REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IVA24/M2/2021

## Mémoire

Présenté pour obtenir le diplôme de master académique en

# Informatique

Parcours : **Images et vie artificielle**

---

# Lancer de rayon optimisé pour les surface de forme libre Bézier

---

**Par :**

**BAIDA YASSINE**

Soutenu le 1 juillet 2021, devant le jury composé de :

| | | |
|---|---|---|
| | | Président |
| Hamida Ammar | MAA | Encadreur |
| | | Examinateur |

## Résumé

L'objectif de ce mémoire de master est d'étudier, de comparer et d'améliorer les approches existantes pour les surfaces de forme libre Bézier ajustées par lancer de rayons directs.

Dans ce rapport, nous montrons comment implémenter le lancer de rayons direct des surfaces de Bézier de manière efficace et plus rapide dans le temps de calcul, Nous utilisons la méthode de Bézier clipping pour trouver l'intersection rayon-surface. Pour que la méthode de Bézier clipping fonctionne en temps de calcul plus rapide, nous devons construire une hiérarchie de volume englobant à chaque intersection rayon -patch, pour la construction de BVH (hiérarchie de volume englobant) de surface de Bézier en utilise la méthode de subdivision de casteljau et les volumes englobant aligné sur les axes (AABB) pour la facilité de mis en œuvre de BVH.



**Mots clés** :

Lancer de rayons, optimisation, la surface de forme libre Bézier, AABB, BVH, Bézier Clipping

## *Remerciements*

*En tout premier lieu, je remercie* ***Allah*** *de m'avoir donné la force et l'audace pour dépasser toutes les difficultés.*

*Je remercie Monsieur* ***Hamida Ammar,*** *Professeur à l'université de Biskra, en tant que Directeur de mon mémoire, il m'a guidé dans mon travail et aidé à trouver des solutions pour avancer.*

*Et je remercie ma Famille surtout ma mère et mon père*

*Enfin Je remercie mes amies tous ceux qui ma connaissent.*

# Table de matière

## Chapitre II : L'optimisation de lancer de rayon pour la surface de Bézier



## Chapitre IV : la conception et l'implémentation

## Table de figure

# Introduction général

La synthèse d'image a connu une évolution fantastique ces deux dernières décennies et son champ d'application est très varié. Toutes les méthodes et algorithmes dérivés de la synthèse d'image sont maintenant utilisés dans des plusieurs domaines tels que : la production des films, les jeux, les simulations d'environnement ...etc.

Le processus de création d'une image de synthèse comprend quatre parties : la modélisation géométrique, la simulation des interactions lumière-matière (ou technique de rendu et la simulation d'éclairage), la visualisation et l'animation (Création de séquences d'images animées).

La modélisation géométrique consiste à représentation mathématiquement des objets et placer dans une scène virtuel. Après l'étape de modélisation, des sources de lumière sont placées dans la scène et la simulation de I' éclairage de cet environnement est réalisée. II s'agit en fait de calculer toutes les interactions entre les sources de lumière et les objets de la scène .une fois la simulation de I' éclairage terminée, le processus de visualisation peut commencer qui est simplement à transformer des images 2D vers des images 3D.

En exploitant soigneusement les ressources du matériel informatique d'aujourd'hui, le lancer de rayons est récemment devenu une réalité, même sur un seul PC de base. Dans la plupart de ces implémentations, les triangles sont utilisés comme seule primitive géométrique. Cependant, le rendu direct de surfaces de forme libre Surtout les surfaces de Bézier serait avantageux pour un grand nombre d'applications, car la tessellation robuste de scènes complexes en triangles est un processus très long. De plus, les scènes constituées de surfaces de forme libre Bézier nécessitent moins de mémoire et offrent une précision beaucoup plus élevée, ce qui entraîne moins d'artefacts de rendu.

Nous allons travailler dans ce mémoire sur l'accélération de technique de lancer de rayon pour les surfaces de forme libre Bézier.

Le travail présent dans ce mémoire est organisé en trois chapitres :

Le premier chapitre est consacré à la présentation du surface de Bézier et le principe de lancer de rayon, puis les différentes méthodes d'intersection rayon-surface de Bézier et à la fin du chapitre, on va discuter sur les avantages et les inconvénients de lancer de rayon.

Dans le deuxième chapitre, nous parlons sur les différent techniques d'accélération les plus couramment utilisée en lancer de rayon, puis nous présentons les différentes méthodes ou les outils qui on utilise dans notre approche de l'optimisation de lancer de rayon pour la surface de Bézier.

Le troisième chapitre, est divisé en deux parties : la première consacré la présentation de l'architecture globale et détaillé de notre approche, la deuxième se fait sur la précision de l'environnement de développement et Les bibliothèques utilisé pour mettre en œuvre notre application et l'analyse des résultats obtenus.

Ce mémoire est terminé par une conclusion générale récapitulant notre système.

## 1. Introduction

La modélisation géométrique est une discipline mathématique qui se charge de construire des modèles géométriques à des objets existants ou à créer, par exemple pour la création d' une fenêtre dans une carrosserie de voiture il suffit à présenter des modèles géométriques plus lisser, continu et possible de manipule, le modèle qui répondent à ce exigence est la modélisation par la surface de Bézier ,elle se présente dans plusieurs aspects applicative : industrie automobile, l'industrie aérospatiale et récemment l'industrie cinématographique , programmes de dessin vectoriel, morphing ( déformation d'image)...etc.

## 2. La modélisation de la surface de Bézier

Avec la révolution industrielle, les machines sont apparues, il a fallu dessiner les pièces pour pouvoir les produire. Pour modéliser les surfaces, les dessinateurs utilisaient des méthodes manuelles à la règle et au crayon. Sans que l'on puisse décrire leurs formes par une formule mathématique.

Puis vers 1950, les machines à commandes numériques sont arrivées, alors il devenait obligatoire à modéliser mathématiquement les courbes et les surfaces complexes, La première approche était d'interpoler linéairement un grand nombre de points mais cela était long et difficile a manipule, il a nécessite à trouver des courbes capables d'être utilisées depuis la conception jusqu'à la réalisation.

Pierre Bézier ingénieur des usines Renault chercha comment traduire mathématiquement une courbe, puis une surface, dessinée à main levée. Il entendait inventer un système complet pour créer un objet en volume à partir d'un dessin, Mais ses recherches n'étaient pas entièrement originales. Dès 1958, un autre ingénieur employé par Citroën, Paul de Casteljau inventa à la même époque un algorithme de numérisation de ces courbes.[1][2]

## 2.1. Courbe de Bézier

### 2.1.1. Généralités

Les courbes de Bézier sont des courbes polynomiales paramétriques inventées initialement dans le cadre de la construction automobile en France à partir des années 60, elles constituent un des premiers essais de création d'une définition à la fois souple et intuitive d'une surface pour la C.A.O (conception assistée par ordinateur). [3]

### 2.1.2. La définition théorique

Une courbe de Bézier de degré $n$ est définie de la façon suivante ou $u$ dans [0 ,1]

$$P(u) = \sum_{i=0}^{n} P_i B_i^n(u) \qquad (1)$$

Dans laquelle les points $P_i$ sont les points de contrôle, et les $B_i^n(u)$ sont les polynômes de Bernstein tels que:

$$B_i^n(u) = C_n^i(1-u)^{n-i}u^i \quad \text{Avec} \quad C_n^i = \frac{n!}{i!(n-i)!} \qquad [4][5]$$

### 2.1.3. L'évaluation des courbes de Bézier par L'algorithme de Casteljau

L'évaluation d'une courbe de Bézier à un **t** donné donne $P(t)$ . Comme **t** varie de **0** à **1**, $P(t)$ trace le segment de courbe. Une façon d'évaluer l'équation de Bézier précédemment définie (1).

C'est probablement la pire méthode pour évaluer un point sur la courbe (L'instabilité numérique, causée par l'élévation de petites valeurs à des puissances élevées, génère des erreurs) ; Un meilleur moyen est l'algorithme de Casteljau.

L'algorithme de Casteljau est un algorithme récursif trouvé par Paul de Casteljau pour approximer efficacement les polynômes écrits dans la base de Bernstein.

Il est rapide et robuste, donne un aperçu du comportement de la courbe de Bézier et mène à des opérations importantes sur les courbes, telles que

- Calcul des dérivées (la dérivée de courbe donne le vecteur tangent en un point).
- Subdiviser la courbe. Il est parfois nécessaire de prendre une seule courbe de Bézier et de produire deux segments de courbe séparés qui ensemble sont identiques à l'original. Pour

ce faire, il est nécessaire de trouver deux ensembles de points de contrôle pour les deux nouvelles courbes. [5] [7][6]

### 2.1.3.1 L'algorithme de Casteljau

**Entrées :** $\boldsymbol{P_0, P_1, \ldots\ldots, P_n}$ .les point de contrôle

**Procédure :** $P_i^j = (1-t)P_i^{j-1}(t) + tP_{i+1}^{j-1}(t)$

Avec **1≤ j≤ n et 0 ≤ i ≤ n-j**

Noter que $P_i^0 = P_i$

**Sortie :** $\boldsymbol{P_0^n(t)}$ le point à paramètre **t** dans la courbe de Bézier $\boldsymbol{P^n}$ **.**

### 2.1.3.2 Le schéma de Casteljau

L'algorithme de Casteljau peut être visualisé comme un schéma de labellisation, Cela formalise la méthode:

- Chaque niveau de récursivité indiqué par un exposant.
- Les points de contrôle sont au niveau zéro et n'ont pas besoin d'être en exposant.
- Chaque niveau successif de récursivité a un point de moins que le niveau précédent.
- Le niveau final $\boldsymbol{P_0^n(t) = P^n}$ est le point sur la courbe (c'est $\boldsymbol{b_0^3}$ dans la figure 1.1.1). [7][5]

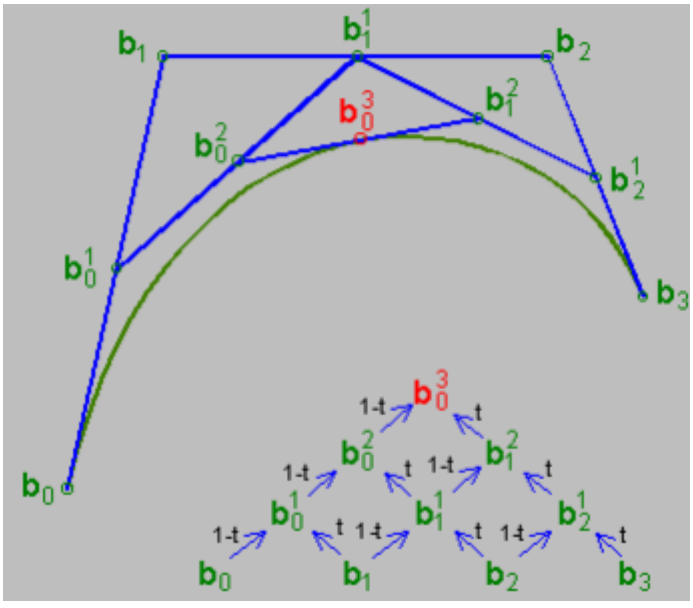



**Figure 1.2.1 : exemple pour un schéma de courbe de Bézier cubique**[7]

### 2.1.4. Propriétés des courbes

- Interpolation des extrémités $\boldsymbol{P_0}$, $\boldsymbol{P_n}$

$$\boldsymbol{P(0) = P_0 \quad et \; P(1) = P_n}$$

- Tangentes ou l'extrémité.
- $\overrightarrow{P_0P_1}$ Est le vecteur tangent à la courbe en $\boldsymbol{P_0}$ .
- $\overrightarrow{P_{n-1}P_n}$ Est le vecteur tangent a la courbe en $\boldsymbol{P_n}$.
- Symétrie, la courbe de Bézier ne varie pas en fonction de l'ordre de point de contrôle, soient dans l'ordre $\boldsymbol{P_0, P_1, \dots, P_n}$ ou dans l'ordre inverse $\boldsymbol{P_n, P_{n-1}, \dots, P_0}$ grâce à la forme suivante :

$$\sum_{i=0}^{n} P_i \, B_i^n(u) = \sum_{i=0}^{n} P_{n-i} \, B_i^n(1-u)$$

- Invariance sous les transformations affines (rotation, réflexion et translation), par exemple si en fait une rotation la courbe ne changer pas sa forme.
- La courbe de Bézier se trouve à l'intérieur de l'enveloppe convexe de ses points de contrôle. [8][5]

## 2.2. Surfaces de Bézier

### 2.2.1. Définition

Les surfaces de Bézier sont une méthode de définition d'une surface par une courbe de Bézier qui se déplace en changeant de forme des courbes de Bézier, elles représentent par une grille rectangulaire de points de contrôle $\boldsymbol{(P_{ij})}$ qui donne la possibilité de manipulations, elles défini par la forme suivant :

$$P(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(s) \cdot B_i^n(t) \cdot P_{ij}$$

Avec $\boldsymbol{s \; et \; t}$ est varier entre **0** et **1.** [4][9]

### 2.2.2. Théorie

A chaque moment, on définit une courbe appelée courbe génératrice par un ensemble de points de contrôle. Le chemin d'un point de contrôle, est aussi une courbe de Bézier qui appelée la courbe directrice.

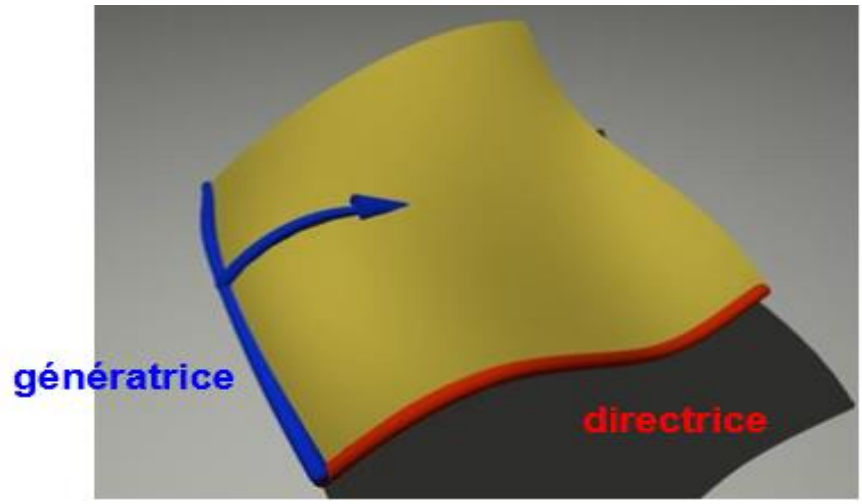



**Figure 1.2.2 : le principe de la modélisation d'une surface de Bézier**

La courbe génératrice de degré **m** peut être écrite de la manière suivante :

$$P(t) = \sum_{i=0}^{m} P_i B_i^m(t)$$

Chaque $P_i$ décrit un courbe directrice de degré **n** donc :

$$P_i(s) = \sum_{i=0}^{n} P_{ij} B_i^n(s)$$

A partir les deux équations, on fait le produit tensoriel de deux courbes, la courbe génératrice et la courbe directrice, on arrive à l'équation de la surface de Bézier : [10][11]

$$P(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(s) \cdot B_i^n(t) \cdot P_{ij}$$

### 2.2.3. Les propriétés

- Les surface de Bézier sont analogues en grande partie les propriétés des courbes du même nom, parmi elles citons : [12][11]
- Une surface de Bézier se trouve toujours dans l'enveloppe convexe définie par ses points de contrôle.
- La surface interpole les coins du quadrillage : $\boldsymbol{P_{00}}, \boldsymbol{P_{0m}}, \boldsymbol{P_{n0}}, \boldsymbol{P_{nm}}$
- Les surfaces de Bézier sont dérivables.
- Il est possible d'augmenter le nombre de points de contrôle sans modifier la surface.

### 2.2.4. Les avantages

- Possibilité de manipulations.
- Simplicité de mise en œuvre.
- Nous pouvons augmenter la dimension de la grille de points de contrôle sans modifier la surface.
- En augmentant la multiplicité de points de contrôle, cela permet de rapprocher la surface de Bézier à ces points.
- Elle est souple et être lisser et continu. [10][13]

### 2.2.5. Les inconvénients

- Le degré d'un courbe est lié au nombre de points de contrôle.
- La complexité augmente avec le nombre de points.
- Contrôle global, Le déplacement d'un point de contrôle entraîne une déformation de toute la surface.
- Limitation obligatoire du degré (9 à 10 en pratique). [10][13]

## 3. Présentation de Lancer de rayon

Il est admis que l'algorithme de lancer de rayons est parmi ceux qui produisent les images de synthèse les plus réalistes, à l'heure actuelle. En effet, cette technique permet de rendre les effets de réflexions multiples et de réfraction.

Avant d'aller plus avant dans la présentation de lancer de rayons, nous commençons par étudier la rastérisation et ses Inconvénients. Par la suite, Nous détaillerons dans la présentation de l'algorithme du lancer de rayons avec les avantages et les inconvénients.

### 3.1. Rastérisation

La première méthode de simulation d'éclairage (ou méthode de rendu) est une méthode espace objet apparue au début des années **70** ; elle est appelée aussi la méthode projective.

Pour générer une image par cette méthode, il appliqué Les différents traitements suivant :

- Transformation de tous les polygones du repère absolu (défini lors de l'tape de modélisation) vers un repère lié à l'observateur.
- L'élimination de tous les polygones ne se trouvant pas dans le champ visuel.
- Détermination des pixels contenus dans la projection d'un polygone et calcul de leurs composantes RVB à l'aide d'un modèle d'illumination local et des informations de texture.
- Calcul des ombres portées.

Les inconvénients de cette méthode est que elle utilise un modèle d'illumination local, ce qui réduit considérablement le réalisme puisque l'éclairage indirecte dans un environnement représente la plus grosse partie de I' éclairage plus de ca ne représente pas la réfractions et la transparence, aussi les objets sont modélisés avec des polygones plans. Donc Nous avons besoin d'une autre approche. [26][14]

## 3.2. Lancer de rayon

### 3.2.1. Définition

Le lancer de rayon c'est une méthode de rendu réaliste (en anglais ray tracing), proposé en 1968 par Appel et fût repris et amélioré par Whitted en 1980. Cela a été la première méthode permettant d'obtenir des images de synthèse photo-réalistes ; elle utilise les lois de l'optique géométrique, l'idée principale consiste à suivre le trajet inverse des rayons lumineux afin de calculer les propriétés géométriques et lumineuses de la scène. [15][16]

### 3.2.2. Le principe

Le lancer de rayon c'est un algorithme récursif, il calcule l'image pixel par pixel ; pour déterminer l'objet vu en un pixel, on lancé un rayon issu de l'œil vers se pixel dans la scène. Les points d'intersection du rayon œil-pixel avec tous les objets de la scène sont calculés, et le plus proche de l'œil est retenu **P** ; ce rayon on appelle un rayon primaire.

Supposons l'existence d'une source lumineuse dans la scène, la couleur de l'objet **O** vu au point **P** dépend de la couleur propre de **O** en ce point, mais aussi si la source éclaire directement l'objet ou non .Pour le savoir, l'algorithme lance un rayon secondaire dans la scène , ce rayon est issu du point **P** et dirigé vers la source lumineuse ; son intersection avec tous les objets de la scène est testée; s'il rencontre un objet opaque situé entre **P** et la source, alors **P** n'est pas éclairé par la source, mais seulement par la lumière ambiante, sinon on calculé la contribution à l'intensité du rayon a base des propriétés de réflexion diffuse et spéculaire et l'intensité de la source. [14]

L'algorithme peut ainsi prendre en compte plusieurs sources lumineuses, il suffit de lancer autant de rayons secondaires qu'il y a de sources lumineuses dans la scène, et d'ajouter la contribution des éclairements au point **P**.

Mais d'autres phénomènes peuvent influer sur la couleur du pixel ; par exemple, si l'objet **O** vu en **P** a une surface réfléchissante ou transparent alors dans le cas de surface réfléchissante la couleur du pixel dépend de la couleur de l'objet réfléchi en **P**. Pour déterminer quel est l'objet qui se reflète en **P**, un rayon secondaire "réfléchi" est lancé dans la scène ; ce rayon est issu de **P** et sa direction est symétrique de celle du rayon incident en **P** (ici le rayon primaire) par rapport à la normale de l'objet en **P**.

Dans le cas de surface transparent la couleur du pixel dépend de la couleur de l'objet que l'on voit à travers **O** au point **P** .Pour le savoir, l'algorithme lance un rayon secondaire réfracté à partir de **P**; sa direction dépend de la direction du rayon d'incidence (ici le rayon primaire) et de l'indice de réfraction de **O** (voir la figure suivant). [15] [26][16]

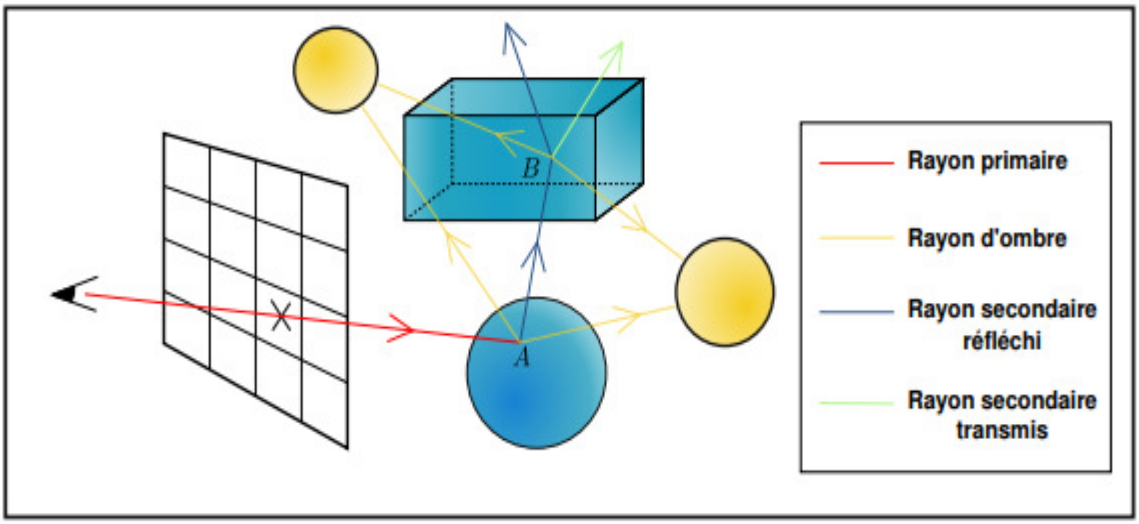



**Figure 1.3.1 : le Principe du lancer de rayon**

Il applique récursivement le même processus pour déterminer les intersections de ces rayons avec d'autres surfaces. Ainsi, pour chaque pixel, on doit être construit un arbre d'intersection (Figure 1.2.2).

Le processus récursif s'arrête dans les cas suivants:

- Quand un rayon quitte la scène (la couleur est celle du fond).
- Quand un rayon intersecté une surface ni spéculaire, ni transparente.
- Quand la contribution du rayon devient négligeable.

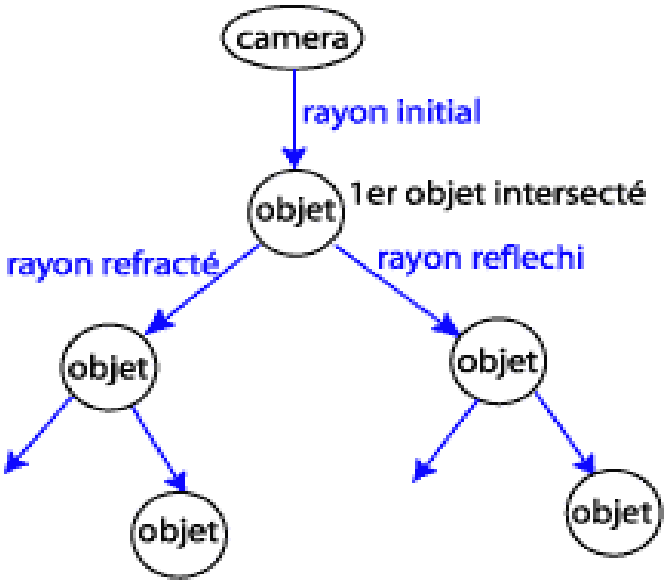



**Figure 1.3.2 : l'arbre d'intersection**

Lorsque le processus récursif est terminé et les arbres d'intersection sont construit, il est parcouru en appliquant une équation à chaque nœud pour calculer l'intensité.

Généralement, l'équation d'illumination est donnée par : [17]

$$\boldsymbol{I = I_a + I_d + I_s + I_t}$$

Où:

- $\boldsymbol{I_a}$ est l'intensité de la source lumineuse ambiante.
- $\boldsymbol{I_d}$ est l'intensité du rayon diffuse quisont calculées comme dans la formule de Phong.
- $\boldsymbol{I_s}$ la lumière spéculaire qui est obtenue par l'équation $\boldsymbol{I_s = K_s \cdot S}$ . Où $\boldsymbol{S}$ est la lumière spéculaire incidente de direction $\boldsymbol{R}$ et $\boldsymbol{K_s}$ l'indice de sécularité (constante).
- On a en plus $\mathbf{I_t}$ une lumière transmise donnée par l'équation $\boldsymbol{I_t = K_t \cdot T}$ Où T est l'intensité du rayon transmis (de direction $\boldsymbol{P}$) et $\boldsymbol{K_t}$ l'indice de transmission (constante).

Pseudo-code d'algorithme de lancer de rayon suivant :

```
scène s
rayon r
tableau_de_couleur teinte
s <- scène à afficher
pour chaque pixel p(x,y) de l'écran
r <- rayon passant par l'observateur et p
teinte[x,y] <- lancerRayon(s,r)
finpour
fin
couleur lancerRayon(scène s,rayon r)
couleur cr,ct,cd
rayon rr,rt,rd
coefficient kr,kt
point p
objet o
o <- objet de la scène s dont l'intersection
avec r est la plus proche de la source de r
    si o n'existe pas  //Aucun objet n'intercepte le rayon.
retourner (noir)
    sinon      //Un ou plusieurs objets interceptent le rayon.
p <- point d'intersection entre r et o
rr<- rayon réfléchi de r en p sur o
rt<- rayon transmis de r en p vis-à-vis de o
kr<- coefficients de réflexion de o
kt<- coefficients de transmission de o
cr<- lancerRayon(s,rr) * kr
ct <- lancerRayon(s,rt) * kt
cd <- noir
       pour chaque lumière sl de s
         rd <- rayon de p vers sl
        si rd non intercepté par un objet
          cd <- cd + diffusion(sl,o,rd)
         finsi
       finpour
retourner(cr  + cd + ct)
```

**Figure 1.3.3 : Pseudo code de l'algorithme lancer de rayon**[15]

## 3.3. Calcul l'intersection

Pour chaque pixel, le tracé de rayons effectue en gros deux types de calculs : des calculs géométriques d'intersection entre un rayon et des objets d'une part, d'autre part des calculs optiques, les calculs les plus important dans tout algorithme de lancer de rayon sont

les calculs géométriques d'intersection rayon-objet, d'autant plus que leur volume croît avec la complexité de la scène.

Dans cette partie, on note $\boldsymbol{p}$ l'origine du rayon lancé, et $\boldsymbol{d}$ sa direction. On dira qu'un point $\mathbf{m}\epsilon\square^{3}$ appartient au rayon si et seulement si $\exists \boldsymbol{t} \geq \mathbf{0}, \boldsymbol{m} = \boldsymbol{p} + \boldsymbol{td}$ .

On se donne maintenant une primitive quelconque, et on cherche l'intersection entre le rayon et cette primitive ; c'est à dire le plus petit $\boldsymbol{t}$ (s'il existe) tel que $\boldsymbol{p} + \boldsymbol{td} \in \boldsymbol{\Omega}$ Pour plus de facilités, on se placera dans le repère local de la primitive.

### 3.3.1. Cas d'un plan simple

Soit un rayon (segment de droite) défini par le point d'origine $\boldsymbol{p(x1, y1, z1)}$ et le vecteur de la direction $\boldsymbol{d(i, j, k)}$ dans l'équation paramétrique suivant :

$$\begin{cases} x = x1 + it \\ y = y1 + jt \\ z = z1 + kt \end{cases}$$

D'autre part, l'équation implicite d'un plan est : $Ax + By + Cz + D = 0$

Pour déterminer le paramètre $\boldsymbol{t}$ de l'intersection (si elle existe) d'un rayon et d'un plan, il suffit de substituer donc les coordonnées (paramétrées) de la droite dans l'équation (implicite) du plan : [18]

$$\mathrm{t} = \frac{\mathrm{A.x1} + \mathrm{B.y1} + \mathrm{C.z1} + \mathrm{D}}{\mathrm{Ai} + \mathrm{Bj} + \mathrm{Ck}}$$

### 3.3.2. Cas d'une sphère

On rappelle que le segment de droite qui représente le rayon lancé est défini par $\boldsymbol{m} = \boldsymbol{p} + \boldsymbol{td}$ D'autre part, l'équation d'une sphère de centre ***(l, m, n)*** et de rayon $\boldsymbol{r}$ est :

$$(x - 1)^2 + (y - m)^2 + (z - n)^2 = r^2$$

En substituant $\boldsymbol{x, y}$ *et* $\boldsymbol{z}$ dans l'équation implicite de la sphère on obtient une équation quadratique en $\boldsymbol{t}$ de la forme : $at^2 + bt + c = 0$

$\Delta<0$ : le segment de droite n'intersecté pas la sphère.

$\Delta=0$ : le segment est tangent à la sphère.

$\Delta > 0$ : les racines réelles donnent les points d'intersection avant et arrière. [14][18]

### 3.3.3. Cas d'une surface de Bézier

Dans cette partie, nous examinerons différentes méthodes existantes pour le lancer de rayons sur les surfaces de Bézier, c'est-à-dire les méthodes pour trouver l'intersection entre un rayon et une surface de Bézier. En plus d'une description, les avantages et les inconvénients de chaque méthode sont discutés.

Avant d'aborder les différentes méthodes, nous allons dire quelques mots sur une technique qui peut être utilisée avec la plupart des méthodes. Si nous, avant d'appliquer la méthode du lancer de rayons, projetons tous les points de contrôle de la surface à deux dimensions, nous pouvons réduire le montant de travail des opérations spécifiques à la surface. Cela ne réduit pas toujours le coût total de recherche de l'intersection, car nous perdons certaines informations de profondeur.

Un rayon peut être écrit sous forme paramétrique comme :

$$r = o + td = (o_x + td_x, o_y + td_y, o_z + td_z)$$

Où **o** est l'origine du rayon, **d** est la direction (normalisée) du rayon et **t** est un scalaire variable qui détermine les positions le long du rayon (ou la longueur du rayon) [21]

#### 3.3.3.1. Subdivision

La subdivision est une méthode simple pour trouver une intersection avec les patches de Bézier. Il s'agit la première méthode utilisée pour les patchs de surface paramétriques de lancer de rayons.

L'idée est simple: diviser la surface en patchs de plus en plus petits, jetant ces patchs qui ne peuvent pas être intersectés par le rayon jusqu'à ce qu'une certaine profondeur maximale soit atteinte. Ensuite, nous retrouvons avec un petit ensemble (espérons-le) de patchs candidats à l'intersection rayon-surface la plus proche.

La division de la surface peut être effectuée efficacement en utilisant l'algorithme classique de Casteljau.

La suppression des patchs est simple: si tous les points de contrôle ont le même signe sur l'une des coordonnées (dans l'espace bidimensionnel défini précédemment), le patch ne peut pas être intersecté par le rayon. (Voir la Figure 1.2.4**)**

Enfin, le maillage du point de contrôle de chaque sous-lot candidat est utilisé comme une approximation de la surface et l'intersection est trouvée en effectuant une simple intersection rayon-triangle sur les triangles générés par le maillage.

Bien que l'idée de base soit simple et facile à mettre en œuvre, la méthode présente certains inconvénients. Tout d'abord, un objet composé de plusieurs surfaces doit utiliser la même profondeur de subdivision pour toutes ses surfaces afin d'éviter les fissures. Deuxièmement, étant donné que nous approchons la surface avec des maillages de points de contrôle, les points d'intersection ne peuvent être trouvés qu'avec un faible degré de précision. Nous pouvons améliorer la précision en augmentant la profondeur de subdivision, mais cela ralentirait la méthode. Enfin, comme nous devons mettre en cache au moins une surface à chaque profondeur de subdivision, la demande de mémoire temporaire est élevée. [19][21]

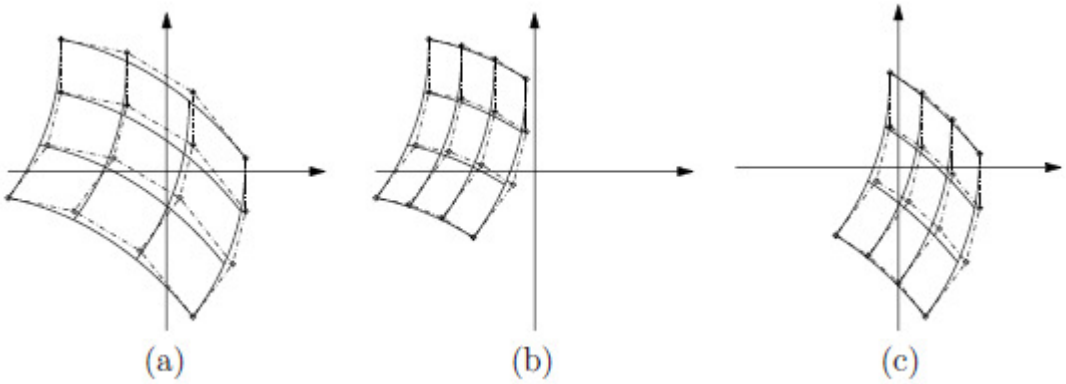



**Figure 1.3.4 : Méthode de subdivision. La surface (a) est divisée en (b) et (c). (b) peut être rejeté puisque tous ses points de contrôle se trouvent d'un côté d'un axe de coordonnées.**

### 3.3.3.2. Bézier clipping

La méthode de Bézier clipping a été introduite à l'origine par Nishita et al (1990). Tout d'abord, nous représentons le rayon comme l'intersection de deux plans perpendiculaires Et que cela surface et ses points de contrôle sont tous projetés sur un troisième plan perpendiculaire à la direction du rayon.

Dans cette projection, les deux plans devient les axe (x, y) et le rayon se projeté vers l'origine du système de coordonnées.

Dans cet exemple, nous allons effectuer un découpage dans la direction **u** paramétrique. Tout d'abord, nous définissons une ligne **L**:

$$ax + by = 0, a^2 + b^2 = 1$$

Par l'origine, parallèle au vecteur **v0** + **v1** (voir la Figure a). On définit ensuite la distance signée de chaque point de contrôle (projeté) $\boldsymbol{p}_{i,j} = (\boldsymbol{x}_{i,j}, \boldsymbol{y}_{i,j})$ à **L** comme

$$d_{i,j} = ax_{i,j} + by_{i,j}$$ (Voir la Figure 1.2.5 b).

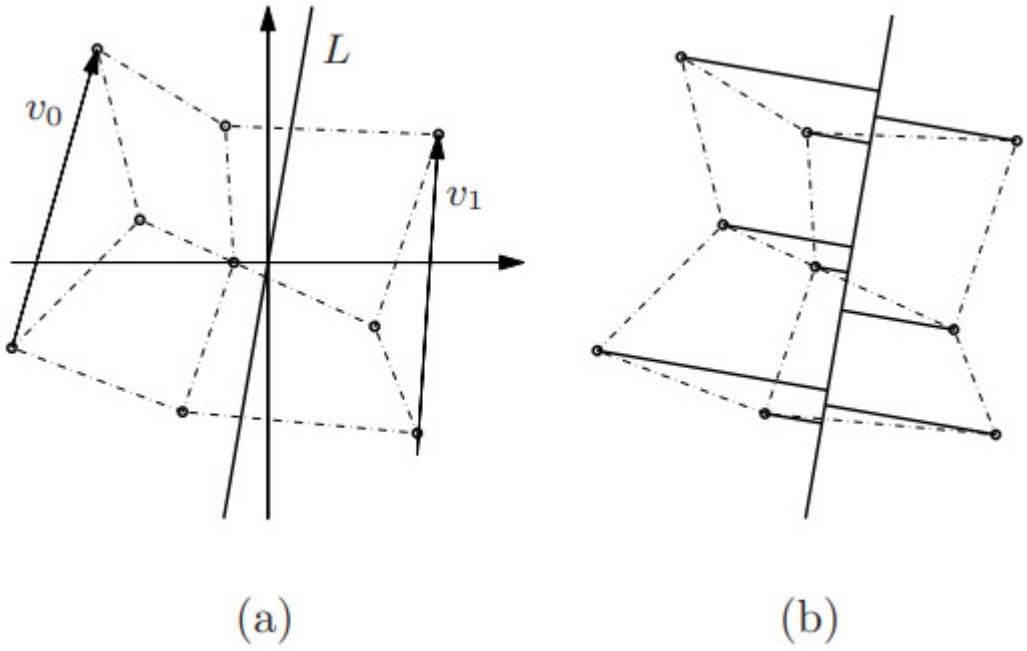



**Figure 1.3.5** : **Méthode de Bézier clipping. Déterminez une ligne L (a) et calculez les distances à L à partir de chaque point de contrôle (b).**

On peut définir un patch explicite dans un espace de coordonnées **2D (u, d),** (vu de côté sur la figure 1.2.6 .En analysant où cette coque convexe coupe l'axe zéro, nous trouvons l'intervalle paramétrique dans lequel le rayon peut intersecté la surface d'origine. Le rayon ne peut couper la surface que dans l'intervalle $\mathbf{[u_{min}, u_{max}]}$.

Enfin, lorsque nous avons trouvé $\mathbf{u_{min}}$ et $\mathbf{u_{max}}$, nous calculons de nouveaux points de contrôle pour le sous-patch correspondant au sous-intervalle du domaine paramétrique d'origine. [19][21][22]

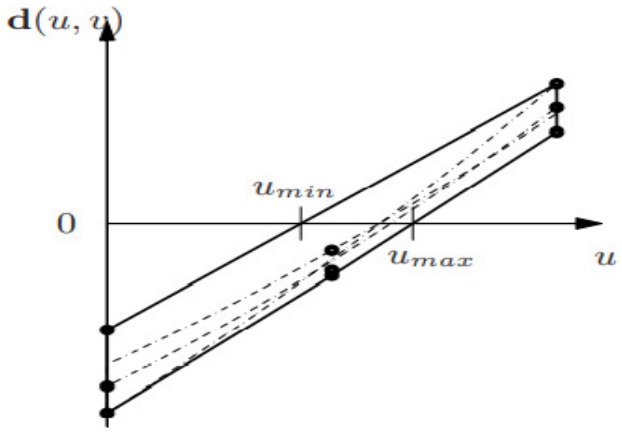



**Figure 1.3.6** : **Coque convexe de surface «à distance».**

Cette méthode d'intersection est stable dans le sens où il est garanti de trouver une intersection si elle existe, mais elle est relativement lente pour les surfaces de degré supérieur.

### 3.3.3.3. Méthode de Newton

Trouver un zéro d'un système d'équations non linéaires est un problème standard en mathématiques computationnelles, et un problème qui apparaît dans de nombreuses applications. L'un des rares outils disponibles pour cette tâche est la méthode de Newton. [23]

Une intersection entre une surface $\boldsymbol{s(u,v)}$ et un rayon $\boldsymbol{r(t)}$ se produit quand et seulement quand

$$f(u,v,t) = s(u,v) - r(t) \qquad (2)$$

Ce système non linéaire peut être résolu en utilisant une itération Newton multi variée en tridimensionnel.

Soit $\boldsymbol{x = (u,v,t)}$ Le système non linéaire (2) peut maintenant être écrit sous forme vectorielle comme $f(x) = 0$

Si $\boldsymbol{Y}$st une matrice non singulière **3 * 3**, alors le schéma de Newton est donné par

$$x_{k+1} = x_k - Yf(x_k)$$

On se réfère au vecteur $\boldsymbol{-Yf(x_k)}$ comme le pas de Newton, et si **Y** est l'inverse Jacobien de $\boldsymbol{f}$en $\boldsymbol{x_k}$, alors géométriquement il représente les coordonnées de l'intersection du rayon et du plan tangent en $\boldsymbol{x_k}$ .

La méthode de Newton peut être utilisée pour trouver le zéro d'un système de manière rapide, mais pas inconditionnellement stable. Pour commencer l'itération, nous avons besoin d'une estimation initiale $\boldsymbol{x_0}$ , et pour que la méthode converge vers une solution, il est important d'avoir une estimation initiale suffisamment proche de la solution. À des fins de lancer de rayons, cela est souvent résolu en utilisant l'hiérarchie des volumes englobant précalculés, qui englobent complètement la surface (Figure 1.2.7 (b)). Chaque volume de la hiérarchie est associé à une ou plusieurs estimations initiales pour la partie de la surface contenue dans ce volume particulier. Si le rayon intersecté l'un des volumes, la probabilité qu'il croise également la surface est élevée et nous pouvons commencer une itération de Newton en utilisant l'estimation initiale du volume actuel. [19][20][21]

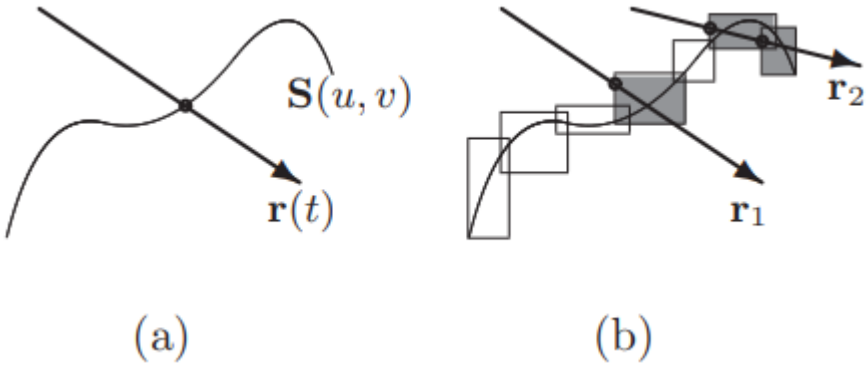



**Figure 1.3.7 : (a) intersection entre un rayon et une surface. (b) en utilisant des volumes englobant des parties de la surface, des coordonnées u, v initiales peuvent être trouvées pour l'itération de Newton.**

## 3.4. Avantage

- le tracé de rayons est une méthode extrêmement simple. Les seuls calculs géométriques nécessaires sont l'intersection entre les rayons et l'objet.
- L'algorithme le plus flexible tel que vous pouvez implémenter n'importe quel effet de rendu (les réflexions et les transparences et l ombrage) précisément dans un lanceur de rayons.
- Les parties cachées sont éliminées de manière intrinsèque. [14][15]

## 3.5. Inconvénient

- Ne tient pas compte des inter-réflexions diffuses (Eclairage diffus mal représenté).
- Le temps requis pour produire une image est fortement affecté par le nombre de sources lumineuses.
- Pas d'éclairage indirect tel que les sources sont visées explicitement.
- Aliassage et disparition des petits objets et des petites ombres. [14][15]

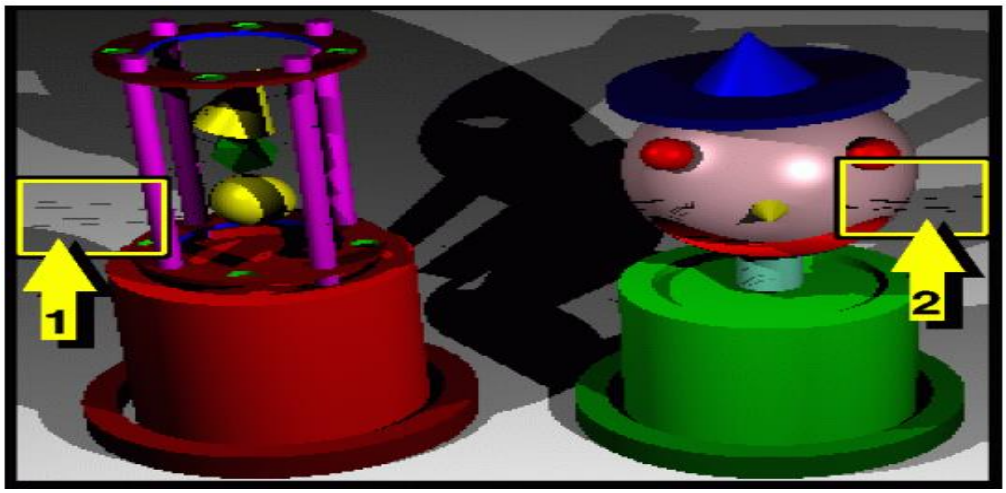



**Figure 1.3.8 : Problème de disparition des petites ombres (1) et des petits objets (2)**

## 4. Conclusion

Malgré que l'algorithme du lancer de rayons est extrêmement puissant et permet de traiter pratiquement tous les aspects du réalisme ; mais les calculs d'intersections sont très coûteux en temps de calcul donc il faut trouver des méthodes d'optimisation.

## 1. Introduction

Depuis les premiers travaux, de nombreux chercheurs se sont penchés sur le problème de l'efficacité du tracé de rayons. Cette méthode de visualisation est très séduisante du fait de sa simplicité à mettre en œuvre presque tous les phénomènes optiques classiques : ombres, réflexions et la réfractions ; mais une telle simplicité a un lourd tribut à payer : le temps de calcul nécessaire. Par exemple pour une scène constituée d'un million de rayons et de cent mille triangles, cent milliards de tests d'intersections seraient à réaliser, ce qui est aujourd'hui impossible en des temps compatibles avec une utilisation interactive.

La première partie de ce chapitre est consacrée à un survol concernant l'état de l'art des techniques d'accélération les plus couramment utilisée en lancer de rayon en donnant les grandes idées qui y sont rattachées. Dans la deuxième partie, nous présentons les différentes méthodes ou les outils qui on utilise dans notre approche de l'optimisation de lancer de rayon dans la surface de Bézier.

## 2. Classification des techniques d'accélération

Face à la tâche d'accélérer le processus de lancer de rayons, il existe trois stratégies bien distinctes à considérer : (1) réduire le coût moyen de l'intersection d'un rayon avec les objets de la scène, (2) diminuer le nombre total de rayons lancés dans la scène, et (3) remplacer les rayons individuels par une entité plus générale. Celles-ci apparaissent sur la figure 2.1.1 comme des « intersections plus rapides », « moins de rayons » et « généralisation de la notion de rayon» respectivement.

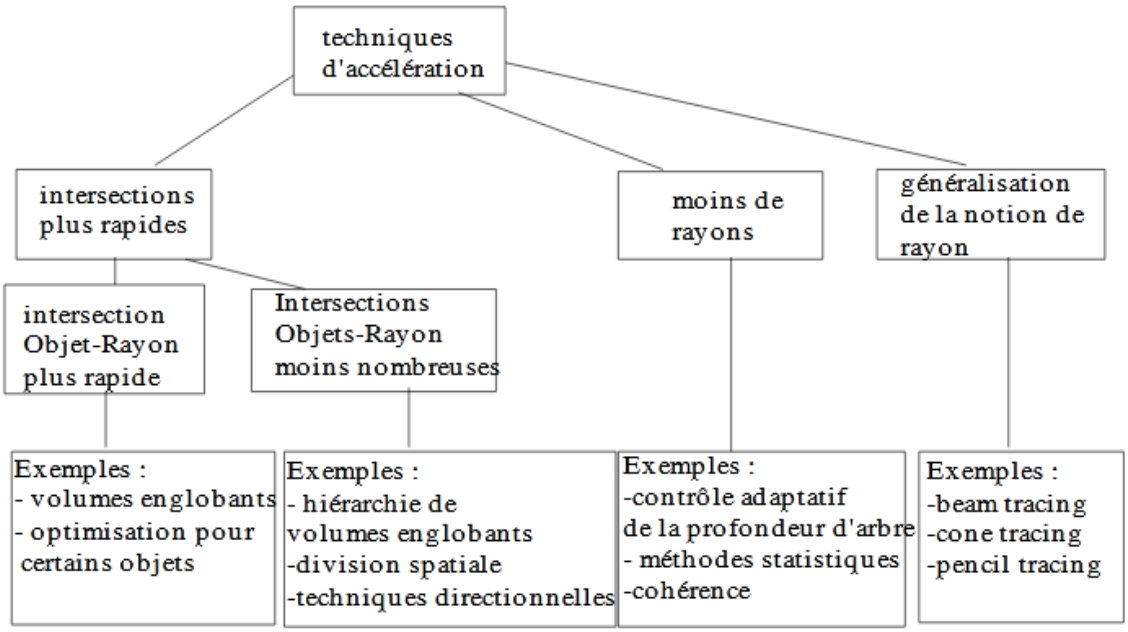



**Figure 2.2.1:classification des optimisations de lancer de rayon**

La catégorie des « intersections plus rapides » se divise en outre sous-catégories d'intersections rayon - objets « plus rapides » et « moins nombreuses ». Le premier consiste en des algorithmes efficaces pour l'intersection de rayons avec des objets primitifs spécifiques, tandis que le second aborde le problème plus large de l'intersection d'un rayon avec des objets d'une scène en utilisant un minimum de tests d'intersection rayon-objet.

La catégorie « moins de rayons » est constituée de techniques qui nous permettent de réduire le nombre de rayons qui doivent être intersectés avec les objets de la scène. Cela inclut les rayons de première génération ainsi que ceux créés par la réflexion, la réfraction et l'ombrage.

La dernière catégorie, appelée « généralisation de la notion de rayon », consiste en un certain nombre de techniques qui commencent par remplacer le concept familier de rayon par une entité plus générale qui subsume les rayons comme un cas spécial.[24][25]

## 2.1. Intersections plus rapides

### 2.1.1. Les volumes englobant

Dès ses premiers travaux sur le lancer de rayon, Whitted constata que plus de 75% du temps de calcul était consacré aux intersections entre les rayons et les objets ; Il eut l'idée d'ajouter des volumes dont l'intersection est très simple pour englober un objet dont l'intersection est-elle compliquée, Afin d'éviter des calculs inutiles tels que l'intersection avec des objets ne se trouvant pas sur le trajet d'un rayon. Cette technique est très efficace si Le volume englobant approximer son objet au mieux (figure 2.2.2).

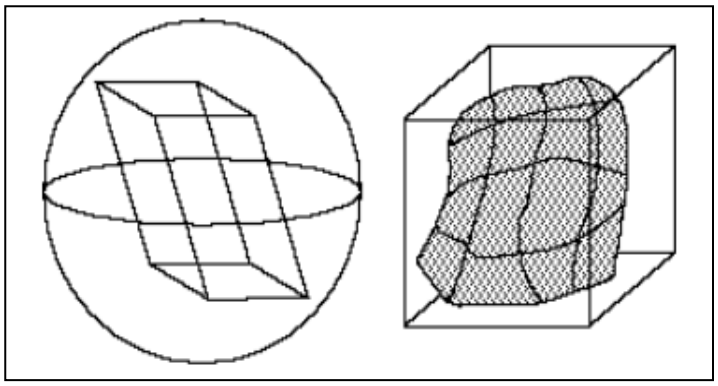


**Figure 2.2.2: efficacités des volumes englobant**

Plus tard, on a développé l'idée d'introduire des volumes englobant hiérarchisés (BVH) sous forme arborescente qui permettent théoriquement d'obtenir une complexité logarithmique en fonction du nombre d'objets de la scène. [27]

L'idée est qu'On utilise en général un arbre binaire dont les nœuds sont des volumes englobant de sous-scènes ; Cette structure pourrait être construite en subdivisant la scène en deux sous-scènes, calculer le volume englobant de chaque sous-scène et créer les nœuds associés. Ce processus est appliqué récursivement à chaque sous-scène jusqu'à ce que les sous-scènes contiennent un nombre d'objets inférieur à un certain seuil.

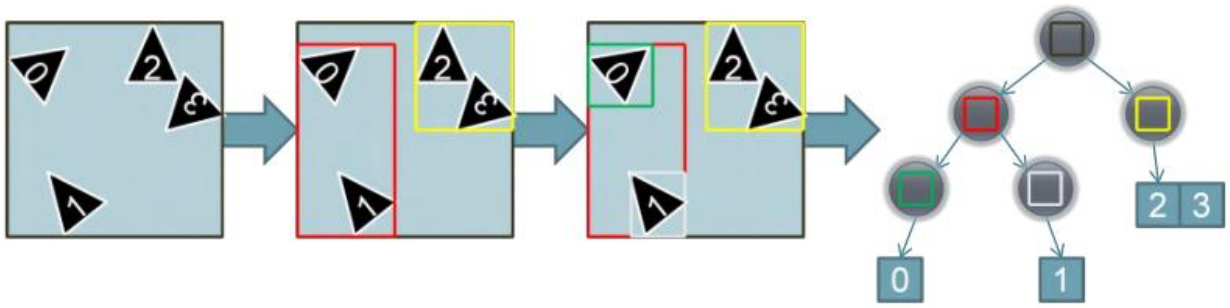



**Figure 2.2.3** : **Vue simplifiée du processus de construction d'une BVH**

Notons que le volume englobant correspondant à la racine de l'arbre binaire englobe toute la scène et que les feuilles englobent un faible nombre d'objets.

Pour évaluer l'intersection entre un rayon et la scène, on parcourt l'arbre de la racine vers les feuilles ; En chaque nœud atteint on teste s'il y a intersection entre le volume englobant associé et le rayon. Si oui on réitère ce processus sur ces deux fils jusqu'à ce qu'on atteigne les feuilles ; sinon on rejeter complètement le sous arbre dont la racine. Dans ce dernier cas toutes les feuilles du sous arbre ne sont pas prises en compte par le calcul d'intersection, ce qui réduit considérablement les temps de calcul. [24][18][15]

## 2.2. Moins de rayons

### 2.2.1. Contrôle adaptatif de la profondeur

Un algorithme de ray-tracing n'arrête de lancer récursivement des rayons dans 3 cas : si le rayon lancé ne touche rien, ou l'objet qui touché par le rayon est mat (non réfléchissante) ou opaque, Ou bien lorsqu'est atteinte une profondeur déterminée dans l'arbre des rayons.

En général, dans une scène, seule une petite partie des objets ont des caractéristiques de transparence ou de spécularité et des lieux géométriques qui nécessitent de relancer les rayons. Ainsi, il existe un potentiel considérable d'économies de calcul si les calculs ne sont effectués que lorsque cela est nécessaire.

Prenons par exemple, l'image d'une pièce en miroir de la figure 2.2.4 **(a)** bien qu'il semble avoir de vastes zones réfléchissantes, seul un petit pourcentage de l'image est en fait constitué de surfaces réfléchissantes (En comparaison avec la figure 2.2.4 **(b)** qui montre la même image générée sans réflexion);Cependant, les réflexions spéculaires sont très importantes pour générer la représentation correcte de la scène car des études ont montré

qu'une profondeur d'arbre de réflexion de 15 devrait être attribuée pour récupérer suffisamment d'informations sur l'image.

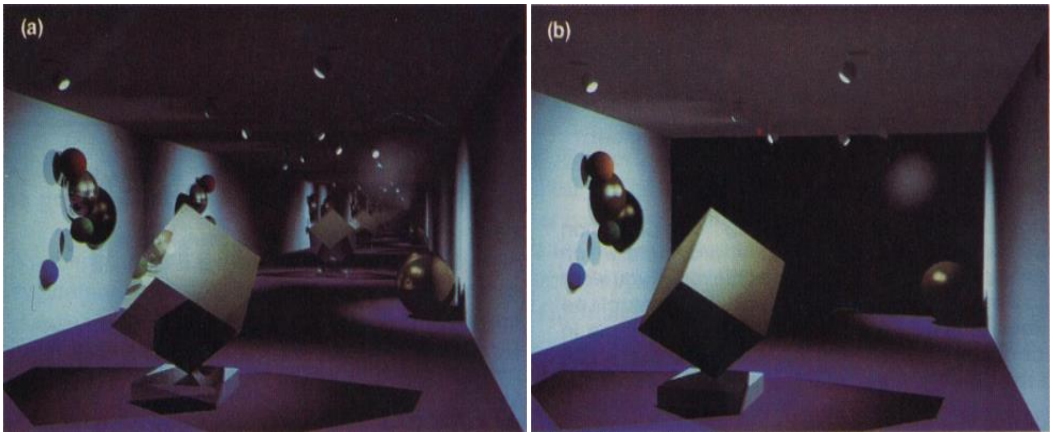



**Figure 2.2.4: Images de la galerie avec des murs en miroir et des sources lumineuses locales**

La technique d'adaptation de la profondeur de récursion de lancer de rayon permettrait d'élaguer l'arbre des récursion et de ne le limiter qu'aux rayons lumineux qui produisent des résultats visibles.

Cette technique peut être mise en place assez simplement en utilisant les atténuations le long du chemin du rayon. Lorsqu'un rayon spéculaire/réfracté est relancé, l'intensité du rayon renvoyé est multiplié par un terme de transmissivité (transmis lors de l'appel récursif) plus petit que **1** (atténuation). Ce processus peut être répété pour les enfants de n'importe quel nœud parent de l'arborescence.si le produit des atténuations est inférieur à un seuil de fixé de coupure on termine le traçage de rayons supplémentaire. [18]

**Remarque** : il faut de plus gérer deux facteurs d'atténuation différents (autant que de rayons relancés) : un pour la réflexion spéculaire et un pour la réfraction.

## 2.3. Généralisation de la notion de rayon

Une autre approche a été envisagée, visant à améliorer la technique de lancer de rayon ; l'idée retenue par cette nouvelle méthode est de définir un concept plus général dont le rayon serait un cas particulier. Sont alors apparus les cônes, faisceaux et pinceaux qui

regroupent un ensemble de rayons et permettent de tracer simultanément plusieurs rayons et d'exploiter ainsi la cohérence de la scène et réduire les problèmes d'aliassage.

Faisons un tour d'horizon un structure d'accélération le plus couramment utilisées dans l'idée de rayons généralisés. [33]

### 2.3.1. Lancer de faisceaux

Cette méthode a été introduite par Heckbert et Hanrahan en 1984. Pour eux, un faisceau est un cône à section polygonale quelconque qui regroupe l'ensemble des rayons passant à l'intérieur du volume ainsi défini. Cependant, avec cette méthode, les objets doivent être entièrement définis par des facettes polygonales et planaires pour conserver la nature polygonale de la section du faisceau.

De façon analogue à l'algorithme du lancer de rayon ; une arborescence de faisceaux réfléchis et transmis est construite mais celle-ci est plus difficile à élaborer que dans le cas des rayons. Les effets de réflexion ou de transmission ne respectant pas nécessairement cette nature polygonale, on réalise une approximation de ceux-ci. [33] [24][15]

## 3. L'optimisation pour la surface de Bézier

Parmi les différent techniques d'accélération de lancer de rayon, on choisi d'utilisation des volumes englobant et le BVH (bounding volumes hiérarchique) avec les surface de Bézier parce que elle est le plus couramment utilisé et les plus faciles à mettre en œuvre.

Dans la suite de cette partie, nous présentons les différents outils utilisés dans le processus d'optimisation de l'intersection ray_patche de Bézier.

## 3.1. La Subdivision de la surface de Bézier

### 3.1.1. Dans le cas de courbes de Bézier

Une courbe de Bézier b est généralement définie sur l'intervalle (le domaine) **[0,1]** et donnée par $\mathbf{b(t)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{P_i}\, \mathbf{B_i^n(t)}$ , mais elle peut également être définie sur n'importe quel intervalle **[0, c].** La partie de la courbe qui correspond à **[0, c]** peut également défini par un enveloppe convexe (polygone de Bézier), comme illustré à la figure 2.3.1 ; La recherche de ce polygone de Bézier est appelée subdivision de la courbe de Bézier ; L'algorithme de Casteljau donne une belle façon de faire l'opération de la subdivise.

Supposons par exemple qu'une courbe de Bézier est coupée à hauteur de **t = c** pour donner deux segments de courbe notés $\mathbf{b_g}$ et $\mathbf{b_d}$ définis respectivement sur **[0, c]** et **[c, 1]**.

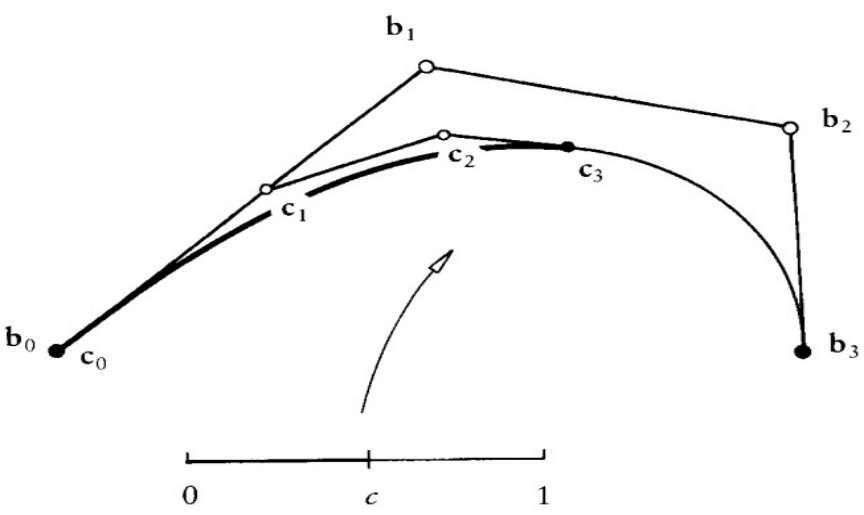



**Figure 2.3.1 : deux polygones de Bézier décrivant la même courbe: l'un (le $b_i$) est associé au paramètre d'intervalle [0,1], l'autre (le $c_i$) à [0, c]**

On doit pour cela trouver les points de contrôle pour $\mathbf{b_g}$ et $\mathbf{b_d}$ ,ce qui résulte de algorithme suivant :

**initialisation**: affecter le tableau des points de contrôle dans les $\mathbf{b_i}$ **($b_0$, $b_1$, …$b_n$).**

**Calcul les point de contrôle des deux segments** : on utilise l algorithme de Casteljau pour calculer les points de contrôle des deux segments de courbe de Bézier .

Les points de contrôle obtenus par la subdivision de $b(t)$ à **t= c** avec **c ϵ [0, 1]** sont :

$$b_g : b_0^0 , b_0^1 , \dots \dots b_0^{n-1} , b_0^n$$

$$b_g : b_0^n , b_1^{n-1} , \dots \dots b_{n-1}^1 , b_n^0$$

**Mémorisation**: on mémorise le point $\mathbf{b_0^n}$ qui est sur la courbe.

**Appel récursif**: on appelle l'algorithme sur les deux segments de courbe de Bézier définies par les points $\mathbf{b_0^i}$ , **i ϵ [0, n]** d'une part, $\mathbf{b_i^{n-i}}$ , **i ϵ [0, n]** d'autre part. [5] [20]

Voici un exemple d'implémentation de l'algorithme en pseudo-code

```
Entrée : tableau b[0][0…N] des coordonnées des points de contrôle.
// Calcul des points intermédiaires ( l algorithme de Casteljau )
Pour i de 1 à N faire
  Pour j de 0 à N-i faire
      b[i][j] = c*b[i+1][j-1] + (1-c)*b[i][j-1]
 Afficher b[N][0] // Afficher (ou stocker) le point milieu
// Construction des courbes de segment bg et bd
 Pour i de 0 à N faire
                    bg [i] = b[i][0]
   bd[i] = b[N-i][i]
    // Appel récursif
 De_Casteljau (bg)
 De_Casteljau (bd)
```

**Figure 2.3.2. : Pseudo code de Subdivision de courbe de Bézier**

### 3.1.2. Dans le cas de surface de Bézier

Une surface de Bézier est définie par une double somme de points :

$$\boldsymbol{P(s,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_j^m(s)\cdot B_i^n(t)\cdot P_{ij}}$$

Le principe de l'algorithme est de réécrire la formule sous la forme

$$\boldsymbol{P(s,t) = \sum_{i=0}^{n}\left[\sum_{j=0}^{m} B_j^m(s)\cdot P_{ij}\right]\ B_i^n(t)}$$

Nous remarquons que la partie de l'équation à l'intérieur des crochets est la représentation d'une courbe de Bézier. Si nous laissons $\boldsymbol{Q_j(s)}$ la valeur entre les crochets, c'est-à-dire

$$\boldsymbol{Q_j(s) = \sum_{j=0}^{m} B_j^m(s)\cdot P_{ij}}$$

On obtient $\boldsymbol{P(s,t) = \sum_{i=0}^{n}\ Q_j(s)\ B_i^n(t)}$

Autrement dit, les quantités $\boldsymbol{Q_j(s)}$ forment les points de contrôle d'une autre courbe de Bézier, et ensemble pour tout ***s*** et ***t,*** elles forment la surface.

Le principe de l'algorithme est :

- calculer les points $\boldsymbol{Q_j\left(\frac{1}{2}\right)}$ et les segments des courbes de Bézier pour chaque **i** par l'algorithme de Casteljau sur les courbes.
- Les ensemble de $\boldsymbol{Q_j(s)}$ tel que $\mathbf{s} \leq \frac{\mathbf{1}}{\mathbf{2}}$ forment les points de contrôle des autre courbes de Bézier, et l'ensemble pour tout ***s*** tel que $\mathbf{s} \leq \frac{\mathbf{1}}{\mathbf{2}}$ et l'ensemble de ***t,*** elles forment un moitié du surface.

  L'autre moitié de la surface peut être obtenue de la même manière. [34][29]

## 3.2. Le volume englobant et la surface de Bézier

### 3.2.1. Le volume englobant en général

L'idée des volumes englobant est de placer chaque objet dans une forme géométrique simple .Plusieurs types de géométries peuvent être utilisées, les plus couramment utilisées sont les sphères, les boîtes isothétiques de même orientation, c'est à-dire alignée sur des axes (AABB:Axis-Aligned Bounding Boxes), les boîtes englobants orientées ou à orientation quelconque (OBB : Oriented Bounding Boxes).Cependant, d'autres volumes plus complexes sont exploitables comme les polytopes à orientation discrète (k_DOP), les tribox ou encore les enveloppes convexes des objet.

Dans cette section, on choisira d'utiliser la boîte alignée avec les axes (AABB, pour axis-aligned bounding box) en raison de sa simplicité (tant pour son calcul que pour les tests d'intersection).

Une boîte englobant alignée sur l'axe (AABB) est simplement un parallélépipède rectangulaire dont les faces sont chacune perpendiculaires à l'un des vecteurs de base. De telles boîtes englobantes surviennent fréquemment dans les problèmes de subdivision spatiale, tels que le lancer de rayons. Une façon courante de spécifier une telle boîte est de fournir deux points $\mathbf{P_{min}} = [\mathbf{x_{min}}\ \mathbf{y_{min}}\ \mathbf{z_{min}}]$ et $\mathbf{P_{max}} = [\mathbf{x_{max}}\ \mathbf{y_{max}}\ \mathbf{z_{max}}]$, comme illustré dans la Figure 2.3.3. [35]

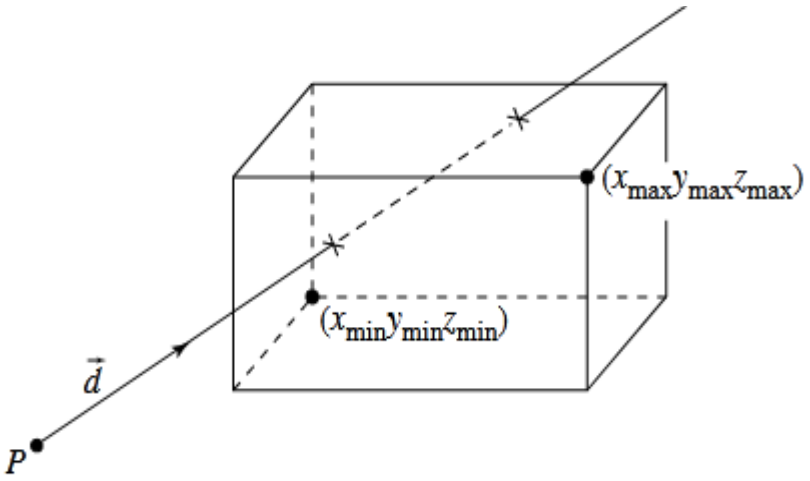



**Figure 2.3.3 : Une boîte englobant alignée sur l'axe (AABB)**

### 3.2.2. L'AABB de la surface de Bézier

Dans l'algorithme d intersection de rayon_ surface de Bézier proposé, les AABB qui englobent un ou plusieurs segments de surface sont utilisés dans le processus de construction de BVH.

Un AABB est construit en utilisant les valeurs minimale et maximale des valeurs de coordonnées de tous les points à inclure. Dans cette étude, Pour construire un AABB pour un objet (un surface ou un courbe de Bézier ) ou une partie de objet , deux approches illustrées sur la figure sont disponibles; un AABB basé sur les objets (les surfaces ou les courbes de Bézier et un AABB basé sur CP ( les points de contrôle ). Un AABB basé sur une surface est construit à partir des valeurs de coordonnées qui doivent être calculées à l'aide d'équation de la surface, de plusieurs points sur la surface, tandis qu'un AABB basé sur CP est construit en utilisant les valeurs minimales et maximales des valeurs de coordonnées de tous les points de contrôle de la surface. (Voire la figure 2.3.4 )

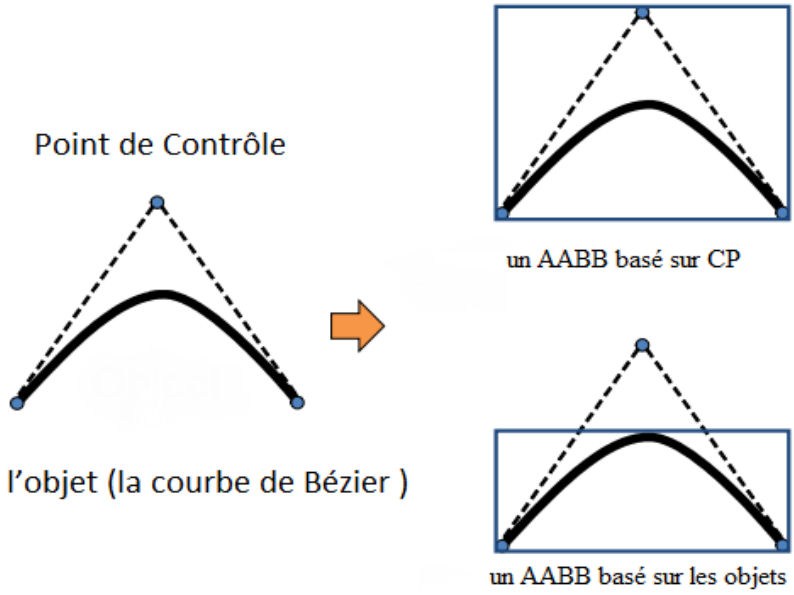



**Figure 2.3.4 : Deux façons de construire l'AABB**

Un AABB basé sur des surfaces est généralement plus compact, mais nécessite plus de charge de calcul pour obtenir les valeurs de coordonnées des points d'échantillonnage, et a une dure difficulté à ne pas enfermer parfois la surface en fonction de la sélection des points d'échantillonnage à utiliser pour la construction du AABB. Contrairement à cela, un AABB basé sur CP peut être construit en utilisant les valeurs de coordonnées des points de contrôle donnés, qui ne nécessitent aucun calcul supplémentaire, et ne manquent jamais de fermer l'objet en raison du fait que l'objet est entouré par la coque convexe des points de contrôle.

Par conséquent, on utilise dans cette étude un AABB basé CP ; La figure 2.3.5 montre un échantillon d'un AABB d'un segment de surface de Bézier. La figure 2.3.5**(b)** montre l'AABB du segment de surface et la figure 2.3.5**(a)** montre les points de contrôle du segment de surface de Bézier. [28][20]

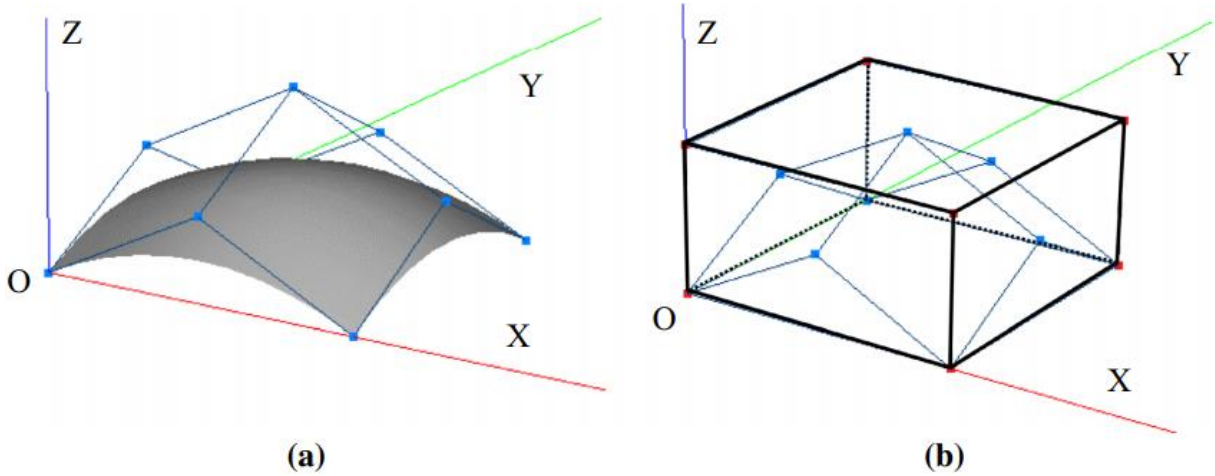



**Figure 2.3.5 : l'AABB de la surface de Bézier**

## 3.3. La Construction du BVH de surface de Bézier

Une hiérarchie de volumes englobant (BVH) est un arbre binaire qui stocke les primitives dans la scène (les patches de Bézier dans notre cas). Chaque nœud de l'arborescence représente un volume englobant.

Dans notre approche Les volumes englobantes en haut de l'arbre représentent par l'AABB de pour toute notre surface de Bézier, tandis que les nœuds en bas de l'arbre représentent les AABB de subpatch de surface de Bézier.

La structure de BVH pourrait être construite en subdivisant la surface en deux subpatch, calculer l'AABB de chaque subpatch et créer les nœuds associés. Ce processus est appliqué récursivement à chaque subpatch jusqu'à un certain seuil. [30][31][32]

## 4. Conclusion

Grâce aux différentes méthodes et les outils mentionnées précédemment, on peut faire l'optimisation de lancer de rayon pour les surface de Bézier afin de l'arrive à un bon temps de calcul.

# 1. Introduction

Les chapitres précédents représente des plusieurs approches et des algorithmes lier avec lancer de rayon qu'été proposé dont le but est d'améliorer les méthodes de création d'image de synthèse on spécifie les surface de Bézier pour la qualité d'image par a rapport la réalité ou le temps de calcul.

Dans ce cadre nous proposons une technique d'optimisation de l'algorithme lancer de rayon pour les surfaces de forme libre Bézier.

Le chapitre de conception explique et montre les étapes de notre amélioration, et dans le chapitre de l'implémentation on parle sur notre application du côté logiciel et présentons les résultats obtenus.

# 2. Conception

## 2.1. Architecture générale

Grace à la nature de l'optimisation de l'algorithme de lancer de rayon il est possible de proposer une méthode pour le calcul de rendu d'un surface de forme libre Bézier, cette méthode permis d'améliorer le temps de calcule entre les rayons et les surface de Bézier, L'architecture générale de notre approche est donnée ci-dessous :

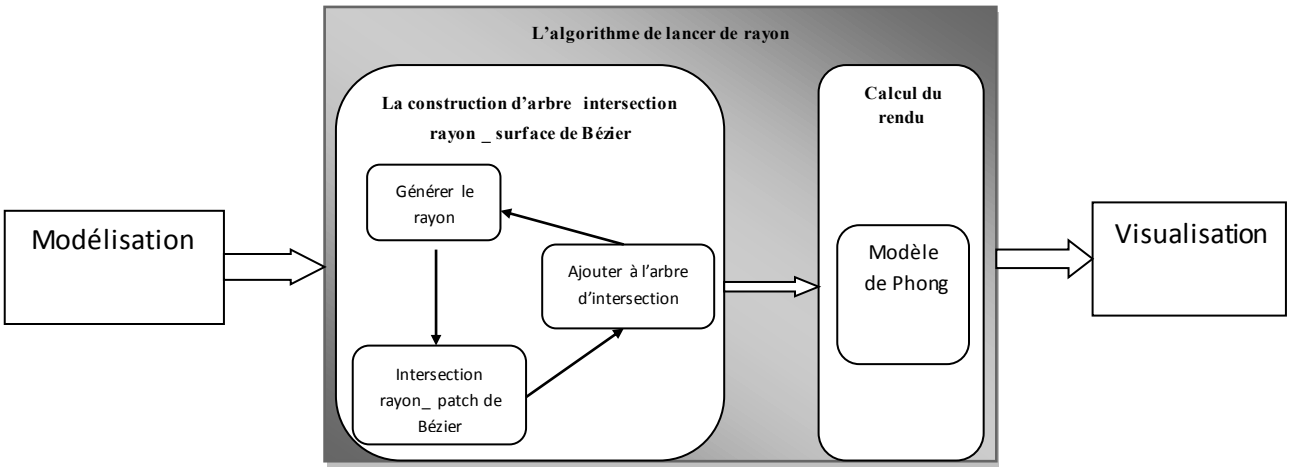



**Figure 3.2.1: L'architecture générale de notre approche**

Comme le schéma montre notre approche contient deux étapes la construction d'arbre intersection rayon _ surface de Bézier et le calcule de rendu, en explique ces processus dans les paragraphes suivant :

### 2. 1.1. Modélisation

À la base de la construction de courbe de Bézier; on modélise notre forme libre Bézier par l'utilisation de principe de l'algorithme de casteljau **[chap I, 1.3.3]** ; tel que à partir les point de contrôle donner, nous obtenons un surface de Bézier.

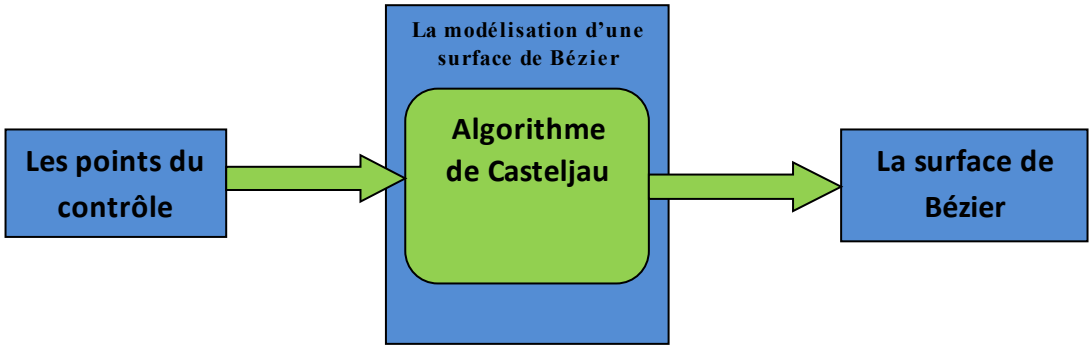



**Figure 3.2.2: Schéma de la modélisation d'une surface de Bézier**

### 2.1.2. Calcul géométriques d'intersection rayon _ surface de Bézier

Dans algorithme original **[chap I, 2.3.2]** arbre est décrivant les intersections rayons-objets est construite au fur et à mesure que le rayon primaire et les sous-rayons se propagent dans la scène, donc la propagation est récursive comme il est montré dans algorithme **[chap I, 2.3.2]**, Dans notre approche, nous concentrons sur l'intersection de la surface de Bézier _ rayon, et nous suivons cet algorithme en trois étapes suivantes :

1) Générer des patchs de Bézier en utilise la subdivision de casteljau **[chap II, 3.1].**

2) Construire BVH en utilisant la borne du patch de Bézier généré **[chap II, 3.3].**

3) Calculer l'intersection de patch Bézier - ray en utilisant la méthode de Bézier clipping **[chap I, 2.4.3.2].**

### 2.1.3. Calcul du rendu

Lorsque le processus récursif est terminé et les arbres d'intersection sont construit, il est parcouru en appliquant une équation à chaque nœud pour calculer l'intensité.

Nous avons utilisé l'équation de modèle de Phong pour calculer l'éclairage **[chap I, 2.3.2]**.

## 2.2. Avantage de notre approche

Les résultats de notre travail donne une surface dynamique de Bézier plus proche à la réalité grâce à les propriétés de l'algorithme de lancer de rayon optimiser par l'utilisation des volume englobant ; on a résout efficacement les problèmes d'aliassage a cause de l'utilisation de volumes englobant hiérarchisés (BVH) , aussi grâce à L'utilisation du volume englobant et le BVH réduisent le temps de calcul, notamment lors de la calcul d'intersection rayon _ patch de Bézier. Finalement notre approche donne des résultats réalistes dans un temps remarquable.

# 3. implémentation

Dans ce chapitre nous allons présenter en premier lieu, l'environnement de développement et les langages de programmation qui on a utilisé avec les différentes bibliothèques pour la modélisation et la programmation de notre application, ainsi que les structures de données choisies pour implémenter ce type de système. Ensuite, nous allons présenter les algorithmes utilisés illustrés par quelques résultats obtenus, nous terminons ce chapitre par une conclusion.

## 3.1. Environnement du développement

La mise en œuvre de n'importe quelle application requiert un environnement et un Langage de programmation de programmation et la nôtre est définie de la manière suivante :

### 3.1.1 Langage de la programmation java

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

Une particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation.

### 3.1.2. Environnement de programmation (NetBeans)

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000, En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).


**Figure 3.3.1: Environnement de programmation NetBeans**

Conçu en Java, NetBeans est disponible sous Windows, Linux, , Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java). L'IDE NetBeans s'appuie sur cette plate forme.

## 3.2. Les bibliothèques utilisées

### 3.2.1. Les bibliothèques java swing

Swing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes (JFC), inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures.

Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant Abstract Window Toolkit (AWT). Il utilise le principe Modèle-Vue-Contrôleur (MVC, les composants Swing jouent en fait le rôle de la vue au sens du MVC) et dispose de plusieurs choix d'apparence pour chacun des composants standards.

### 3.2.2. Les bibliothèques *java.awt*

Les bibliothèques ***java.awt*** sont un ensemble de classes fournies par Java afin de dessiner des formes sur une fenêtre. L'abréviation AWT signifie Abstract Windowing Toolkit. Aujourd'hui, la bibliothèque a été convertie en un énorme ensemble de classes qui permet à l'utilisateur de créer une application entièrement basée sur une interface graphique. L'apparence visuelle de ces classes dépend de la plate-forme sur laquelle l'application s'exécute.

Les bibliothèques de **java.awt** les plus importants qui est utilisé dans notre programme suivant :

- java.awt.Color;
- java.awt.Graphics2D;
- java.awt.Point;

## 3.3. Interface du système

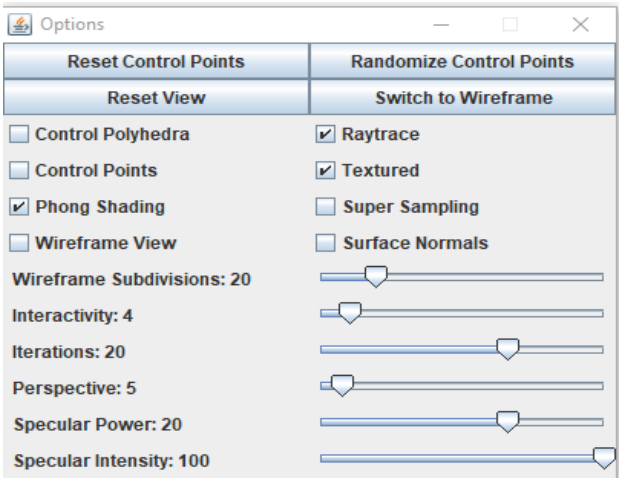



**Figure 3.3.2: Interface du système**

## 3.4. Résultats

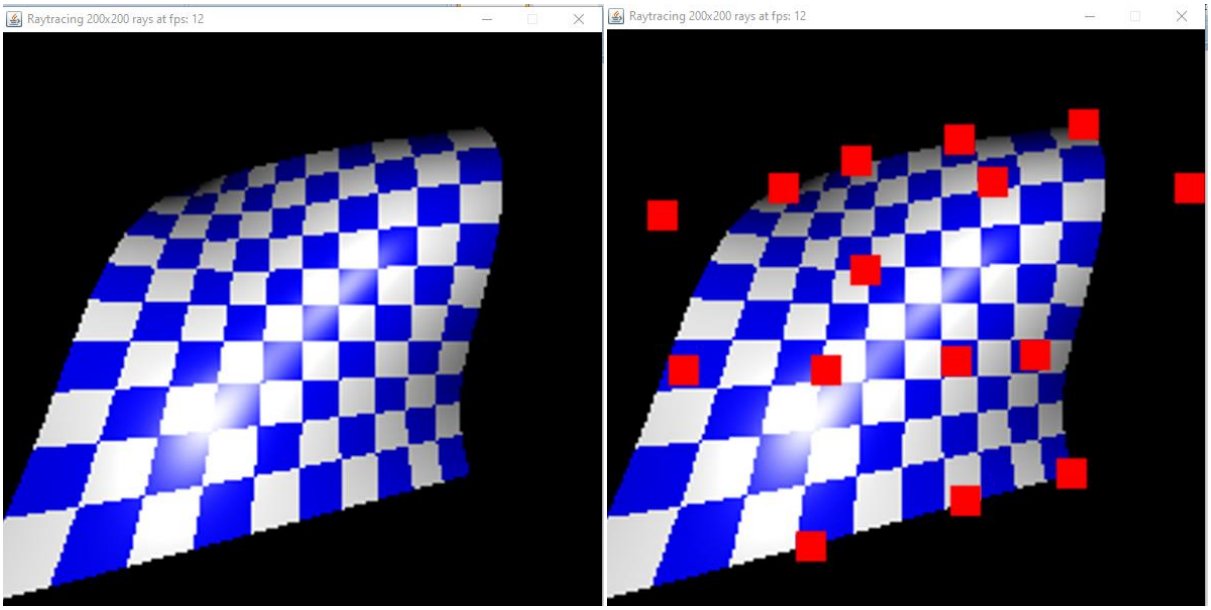



**Figure 3.3.3: le résultat, sans point de contrôle (à gauche), avec les points de contrôle (à droit)**

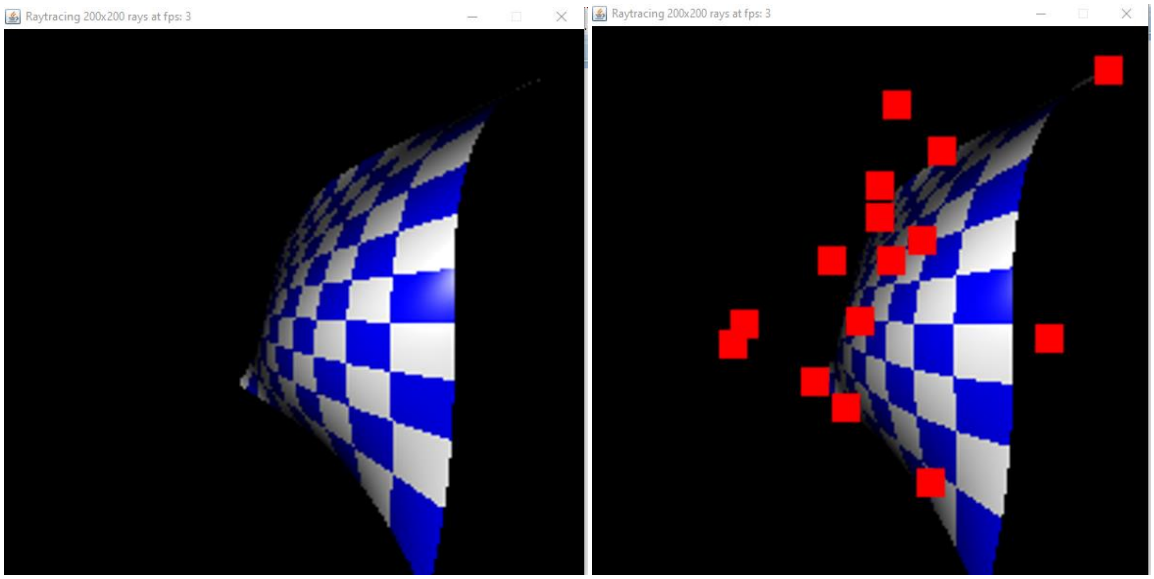



**Figure 3.3.4: le résultat dans autre dimensions, sans les points de contrôle (à gauche), avec point de contrôle (à droit)**

## Bibliographie


[1] Aymar, " Chapitre 16: Courbes de Bézier" , Saint-Seine ; 2011–2012.

[2] Jean-Paul Bécar, Jean Vareille, " Des courbes et surfaces « Bézier » ; Une histoire de géométrie polaire brûlante d'actualité " ; 2011.

[3] Claudia NEGULESCU, " Interpolation et approximation : Courbes de Bézier CMI ", Université de Provence ; Année 2007-2008.

[4] Vincent Daval, Alban Bajard, Frederic Truchetet, Olivier Aubreton, " Estimation de surface de Bézier à partir d'images de lumière structurée dans une optique de compression " ; 9 Jan 2014.

[5] Bensid Yazid, " étude des courbe de Bézier et des b_splines " , université d'ORAN ; mai 2011 .

[6] Daniel Perrin, " Les courbes de Bézier " , universite-paris-saclay.

[7] Rockwood, "Interactive Curves and Surfaces", JDill de Casteljeau.doc, The de Casteljau Algorithm for Evaluating Bézier Curves, Frome Rockwood "Interactive Curves and Surfaces" ; 29 Oct 2000.

[8] G. Demengel, J.P. Pouget, " Modèles de Béziers, de B-splines et de NURBS: Mathématiques des courbes et des surfaces ".

[9] Ahmed Zidna, " Contribution à la modélisation des carreaux troués " , Université Paul Verlaine – Metz ;1990.

[10] Piere Benard ; URL : http://www.labri.fr/perso/pbenard/teaching/pghp, visiter le : 05-04-2021.

[11] Thomas Guillod, " Interpolations, courbes de Bézier et B-splines ", réalisé au Lycée Blaise-Cendrars sous la direction de Jean-Bernard Mathey .

[12] Severine Dubuisson, " Courbes et surfaces Algorithmique Graphique et Modélisation " ; mars 2007.

[13] Sylvain Brandel, " Modélisation géométrique 2 ", mars 2019.

[14] Fabrice Aubert ," Lancer de Rayon et structures de partitionnement ",Université de Lile ; 2012-2013.

[15] Kouda Oussama, " Implémentation optimisée de l'algorithme du Lancer de faisceaux ", Mémoire de fin d étude, Université Mohamed Khider – BISKRA ; avril 2015.

[16] Jacqueline Argence, "Algorithmes pour le trace de rayons dans le cadre d'une modélisation par arbre de construction ", Université Jean Monnet .

[17] Christian Nguyen, "Le lancer de rayons (raytracing) - Concepts fondamentaux- ", Université de Toulon .

[18] PASCAL MIGNOT, " Chapitre 1 : Le lance de rayon ", Université de Reims.

[19] Frédéric Claux, " Rendu interactif de modéles B-Rep sur GPU ", Doctorat de L'Université de Toulouse ; 2014.

[20] Josef Pelikán , " Ray × Bézier surface intersection " , computer Graphic cherles university.

[21] Joakim Löw, " Ray Tracing Bézier Surfaces on GPU ",Linköping Universitet .

[22] S. H. Martin Roth, Patrick Dezi†, Markus H. Gross, " Triangular Bézier Clipping ", Swiss Federal Institute of Technology (ETH).

[23] Michael T. Heath, " An Introductory Survey " , McGrawHill - second edition - ; 2002.

[24] Philippe RIS ," classification et définition- chapitre 3- " , Théze de doctorat .

[25] Andrew S. Glassner , " An introduction to ray tracing sous la direction de Londres ", Academic press limited, United Kingdom ; July 1989.

[26] Gaël Guennebaud, " Lancer de Rayon & Structures de partitionnement (ray-tracing ) " .

[27] Kay (T.L.), Kajiya (J.), " Ray tracing complexe scènes " ,Computer Graphics , 20(4) ; Août 1986.

[28] Atsuya Oishi1 · Genki Yagawa , " A surface-to-surface contact search méthode en hanced by deep Learning ", 9 January 2020.

[29] Kenneth I. Joy," On-Line Geometric Modeling Notes, bézier patch subdivision ", University of Californie, Davis .

[30] Shi-Min Hu, " Computer Graphics ", Tsinghua Université.

[31] wiki ; URL : https://fr.xcv.wiki/wiki/Bounding_volume_hierarchy, visiter le: 27 mai 2021.

[32] Brian Fischer (bfischer),Ian Huang (ijhuang) , "Parallèle BVH Construction on a CPU".

[33] Heckbert (P.S.), Hanrahan (P.), " Beam tracing polygonal objectes, Computer Graphics", 18(3) ; juillet 1984.

[34] mtu; URL : https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/surface/bezier-de-casteljau.html, visiter le : 19 mars 2021.

[35] Boucetta Abdelkader, "simulation pre-clinique pour optimisation de la prothese femorale en appliquant les modeles de la CAO ", Mémoire de fin d étude, université des sciences et de la technologie houari boumediene.

# L’annexe

## 1. La fenêtre de rendu

La fenêtre de rendu montre le résultat final du traitement en affiche le frame buffer à l'écran. En plus de la surface, vous pouvez visualiser les points de contrôle et le polygone, ainsi que l'interpolation des points de contrôle en 2D.

### 1.1. Interaction avec la souris

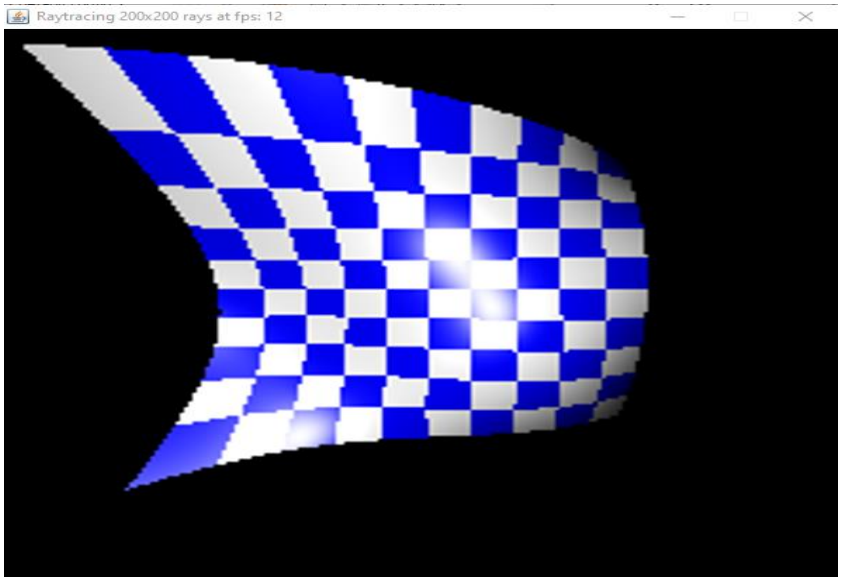
**Points de contrôle** : les points de contrôle peuvent être sélectionnés et déplacés lorsqu'ils sont visibles.

**Zoom** : avec la molette de la souris, il est possible de modifier la distance de la surface au centre de projection des rayons, modifiant ainsi la taille perçue sur l'écran.

**Rotation** : en cliquant et en faisant glisser une zone vide de la fenêtre de rendu, il est possible de faire pivoter la surface de Bézier selon les axes X et Y.

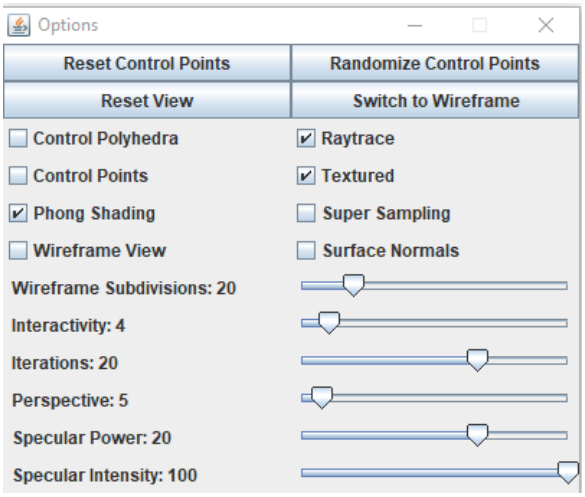
### 1.2. Source de lumière

La source lumineuse est une lumière fixe positionnée à (0, 0, 0) dont la couleur est le blanc. Celui-ci émet en effet une lumière blanche qui rebondit sur la surface révèle ses formes et ses couleurs.



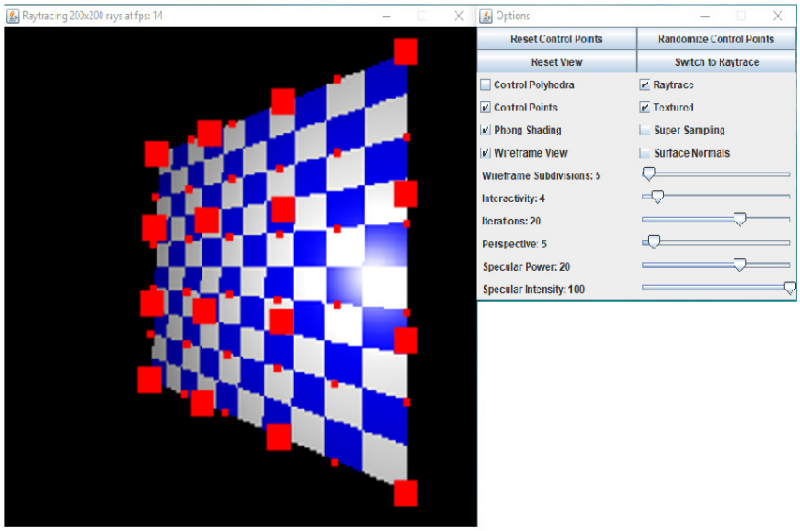
## 2. Les options de Boîte à outils

Ce panneau nous permettons de contrôler les paramètres du moteur de lancer de rayons et de choisir les informations affichées dans la fenêtre de rendu.



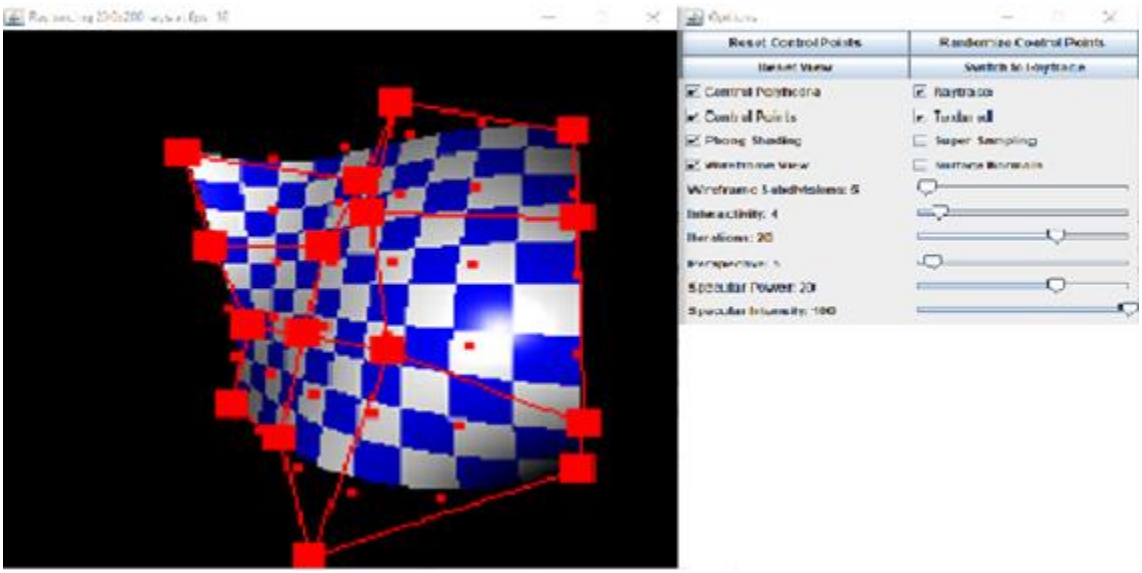

## 2.1. Reset Control Points

Ce bouton réinitialise tous les points de contrôle pour former une surface de forme carrée. De cette façon, tous les points se trouvent sur le même plan.



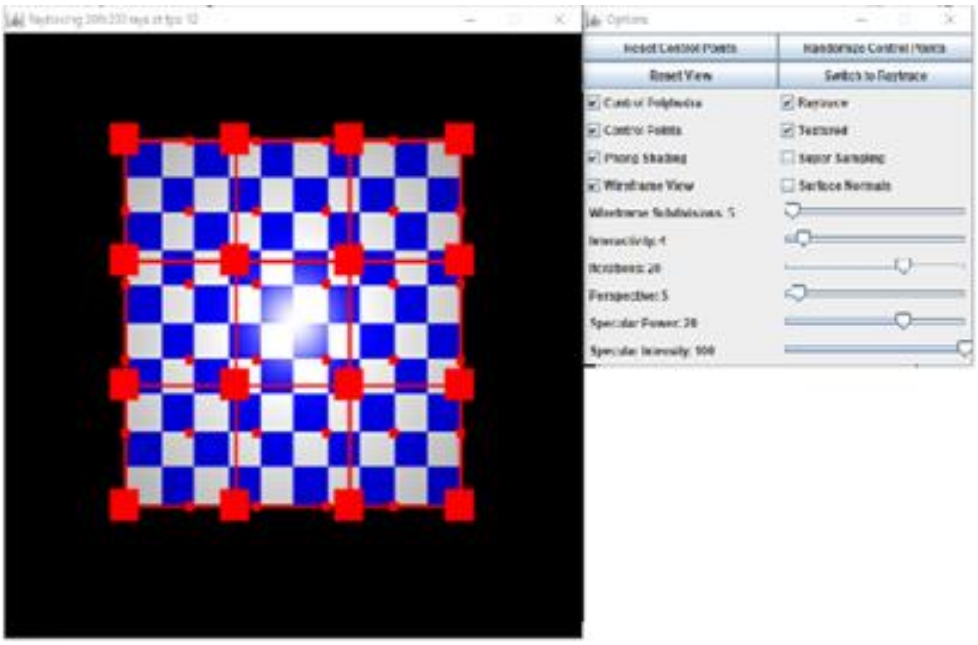
## 2.2. Randomize Control Points

Ce bouton agit également sur les points de contrôle en perturbant leur position par rapport à l'axe Z. Les valeurs sont calculées de manière aléatoire dans une plage prédéterminée. Cette fonctionnalité vous permet de modifier la surface de manière très simple pour tester rapidement différentes formes.



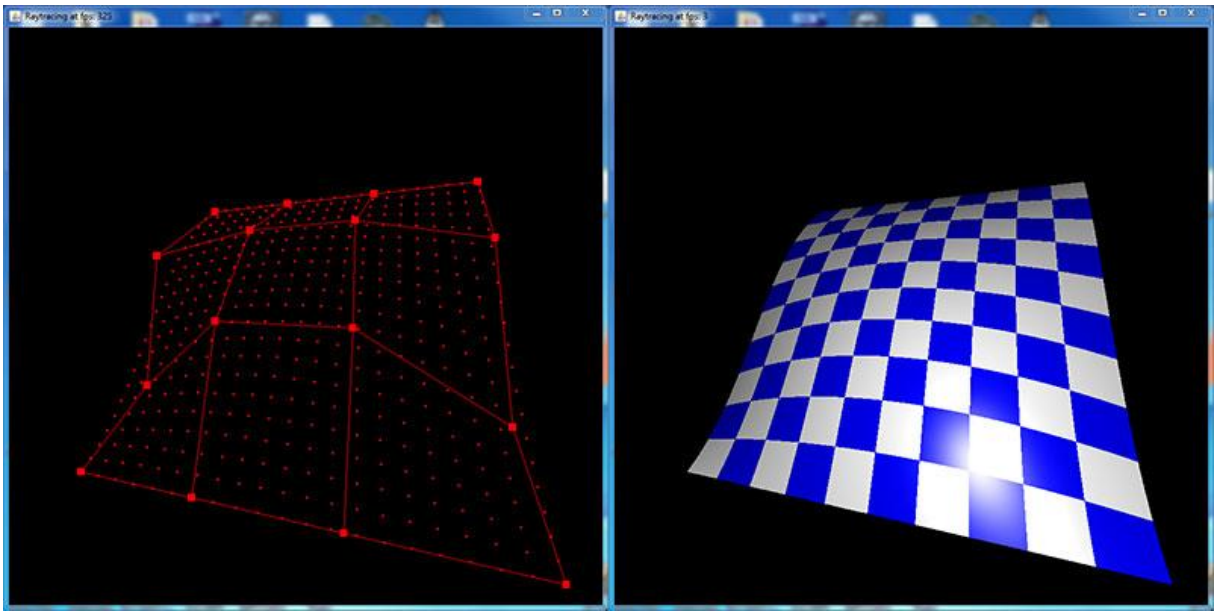

## 2.2. Reset View

Lorsque la rotation de la surface est modifiée, ce bouton vous permet de revenir à la valeur d'origine.



## 2.3. Switch to Wireframe/Raytrace

Modifiez les paramètres en même temps pour passer du mode filaire (Wireframe), dans lequel seuls le polygone et les points de contrôle sont affichés avec les points d'interpolation 2D, au mode lancer de rayons dans lequel seule la surface est affichée.



## 2.4. Control Polygon

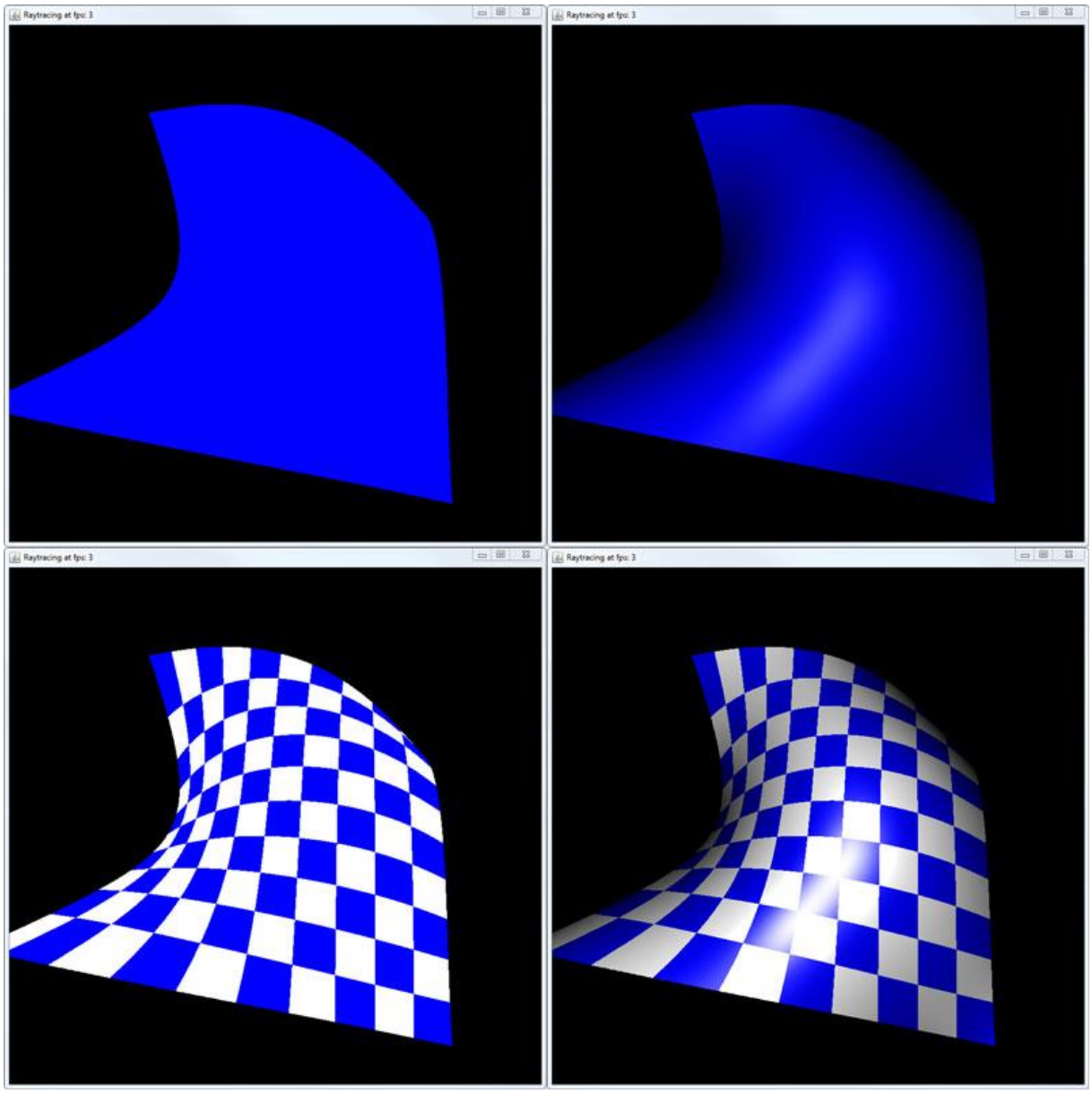
Lorsqu'il est sélectionné, il affiche le polygone de contrôle.

## 2.5. Control Points

Lorsqu'il est sélectionné, il affiche les points de contrôle associés à la surface.

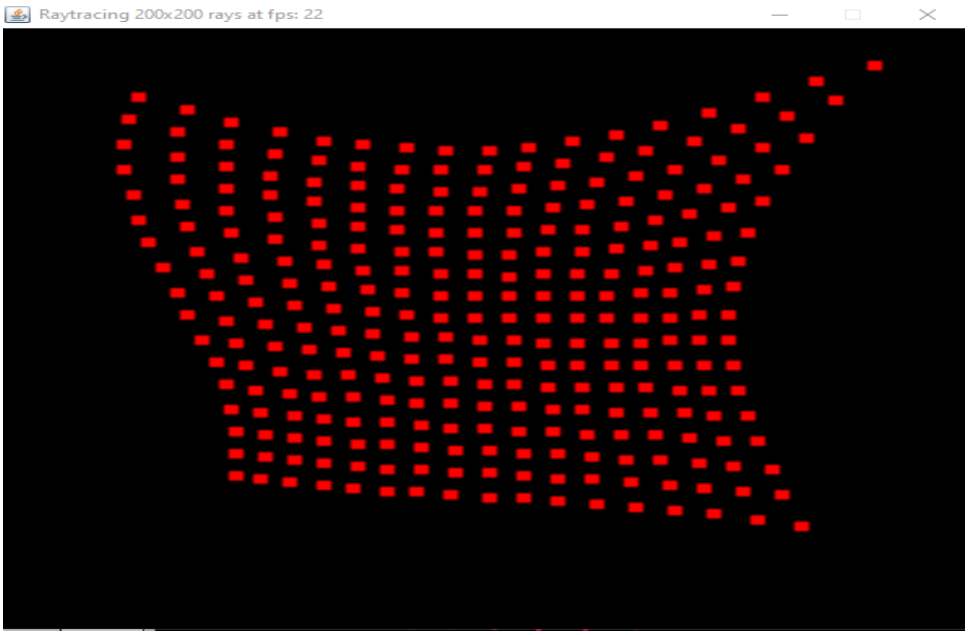
## 2.6.Phong Shading

Cette balise spécifie le mode d'ombrage, Lorsqu'elle est active, la surface est éclairé par la modèle de Phong.



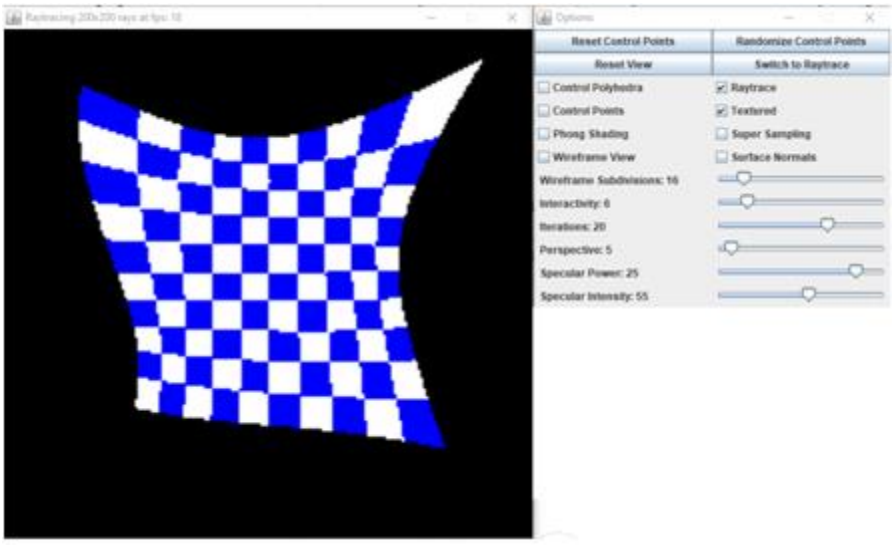

## 2.7. Wireframe View

La visualisation filaire est obtenue en interpolant les points de contrôle projetés sur le framebuffer, elle a donc un faible coût de calcul. En utilisant De Casteljau sur les points de contrôle dans l'espace 2D du framebuffer, il est possible d'obtenir une représentation temporaire de la surface.




## 2.8. Texture

Une texture est généralement une image qui est appliquée sur une surface plane pour l'enrichir de détails. Chaque point de la surface a une paire de coordonnées (u, v) qui sont ensuite mappées dans l'image source. Dans ce cas, la source n'est pas une image raster mais un damier généré dynamiquement, nous avons donc une texture procédurale. L'utilisation de cette texture permet d'avoir une idée de la répartition des points sur la surface, donnant également à l'utilisateur une meilleure perception de la tridimensionnalité de la surface.

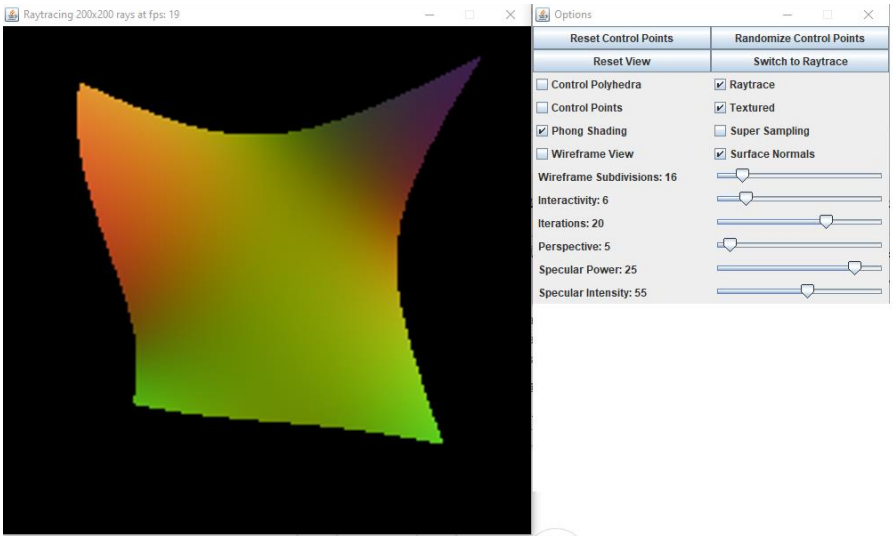


## 2.9. Super-Sampling

L'activation de ce drapeau active la méthode d'optimisation de lancer de rayons pour la surface de Bézier.

## 2.10. Surface Normals

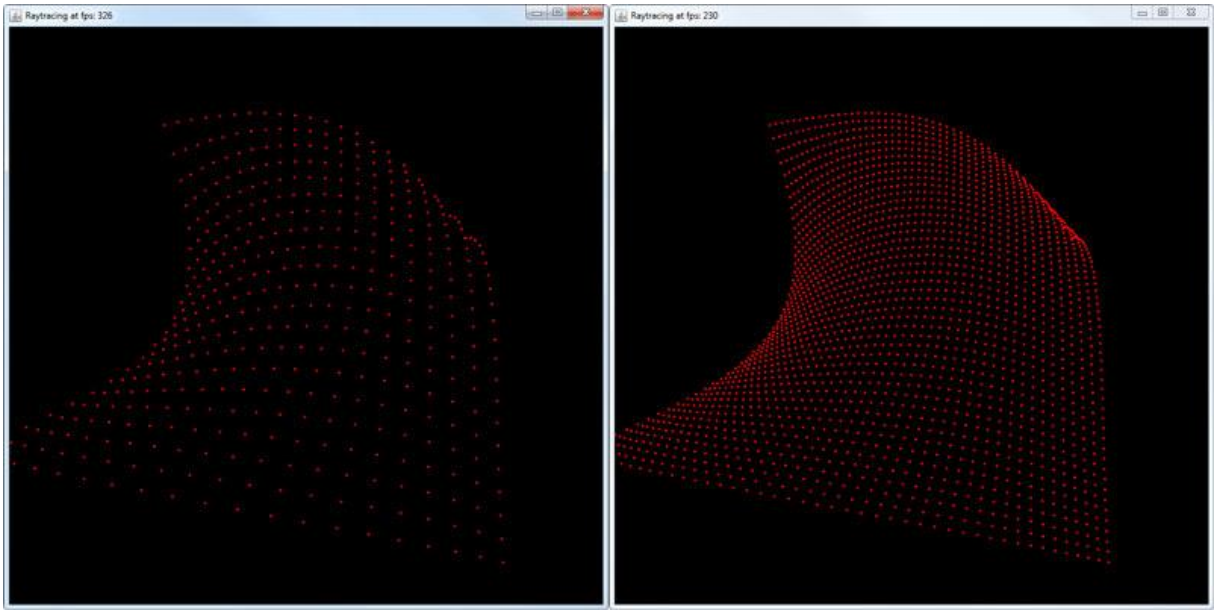
Cette représentation permet de visualiser en rgb codant la direction des vecteurs normaux pour chaque point de la surface.




## 2.11. Wireframe Subdivision

Contrôle la granularité avec laquelle les points sur la surface 2D sont évalués. Cet outil représente une bonne approximation de la surface, pour une utilisation limitée des ressources.

Ce curseur vous permet de définir le nombre d'étapes nécessaires pour rendre une seule image, cela augmente la vitesse de réponse de l'application aux entrées de l'utilisateur, mais d'un autre côté, cela ralentit le processus de rendu.

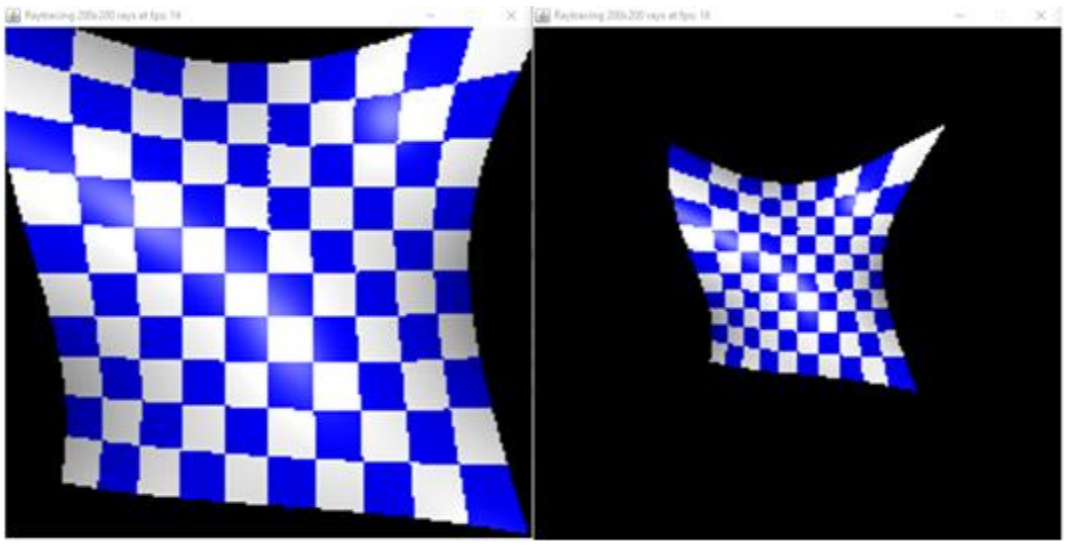



## 2.12. Interactivity

Ce curseur vous permet de définir le nombre d'étapes nécessaires pour rendre une seule image, cela augmente la vitesse de réponse de l'application aux entrées de l'utilisateur, mais d'un autre côté, cela ralentit le processus de rendu.

## 2.13. Perspective

En changeant la divergence des rayons par rapport à la direction de l'axe Z, il est possible de changer la perspective de la vision, changeant ainsi la perception de la profondeur.



## Note :

Pour l'exécution de notre application, il faut d'installer l'environnement de programmation (NetBeans) ,nous cliquons sur File, Open Project puis en sélectionner notre projet (Projet_Final).