Project 2 – Spatial Data Structure

CSE 168: Rendering Algorithms, Spring 2014

# Description

Add a spatial data structure (such as an AABB tree) to your renderer. It should be derived off of the Object base class and be instance-able. Also, add cast shadows to the renderer. It should be able to run the code listed below in the *Project 2 Function* section, and generate the image in the *Sample Image* section.

In addition, add some timing functions to both the BoxTree::Construct() and Camera::Render() functions and output the resulting times to the console. For the sample data provided, it should only take a few seconds to generate the data structure and a few more seconds to render.

# Mesh Format and Sample Data

You can use the dragon.ply file from the Stanford model library. A simple PLY file loader is also provided below.

# Project 2 Function

Project 2 should be able to be run with the following sample code (or something very similar):

void project2() {

// Create scene

Scene scn;

scn.SetSkyColor(Color(0.8f, 0.8f, 1.0f));

// Create ground

MeshObject ground;

ground.MakeBox(5.0f,0.1f,5.0f);

scn.AddObject(ground);

// Create dragon

MeshObject dragon;

dragon.LoadPLY("dragon.ply");

dragon.Smooth();

BoxTreeObject tree;

tree.Construct(dragon);

scn.AddObject(tree);

// Create instance

InstanceObject inst(tree);

Matrix34 mtx;

mtx.MakeRotateY(PI);

mtx.d.Set(-0.05f,0.0f,-0.1f);

inst.SetMatrix(mtx);

scn.AddObject(inst);

// Create lights

DirectLight sunlgt;

sunlgt.SetBaseColor(Color(1.0f, 1.0f, 0.9f));

sunlgt.SetIntensity(1.0f);

sunlgt.SetDirection(Vector3(2.0f, -3.0f, -2.0f));

scn.AddLight(sunlgt);

PointLight redlgt;

redlgt.SetBaseColor(Color(1.0f, 0.2f, 0.2f));

redlgt.SetIntensity(0.02f);

redlgt.SetPosition(Vector3(-0.2f, 0.2f, 0.2f));

scn.AddLight(redlgt);

PointLight bluelgt;

bluelgt.SetBaseColor(Color(0.2f, 0.2f, 1.0f));

bluelgt.SetIntensity(0.02f);

bluelgt.SetPosition(Vector3(0.1f, 0.1f, 0.3f));

scn.AddLight(bluelgt);

// Create camera

Camera cam;

cam.LookAt(Vector3(-0.1f,0.1f,0.2f),Vector3(-0.05f,0.12f,0.0f));

cam.SetFOV(40.0f);

cam.SetAspect(1.33f);

cam.SetResolution(800,600);

// Render image

cam.Render(scn);

cam.SaveBitmap("project2.bmp");

}

# Sample Image

The sample code above should generate the following image:

# 

# Grading

This project is worth 12 points:

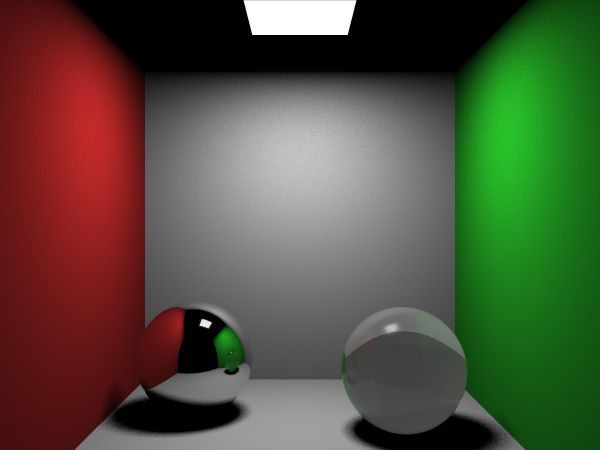
* Write code to build & traverse data structure 2
* Image renders correctly using data structure 4
* Construction & render time displayed 1
* Render with correct shadows 2
* Total runtime under 1 minute 3
* Total 12

# Extra Credit

For 1 extra credit point total, you must do *all* of the following:

* Add area lights
* Add Fresnel dielectrics and metals
* Add pixel supersampling for antialiasing

And then render an image something like this:



# PLY File Loader

Here is a basic PLY file loader. It compiles in VisualStudio but hasn’t been tested on other systems. You may need to move the #define line above any other #include’s at the top of the file.

#define \_CRT\_SECURE\_NO\_WARNINGS

bool MeshObject::LoadPLY(const char \*filename,Material \*mtl) {

// Open file

FILE \*f=fopen(filename,"r");

if(f==0) {

printf("ERROR: MeshObject::LoadPLY()- Can't open '%s'\n",filename);

return false;

}

// Read header

char tmp[256];

int numverts=0,numtris=0;

int posprop=-99,normprop=-99;

int props=0;

while(1) {

fgets(tmp,256,f);

if(strncmp(tmp,"element vertex",14)==0)

numverts=atoi(&tmp[14]);

if(strncmp(tmp,"element face",12)==0)

numtris=atoi(&tmp[12]);

if(strncmp(tmp,"property",8)==0) {

int len=strlen(tmp);

if(strncmp(&tmp[len-3]," x",2)==0) posprop=props;

if(strncmp(&tmp[len-3],"nx",2)==0) normprop=props;

props++;

}

if(strcmp(tmp,"end\_header\n")==0) break;

}

if(posprop==-1) {

printf("ERROR: MeshObject::LoadPLY()- No vertex positions found\n");

fclose(f);

return false;

}

// Read verts

int i=0;

if(numverts>0) {

NumVertexes=numverts;

Vertexes=new Vertex[NumVertexes];

for(i=0;i<NumVertexes;i++) {

fgets(tmp,256,f);

char \*pch=strtok(tmp," ");

int prop=0;

while(pch) {

if(prop==posprop) Vertexes[i].Position.x=float(atof(pch));

if(prop==posprop+1) Vertexes[i].Position.y=float(atof(pch));

if(prop==posprop+2) Vertexes[i].Position.z=float(atof(pch));

if(prop==normprop) Vertexes[i].Normal.x=float(atof(pch));

if(prop==normprop+1) Vertexes[i].Normal.y=float(atof(pch));

if(prop==normprop+2) Vertexes[i].Normal.z=float(atof(pch));

pch=strtok(0," ");

prop++;

}

}

}

// Read tris

if(numtris>0) {

if(mtl==0) mtl=new LambertMaterial;

NumTriangles=numtris;

Triangles=new Triangle[numtris];

for(i=0;i<numtris;i++) {

int count,i0,i1,i2;

fscanf(f,"%d %d %d %d\n",&count,&i0,&i1,&i2);

if(count!=3) {

printf("ERROR: MeshObject::LoadPLY()- Only triangles are supported\n");

fclose(f);

return false;

}

Triangles[i].Init(&Vertexes[i0],&Vertexes[i1],&Vertexes[i2],mtl);

}

}

// Smooth

if(normprop<0) Smooth();

// Close file

fclose(f);

printf("Loaded %d triangles from file '%s'\n",numtris,filename);

return true;

}

void MeshObject::Smooth() {

int i,j;

for(i=0;i<NumVertexes;i++)

Vertexes[i].Normal.Zero();

for(i=0;i<NumTriangles;i++) {

Triangle &tri=Triangles[i];

Vector3 e1=tri.GetVtx(1).Position-tri.GetVtx(0).Position;

Vector3 e2=tri.GetVtx(2).Position-tri.GetVtx(0).Position;

Vector3 cross;

cross.Cross(e1,e2);

for(j=0;j<3;j++)

tri.GetVtx(j).Normal.Add(cross);

}

for(i=0;i<NumVertexes;i++)

Vertexes[i].Normal.Normalize();

}