again and animate the camera to the left to check your answer.

Exercise 5

Note that we computed box extents around untransformed mesh objects to speed up their rendering. However, alternatively, we could have computed box extents around the transformed mesh objects.

a) Think about how such an approach would change your current implementation.

b) Think about the pros and cons of such an approach.

(You do not have to implement this alternative approach.)

Exercise 6

Would it be possible to add extents to boolean objects? Explain your answer in detail.