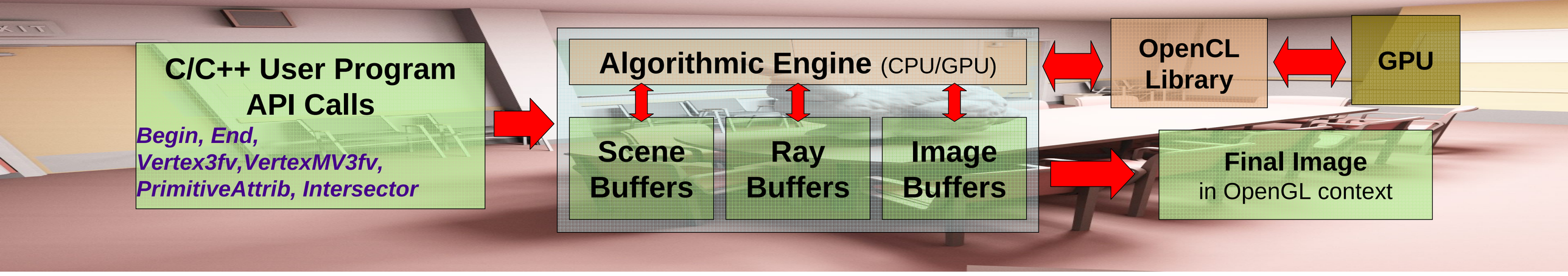
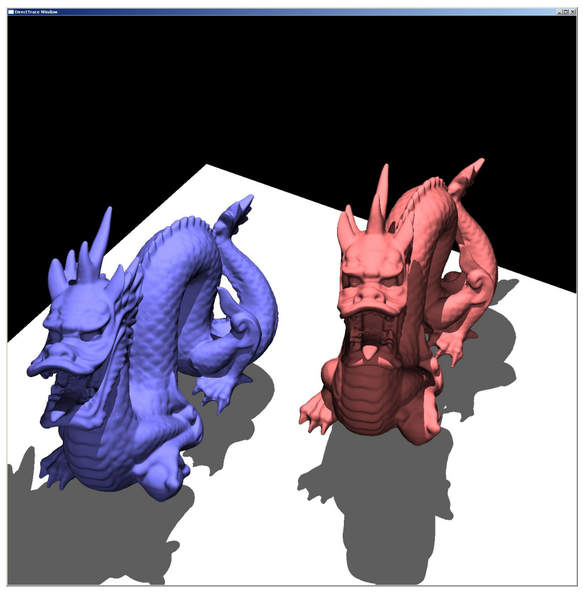


The Direct-Trace Library: Ray-Tracing for the masses

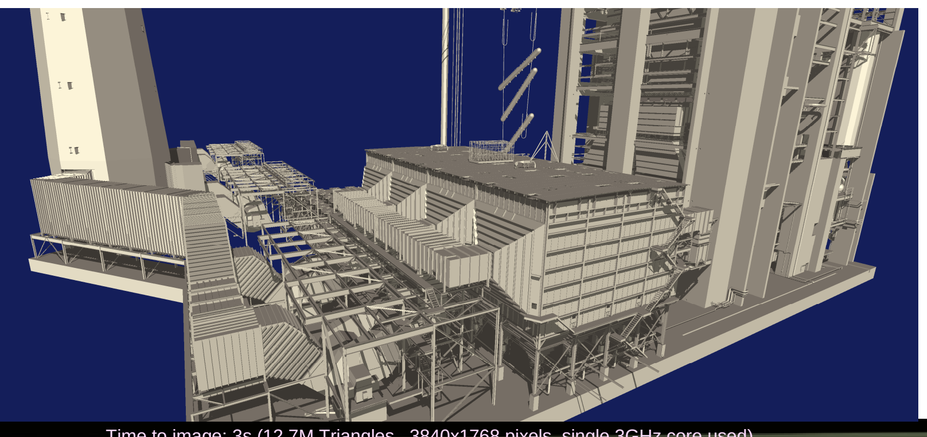
- **Simple Programming API**
- **State-Of-The-Art Performances**
  - **Without spatial subdivision data structures!**
  - **Quick processing of fully dynamic content!**
  - **~1M random Rays / Core (current).**
  - **~10M frustum Rays / Core.**
  - **OpenCL support.**
  - **Auto Management of Resources.**
  - **Increased Productivity.**
- **Interactive Global Illumination**
- **OpenGL-like programming style**
  - **With much improved usability.**
  - **Materials & soon Texturing.**
- **Shader Support**
  - **OpenCL & X86.**
- **EPSRC-funded Proof-Of-Concept**
  - **Availability Summer 2011.**
  - **www.DirectTrace.org**

```
//1. Declarations + setup
#include "DirectTraceAPI.h"
DTRayBuffer *primaryRays;
DTImage *finalImage;
DTImage *normalAtIntersection;
DTImage *uvAtIntersection;
DTScene *scene;
…
dtAPI=new DirectTraceAPI(CL);
primaryRays=new DTRayBuffer(*dtAPI);
normalAtIntersection=new DTImage(*dtAPI);
uvAtIntersection=new DTImage(*dtAPI);
finalImage=new DTImage(*dtAPI);
primaryRays->Resize(imageDimX,imageDimY);
normalAtIntersection->Resize(imageDimX,imageDimY,4);
uvAtIntersection->Resize(imageDimX,imageDimY,2);
finalImage->Resize(imageDimX,imageDimY,4);
…
//2. Setting up the scene
scene->Reset();
int redishId=scene->NewMaterial(sizeof(FloatVector3),3*sizeof(FloatVector3),0,nbTriangles,0);
int bluishId=scene->NewMaterial(sizeof(FloatVector3),3*sizeof(FloatVector3),0,nbTriangles,0);
int whiteId=scene->NewMaterial(sizeof(FloatVector3),3*sizeof(FloatVector3),0,2,0);
scene->MaterialAttrib(redishId,&redish,sizeof(redish));
scene->MaterialAttrib(bluishId,&bluish,sizeof(bluish));
scene->MaterialAttrib(whiteId,&white,sizeof(white));
scene->AllocateSpaceForNPrimitives(DT_TRIANGLES,2*nbTriangles);
//Set ground
scene->Begin(DT_TRIANGLES,whiteId);
scene->PrimitiveAttribMV3fv(groundNormal);
for (int i=0;i<6;i++)
        scene->VertexMV3fv(groundVertices[i]);
scene->End();
//Set two models
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glTranslatef(objectRadius*0.4,0,objectRadius*0.4);
scene->Begin(DT_TRIANGLES,redishId);
for (int j=0;j<2;j++)
{
        for (int i=0;i<nbTriangles;i++)
        {
                scene->PrimitiveAttribMV3fv(normals+3*triangles[3*i]);
                scene->PrimitiveAttribMV3fv(normals+3*triangles[3*i+1]);
                scene->PrimitiveAttribMV3fv(normals+3*triangles[3*i+2]);
                scene->VertexMV3fv(vertices+3*triangles[3*i]);
                scene->VertexMV3fv(vertices+3*triangles[3*i+1]);
                scene->VertexMV3fv(vertices+3*triangles[3*i+2]);
        }
        scene->End();
        glTranslatef(-0.8*objectRadius,0,-0.8*objectRadius);
        scene->Begin(DT_TRIANGLES,bluishId);
}
scene->End();
glPopMatrix();
//3. Running the Intersector.
primaryRays->SetFrustumRaysFromGLProjectionMatrix();
scene->Intersector(*primaryRays);
//4. Shading.
DTImage origin(primaryRays->Centers());
DTImage dir(primaryRays->Directions());
Handle *handles[5]={finalImage,normalAtIntersection,uvAtIntersection,&dir,NULL};
Handle *paramRecast[4]={&origin,&dir,normalAtIntersection,NULL};
uvAtIntersection->UVAtIntersection(*scene,*primaryRays);
primaryRays->Run(ShaderPhong,handles);
normalAtIntersection->NormalsAtIntersection(*scene,*primaryRays);
primaryRays->Run(RecastFromLightSource,paramRecast);
primaryRays->Touch();
scene->Intersector(*primaryRays);
primaryRays->Run(AddLightSourceContribution,handles);
//5. Display.
finalImage->DeInterleave(3);
finalImage->MapImageToWindow(NO_INTERPOLATION);

int RecastFromLightSource(void *materialAttrib, void *primAttrib, float *arrays[])
{
        float *origin=arrays[0];
        float *direction=arrays[1];
        float *normalAtIntersection=arrays[2];
        float t=direction[3];
        for (int i=0;i<3;i++)
        {
                direction[i]=(origin[i]+t*direction[i])-pointLightCoord[i];
                origin[i]= pointLightCoord[i];
        }
        bool  frontFaceDetection=
                normalAtIntersection[0]*direction[0]+
                normalAtIntersection[1]*direction[1]+
                normalAtIntersection[2]*direction[2]<0;
        return frontFaceDetection;
}
0
```



Example from SDK



Time to image: 3s (12.7M Triangles, 3840x1768 pixels, single 3GHz core used)