Image multi-scène par intégration d'un autostéréogramme dans une scène 3D

Julien Dehos, Pierre-Alexandre Hébert, Christophe Renaud

Revue Électronique Francophone d'Informatique Graphique, Volume 8, Numéro 2, pp. 67–78, 2014

Image multi-scène par intégration d'un autostéréogramme dans une scène 3D

Julien Dehos, Pierre-Alexandre Hébert, Christophe Renaud

Université Lille- Nord de France, ULCO
LISIC - BP 719 - 62228 Calais cedex

Résumé

In this paper, we propose a method to insert a single-image-stereogram (SIS) into a 3D scene. Thus, the final displayed image contains both the main 3D scene and the included SIS scene. The viewer can see the main scene, when focusing his eyes on the display screen, or the SIS scene, when focusing behind. The method is implemented on the GPU and produces Full-HD images in real-time.

Dans ce papier, nous proposons d'intégrer un autostéréogramme dans une scène 3D de telle sorte que l'image finalement affichée contienne à la fois la scène 3D principale et la scène de l'autostéréogramme. En focalisant son regard sur l'écran d'affichage ou derrière celui-ci, l'utilisateur peut alors voir l'une ou l'autre des deux scènes, réalisant ainsi une sorte de "contrepèterie visuelle". La méthode s'implémente en quelques passes de rendu sur GPU, et permet d'obtenir un affichage Full-HD en temps-réel.

Mots clé : Informatique Graphique, Autostéréogramme, Rendu, GPU, image multi-scène

1. Introduction

Dans ce papier nous nous intéressons à la réalisation d'images représentant plusieurs scènes à la fois. Ainsi, une même image peut être interprétée selon des sens différents.

Pour réaliser ce genre d'images, il existe différentes techniques dont certaines sont utilisées depuis longtemps par des artistes. Par exemple, la scène principale de l'image peut être organisée de telle sorte qu'elle produise une ombre dont la forme fait penser à un objet particulier.

Les autostéréogrammes peuvent également être considérés comme des images multi-scènes. En effet, à première vue, un autostéréogramme ressemble à une image de texture mais, vu d'une façon particulière, il permet de percevoir une forme en relief. Cependant, l'aspect multi-scène d'un autostéréogramme est relativement limité dans la mesure où la scène principale se résume toujours à une image de texture.

L'idée proposée dans ce papier est d'intégrer un autostéréogramme dans une scène virtuelle 3D classique. Ainsi, le rendu obtenu permet de voir à la fois la scène 3D et la forme en relief contenue dans l'autostéréogramme. Pour cela, il suffit d'utiliser l'autostéréogramme comme une texture classique dans la scène 3D mais ceci nécessite d'adapter le calcul de l'autostéréogramme. La méthode s'implémente sur GPU et permet un rendu temps-réel. Elle peut donc être utilisée pour des applications interactives (par exemple, pour proposer une énigme dans un jeu vidéo) aussi bien que pour des effets purement artistiques.

Nous apportons principalement deux contributions. Tout d'abord, nous proposons une formulation simple et explicite du calcul d'un autostéréogramme, en séquentiel et en parallèle, applicable pour différents types d'autostéréogrammes. En effet, les formulations que nous avons trouvées dans la littérature sont limitées à un type d'autostéréogramme particulier ou données implicitement à travers une implémentation particulière difficile à reprendre. Enfin, nous proposons une méthode pour intégrer un autostéréogramme dans une scène 3D virtuelle et ainsi réaliser des images multi-scènes. Cette méthode permet un rendu en temps-réel et peut être implémentée dans une application 3D interactive de type jeu vidéo ou "serious game".

La suite de l'article est organisée comme suit : la section 2 présente des approches existantes (dont les autostéréogrammes) permettant de réaliser des images multi-scènes. La section 3 détaille le principe des autostéréogrammes et des méthodes pour les calculer puis la section 4 décrit notre méthode d'intégration d'un autostéréogramme dans une scène 3D. Enfin, la section 5 présente quelques résultats et la section 6 quelques discussions.

Figure 1: *"Vertumne" (Giuseppe Arcimboldo). Exemple d'image multi-scène par collage : dans son ensemble, l'image représente un portrait mais dans le détail, il s'agit d'un assemblage de végétaux.*

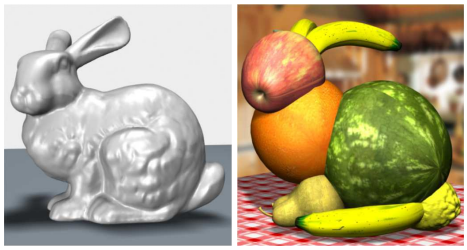


Figure 2: *Collage automatique d'objets 3D [GSP*07]. À gauche : l'objet à représenter. À droite : l'assemblage d'objets de base obtenu par la méthode.*

2. Travaux existants

2.1. Images multi-scènes

Il existe différentes méthodes pour réaliser des images multi-scènes. Il est intéressant de remarquer que ces méthodes sont souvent inspirées par des travaux d'artistes, parfois vieux de plusieurs siècles.

La technique des collages consiste à regrouper des images de telle sorte que l'assemblage complet constitue elle-même une image à part entière. Les oeuvres d'Arcimboldo en sont certainement les exemples les plus connus (voir Figure 1). En reprenant cette idée, Huang et al. proposent une méthode de collage d'images [HZZ11] et Gal et al. une méthode de collage d'objets 3D [GSP*07] (voir Figure 2).

Une autre technique pour réaliser des images multi-scènes consiste à disposer les objets et sources de lumière de la scène principale de telle sorte que les ombres alors produites représentent elles-mêmes d'autres scènes [Cas02] (voir Figure 3). Mitra et Pauly [MP09] proposent une méthode pour calculer un objet 3D produisant des ombres données selon un éclairage particulier (voir Figure 4).

La technique de l'anamorphose consiste à placer, dans une

Figure 3: *"La Découverte de l'ombre" (Roberto Casati) [Cas02]. Image multi-scène utilisant les ombres. La scène principale (les lapins) contient une ombre qui représente en fait une seconde scène (une main).*

Figure 4: *Calcul d'un objet 3D à partir des ombres voulues [MP09].*

scène, une image déformée pour un point de vue fixé. Ainsi, si on observe la scène depuis ce point de vue, l'effet de perspective compense la déformation de telle sorte que l'anamorphose semble exister réellement dans la scène (voir Figure 5). Les anamorphoses sont utilisées depuis longtemps, par exemple pour réaliser des trompe-l'oeil, des marquages au sol de signalisation routière... Il s'agit souvent d'un dessin ou d'une peinture réalisés sur des structures urbaines (rue, batiment...). Différents travaux utilisent les anamorphoses, par exemple pour reconstruire des scènes réelles à partir de photographies [VRC*13] ou pour visualiser des objets virtuels sur un dispositif d'affichage portable [EO11] (voir Figure 6).

Figure 5: *Anamorphose dessinée à la craie dans la rue (Julian Beever). À gauche : l'anamorphose est observée depuis son point de vue de référence ; l'escargot semble exister réellement. À droite : depuis un point de vue différent, l'effet ne se produit pas et on perçoit le dessin déformé.*

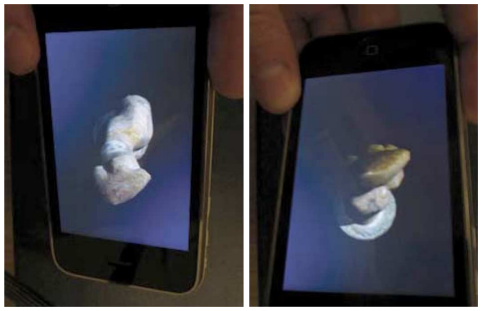


Figure 6: *Utilisation de l'anamorphose pour simuler la visualisation d'un objet 3D virtuel sur un smartphone en fonction de l'orientation de celui-ci [EO11].*

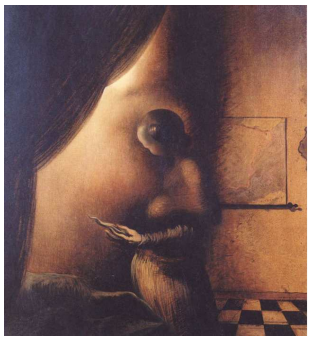


Figure 7: *"L'image disparaît" (Salvador Dali). Exemple de camouflage : l'image peut être interprétée à la fois comme une femme de profil ou comme la tête d'un homme.*

La technique du camouflage consiste à créer une image, généralement figurative, qui peut être interprétée de différentes façons. Cette technique a été largement utilisée par les artistes (par exemple, voir Figure 7). Certains travaux utilisent cette technique pour réaliser des animations stylisées [MCL*09] ou pour intégrer des images dans des photographies [CHM*10] (voir Figure 8).

Enfin, les images hybrides sont des images dans lesquelles on perçoit des scènes différentes selon la distance d'observation (voir Figure 9). Une image hybride peut se calculer en combinant deux images, l'une filtrée passe-bas et l'autre filtrée passe-haut. Ainsi le spectateur perçoit l'image de haute fréquence lorsqu'il est proche de l'image hybride et l'image de basse fréquence lorsqu'il s'en éloigne [OTS06, IDW*13] (voir Figure 10).

2.2. Autostéréogrammes

Un autostéréogramme est une image construite de telle sorte que l'on peut y percevoir une forme en relief lorsqu'on la regarde d'une façon particulière (en focalisant le regard devant ou derrière le plan réel de l'image). Il peut être affiché

Figure 8: *Image multi-scène par camouflage [CHM*10]. À gauche : l'image principale (le paysage) avec les sous-images à camoufler (les quatres têtes de lion). À droite : l'image résultat.*

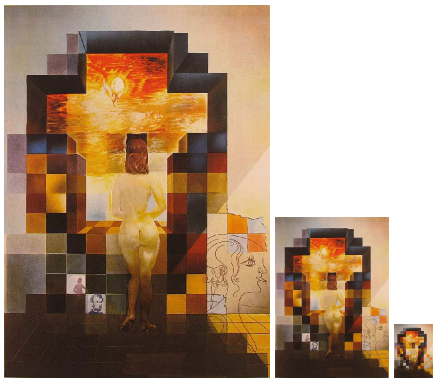


Figure 9: *"Gala regardant la mer Méditerranée qui à vingt mètres se transforme en portrait d'Abraham Lincoln" (Salvador Dali). Exemple d'image hybride.*

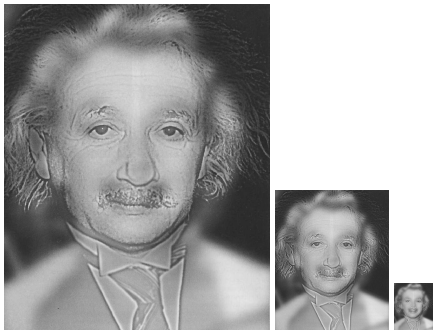


Figure 10: *Exemple d'image hybride [OTS06]. De près, le spectateur perçoit le portrait d'Albert Enstein (image de haute fréquence), de loin, celui de Marilyn Monroe (image de basse fréquence).*

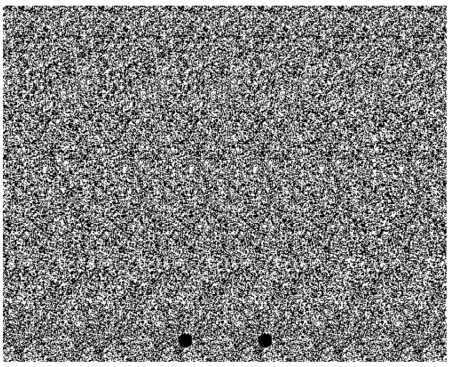


Figure 11: *Autostéréogramme généré à partir de points aléatoires [TIW94]. En focalisant le regard derrière l'écran, on aperçoit une forme sphérique en relief. Les deux pastilles noires permettent d'aider à percevoir l'autostéréogramme.*

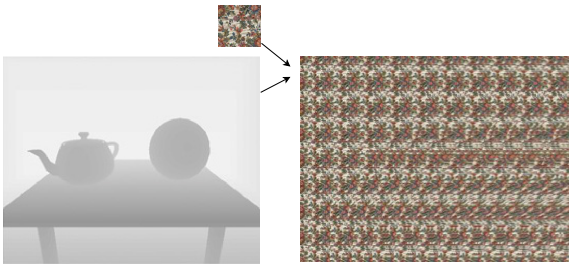


Figure 12: *Autostéréogramme généré à partir d'une image de texture [Pol04]. La méthode utilise une image de texture (en haut à gauche) et une carte de profondeur (en bas à gauche) pour générer l'autostéréogramme (à droite) en temps-réel, via le GPU.*

sur un écran ou imprimé sur un support papier et ne nécessite aucun appareil de vision particulier.

Il existe plusieurs types d'autostéréogrammes : ceux générés à partir de points aléatoires (voir Figure 11), ceux générés à partir d'une image de texture (voir Figure 12) et ceux réalisés par des répétitions de motifs au sein d'une même image (voir Figure 13).

Différentes méthodes permettent de calculer des autostéréogrammes. Dans [TC90], Tyler et Clarke proposent de moduler la fréquence de répétition d'un motif aléatoire pour contrôler le relief perçu. Ils présentent les éléments de base qui sont utilisés dans les méthodes récentes de calcul d'autostéréogrammes sur GPU. Cependant, leurs travaux se limitent à des autostéréogrammes basés points aléatoires. De plus, ils détaillent leur méthode de calcul mais n'en donnent pas de justification rigoureuse.

Thimbleby et al. proposent une méthode différente pour calculer des autostéréogrammes basés points aléatoires [TIW94]. Il s'agit, dans une première passe, de déterminer

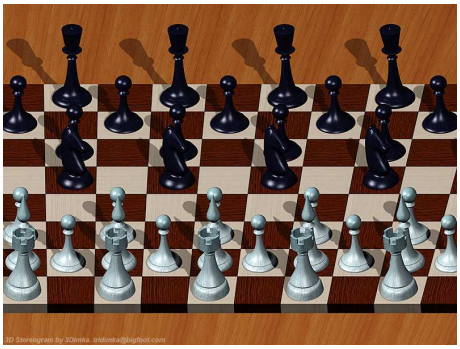


Figure 13: *Autostéréogramme réalisé par un artiste en répétant des motifs à l'intérieur de l'image [3Di]. En focalisant correctement, on perçoit réellement la scène en relief.*

les ensembles de pixels devant avoir la même couleur puis, dans une seconde passe, d'affecter des couleurs aux pixels. Contrairement à la méthode de Tyler et Clarke, leur algorithme fonctionne de manière symétrique, c'est-à-dire sans biais issu d'un sens de parcours privilégié. Malheureusement l'étape de partitionnement de l'ensemble des pixels est coûteuse et peu adaptée à une implémentation GPU efficace.

L'implémentation sur GPU du calcul des autostéréogrammes a fait l'objet de différents travaux. Les implémentations proposées sont basées sur la méthode de Tyler et Clarke [TC90] et consistent à calculer un autostéréogramme par bandes verticales successives, en plusieurs passes de rendu. Dans [Ges], Geselowitz rappelle la méthode de base puis en propose une implémentation GPU. Cependant, il manque toujours une justification rigoureuse de cette méthode et en particulier des approximations qu'elle réalise. Enfin, Geselowitz présente son implémentation à travers un code assembleur GPU, ce qui la rend difficile à comprendre.

Dans [Pol04], Policarpo propose également une implémentation GPU fonctionnant par bandes verticales successives. Il utilise une formulation originale du calcul mais n'en donne aucune justification. Enfin, dans [PGM03], Petz et al. formalisent le problème de façon rigoureuse puis en déduisent un algorithme et une implémentation GPU (shader en code assembleur). Cependant, leur implémentation réalise exactement le calcul formalisé sans considérer d'approximation possible pour améliorer le temps de calcul.

Les méthodes existantes calculent des autostéréogrammes destinés à être affichés sur tout l'écran. Les images ainsi obtenues permettent de percevoir la forme en relief facilement mais l'aspect multi-scène est limité (une scène en relief et une image de texture répétitive ressemblant à du papier-peint). À notre connaissance, personne n'a encore proposé d'utiliser les autostéréogrammes en les intégrant dans une autre scène, comme proposé dans ce papier.

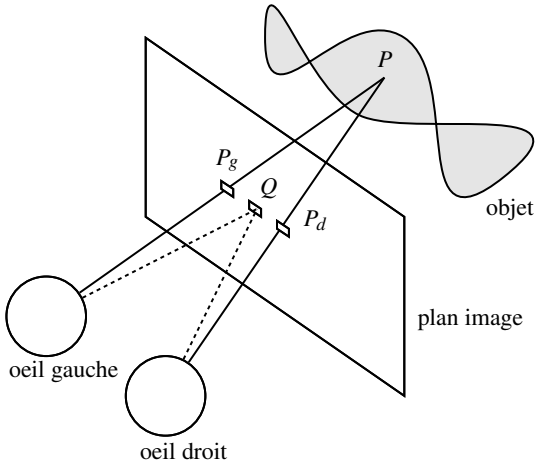


Figure 14: *Principe des autostéréogrammes. Dans une image classique, les yeux focalisent sur des points/pixels Q d'un même plan et on ne perçoit donc aucun relief. Dans le cas d'un autostéréogramme, on affecte une même couleur à deux pixels P_g et P_d si bien que lorsque les yeux gauche et droit sont dirigés respectivement vers P_g et P_d, on perçoit en fait un point P situé derrière le plan image. L'écart entre P_g et P_d définit la profondeur de P, ce qui permet de donner une impression de relief sur l'ensemble de l'image.*

d	diagonale de l'écran (53 cm)
a	rapport de cadre de l'écran ($16/9$)
w_p	largeur de l'écran en pixels (1920 px)
w	largeur de l'écran en cm
f	distance de l'écran (100 cm)
f'	profondeur du point P à simuler
e	écart pupillaire (6.5 cm)
e_p	écart pupillaire en pixels
e'	écart en cm entre les pixels simulant le point P
e'_p	écart en pixels entre les pixels simulant le point P

Table 1: *Notations utilisées Figure 15. Les valeurs données ici correspondent à un écran Full-HD de 21 pouces placé à 1 m de l'utilisateur.*

3. Fondements des autostéréogrammes

Dans cette section, nous détaillons le principe des autostéréogrammes, comment les percevoir et comment les construire. Notre approche est valable pour des autostéréogrammes basés points aléatoires et pour des autostéréogrammes basés images de texture. Enfin, nous présentons une méthode de calcul séquentielle et une méthode parallèle.

Les résultats donnés ici ont, pour la plupart, déjà été publiés [PGM03, Ges, Pol04]. Cependant, comme expliqué dans la section précédente, les travaux existants se limitent généralement à un type d'autostéréogramme ou à un algorithme particulier, manquent de justifications ou présentent leur méthode à travers une implémentation difficilement compréhensible (code assembleur GPU). C'est pourquoi, nous tentons, dans cette section, de présenter les autostéréogrammes de façon claire (sans détails d'implémentation), complète (du principe de base aux méthodes de calculs) et rigoureuse (notamment en explicitant les approximations réalisées).

3.1. Principe

Le principe d'un autostéréogramme est de regarder derrière (ou devant) l'image réelle (voir Figure 14). En effet, avec une image classique, on focalise le regard sur les pixels de l'image physiquement affichée, c'est-à-dire des points d'un même plan. Il n'y a donc pas de relief. Le but d'un autostéréogramme est de représenter un objet 3D situé derrière le plan image, par exemple au point P. Pour cela il suffit de trouver les pixels P_g et P_d, vus respectivement par les yeux gauche et droit lorsqu'ils focalisent sur P, et de leur affecter une même couleur. L'écart entre P_g et P_d est lié à la profondeur de P ; donc, en faisant varier cet écart dans l'image, on peut donner une impression de relief.

Plus précisément, en utilisant les notations présentées Figure 15 et Table 1, l'écart e' (en cm) entre deux pixels correspondant à une profondeur f' est :

$$e' = \frac{f'e}{f+f'} \tag{1}$$

On retrouve ce résultat dans [PGM03]. De plus, en remarquant que la largeur de l'écran en cm est :

$$w = \frac{ad}{\sqrt{1+a^2}} \tag{2}$$

alors on peut calculer l'écart e'_p (en pixels) en fonction de la profondeur f' (en cm) :

$$e'_p = \frac{w_p e'}{w} = \frac{f' w_p e}{(f+f')w} \tag{3}$$

Réciproquement, on obtient f' en fonction de e'_p avec :

$$f' = \frac{e'_p w f}{w_p e - e'_p w} \tag{4}$$

Avec les valeurs numériques proposées Table 1, l'écart maximal (quand $f' \to \infty$) est $e'_{p\infty} = e_p \approx 277$ px.

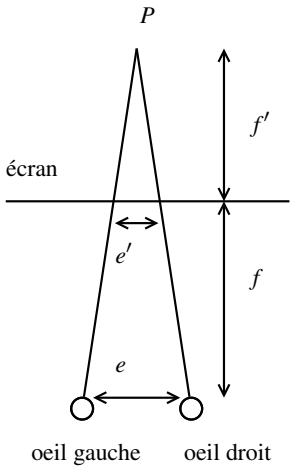


Figure 15: *Simuler un point P à une profondeur f', revient à calculer la distance e' sur l'écran et à affecter la même couleur aux deux pixels correspondants (voir les notations Table 1).*

3.2. Perception

Pour visualiser un autostéréogramme, il suffit de focaliser son regard derrière l'image réelle. Malheureusement, l'application pratique de ce principe est nettement moins facile et demande un apprentissage qui peut être aussi frustrant que l'effet de profondeur, une fois perçu, peut être saisissant [N.E93].

Il existe différentes méthodes pour acquérir la "technique" de visualisation. Une première méthode consiste à "regarder dans le vide", à travers l'image réelle. Pour aider à trouver la focalisation, on ajoute parfois deux pastilles colorées (voir Figure 28, à la fin du papier). Lorsque l'on commence à focaliser derrière l'image, ces pastilles se dédoublent puis deux de ces pastilles "fantômes" se regroupent au centre lorsque la focalisation correcte est atteinte. On peut alors, en gardant la focalisation, percevoir le relief sur le reste de l'image.

Une autre méthode consiste à se placer devant une fenêtre et à tenir un autostéréogramme (sur papier) à bout de bras, face à soi. On regarde tout d'abord au loin à travers la fenêtre, au-dessus de l'autostéréogramme. Puis on descend le regard sur l'autostéréogramme en essayant de garder la focalisation.

Une dernière méthode, consiste à "coller son nez sur l'autostéréogramme". La focalisation sur le plan image est alors physiologiquement impossible ce qui force à regarder à travers l'image réelle. Il suffit alors de reculer en conservant le même regard ; l'autostéréogramme va alors apparaître au bout d'une certaine distance.

3.3. Calcul séquentiel

Pour construire un autostéréogramme, on utilise généralement une image de profondeur, qui détermine les écarts entre les paires de pixels, et une texture, qui définit les couleurs des paires de pixels [TC90, TIW94, Ges, Pol04]. La texture peut être une image de texture classique mais il n'est pas trivial d'en choisir une qui produira un autostéréogramme de bonne qualité. Une solution est d'utiliser une image de points aléatoires, ce qui produit généralement des autostéréogrammes satisfaisants (voir Figure 28, à la fin du papier).

Pour calculer un autostéréogramme, il suffit de parcourir une ligne de pixels et d'affecter au pixel courant la couleur d'un pixel précédent dans la ligne, selon l'écart donné par l'image de profondeur [Ges] (voir Figure 16 et Figure 17).

Cette méthode de calcul n'est pas adaptée à une implémentation parallèle GPU car lors du parcours d'une ligne, le calcul du pixel courant nécessite que tous les pixels précédents de la ligne aient été calculés. Ainsi, même en calculant toutes les lignes en parallèle, il faut calculer toutes les colonnes de pixels séquentiellement, ce qui revient à effectuer de l'ordre de 2000 passes de rendu pour une image Full-HD.

3.4. Calcul parallèle

Petz et al. [PGM03] et Policarpo [Pol04] ont proposé des implémentations parallèles GPU pour calculer des autostéréogrammes. La méthode sous-jacente consiste à calculer des bandes verticales successives où les pixels d'une bande dépendent uniquement des pixels des bandes précédentes et peuvent donc être calculés en parallèle. Ainsi, avec des bandes de 100 pixels de large, il suffit alors de 20 passes de rendu pour calculer une image Full-HD.

Dans le calcul séquentiel, un pixel (x,y) est calculé en recopiant un pixel précédent dans la ligne, à un écart $\alpha D[x,y]$ où α est un paramètre de l'algorithme (facteur de profondeur) et D l'image de profondeur à valeurs dans $[0,1]$. Ainsi, l'écart entre deux pixels gauche et droit est compris dans $[0,\alpha]$.

Le principe du calcul parallèle est d'utiliser un écart $W_s + \beta D[x,y]$ où W_s (largeur de bande) et β (facteur de profondeur) sont des paramètres de l'algorithme. Ceci limite la profondeur représentable à $[W_s, W_s+\beta]$ (c'est-à-dire une profondeur minimale non nulle) mais assure qu'un pixel ne dépend pas de ses W_s voisins précédents. Par conséquent, on peut calculer une bande de W_s pixels en parallèle (voir Figure 18, Figure 19 et Algorithme 1).

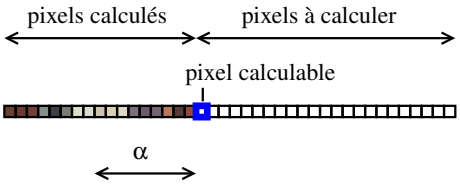


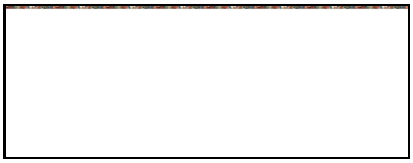
Figure 16: *Calcul séquentiel (de gauche à droite) d'un autostéréogramme. On affecte au pixel courant (en bleu) la couleur d'un pixel précédent de la ligne, à un écart compris dans* $[0,\alpha]$.

1. Calculer l'image de profondeur D
   où $D : \mathbb{N}^2 \to [0,1]$

2. Initialiser les premiers pixels de la ligne : $x < \alpha$
   où $\alpha \in [0, e_p]$ (facteur de profondeur)

3. Calculer séquentiellement les pixels de la ligne :
   $S[x,y] \leftarrow S[x-\alpha D[x,y],y]$

4. Passer à la ligne suivante



5. Image finale

Figure 17: *Calcul séquentiel (de gauche à droite) d'un autostéréogramme. Le calcul d'un pixel nécessite que tous les pixels précédents de la même ligne aient été calculés. Les lignes peuvent être calculées en parallèle.*

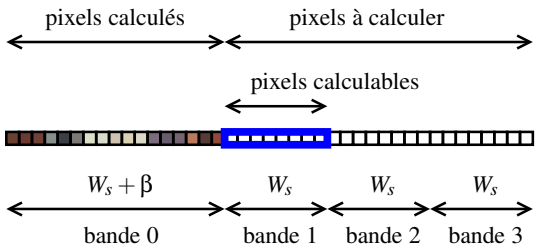
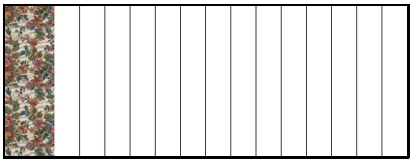


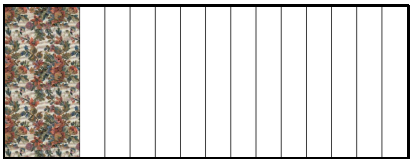
Figure 18: *Calcul parallèle d'un autostéréogramme par bandes successives (de gauche à droite). On affecte à un pixel la couleur d'un pixel précédent de la ligne, à un écart compris dans* $[W_s, W_s+\beta]$. *On peut donc calculer en parallèle une bande de* W_s *pixels (en bleu).*

1. Calculer l'image de profondeur D

2. Initialiser la première bande verticale : $x < W_s+\beta$
   où $\beta \in [0, e_p - W_s]$ (facteur de profondeur)



3. Calculer en parallèle les pixels de la bande suivante :
   $S[x,y] \leftarrow S[x-W_s-\beta D[x,y],y]$



4. Image finale

Figure 19: *Calcul parallèle d'un autostéréogramme par bandes successives (de gauche à droite). Tous les pixels d'une même bande sont calculés en parallèle.*

Input :
$T : [0, W_T[\times [0, H_T[\rightarrow [0,1]^3$ (image de texture)
$D : [0, W_D[\times [0, H_D[\rightarrow [0,1]$ (image de profondeur)
$W_s \in]0, e_p[$ (largeur des bandes)
$\beta \in]0, e_p - W_s[$ (facteur de profondeur)

Output :
$S : [0, W_D[\times [0, H_D[\rightarrow [0,1]^3$ (autostéréogramme)

begin
  {calcule la première bande verticale}
  {section parallélisable}
  for x ***from*** 0 ***to*** $W_s + \beta - 1$ **do**
    for y ***from*** 0 ***to*** $H_D - 1$ **do**
      $x' \leftarrow (x - \beta D[x,y]) \mod W_T$
      $y' \leftarrow y \mod H_T$
      $S[x,y] \leftarrow T[x',y']$
    end
  end
  {calcule les bandes restantes}
  for x_s ***from*** $W_s + \beta$ ***to*** $W_D - 1$ ***step*** W_s **do**
    {section parallélisable}
    for x ***from*** x_s ***to*** $\min(x_s + W_s - 1, W_D - 1)$ **do**
      for y ***from*** 0 ***to*** $H_D - 1$ **do**
        $x' \leftarrow x - W_s - \beta D[x,y]$
        $S[x,y] \leftarrow S[x',y]$
      end
    end
  end
end

Algorithme 1 : Calcul par bandes d'un autostéréogramme

Pour calculer un autostéréogramme, on n'utilise pas directement l'équation 3 mais plutôt une approximation linéaire, ce qui permet de simplifier les calculs sans réellement dégrader le résultat final. En effet, si on se limite, par exemple, à $e'_p \in [100, 150]$ (ce qui correspond à $W_s = 100$ et $\beta = 50$), alors on peut simuler une profondeur $f' \in [74, 175]$ avec l'approximation illustrée Figure 20.

4. Intégration dans une scène 3D

4.1. Problématique

Habituellement, un autostéréogramme est calculé pour être directement affiché. L'implémentation GPU du calcul se résume donc simplement à faire le rendu d'une surface correspondant à l'écran d'affichage (voir Figure 21).

Ici, l'idée est d'intégrer un autostéréogramme dans une scène 3D virtuelle (par exemple, dans une application 3D interactive). L'image ainsi obtenue est alors multi-scène car l'utilisateur peut toujours percevoir la scène 3D principale mais également la scène contenue dans l'autostéréogramme, en focalisant son regard derrière l'écran d'affichage.

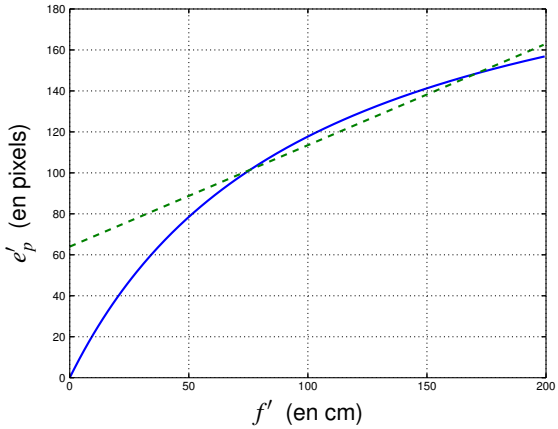


Figure 20: *Écart entre les deux pixels (gauche et droit) en fonction de la profondeur simulée (équation 3) pour les valeurs données Figure 15 (écran Full-HD de 21 pouces placé à 1 m de l'utilisateur). En pointillé, l'approximation linéaire utilisée en pratique quand on se limite à $e'_p \in [100, 150]$ (ce qui correspond à $f' \in [74, 175]$).*

Pour réaliser cela, il faut calculer un autostéréogramme que l'on utilise ensuite dans la scène 3D pour texturer un objet. Le problème est que l'autostéréogramme doit être calculé dans l'espace écran (car c'est au niveau de l'écran d'affichage que se produit l'effet de focalisation des yeux). Or, c'est le rendu de la scène 3D qui permet de passer de l'espace objet vers l'espace écran. En d'autres termes, on doit faire un rendu de la scène 3D avant de pouvoir calculer l'autostéréogramme mais on doit avoir l'autostéréogramme pour pouvoir faire le rendu de la scène 3D complète.

4.2. Méthode proposée

La méthode proposée pour intégrer un autostéréogramme dans une scène 3D consiste à transformer les éléments nécessaires pour calculer l'autostéréogramme (image de profondeur et image de texture) dans l'espace écran. On y applique ensuite la méthode de calcul classique puis on fait un rendu de la scène 3D en utilisant l'autostéréogramme comme une texture définie dans l'espace écran.

La méthode est détaillée Figure 22. Elle nécessite une scène et une image de texture pour l'autostéréogramme, et une scène 3D principale accueillant l'autostéréogramme. Le rendu final correspond à l'image multi-scène contenant la scène 3D et la scène de l'autostéréogramme.

Les passes de rendu à réaliser sont les suivantes :
— calculer l'image de profondeur à partir de la scène qui doit être visible dans l'autostéréogramme ;
— projeter l'image de profondeur dans l'espace écran

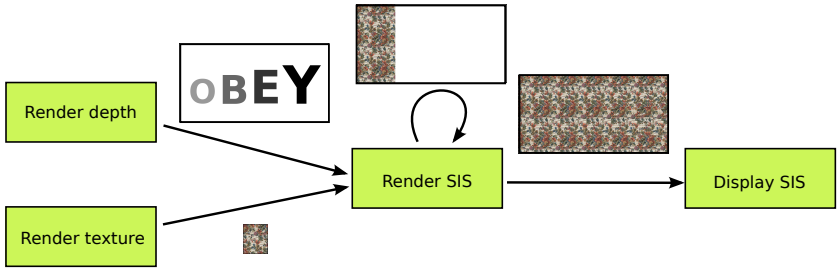


Figure 21: *Pipeline de rendu permettant d'implémenter le calcul d'un autostéréogramme par bandes, sur GPU (Algorithme 1). Chaque flèche correspond à une passe de rendu. Selon l'application visée, l'image de profondeur et l'image de texture peuvent être précalculées.*

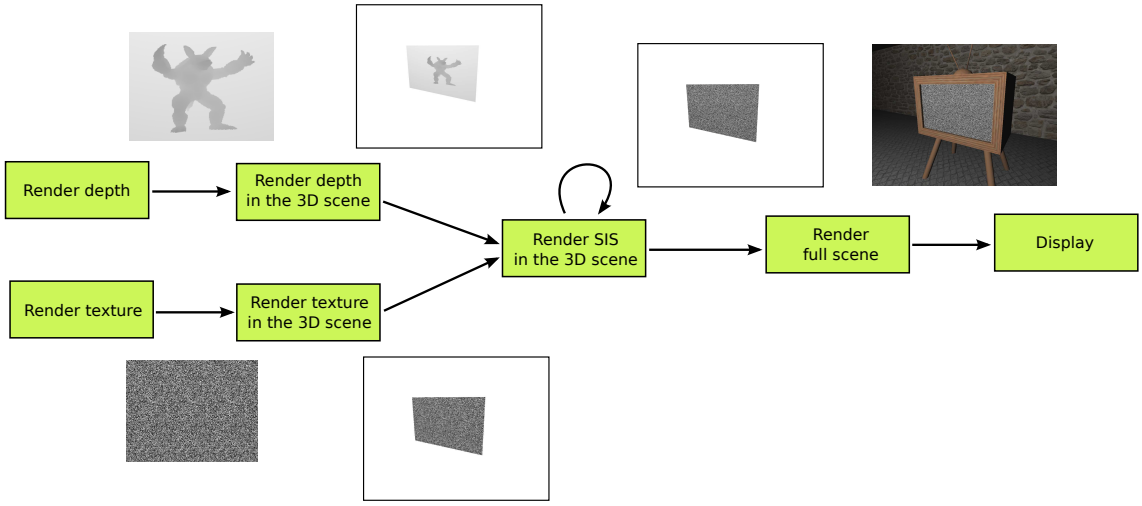


Figure 22: *Pipeline de rendu GPU permettant d'intégrer un autostéréogramme dans une scène 3D. Chaque flèche correspond à une passe de rendu. Lors du rendu final, l'autostéréogramme est utilisé comme une texture définie dans l'espace écran.*

(c'est-à-dire faire un rendu de l'objet 3D dans la scène principale, texturé avec cette image) ;
— calculer l'image de texture à utiliser pour l'autostéréogramme ;
— projeter l'image de texture dans l'espace écran ;
— calculer l'autostéréogramme dans l'espace écran en utilisant l'algorithme 1 (multi-passe) ;
— calculer le rendu de la scène 3D principale complète (en utilisant l'autostéréogramme comme texture dans l'espace écran pour l'objet 3D correspondant).

4.3. Extensions

Le pipeline de rendu proposé Figure 22 permet d'implémenter quelques extensions intéressantes. Par exemple, calculer l'image de texture dans une passe de rendu permet de réaliser des textures procédurales ou des textures animées.

De même, l'image de profondeur est calculée dynamiquement, à partir d'une scène 3D virtuelle. Il est donc possible d'animer cette scène (et donc la forme en relief qui sera visible dans l'autostéréogramme final).

Il est également possible de faire interagir la scène de l'autostéréogramme avec la scène 3D principale. Par exemple, on peut récupérer la position de l'utilisateur dans la scène principale et s'en servir comme transformation géométrique pour le rendu de l'image de profondeur. Typiquement, l'utilisateur peut tourner autour de la scène visible dans l'autostéréogramme en tournant autour de l'objet correspondant dans la scène principale.

Enfin, la texture que constitue l'autostéréogramme dans la scène finale peut être utilisée de différentes façons dans le rendu final. Notamment, les calculs d'éclairage peuvent être appliqués sans perturber la perception de l'autostéréogramme (dans la limite où ils n'introduisent pas de variation haute-fréquence).

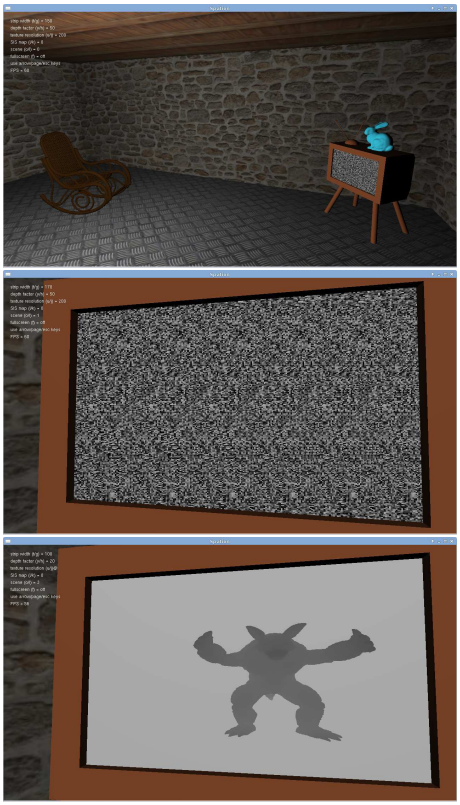


Figure 23: *Intégration d'un autostéréogramme basé points aléatoires dans une scène 3D. En haut : la scène principale (un poste de TV dans un "salon"). Au milieu : utilisation d'un autostéréogramme comme image du poste de TV. En bas : l'image de profondeur de l'autostéréogramme. Voir l'agrandissement Figure 29.*

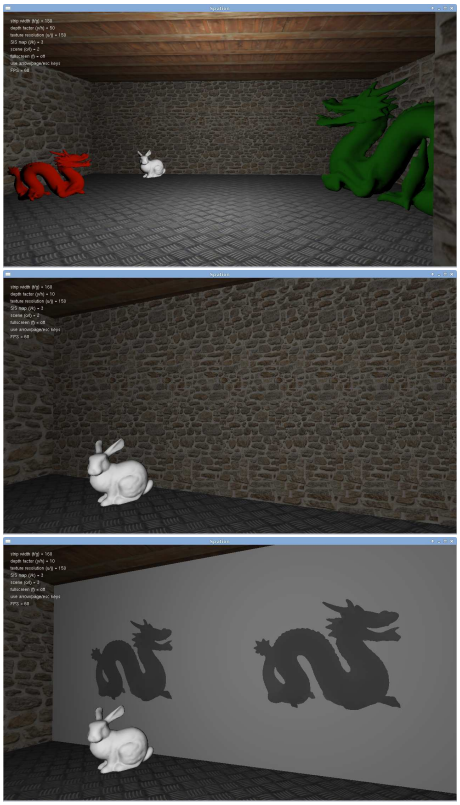


Figure 24: *Intégration d'un autostéréogramme basé texture dans une scène 3D. En haut : la scène principale. Au milieu : utilisation d'un autostéréogramme comme texture d'un mur. En bas : l'image de profondeur de l'autostéréogramme. Voir l'agrandissement Figure 30.*

5. Résultats

Nous avons modifié une application de promenade virtuelle 3D pour y intégrer des autostéréogrammes basés points ou textures (figures 23 à 26). L'implémentation a été réalisée en OpenGL et testée sur un GPU Nvidia Quadro 2000M. Avec les scènes de test et paramétrages utilisés, nous avons toujours obtenu un rendu Full-HD temps-réel (plus de 30 images par seconde). Le code source est disponible à l'adresse suivante :

`http://www-lisic.univ-littoral.fr/~dehos/recherche/spation`

Cette application réalise bien un affichage multi-scène. En effet, nous avons une scène principale où l'autostéréogramme apparaît comme une texture classique (plaquée sur un objet et respectant l'éclairage), et une scène secondaire, visible dans l'autostéréogramme en focalisant le regard derrière l'écran d'affichage. Comme l'autostéréogramme est calculé dans l'espace écran, la scène secondaire est visible quelle que soit la position de l'utilisateur dans la scène principale (du moment qu'il reste raisonnablement proche de l'autostéréogramme).

La première scène de l'application est présentée Figure 23. Ici, un autostéréogramme basé points aléatoires est utilisé pour représenter la neige électronique d'un poste de TV et cache une scène du modèle 3D Armadillo. L'image de neige est calculée dynamiquement lors du rendu de l'image de texture à utiliser pour l'autostéréogramme.

La scène présentée Figure 24 utilise un autostéréogramme basé texture : la texture du mur, derrière le lapin est en fait un autostéréogramme. Dans la scène principale, cette texture semble plutôt réaliste si bien que l'autostéréogramme passe presque inaperçu. Évidemment, lorsque l'utilisateur se déplace, l'autostéréogramme est mis à jour et l'utilisateur se rend compte que cette texture cache quelque chose.

Enfin, la Figure 25 illustre l'interaction possible entre la scène principale et la scène de l'autostéréogramme. Ici, la scène secondaire est orientée en fonction de la position de l'utilisateur dans la scène principale. Ainsi, lorsque l'utilisa-

Figure 26: *Utilisation de la méthode pour réaliser une énigme humoristique : pour qu'il goûte la farce, l'utilisateur doit passer entre les caisses, regarder dans le fût et y mater l'abysse.*

teur se déplace autour du sol (scène principale), il se déplace également autour du Armadillo au niveau de l'autostéréogramme (scène secondaire).

6. Discussion

La méthode proposée pour intégrer un autostéréogramme dans une scène 3D peut avoir différentes applications : donner un effet artistique, réaliser une énigme dans un jeu (voir Figure 26)... Cependant, elle souffre de quelques limitations issues de l'utilisation d'un autostéréogramme ou du fait que celui-ci est calculé au niveau de l'écran puis plaqué dans la scène 3D. En effet, comme le calcul de l'autostéréogramme revient à réutiliser des pixels dans l'image affichée à l'écran, les effets de perspective prononcés ne peuvent pas être correctement représentés (voir Figure 27). Pour la même raison, l'autostéréogramme calculé doit être plaqué sur une surface relativement plane de la scène principale.

Autre limitation, lorsque l'autostéréogramme est utilisé pour le rendu de la scène principale, il ne doit pas subir d'effet "haute-fréquence". Le cas typique est le calcul d'éclairage : un ombrage diffus sur l'autostéréogramme ne perturbera pas la perception de la scène qui y est cachée mais une ombre dure sera gênante.

Enfin, l'algorithme de calcul d'autostéréogramme utilisé ici fonctionne selon une direction privilégiée (de la gauche vers la droite). Ceci implique des rendus différents selon la direction d'observation, notamment dans les situations de forte perspective.

Figure 27: *Les principales limitations de la méthode proposée sont dues au fait que l'autostéréogramme doit être calculé au niveau de l'écran d'affichage avant d'être utilisé dans la scène 3D, ce qui perturbe notamment le rendu de la perspective.*

7. Conclusion

Dans ce papier, nous avons proposé une méthode qui utilise des autostéréogrammes pour calculer des images multi-scènes. Tout d'abord, nous avons rappelé le principe des autostéréogrammes et comment les calculer. Puis nous avons détaillé des algorithmes de calcul, séquentiel et parallèle multi-passe, utilisables en pratique. Enfin, nous avons proposé une méthode permettant d'étendre l'utilisation des autostéréogrammes pour réaliser des images multi-scènes.

La méthode proposée permet d'intégrer des autostéréogrammes dans une scène 3D classique. Ainsi, dans l'image finale, on peut percevoir à la fois la scène 3D principale et la scène représentée dans l'autostéréogramme. La perception de l'une ou l'autre de ces deux scènes dépend de la façon de regarder l'image (en focalisant le regard sur l'écran d'affichage ou derrière celui-ci). Le principe de cette méthode est de réaliser le calcul de l'autostéréogramme au niveau de l'écran d'affichage puis d'utiliser l'image obtenue comme une texture dans la scène 3D. Une implémentation multi-passe sur GPU permet d'obtenir un rendu Full-HD en temps-réel.

La principale limitation de la méthode est que le rendu final n'est plus cohérent lorsque les deux espaces de calculs (écran et scène 3D) sont trop différents (typiquement, lorsque la surface texturée avec l'autostéréogramme n'est plus suffisamment parallèle à l'écran). Il serait intéressant, dans des futurs travaux, de modifier la méthode de calcul de l'autostéréogramme ou d'adapter dynamiquement certains paramètres pour compenser le problème.

8. Remerciements

Les auteurs tiennent à remercier les relecteurs pour leurs remarques à propos du manuscrit et pour avoir mis tant de coeur dans la lecture de son contenu.

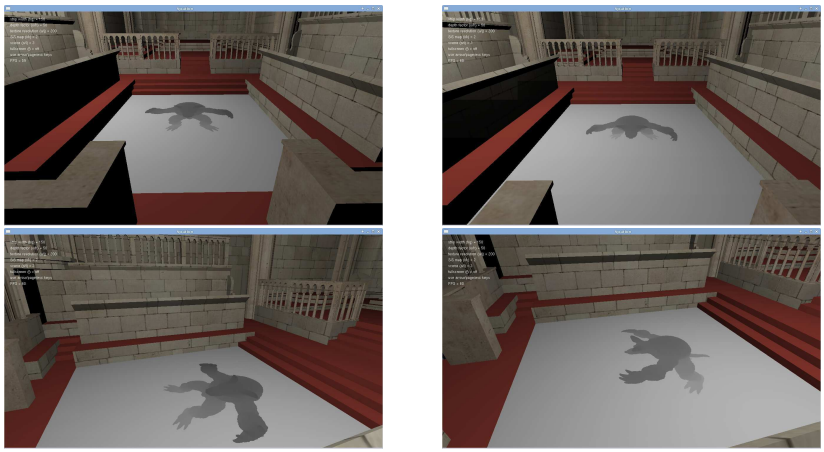


Figure 25: *Utilisation de la position du point de vue dans la scène principale pour orienter la scène de l'autostéréogramme. Colonne de gauche : sans interaction, la scène secondaire (image de profondeur) apparait comme une image plaquée au sol. Colonne de droite : avec interaction, la scène secondaire est perçue comme une scène 3D.*

Références

[3Di] 3DIMKA : Chess single-image-stereogram. http://3dimka.deviantart.com.

[Cas02] CASATI R. : *La Découverte de l'ombre*. Albin Michel, 2002.

[CHM*10] CHU H.-K., HSU W.-H., MITRA N. J., COHEN-OR D., WONG T.-T., LEE T.-Y. : Camouflage images. *ACM Transactions on Graphics. Vol. 29*, Num. 3 (2010).

[EO11] ERICSSON F., OLWAL A. : Interaction and rendering techniques for handheld phantograms. In *ACM CHI Conference on Human Factors in Computing Systems* (2011), pp. 1339–1344.

[Ges] GESELOWITZ L. : The AbSIRD Project : To create real-time SIRDs. http://www.lewcid.com/lg/download/SIRD/AbSIRD/essay.html.

[GSP*07] GAL R., SORKINE O., POPA T., SHEFFER A., COHEN-OR D. : 3D collage : Expressive non-realistic modeling. In *ACM NPAR Non-Photorealistic Animation and Rendering* (2007), pp. 7–14.

[HZZ11] HUANG H., ZHANG L., ZHANG H.-C. : Arcimboldo-like collage using internet images. *ACM Transactions on Graphics. Vol. 30*, Num. 6 (2011), 1–8.

[IDW*13] ISENBERG P., DRAGICEVIC P., WILLETT W., BEZERIANOS A., FEKETE J.-D. : Hybrid-image visualization for large viewing environments. *IEEE Transactions on Visualization and Computer Graphics. Vol. 19*, Num. 12 (2013), 2346–2355.

[MCL*09] MITRA N. J., CHU H.-K., LEE T.-Y., WOLF L., YESHURUN H., COHEN-OR D. : Emerging images. *ACM Transactions on Graphics. Vol. 28*, Num. 5 (2009), 1–8.

[MP09] MITRA N. J., PAULY M. : Shadow art. *ACM Transactions on Graphics. Vol. 28*, Num. 5 (2009).

[N.E93] N.E. THING ENTERPRISES (Ed.) : *Magic eye : A new way of looking at the world*. Andrews And McMeel Publishing, 1993.

[OTS06] OLIVA A., TORRALBA A., SCHYNS P. G. : Hybrid images. *ACM Transactions on Graphics. Vol. 25*, Num. 3 (2006), 527–532.

[PGM03] PETZ C., GOLDLUCKE B., MAGNOR M. : Hardware-accelerated autostereogram rendering for interactive 3d visualization. In *SPIE Stereoscopic Displays and Virtual Reality Systems X* (2003), vol. 5006.

[Pol04] POLICARPO F. : *GPU Gems : Programming Techniques, Tips and Tricks for Real-Time Graphics, Chapter 41 : Real-Time Stereograms*. Pearson Higher Education, 2004.

[TC90] TYLER C. W., CLARKE M. B. : The autostereogram. In *SPIE Stereoscopic display and applications* (1990), vol. 1256, pp. 192–197.

[TIW94] THIMBLEBY H. W., INGLIS S., WITTEN I. : Displaying 3d images : Algorithms for single image random dot stereograms. *IEEE Computer. Vol. 27* (1994), 38–48.

[VRC*13] VANGORP P., RICHARDT C., COOPER E. A., CHAURASIA G., BANKS M. S., DRETTAKIS G. : Perception of perspective distortions in image-based rendering. *ACM Transactions on Graphics. Vol. 32*, Num. 4 (2013), 1–12.

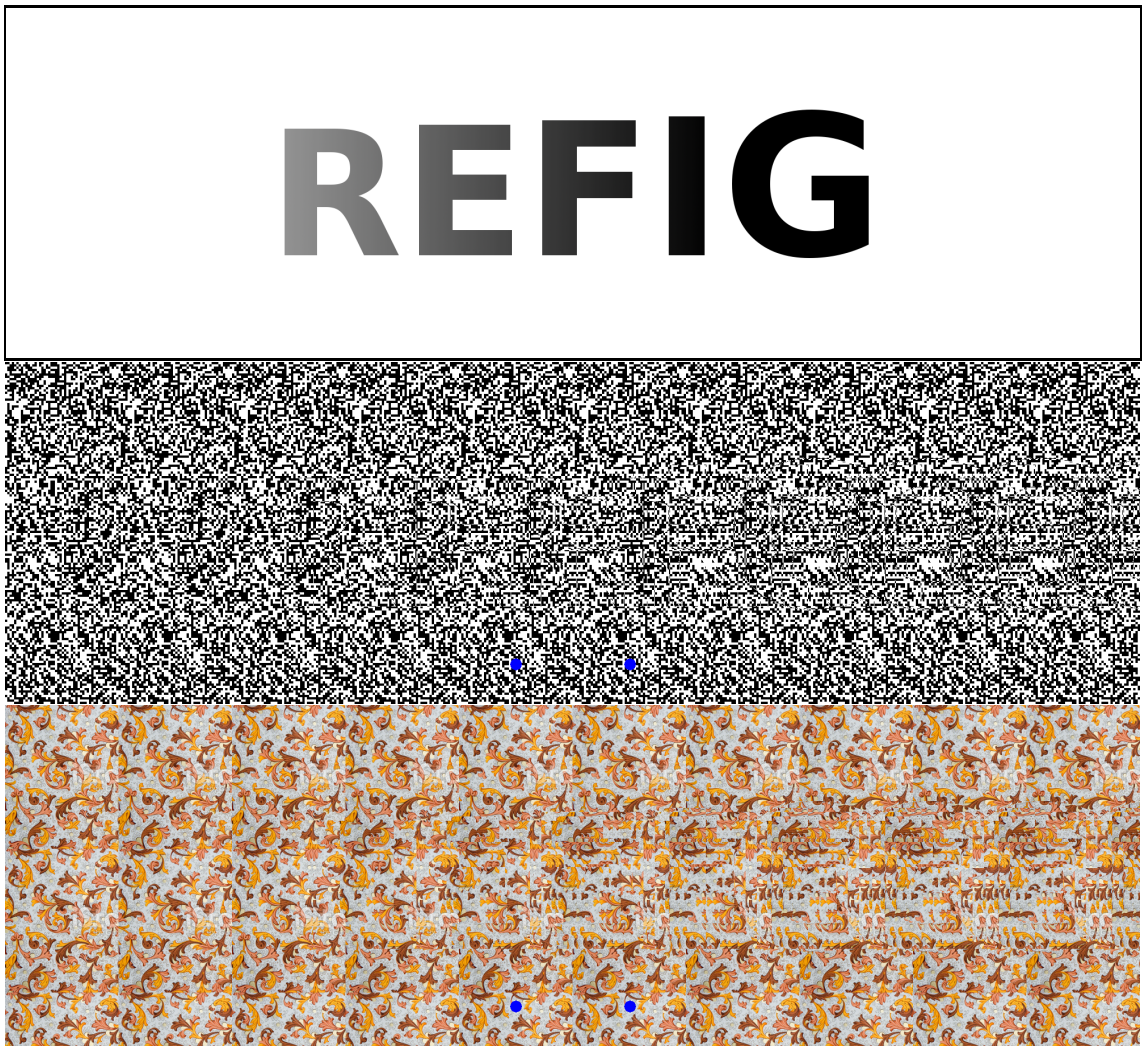


Figure 28: *Exemple d'autostéréogramme : image de profondeur (haut), autostéréogramme basé points (milieu), autostéréogramme basé texture (bas). Pour visualiser un autostéréogramme, il faut focaliser le regard derrière le plan image réel ; on perçoit alors le relief correspondant à l'image de profondeur.*

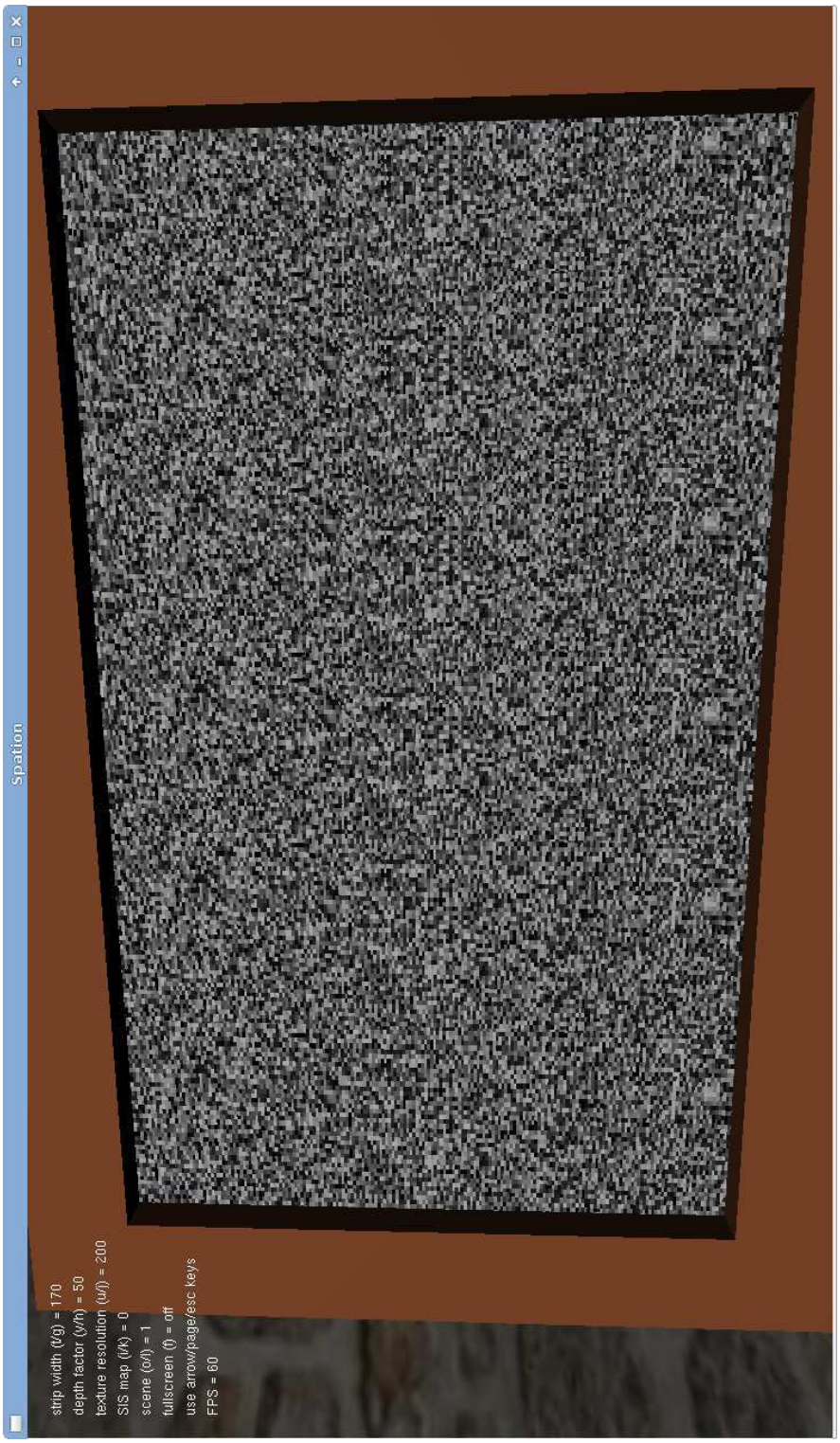


Figure 29: *Agrandissement de la Figure 23.*

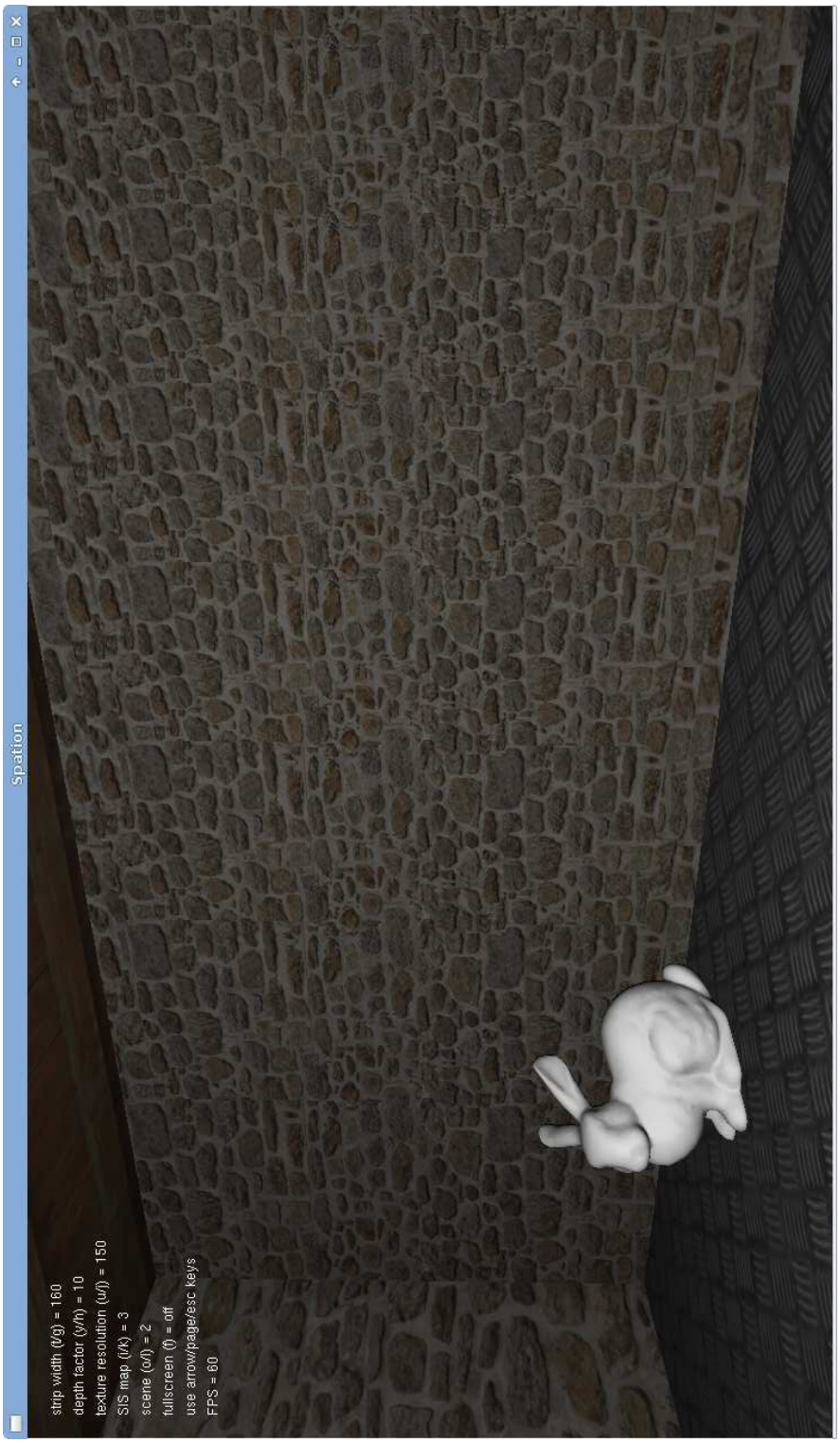


Figure 30: *Agrandissement de la Figure 24.*