

Rapport du TP5 du module GPU

Apporter du relief sur une texture, techniques de *mapping*

Antoine BRALET et Guillaume DURET

06 Novembre 2020

1 Introduction

Ce rapport vise à expliquer notre démarche ainsi que nos résultats lors du TP n°5 du module de GPU. Celui-ci visait à donner du relief à une texture afin qu'elle soit la plus réaliste possible. Pour se faire plusieurs solutions vont être mises en œuvre dans ce TP et qui ne sont pas forcément incompatibles les unes avec les autres. Les compatibilités de chacune de ces méthodes seront bien sûr détaillées au fur et à mesure du rapport au sein de chaque partie. Néanmoins nous pouvons d'ores et déjà citer deux grandes méthodes distinctes et qui donnent chacune des résultats très intéressants : la modification de la géométrie et l'adaptation de texture. Ces deux méthodes sont décrites ci-après. Néanmoins, notre TP visant à être le plus réaliste petit à petit, nous avons suivi la ligne directrice suivante : mettre en place une illumination de Phong afin d'avoir des effets de lumières sur la texture, modifier le géométrie à l'aide de cartes de hauteurs pour avoir des effets de reliefs et modifier les normales en chaque point pour avoir un effet de lumières plus réaliste jusqu'à utiliser une carte d'occultation ambiante permettant d'ajouter des ombres dans les endroits cachés et difficiles d'accès (au sens de la lumière). Enfin nous étudierons aussi le fonctionnement et l'effet de l'adaptation de texture permettant d'ajouter du relief par simple illusion d'optique en choisissant bien les coordonnées de textures en fonction de l'emplacement de la caméra.

Toutes ces expériences seront réalisées en appliquant une texture sur un carré sur lequel une grille de 100x100 a été disposée afin d'ajouter de la complexité à la forme et pouvoir la modeler. Il semble important ici de noter que tout changement de taille de la grille sera notifié au besoin dans le rapport. Si tel n'est pas le cas nous considérerons que la grille de 100x100 fait référence. De plus la texture appliquée à l'objet ne sera pas la même pour les modifications géométriques et l'adaptation de texture, vous trouverez en Figure 1 la texture respectivement gauche et droite utilisée pour ces méthodes.

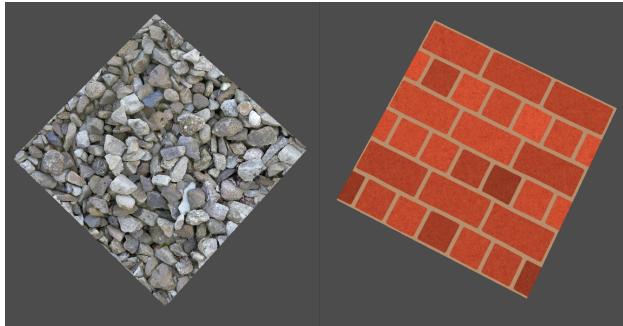


Figure 1: De gauche à droite la texture utilisée lors de la phase de modification de géométrie et la texture de la phase d'adaptation de texture.

2 Illumination de Phong

Pour la simulation de la lumière, l'illumination de Phong est utilisée. Cette illumination est une somme pondérée de 3 illuminations différentes comme cela est illustré dans la Figure 2

En prenant un point p de couleur c , il est possible d'évaluer les 3 illuminations grâce aux formules (1),(2),(3) et ainsi obtenir la couleur résultante c_{phong} avec l'équation (4) telle que les vecteurs sont définis dans la Figure 3

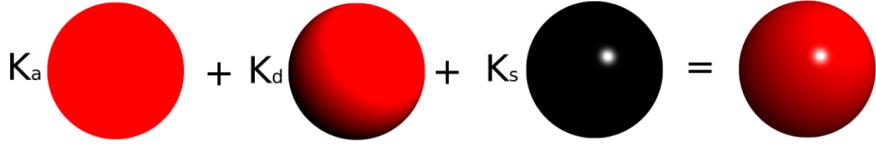


Figure 2: Exemple d'illumination dans le cas d'une sphère. Les 3 illuminations : ambiante, diffuse et spéculaire sont montrées séparément. L'image finale étant obtenue comme la somme pondérée de ces 3 termes.

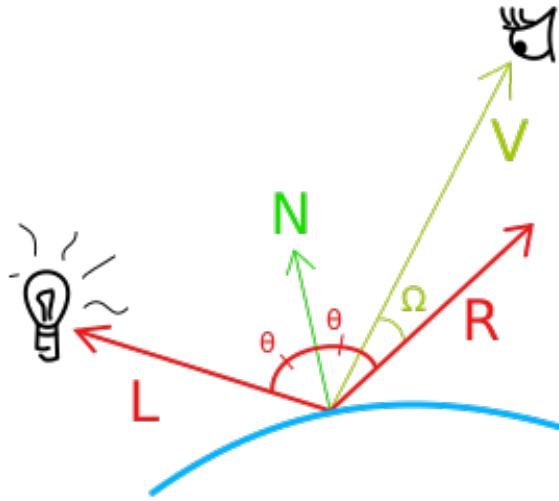


Figure 3: Définition des différents vecteurs. On note \vec{u}_L le vecteur unitaire pointant de p vers la source de lumière, \vec{r} le symétrique de \vec{u}_L par rapport à la normale \vec{n} et \vec{v} le vecteur unitaire pointant de p vers la caméra.

$$I_a = K_a \quad (1)$$

$$I_d = K_d \langle \vec{n}, \vec{u}_L \rangle_{[0,1]} \quad (2)$$

$$I_s = K_s \langle \vec{r}, \vec{v} \rangle_{[0,1]}^{e_s} \quad (3)$$

$$c_{phong} = (I_a + I_d)c + I_s \quad (4)$$

Les paramètres choisis pour cette illumination sont $K_a = 0.2$, $K_d = 0.8$, $K_s = 0.6$ et $e_s = 256$ qui permettent d'obtenir la Figure 4 sur laquelle un halo de lumière créé par l'illumination est remarquable sur la texture.

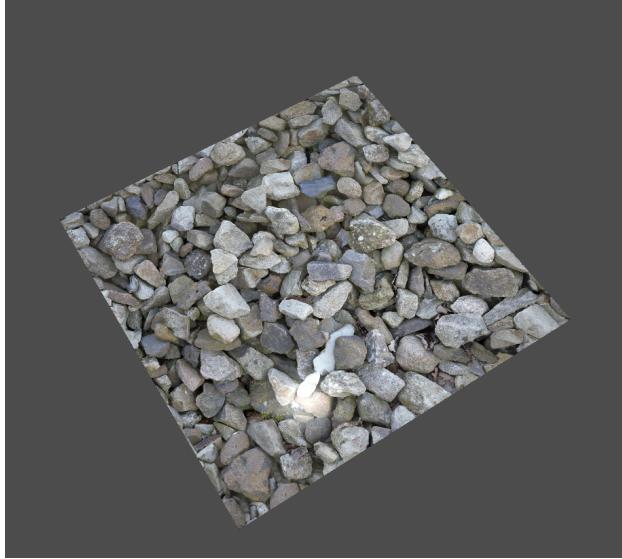


Figure 4: Résultat de l'illumination de Phong sur la texture avec une source de lumière située au point $(0,2,0)$ en coordonnées mondes

3 Modification de géométrie, *Height Maps*

La première méthode que nous mettrons en place afin de donner du relief à une texture consiste en une modification de la géométrie de la grille de référence. Effectivement, qu'y a-t-il de plus simple pour créer du relief que de modifier la composante "y" de la géométrie afin qu'elle soit bien définie comme une forme en trois dimensions et non pas simplement un plan en deux dimensions ? Pour ce faire il est alors nécessaire d'avoir une carte de hauteurs (*Height map* en anglais) permettant de renseigner pour chaque coordonnées de texture la hauteur normalisée de la texture. Il semble important de noter que l'illumination de Phong est parfaitement compatible avec cette méthode.

Aussi simple que ceci puisse paraître, la carte des hauteurs est, elle-même une texture que nous devions envoyer au GPU afin qu'il puisse avoir les données de hauteurs et pour que nous puissions modifier la hauteur de la grille au sein même du vertex shader. Pour ce faire nous avons alors dû manipuler les indices de textures ainsi que comprendre la gestion de texture en mémoire GPU :

Chaque texture en mémoire GPU est stockée dans l'espace de *texture active*. Afin de sélectionner l'unité dans laquelle nous souhaitons que la texture soit mémorisée, il suffit de renseigner un numéro qui correspond à l'indice mémoire. Cet entier est fixé par l'utilisateur. Une fois l'indice choisi et la texture mémorisée par le GPU, il suffit alors de récupérer le *GLuint* correspondant à la localisation de la variable de la texture dans les shaders puis de lui attribuer l'indice de l'unité ayant mémorisé la texture afin que la variable soit directement associée à la texture que nous avons chargée sur le GPU. En procédant de cette façon, il nous est alors possible de charger plusieurs textures et de s'assurer que celles-ci sont bien associées aux variables mentionnées dans les shaders.

En appliquant cette méthode il nous a alors été possible de passer la carte de hauteurs en mémoire GPU et de l'utiliser à partir des shaders (en l'occurrence le vertex shader ici) permettant de modifier la géométrie de la grille après remise à l'échelle (car la carte des hauteurs est normalisée sur $[0, 1]$) de 0.2. Une telle opération permet d'obtenir les résultats de la Figure 5.



Figure 5: Résultats obtenus après modification de la hauteur des points de la grille afin de créer du relief

On peut alors constater que l'on a bien une géométrie bosselée ce qui était particulièrement attendu. Néanmoins, on peut noter en regardant la frontière entre deux cailloux que celle-ci n'est pas du tout naturelle. Nous avons essayé d'améliorer la résolution de la grille (1000x1000 et 10000x10000) afin de voir l'effet sur le résultat final. Une telle expérience n'a pas amélioré l'effet mentionné précédemment et a allongé le temps de calcul. Aussi améliorer la résolution n'est pas la solution. Cet effet semble particulièrement difficile à régler sans avoir à complexifier la forme de par le fait que la texture est simplement une vue de dessus, aussi il paraît normal que les raccords entre deux pierres soient particulièrement visibles lorsque l'on a une vue de côté.

4 Modification de normales, *Normal Mapping*

Nous cherchons ici à améliorer les résultats du point de vue de la lumière reflétée par l'objet et reçue par la caméra. En effet, le halo lumineux qui se reflète sur la surface avec l'illumination de Phong n'est pas réaliste sur ces cailloux. Ceci est dû au fait que l'ensemble des normales de notre surface pointe à la verticale et par conséquent la lumière se reflète exactement de la même manière sur tous les points de la surface, même si leur position a été modifiée au préalable. Le changement que nous nous apprêtons à décrire ici est compatible avec l'illumination de Phong et la modification de géométrie.

Afin de rendre les effets de lumières plus réalistes, nous allons modifier les normales en chaque point de la grille de la surface. Sachant que ces normales dépendent directement de la texture qui est appliquée à notre objet, il nous est nécessaire de nous référer à la carte de normales spécifique de notre texture. Tout comme la carte des hauteurs, cette carte de normales est enregistrée sous la forme d'une texture, ainsi les mêmes modifications algorithmiques décrites précédemment nous permettent de faire passer cette carte de normales dans la mémoire GPU afin que nous ayons accès à elle depuis les shaders.

Néanmoins, une subtilité subsiste. Non seulement les cartes de normales ont des valeurs dans l'intervalle $[0, 1]$ ce qui implique que nous devons réadapter les valeurs afin qu'elles parcourent l'intervalle $[-1, 1]$ mais en plus elles ne sont pas enregistrées dans l'espace monde, espace ambiant dans lequel se situe notre surface, notre caméra et notre lumière. Effectivement ces normales sont enregistrées dans l'espace tangent du point en lequel elles ont été calculées : chaque normale a un espace différent. Ainsi pour pouvoir repasser dans l'espace monde, il est nécessaire de connaître la matrice de passage qui nous permet de passer de l'espace tangent à l'espace monde. Cette matrice est définie comme telle :

$$M = (T \quad B \quad N) \quad (5)$$

avec T la tangente en le point, B sa bitangente et N sa normale sous forme de vecteur colonne dans l'espace monde et comme défini sur la Figure 6.

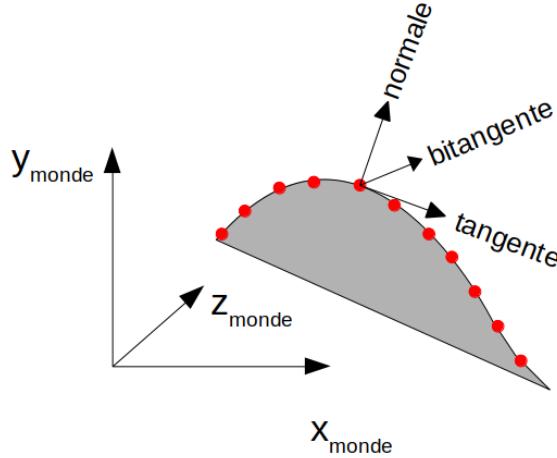


Figure 6: Illustration de la position des axes du repère tangent d'un point dans le repère monde. Le repère tangent en un point est composé par la tangente en ce point, sa bitangente et sa normale.

Ainsi donc, pour passer un vecteur \vec{v} de l'espace tangent à l'espace monde, il suffit de multiplier \vec{v} à droite de M pour obtenir les coordonnées dans l'espace monde. Cependant, en pratique, il s'avère plus efficace de passer l'ensemble des objets (lumière, caméra, ...) dans l'espace tangent afin de faire les calculs. Sachant alors que la matrice M de l'équation 5 est orthonormée car formée par trois vecteurs formant une base orthonormée de l'espace, alors la matrice inverse de M est sa propre transposée. Ainsi :

$$M^{-1} = M^T = \begin{pmatrix} T^T \\ B^T \\ N^T \end{pmatrix} \quad (6)$$

Cette matrice est appelée "TBN" dans notre code shaders et nous permet donc bien de passer de l'espace objet à l'espace tangent afin que l'ensemble des vecteurs soient bien placés dans le même repère. Ainsi donc, en modifiant la normale en fonction de la valeur présente dans la carte des normales, il est possible d'obtenir le résultat de droite présent dans la Figure 7. On peut ici observer que le réalisme de notre résultat en comparaison avec le résultat précédemment obtenu avec l'illumination de Phong seule (à gauche sur la Figure) est grandement amélioré. Effectivement, le halo lumineux a disparu et les pierres reluisent essentiellement sur le dessus, faisant penser qu'il a plu peu de temps auparavant. À noter que nous avons conservé ici en plus la modification de géométrie pour donner un aspect de relief d'autant plus important.

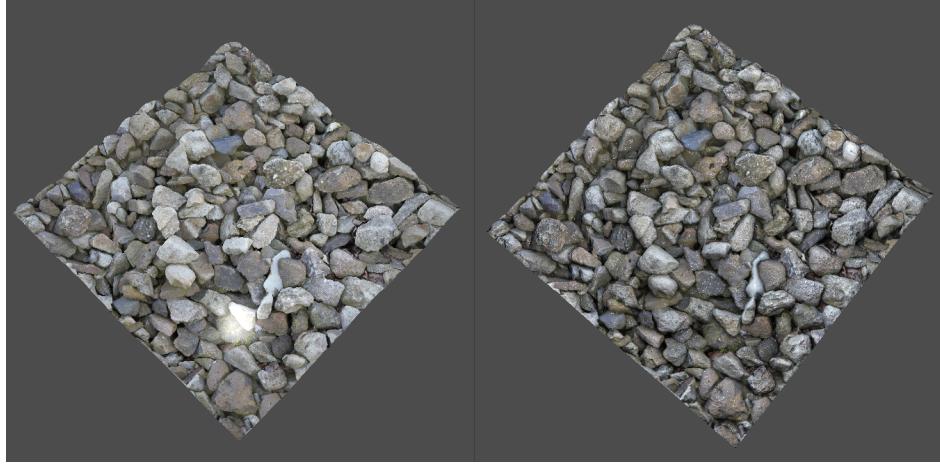


Figure 7: À gauche le résultat obtenu uniquement avec l'illumination de Phong, et à droite le résultat obtenu avec le *Normal Mapping* autrement dit la modification de normales. On peut bien voir que le résultat semble beaucoup plus réaliste avec le *Normal Mapping* puisque le halo lumineux a complètement disparu et les pierres semblent briller comme si celles-ci étaient mouillées.

5 Effets d'ombres et de lumières, *Ambiant occlusion*

Étant donné les bons résultats obtenus à l'étape précédente, nous pourrions arrêter notre recherche ici. Cependant ce résultat est toujours perfectible, et une manière simple et rapide de l'améliorer et de prendre en compte l'occultation ambiante. Effectivement, le résultat précédent considère que chaque point de texture voit approximativement la même dose de lumière provenant du "ciel" de notre monde 3D. Or une pierre située en dessous d'une autre pierre voit beaucoup moins de lumière que celle du dessus. Pour cette raison, il est possible d'utiliser une carte d'occultation ambiante permettant de renseigner si un point de texture est très occulté ou au contraire très peu. Cette information est également compatible avec les trois modifications préalablement mentionnées.

De nouveau, la carte d'occultation ambiante est renseignée sous la forme d'une texture que nous passons en mémoire GPU suivant la méthode algorithmique décrite dans la section 3. Afin de le prendre en compte dans nos calculs, puisqu'il s'agit d'une occultation ambiante nous choisissons de modifier uniquement le terme d'illumination ambiante de l'illumination de Phong. Ainsi donc, le terme d'occultation ambiante étant compris dans l'intervalle $[0, 1]$ et prenant pour valeur 0 si la texture est très occultée et 1 si elle ne l'est pas du tout, en multipliant le terme d'illumination ambiante par le terme d'occultation ambiante, on peut obtenir les résultats présents en Figure 8 qui donne une vue d'ensemble du résultat (à droite) et en Figure 9 où l'on a zoomé sur des zones sombres de l'espace afin de mieux percevoir les changements avec (à droite) et sans (à gauche) l'utilisation de l'occultation ambiante.



Figure 8: Comparaison entre la prise en compte de l'occultation ambiante (à droite) et sans celle-ci (à gauche). Dans les deux cas, l'éclairage de Phong, la modification de géométrie et de normales sont également utilisées. On peut alors bien observer que les zones placées sous les pierres sont plus sombres qu'elles ne l'étaient auparavant.



Figure 9: Zoom sur les zones sombres afin de mieux percevoir les zones plus ombragées lorsque l'occultation ambiante est mise en place (à droite) en comparaison sans sa mise en place (à gauche).

On peut alors observer que les zones de la texture situées sous des pierres sont belles et bien assombries ce qui permet de donner encore plus de profondeur et donc d'ajouter du relief à la texture. On peut en particulier visualiser cet effet sur la Figure 9 dans la zone haute droite des deux images, on peut y voir une zone initialement assez claire à gauche (sans occultation ambiante) devenir bien plus sombre à droite (avec occultation ambiante).

On peut ainsi se retrouver avec une texture de pierre mise en relief grâce à la modification de la géométrie, illuminée naturellement grâce à l'éclairage de Phong et à la modification de normales et enfin cohérente sur les zones d'ombres grâce à l'occultation ambiante. Bien sûr il est toujours possible d'améliorer ce résultat mais il est tout de même particulièrement satisfaisant.

6 Adaptation de textures, *Parallax Mapping*

Néanmoins, il est également possible de procéder d'une autre manière afin de créer du relief sur la texture : l'adaptation de texture à l'aide du *Parallax Mapping*. Si l'illumination de Phong, le *Normal Mapping* et l'occultation ambiante sont parfaitement compatibles avec cette nouvelle méthode, elle n'est par contre pas compatible avec la modification de géométrie. En effet, le *Parallax Mapping* vise à donner une impression de relief en adaptant la texture en fonction de la position de la caméra. Ajouter en plus une modification de géométrie risquerait de complètement annuler l'effet créé par le *Parallax Mapping* voire même rendre la texture complètement incohérente avec la géométrie modifiée.

Ainsi donc nous conservons par la suite l'illumination de Phong et le *Normal Mapping* mais pas l'occultation ambiante car la carte d'occultation ambiante n'est pas disponible dans nos ressources pour le faire. De plus, elle aurait très peu d'impact sur le résultat final puisque nous allons utiliser la texture de droite de la Figure 1 qui est simplement un mur de briques donc il n'y a pas de réel intérêt à l'occultation ambiante puisqu'il n'y a pas réellement de zones "cachées" ici contrairement à la texture précédente.

Il est à noter que malgré le fait que nous ne modifions pas la géométrie, nous allons tout de même nous servir de la carte des hauteurs pour cette méthode. En effet, comme illustré en Figure 10, le *Parallax Mapping* vise à récupérer les coordonnées de texture censées être appliquées en B (projété sur l'objet) et les appliquer au point A afin de donner un effet de relief. Cette modification est particulièrement dépendante de la position de la caméra. Néanmoins, le point B est très difficile à trouver en pratique mais peut être approximé. En effet, on sait que celui-ci se trouve le long du vecteur \vec{v} normalisé. Ainsi en considérant h la hauteur en le point A, il est possible d'opérer une mise à l'échelle du vecteur \vec{v} afin qu'en prenant la texture à cet endroit, on ait une bonne approximation du point B. Ce qui permet d'obtenir l'équation 7 permettant de récupérer les coordonnées de texture p_{xy}^A à appliquer en le point A à partir des coordonnées de texture initiales p_{init}^A . Les mentions "xy" et "z" spécifient que l'on ne se sert que de la coordonnée spécifiée et la variable h_{adapt} est le facteur d'échelle permettant de mettre les hauteurs présentent dans la carte des hauteurs à la bonne échelle par rapport à notre objet.

$$p_{xy}^A = p_{init}^A + (h \cdot \vec{v})_{xy} / \vec{v}_z \quad (7)$$

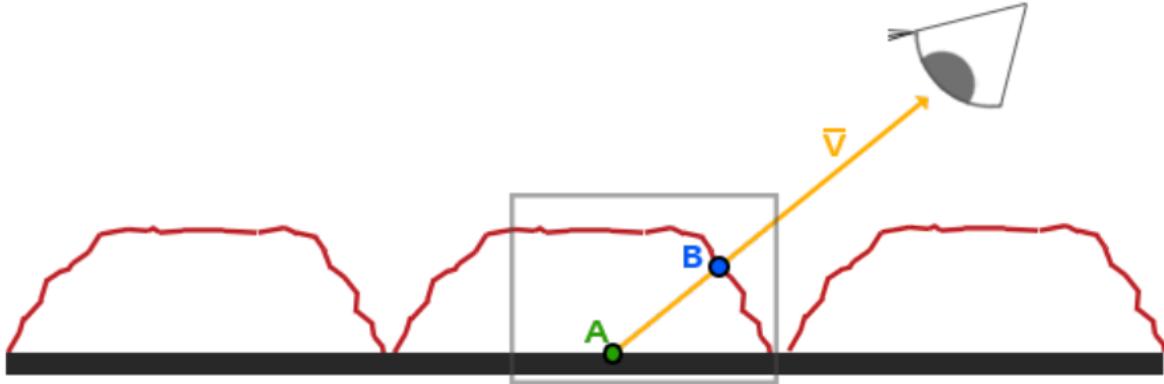


Figure 10: Illustration trouvé sur <https://learnopengl.com/Advanced-Lighting/Parallax-Mapping> du *Parallax Mapping*. On peut y observer un descriptif de la méthode qui consiste à trouver les coordonnées de textures en le point B afin de les attribuer en le point A. La localisation du point B est particulièrement dépendante de la position de la caméra ainsi que de la hauteur en le point A (symbolisée ici par la ligne rouge).

L'équation 7 mentionne la division par la composante en "z" du vecteur \vec{v} . Cette division vise à rendre le résultat plus cohérent et plus réaliste en tenant en compte la direction de la caméra plus encore. Nous

verrons son impact sur les résultats ci-après. Mais commençons tout d'abord par visualiser les résultats obtenus sans cette composante. Il est à noter que l'ensemble des calculs sont effectués dans l'espace tangent comme décrit dans la section 4. La Figure 11 nous permet de comparer les résultats en utilisant l'illumination de Phong et le *Normal Mapping* et avec (à droite) ou sans (à gauche) l'utilisation du *Parallax Mapping*. On peut alors très facilement observer le bon fonctionnement de l'algorithme de par l'impression de relief et de profondeur qu'il s'est formé sur l'image de droite, soit, avec l'utilisation du *Parallax Mapping*. L'effet est d'autant plus notable que l'illumination de Phong et la modification des normales le rendent d'autant plus réaliste.

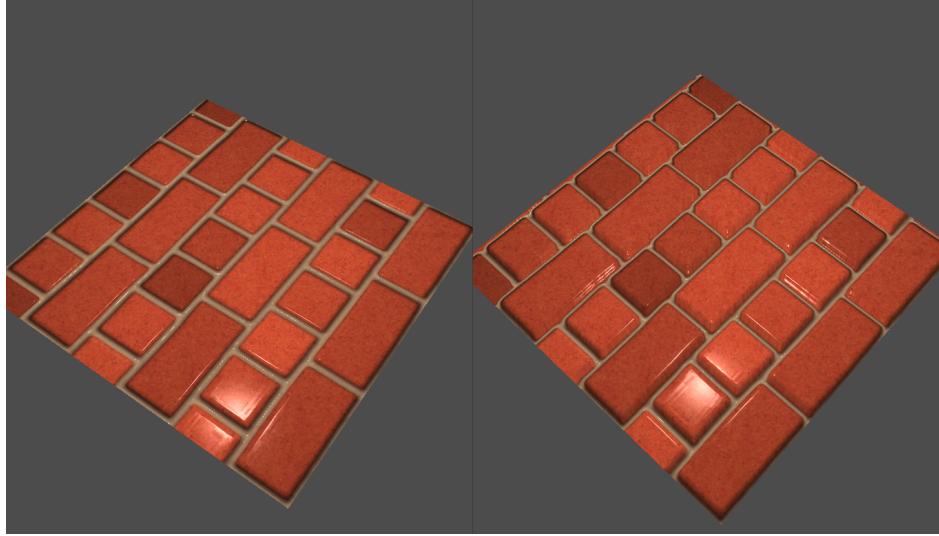


Figure 11: Comparaison avec (à droite) et sans (à gauche) l'utilisation du *Parallax Mapping*. On peut bien visualiser une nette impression de relief par illusion d'optique créé par l'adaptation de texture.

Cependant, un problème apparaît lorsque la caméra devient rasante par rapport à l'objet comme illustré sur la Figure 12. Ce problème est un effet de bord : lorsque la caméra est trop rasante, on peut observer des "coulures" au niveau des intersections. Cet effet peut être réduit en modifiant la valeur du facteur d'échelle, en l'occurrence en le réduisant. Attention cependant car une trop forte réduction de ce paramètre risquerait de faire perdre complètement l'effet de relief que nous souhaitions de primes abords. De plus, ce phénomène apparaîtra toujours à partir d'un certain angle, et la division par la composante en "z" est censée réduire grandement ce problème.

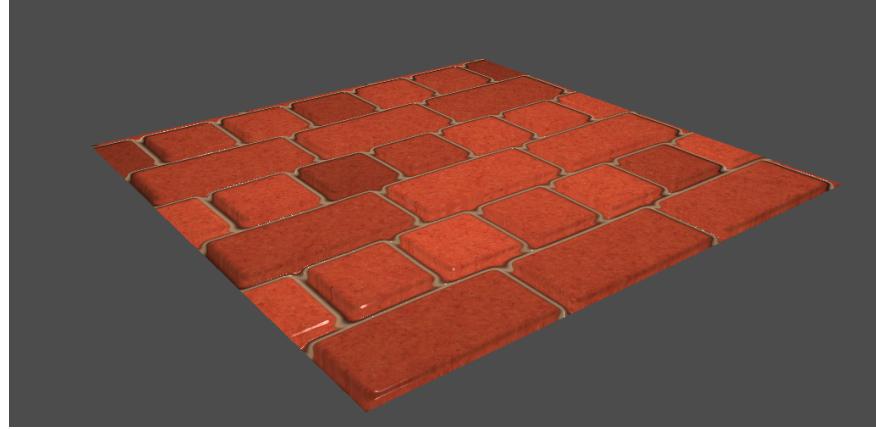


Figure 12: Illustration des problèmes de bords créés lorsque la caméra est trop rasante par rapport à l'objet : on observe au niveau des intersections des "coulures" rendant l'effet beaucoup moins réaliste. Ce phénomène peut être atténué en diminuant le facteur d'échelle.

Aussi il est possible de mettre en place les calculs décris dans l'équation 7 et d'obtenir les résultats de droite de la Figure 13. L'image de gauche représente le résultat sans la division par la composante "z" avec le même facteur d'échelle et suivant approximativement le même angle rasant. Contrairement aux résultats attendus, il semblerait que le résultat soit beaucoup moins bon. Certes il est toujours possible de réduire le facteur d'échelle afin de trouver de meilleurs résultats mais ceci donnerait également de meilleurs résultats sans la division. Aussi, sachant que dans la théorie ces calculs devraient améliorer les résultats, nous sommes confrontés à un problème soit d'ordre algorithmique : une erreur dans l'algorithme ferait que le résultat est moins bon, soit d'ordre pratique : la théorie peut en effet parfois ne pas s'appliquer dans la pratique. Dans les deux cas, avec les résultats que nous obtenons, il semblerait que le résultat soit meilleur sans utiliser la division.

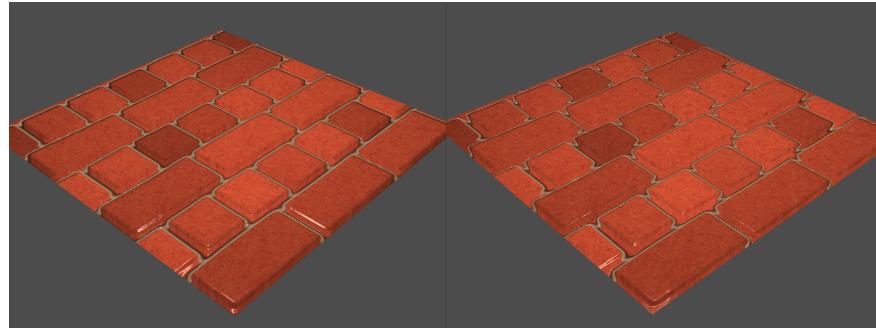


Figure 13: Comparaison des résultats pour un angle assez rasant, avec (à droite) et sans (à gauche) la division par la composante en "z". À la vue de ces résultats, il semblerait que diviser par la composante en "z" n'améliore pas les problèmes de bords mais les empire car les coulures au niveau des intersections semblent beaucoup plus prononcées qu'elles ne le sont sans la division.

Afin de ne percevoir que très peu cet effet, il serait donc conseillé de faire en sorte (en pratique, dans un jeu vidéo par exemple) que la caméra ne puisse pas adopter de tels angles très rasants (ce qui semble assez cohérent et raisonnable : il n'y a que très peu de raisons pour lesquelles un personnage jouable viendrait se coller au mur et le regarder de manière rasante)

Il est tout de même possible d'améliorer les résultats non pas pour les angles rasants mais sur les bords de l'objet. En effet, on peut noter sur la Figure 14 de gauche que sur les bords de notre objet, le mur semble complètement coupé ce qui n'est pas naturel, en effet on devrait pouvoir voir le fond lorsque l'on est dans un creux du bord de l'image et non pas recommencer la texture. Pour ce faire, il suffit de ne pas afficher le fragment lorsque les coordonnées de textures du point B obtenues ne sont pas dans la texture strictement. Cette opération nous permet d'obtenir l'image de droite de la Figure 14 et donc rendre le résultat d'autant plus réaliste.

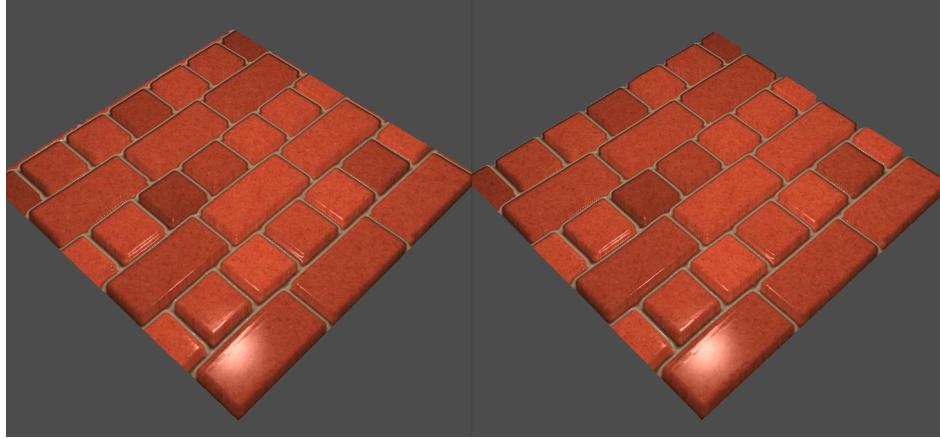


Figure 14: Comparaison des résultats avec (à droite) et sans (à gauche) suppression des fragments dont les coordonnées de texture ne sont pas compris strictement dans la texture. On peut alors observer que le réalisme sur les bords est bien meilleur en supprimant les fragments en question.

7 Conclusion

Plusieurs méthodes peuvent être utilisées pour pouvoir améliorer une texture ou dans le cas général améliorer le rendu d'une image. Dans ce TP plusieurs méthodes ont été vu comme l'illumination de Phong pour simuler l'effet de lumière, la modification des hauteurs pour modifier la géométrie et ajouter du relief à la texture, la modification des normales pour améliorer le rendu et le réalisme du calcul de la lumière, l'effet d'occultation ambiante pour ajouter un effet d'ombre pour les zones moins exposé à la lumière ou encore l'adaptation de texture pour simuler un relief sans avoir à modifier la géométrie. Ces méthodes comme il a été montré ont parfois de très bons résultats mais ont un cout qu'il ne faut pas négliger. En effet dans les exemples étudiés il est possible de citer le calcul de lumière qui ajoute du calcul et donc du temps GPU. La plupart des méthodes nécessitent notamment l'utilisation de textures supplémentaires pour obtenir des informations (hauteurs, normales...) qui ne sont pas présentes dans la texture d'origine ce qui utilise de la mémoire et qui, dans le cas de grandes scènes pourrait être un problème. Il faut ajouter à cela le travail pour obtenir ces textures d'information. Il est tout de fois notable que les problèmes cités sont réduits par le fait que l'utilisation d'une même texture d'information peut servir à plusieurs méthodes cumulables ce qui augmente la rentabilité de la texture. Il est ainsi possible d'avoir pour quelques très raisonnables coûts supplémentaires de très largement améliorer le rendu de la texture.