Object-oriented Decomposition of Global Illumination

Radek Ošlejšek
oslejsek@informatics.muni.cz

Faculty of Informatics
Masaryk University
Brno, Czech Republic

Abstract

The paper deals with the models of object-oriented decompositions of rendering process. Three simple architectures with the description of communication will be presented here. They serve as the examples of our approach. Our research goal is to design and study complex rendering architectures using the benefits of current OO technology.

KEYWORDS: Object-oriented decomposition, global illumination, ray tracing, radiosity.

1 Introduction

OO graphics architectures were thorougly studied in the last decade. The published systems (eg. [Chen95], [Fell96], [Slus95]) have shown promising results. Current OO technology offers many possibilities to reimplement traditional rendering techniques in OO class hierarchies. It also enables us to research new global renedering methods and experiment with them in OO environment. The paper is tructured as follows: First we sketch briefly the classes we use in our experimental architectures. Then we described 3 classical methods converted to our OO environment. In the last section we will mention our future work based on real world paradigm.

2 General description

In Figure 1 and Figure 2 there are presented association and collaboration graphs of general approach of OO decomposition.

We use the following classes:

Viewer – Help class which only starts process of rendering.

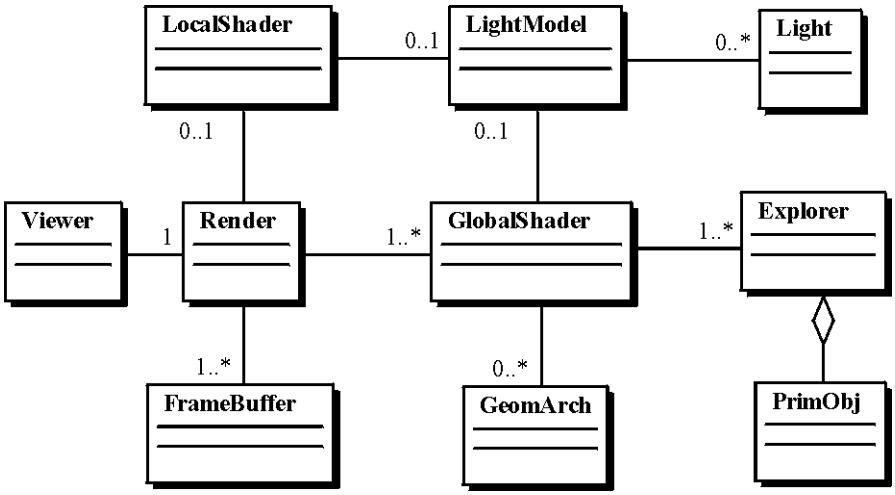

Figure 1: General decomposition - association

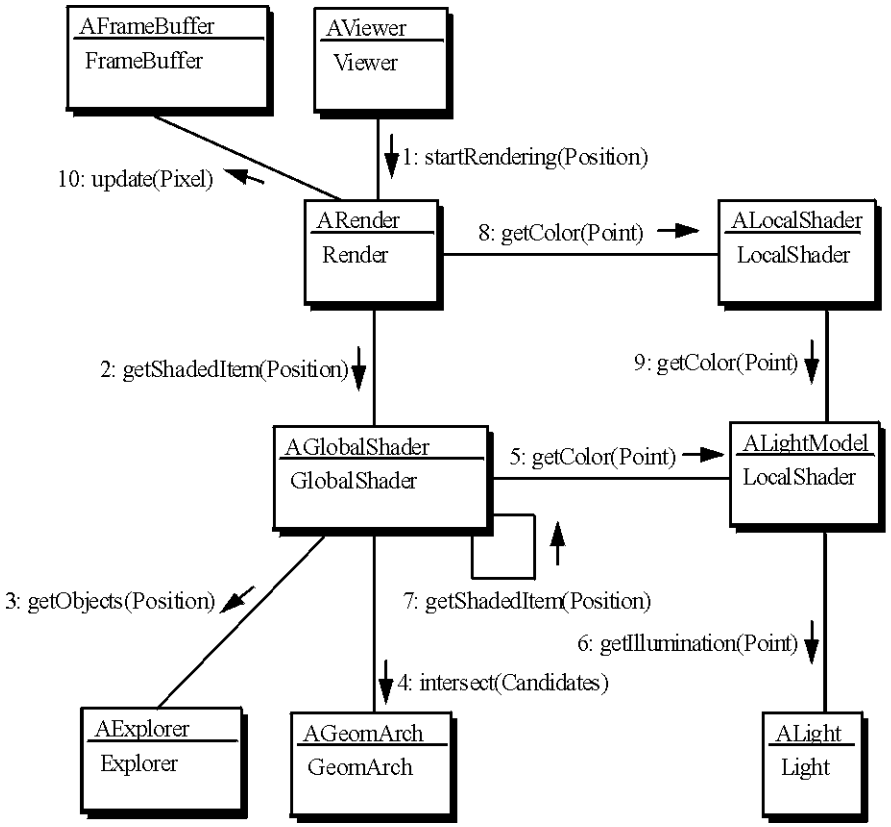

Figure 2: General decomposition - collaboration

Render – Manages operations used to create 2D picture, sends informations about avatar's position and view direction to global shader, updates frame-buffer etc.

GlobalShader – Solves global illumination. This object usually prepares informations for local shader.

LocalShader – Computes color of entire patch. Computation uses informations taken from light model.

LightModel – Computes color of one point.

Light – Light source. Can be dot light, directional light or spot light. Instance of this class returns light intensity in given 3D point.

GeomArch – Geometric architecture. This object is specialized to compute quickly the intersections of ray with objects in scene.

Explorer – Walks in space and returns interesting objects such as candidates for intersection with ray. Explorer knows about all objects in scene.

PrimObj – Primitive object. It is stored in Explorer.

3 Case study I - Flat Model

At the beginning the object **Render** sends message **GlobalShader**::*beginShading()* to initialize rendering process. The method *beginShading()* has an argument describing view direction. It is also capable to discover ordered objects using **Explorer**. This is done by method **Explorer**::*objectsAhead()*. In the next steps **Render** sends vantages to **GlobalShader**. **GlobalShader** inspects lights and returns actual primitive with precomputed informations for **LocalShader** such as colors in vertices or just one color per surface. **Render** sends the selected primitives to **LocalShader**. **LocalShader** returns colored pixels which are written by **Render** to screen.

The collaboration diagram of the rendering process is shown in Figure 3.

4 Case study II - Ray Tracing

Render generates primary rays and takes the resulting colors of pixels from **GlobalShader**. **Render** controls the sampling strategy. It may cast one ray for each pixel, or more sapling rays per pixel with subsequent weighted color averaging. Using camera parameters **Render** sets inital ray direction according projection method.

GlobalShader is invoked by *beginShading(primary_ray)()* message. From **Explorer** he takes candidates for intersection and from object **GeomArch** it takes differencial neighborhood for one real intersection.

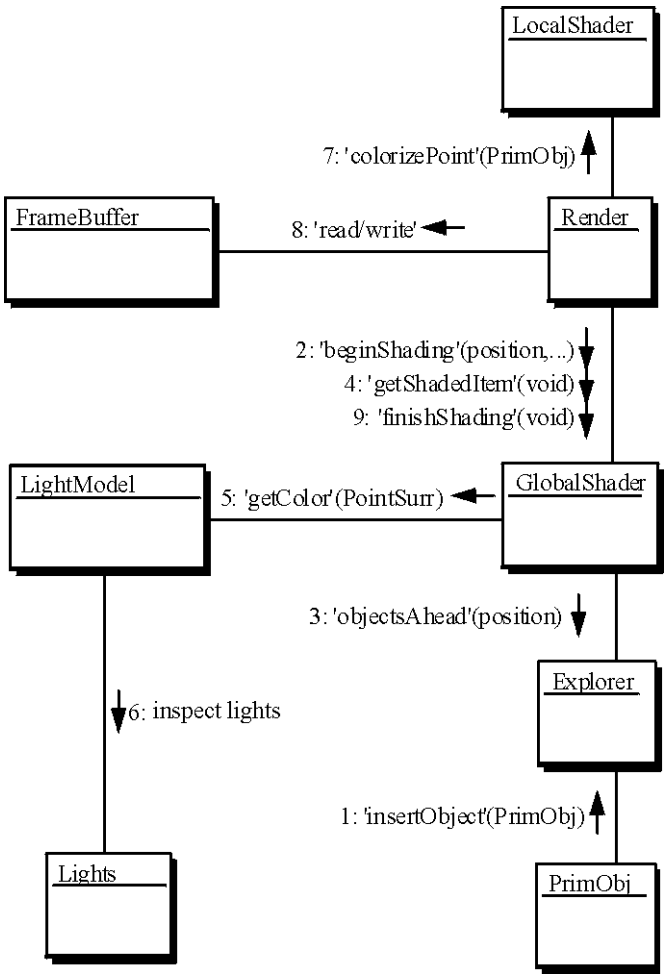

Figure 3: Communication protocol for flat model

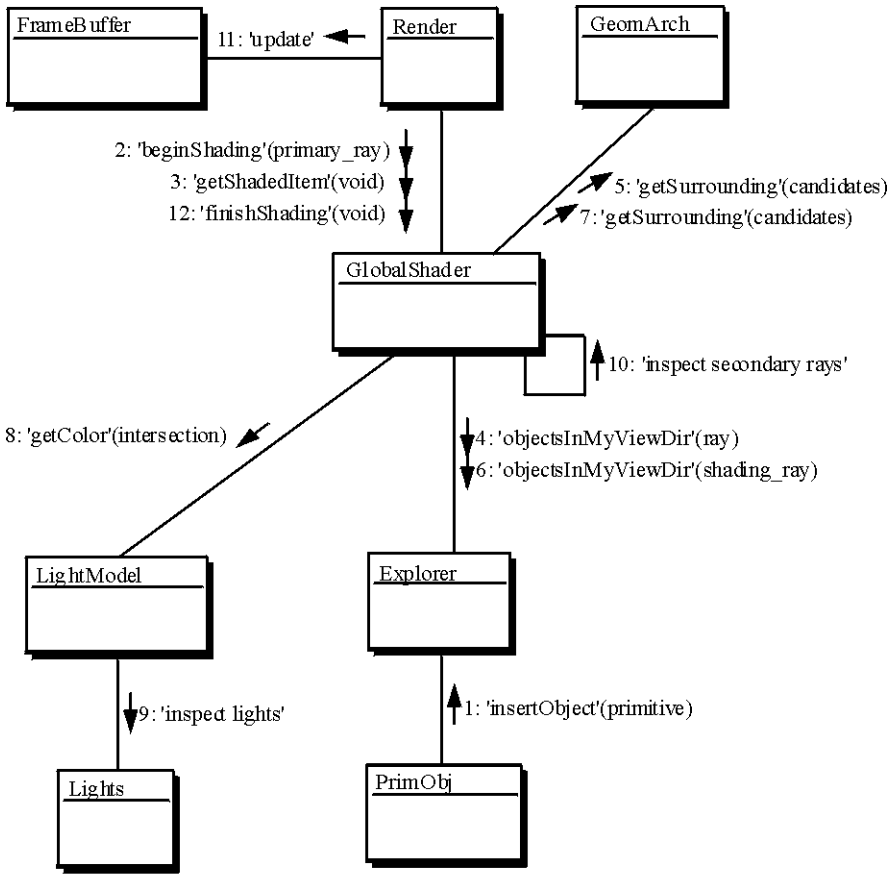

Figure 4: Communication protocol for ray tracer

The next step is the computation of color contribution to ray. This is done by inspection of lights. **GlobalShader** casts the shading rays to light sources and intersection computation of these rays with objects in scene is done (via cooperation of **Explorer** and **GeomArch**). When intersection with some object exists and if it is nearer than light source then the tested surface point is shaded by this object.

In the last step **GlobalShader** generates reflection and transparency rays and calls itself recursively for these secondary rays.

5 Case study III - Progressive Radiosity

Graphic system implementing simple version of progressive radiosity can work by the next schedule.

In the begin **Render** calls method **GlobalShader**::*beginShading()* and it starts rendering process. **GlobalShader** gets the position of light sources. These are used as starting surfaces for energy distribution. In the next step **GlobalShader** takes visible objects from **Explorer**, computes form factors and deposits shooted energy to patches. Patches with deposited energy are stored locally and they are used in the next iteration step.

Next steps are similar to flat model. **Render** takes precomputed primitives by method **GlobalShader**::*getShadedItem()*, these primitives are sent to **LocalShader** and colored pixels are drawn to frame-buffer.

6 Advanced techniques and future work

In the real world we may find the following situation: The world contains many static objects such as houses or roads. Adventurer going to some unknown piece of land starts with the empty map (blank sheet of paper) and he draws the dicovered objects to map - first with low precision and without details, later he can add precise measurements and his map is getting more useful. First this operation takes some time and so he proceeds slowly. But the next travels through already mapped area should be faster and more effective.

Our idea is to use the above described situation as paradigm for the dynamic scene with global rendering. We have already proposed and implemented the model in which explorer is looking at map during his travel in space. It offers many possibilities to experiment with different space sorting structures, bounding volumes speed-up techniques etc. The research of these advanced models is the topic our current and future work.

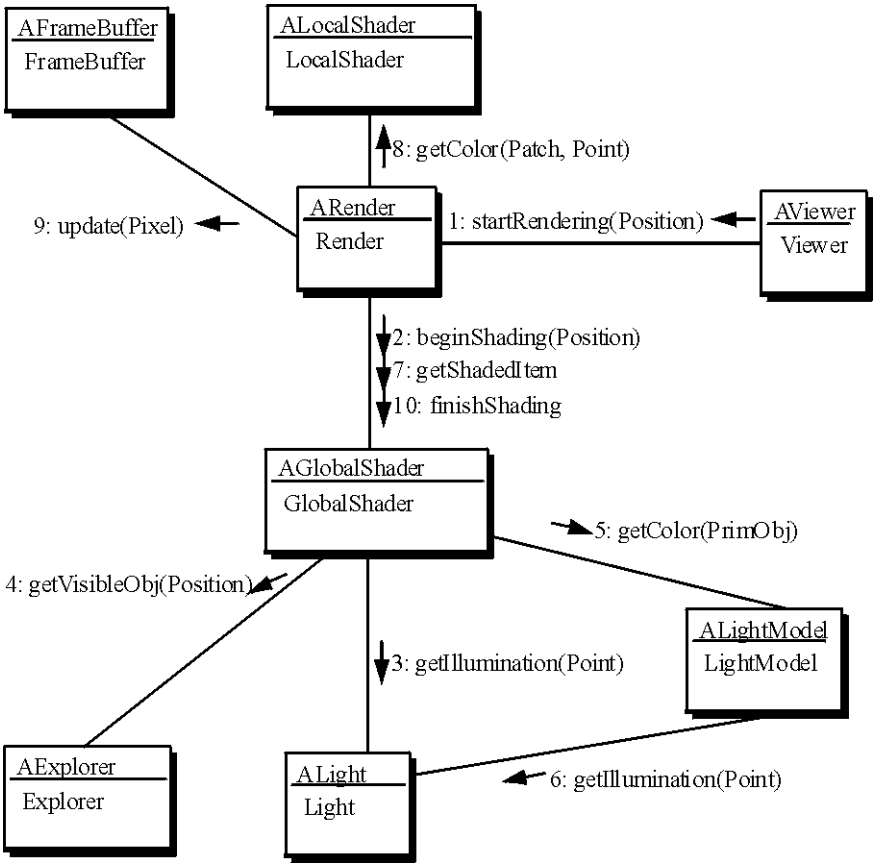

Figure 5: Progressive radiosity communication protocol

References

[Chen95] Chen S.E., Turkowski K., Turner D.: *An Object–Oriented Testbed for Global Illumination.* In: Laffra et al. (Eds.) Object–Oriented Programming for Graphics. Springer–Verlag, 1995, pp.155–166

[Fell96] Fellner,D.W.: Extensible Image Synthesis. In: *Object-Oriented and Mixed Programming Paradigms*, Wisskirchen P., (Ed.), Focus on Computer Graphics, Springer, Feb. 1996

[Slus95] Slussalek,P., Seidel,H.P.: Vision - An Architecture for Global Illumination Calculations. In: IEEE Trans. *Visualization & Computer Graphics* 1(1), 1995