Conférence Internationale en Automatique & Traitement de Signal (ATS-2017)
Proceedings of Engineering and Technology – PET
Vol.22 pp.7-12

Rendering virtual world

[1] khemliche sarra, [2]Babahenini Mohamed Chaouki
LESIA laboratory
Mohamed Kheider University
Biskra, Algeria
[1]s.khemliche@univ-biskra.dz
[2]chaouki.babahenini@gmail.com

Abstract—**our work is subscribe in the field of many light rendering based on virtual point light (VPL) to calculate the global illumination, we interested in introducing a new instant radiosity method different from the one which introduced by keller for the model of the global rendering , we improve the rendering by using the deffered rendering rather than the forward one to accelerate the computing enhanced by the increase number of objects and light sources. To well perform these techniques we used the shader compute shader as allowing us to implement ray tracing to find the points of intersection and the creation of VPLs. We systematically study simulation of complex effects on image quality and material appearance of global illumination approximations made by VPL algorithms to provide a fast and smoother complex illumination .Efficiently, the computational effort is heavily reduced.**

Keywords— Global Illumination, locale illumination, instant radiosity, deffered shading, forward shading.

I. INTRODUCTION

Computer graphics has a big impact on a wide range of various fields, such as film, video game and design, to reproduce a virtual world as complex as the reality around us.
Realism is an essential element to generate an image ideally indistinguishable from a photograph of the real world. Many aspects must be balanced to achieve this goal including the quality of modeling three-dimensional objects that appear in the scene, the number and fineness of detail shown on the objects and / or the stage and finally the quality of lighting which is applied in the simulated environment.
The image synthesis process involves two main steps:
• **modeling** the stage of storing the various information necessary for the construction of the stage (usually 3D) such as geometric coordinates of objects, light sources, texture and material properties.
•**Calculation of illumination** which is to see the different objects in the scene by simulating the lighting and calculate the different interactions between light and matter as transparency and reflection.
Considered a key element for the calculation of realistic images, global illumination techniques have experienced in recent years a significant improvement in visual terms as well as the calculation time. State of the art usually relay on traditional methods ; radiosity, ray tracing, the path tracing, the metropolis light transport and photon mapping, a combination of these algorithms can be used.
To make photo-realistic images with global illumination techniques, the exact calculation of the distribution of the light in the 3D scene is the most important element to take into account.for this end Kajiya [4] introduced in 1986 the rendering equation to model the distribution of light in a scene. an exact solution of this equation is impossible since there are an infinite number of incident light. In order to solve this equation several techniques have been developed. However, all approaches suffer from computing time, which is still insufficient to meet the requirements of industries such as video games.

The objective of our work is to simulate the light propagation in the scene with less time consuming.

The organization of this paper is structured as follows:
Section 2 gives an overview of background that used in the field of real time global illumination. In the section 3 we will detail the implementation and testing of our application, and balance sheets, where we will explain the implementation of our application. Finally, we close this paper with a conclusion and perspectives.

II. BACKGROUND

In the field of computer graphics, generating photorealistic images is requires a precise calculation of the distribution of the light in the 3D scene, this latter is difficult to be attend because of its complexity in the real world where the light emitted by a source bounces of objects that constitute this world, in fact, the calculation of the color of a point in a 3D scene requires the integration of all the incident light. Several methods have been developed for the lighting calculations. Generally, we can classify them under two major categories: the first is ***local illumination*** lighting which calculates by considering the interactions between the light sources and the object, while the second is ***global illumination*** that not only recognizes objects-sources interactions but also objects-objects interactions.
In this section we will give a general definition of local illumination, determine its limits and motivations for global illumination, inducing the various basic concepts of radiometry, the light-to-surface interaction and finally the equation made with methods of resolution.
The local illumination is a method of calculation used in illumination 3D image synthesis. It calculates the apparent brightness of a volume, considering only the interactions between light sources and objects. [3].
Unlike local illumination that considers the interaction of light between the object and the light sources, global illumination take into account not only the object-source interaction but

ISSN 2356-5608

also the subject-object interaction (interaction between the reflected object and the light from other objects in the scene).
The role of global illumination is to calculate the energy distribution of light in a scene. The images rendered using the global illumination techniques are more realistic as rendered images by using local illumination techniques. However, they are also much slower and more costly in terms of computing time.
Rendering photo-realistic images with global illumination techniques requires the basic concept of knowledge of radiometry and the way light is reflected by the object, i.e the light object interaction. In the following figure, we will introduce some basic definitions to arrive at the end of this section to define rendering equation in detail.

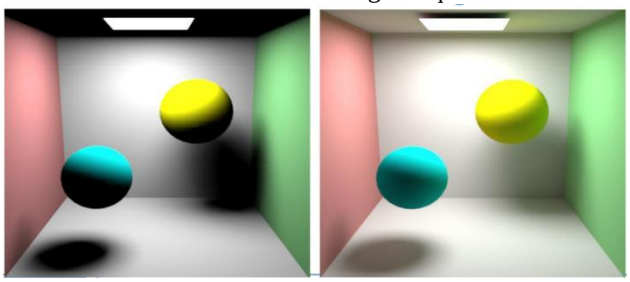


Figure 1 : direct illumination vs global illumination

In computer graphics, the distribution of energy in a scene was mathematically formulated first by Kajiya [4] ,in the form of an equation called rendering equation. The latter calculates the outgoing luminance of a point of the surface based on the luminance emitted by the surface, the incident luminance and material properties of the surface.

It says that at each particular point that has a position and direction on a surface, the outgoing light L_o at a surface location x in direction ω is the sum of emitted radiance L_e and reflected radiance L_r(Eq.1).

$$L_o(x,w) = L_e(x,w) + L_r(x,w) \qquad (1)$$

The reflected radiance is computed as

$$L_r(x,w) = \int_{\Omega^+} Li(x,w_i) f_r(x,w_i \to w) < N(x), w_i >^{-} dw_i \quad (2)$$

Where Ω^+ is the upper hemisphere oriented around the surface normal N(x) at x, f_r the bidirectional reflectance function (BRDF) and<>$^+$ a dot product that is clamped to zero. To determine the incident radiance, L_i, the ray casting operator is used to determine from which other surface location this radiance is emitted and reflected. The goal of global illumination algorithms is to compute L_o(x, ω) for a given scene; materials and lighting L_e [6].It is a recursive equation which makes it unsolved analytically by exact method for this end several methods have been developed to solve the rendering equation.

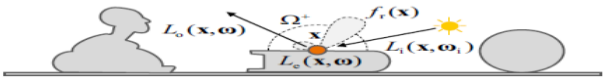


Figure.2 rendering equation

Calculate an image of a 3D scene physically realistic interactive in real-time is a key objective of current research in the field of photorealistic rendering. For this it is necessary to calculate all light interactions in the scene. The algorithms are algorithms realizing this global illumination. Achieving this goal would allow a variety of applications, in this context; many algorithms have been presented, taking into account all or part of the effects.

In this paper, we present an implementation of the reconstruction of one of them, the algorithm called "instant radiosity". [2] It belongs to the family of algorithms using virtual point light (vpl) sources to simulate indirect lighting in diffuse scenes. VPLS are valid only in the context of diffuse scenes because their issue is supposed lambertian. This algorithm is not a general algorithm: scenes with reflections and refractions are not supported, but it is very effective for diffuse scenes.

III. INSTANT RADIOSITY ON GPUS: IMPLEMENTATION, TESTING, AND ASSESSMENT RESULTS

This section aims to describe the different stages of implementation of our system, it will allow us to define the process of realization, then present some optimization techniques instant radiosity. Finally get to mention some results.
Our main objective is the establishment of a real-time application to diffuse global illumination of a 3D scene, we choose it to achieve instant radiosity technique, since it has several advantages and it is achievable via shaders, using the graphics pipeline OpenGL and features new graphics cards.
Our choice for carrying out the technique of the instant radiosity is motivated by:

- Improve the realism
- Minimize the time and processing power to render
- Applicable in a real time frame unlike other methods

Our main objectives are:

- Find the intersections of the rays with objects in the scene and the creation of VPLS.
- Speed up the calculation time with the use of GPU.
- •Application and comparison of the various shading techniques

Conducting the instant radiosity approach: To understand the idea of our system we have developed the following diagram:

ISSN 2356-5608

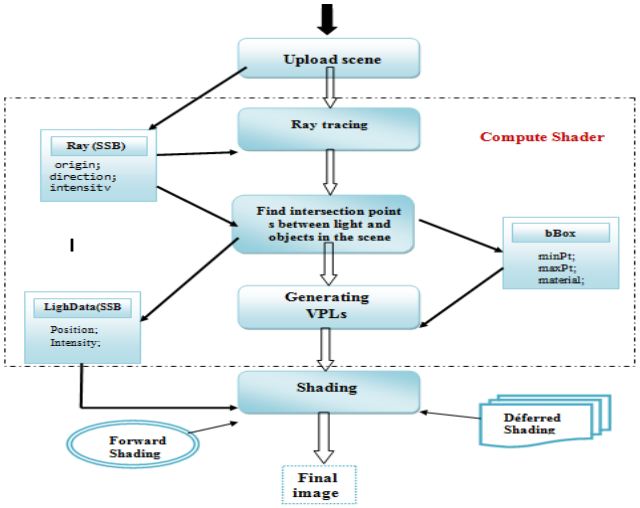


Figure 3: stages of our application

In this section we will present the various stages of implementation of instant radiosity using different techniques fit into the programming framework, for this we must do the following:

- **Specify the entry**: define the description file of the scene
- **Modeling**: To describe the geometry of the scene
- **Add VPLs**: shooting rays and find the intersections
- **Rendering**: allowing us to calculate the lighting parameters (Forward Shading / Shading DEFFERED)
- **Output**: use the result in an .exe file

1. STRUCTURES AND TECHNIQUES USED:

- **Compute Shader**: perform ray tracing and creation of VPLS of the scene, it is a solution that would work on most graphics cards
- **Datalight / Ray:** We have created structures to store information on each ray to be traced by the Compute Shader (Ray) and to store information on the location of the VPL in the scene (LightData). Ray structure consists of an origin, a direction and an intensity value that will be assigned to each VPL that will be created at the intersection of this ray with an object in the scene. The LightData consists of two elements, the position and intensity, intensity is a value vec4 wherein we store the color of the VPL in the r, g and b components, and the actual intensity in the component.
- **Shader Storage Buffer (SSB)**: A structure for transferring data between the rendering program (the host side code) and shaders.

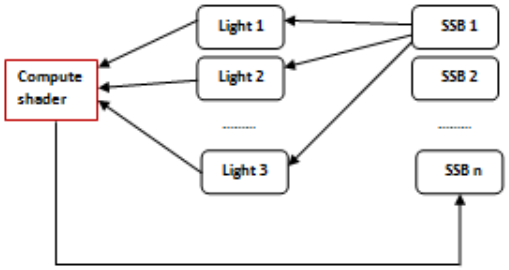


Figure 4.Communication compute shader SSB

- **Setting the scene description file**

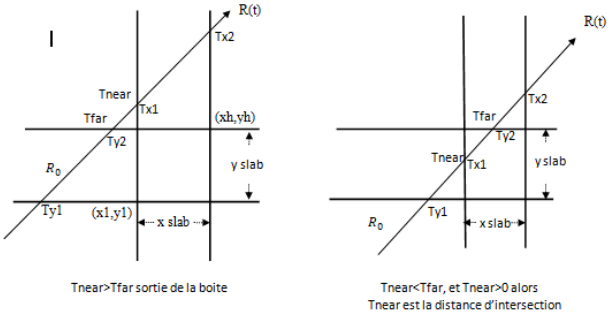
We entered as a scene in the form of a file (.obj) that contains the information necessary to the scene geometry (positions, faces, normal ...) and file (.mtl) (complement format (.obj)) which contains the definition of material 3D objects (components: diffuse, ambient ... etc), you will find below an example of .obj.

```
mtllib cube.mtl
v 0.000000 500.0000 500.0000
v 0.000000 0.000000 500.0000
v 500.0000 0.000000 500.0000
v 500.0000 500.0000 500.0000
v 0.000000 500.0000 0.000000
v 0.000000 0.000000 0.000000
v 500.0000 0.000000 0.000000
v 500.0000 500.0000 0.000000
# 8 vertices
g front cube
usemtl white
f 1 2 3 4
g back cube
# expects white material
f 8 7 6 5
g right cube
usemtl red
f 4 3 7 8
g top cube
usemtl white
f 5 1 4 8
```

Next we create the necessary structures to store information and used in order to calculate the coordinates of the point, color and other stages of completion. Once the modeling stage is completed we can start treating our problem.

- **Add VPLs:**

To find the intersection points we used the method kay-Kajiyaslabs (intersection between the radius r and the box (box) specified by boxMin and boxMax)



Algorithm :

The box is defined by two coordinate: minimum measurement box BI = [x1, y1, z1] and maximum measuring the Bh box [xh, yh, zh] .The radius R is defined by R0, Rd

ISSN 2356-5608

```
Real Tnear = -∞,  Tfar = +∞;
for each pair P of plans associated to  X,Y,Z do
        if  (R parallele to X plan)then
            if origine X0 isn't between slabs (X0<X1 or  X0>Xh) then
                    return false ;
            end if
    else
            //calculate the distance of the intersection plans
            T1=(X1-X0)/Xd;
             T2=(Xh-X0)/Xd;
           if (T1>T2) then  swap (T1,T2);    // T1 the intersetion with the plan X;
                    if (T1>Tnear) then  Tnear=T1;
                    if (T2<Tfar) then  Tfar=T2;
                     if (Tnear>Tfar)  then  box unceccecful return false ;
                     if (Tfar<0)  then   box behind  R return false ;
    end if
                      Return  (Tnear,Tfar);
End for
```

2. RENDERING:

- **Forward Shading**: direct illumination is computed contributed by each stage light for each fragment in the fragment shader directly following the call to the vertex shader. The progress of this technique in the rendering pipeline is as followings:

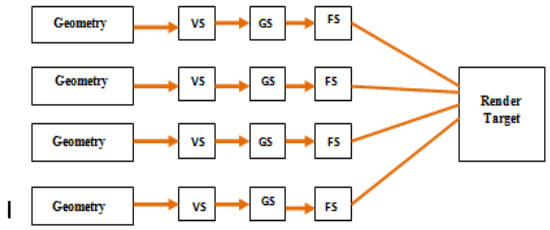


Figure 5:Running forward shading in the rendering pipeline

Algorithm :

```
for   each real light source  do
        output color  ←—  old output color +
                (color SL* intensity SL*difusion factor);
endfor
 for    each VPL do
        output color   ←—   old output color     +
                (color VPL* intensity VPL* difusion factor);
endfor
```

- **Deffered Shading :** Deferred Shading or G Buffer is a popular technique in many applications (games). The key point behind DEFFERED shading is the decoupling of the geometry calculations (position and normal transformations) and lighting calculations.

In our application we have used this technique to display the virtual light sources and some buffers.The following diagram shows the general architecture of this method:

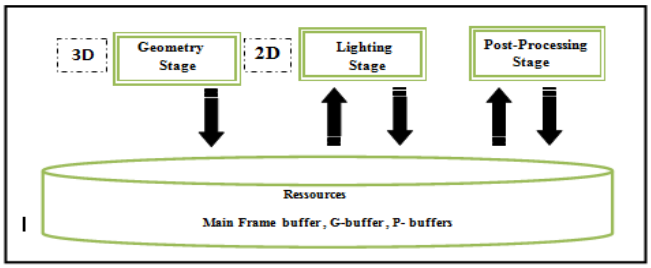


Figure 6: deferred rendering architecture

These parts (stages) using the programmable pipeline functionality, each party communicates with others through a shared memory area organized in buffers (buffers) that can be render target (writing in video memory on the graphics card) or texture (read)

- Geometry course: it is responsible for filling the g-buffer with the information needed to use them in the next steps.
- Lighting placement: has as input the g-buffer and the information of light sources and lighting accumulate in P-buff Post-processing training: several treatments are performed to improve the image result of P-buffer. Finally, the image will turn a hand frame buffer to be displayed on the screen.
- The exchange of information between steps through the resource structure (video memory)

3. RESULTS AND BALANCE SHEET

Established results were achieved using the tools and the following configurations:

- **Tool and platform used**
  - ***OpenGL 4.3***

Several new features in OpenGL 4.3 specification:
• GPU compute shaders that exploit the parallel computing (image, volume and geometry processing in the context of the graphics pipeline);
• shader storage buffer objects (shader storage buffer) that enable vertex, geometry, and fragment shaders calculation to read and write large amounts of data and transmit important data between shader stages;
• increased memory safety that guarantees application can't read or write outside its own stamps in another application data, etc

  - ***Visual C ++ 2010***:C ++ has been one of the most popular commercial programming languages. Microsoft Visual C ++ 2010 Express offers a powerful code editor with support for syntax highlighting. Finally, the tool is particularly widely used in the creation of open source projects for Windows.
  - ***GLSL (OpenGL Shading Language)***

GLSL shader is a programming language. These allow advanced control of the pipeline of the graphics card. The GLSL was developed by the OpenGL Architecture Review Board to facilitate shader programming with the OpenGL API.

ISSN 2356-5608

- **Configuration**

  For a better assessment of our application, we have tested two types of machines, a graphics workstation and work PC. We will present in the following hardware and software configuration of the two machines.

  - **hardware Requirements**

- Graphic Station: the graphics station is equipped with an Intel i7 CPU core, 2.8GHz, RAM 12 GB, and two NVIDIA GeForce GTX 480.
- Work PC: PC with an Intel Pentium CPU, 2.00GHz, 2GB RAM, and a NVIDIA GeForce GT 635.

1. RESULTS :

Local illumination

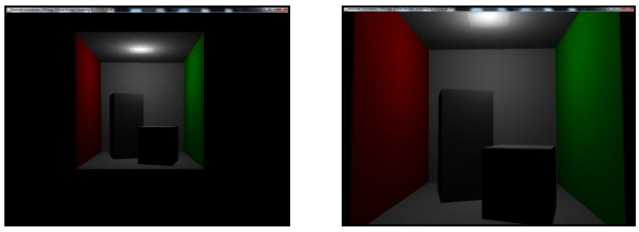
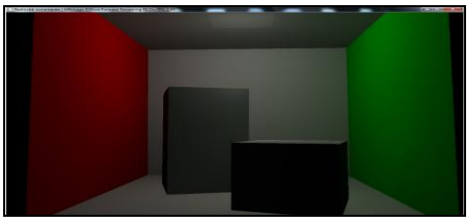
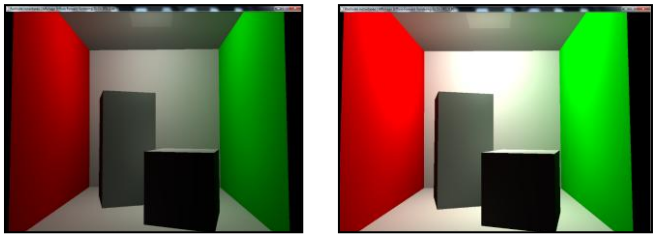


Figure7:Display Forward rendering, global illumination: DisableFPS ≅ 60

Global illumination:



(a)



(b) **(c)**

Figure8: Display : Forward Rendering ,Global illuination: active
numVPLs : (a) = 250 , (b) = 500 , (c) = 900
FPS : (a) ≅ 3 , (b) ≅ 2 , (c) ≅ 1

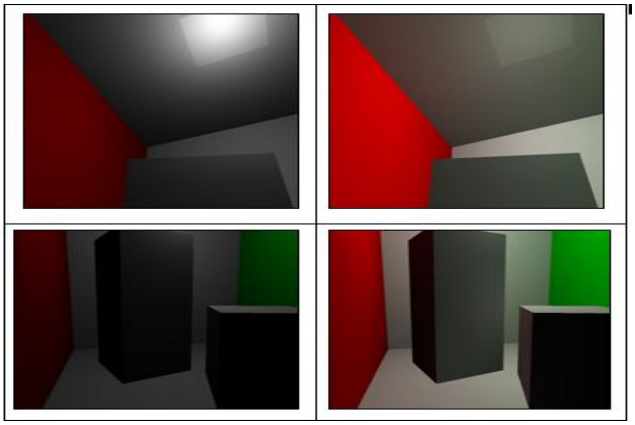


Figure 9 :Comparison global/locale Illumination

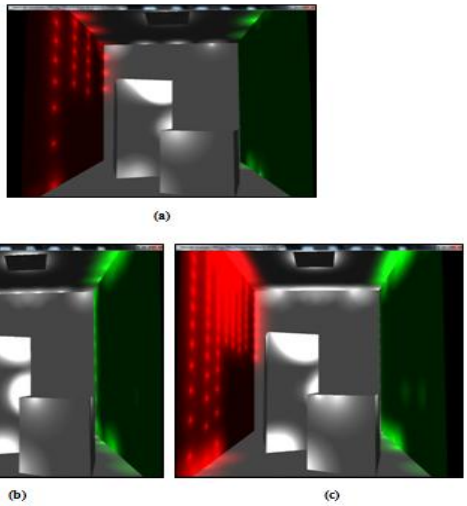


Figure10:Display VPLs:numVPLs : (a) = 250 , (b) = 500 , (c) = 900

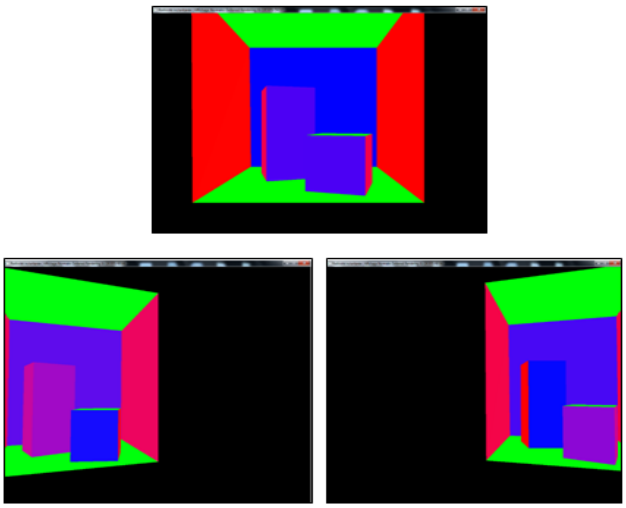


Figure11: Display buffer normal of G-buffer

DISCUSSION:

ISSN 2356-5608

The number of VPLs plays a very important role in determining the quality of the scene, if you increase this value, realistic scenes is obtained because they represent most of the interactions between the surfaces, we also note that in the case of Forward shading: if we increase the number of VPLS calculation time also increases very quickly, so you have to use a method which minimizes the computation time while maintaining a scene of good quality, this method not only solves the problem of a large number of VPLS, but also solves the problem of the complexity of the scene according to the number of objects and light sources the DEFFERED shading.
To understand this change we make a comparison between the two techniques in terms of number of VPLS.

Numberof VPLs	Forward Shading	Deffered shading
20	38.09	40.16
50	17.65	39.45
90	9.62	30.01
200	4.51	19.55
500	1.81	9.87
800	1.11	6.70

Table1: results of performance on NVIDIA GT 635

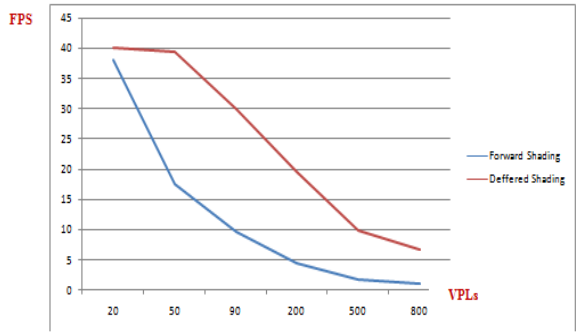


Figure6 :FPS change depending on number of VPLs

This graph shows the two time curves calculation made by Forward Shading and DEFFERED Shading .We can observe that the calculation time is changing so we increase the number of VPLS, thus the computing time is minimized observable way by the method of DEFFERED shading relative to the forward shading by that calculation with DEFFERED shading is divided into several stages (stages).

IV. CONCCLUSION AND OUTLOOK

The work done in this paper is in the context of the overall record with Instant radiosity technique. We have implemented a technique that consists of two phases: the first for ray tracing find intersections then create the VPLS and the other for rendering. The two phases were implemented according to the kay and Kajiya algorithm and Keller respectively. [5] [1]

This work allowed us to conduct a study on all the rendering illumination techniques namely: local illumination and global illumination, to emerge in the technique of instant radiosity and expand to other rendering techniques, we can mention: ray tracing, DEFFERED shading, shading forward.

In the end, we can say that the result of this work has fulfilled its role perfectly for study and implementation of the algorithm of instant radiosity, we draw to the end make more realistic image.

Outlook:

This work is a first step we are planning improvements and extensions to the following points:

- Selects sampling to control the roads and intersections points rays with objects in the scene.
- improve the implementation of the DEFFERED shading technique for indirect illumination
- Extend scene to multiple objects with complex geometries and variable.

REFERENCES

[1] Alexander Keller Instant Radiosity , 1997
[2] Anthony Pajot, Loïc Barthe, Mathias Paulin, Reconstruction sur GPU de Metropolis Instant Radiosity, 2012
[3] Eisemann .E, Cours synthèse d'image, Inria alpes, 2007
[4] Kajiya .J, The rendering equation, Siggraph Computer Graphics 1986.s
[5] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes,Computer graphics and interactive techniques, 1986
[6] T,Ritschel. C ,Dachsbacher.T, Grosch, & Kautz, J. (2014, February). The state of the art in interactive global illumination. In *Computer Graphics Forum* (Vol. 31, No. 1, pp. 160-188). Blackwell Publishing Ltd.

ISSN 2356-5608