Ray-Tracing - Adds-on

- Ombres
- Luminosité et brillance
- Objets limités
- Réflexions
- Transparence / modification des ombres
- Lumière directe
- Objets composés
- Perturbations / textures
- Divers : objets négatifs, lumière d'ambiance

Modification de la couleur

- La technique de Raytracing vue jusqu'ici nous a permis
  - de nous déplacer dans notre espace
  - de placer des objets dans une position aléatoire
  - de trouver pour un pixel donné, quel objet croise notre regard.

- Nos images restent pour le moment peu réalistes. En effet, la couleur de l'objet est simplement transmise au pixel concerné.

- Les techniques d'ombre, luminosité et brillance vont uniquement agir sur la couleur de l'objet afin d'obtenir les effets voulus. Cette phase se passe donc après avoir déterminé les paramètres de l'intersection rayon/objet.

Les ombres

- Les sources de lumière vont entrer en jeu. Commençons par un spot S placé en $(500, 500, 500)$.

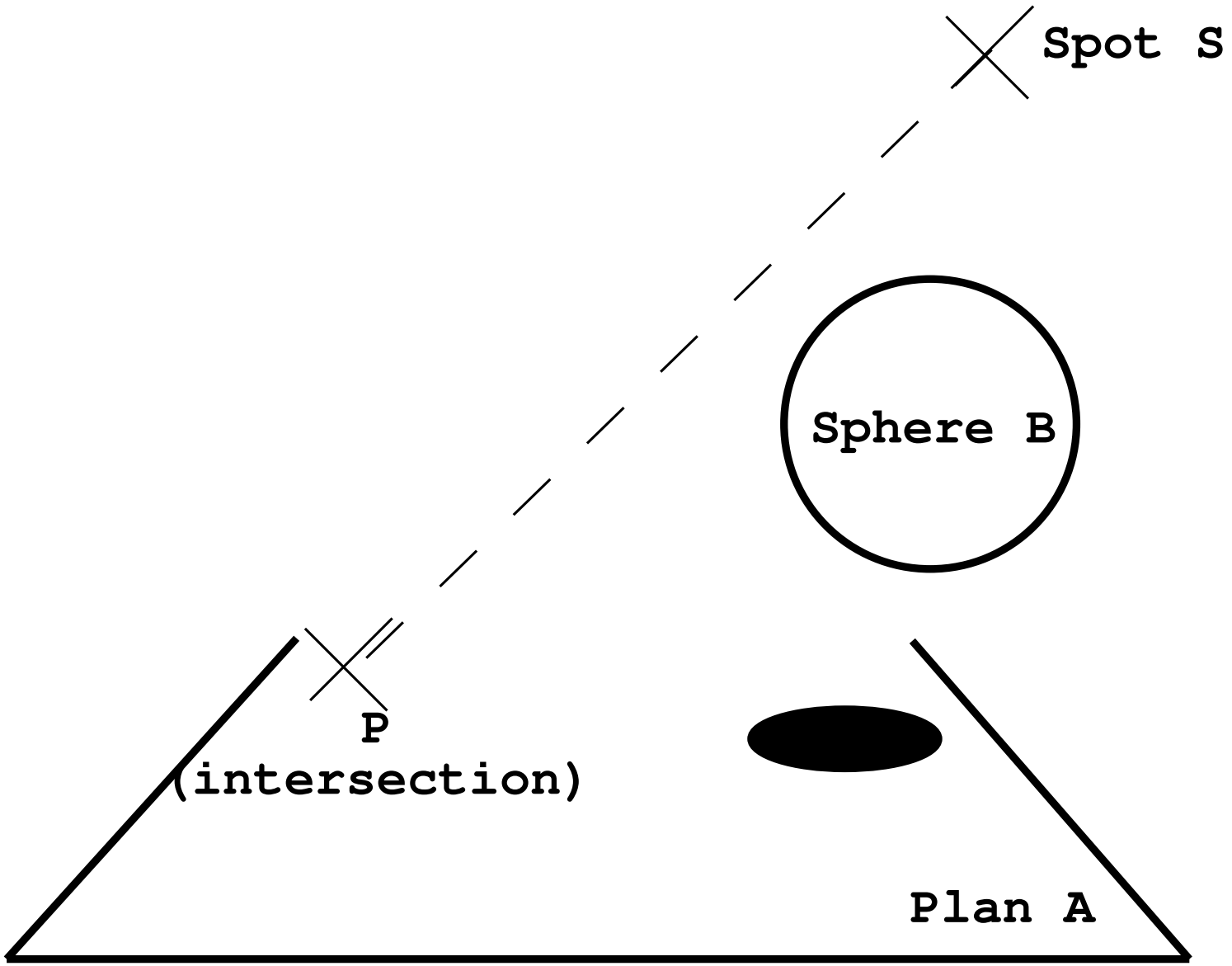
- Ce spot éclaire dans toutes les directions.

- Dans quel cas y-a-t-il une ombre ? Lorsqu'un objet B se situe entre le spot et l'objet A que l'on regarde.

- A un pixel (X, Y) donné, le principe de ray-tracing nous permet d'y associer un point P sur l'objet A (l'intersection entre l'objet et notre droite de vision). Nous obtenons alors une unique droite passant par ce point et le spot.

- Intersection droite-objet, ça vous rapelle quelque-chose ?
Il s'agit ici de ne conserver QUE les intersections qui se trouvent entre notre objet A et le spot S.

Ombres...



Droite d'ombres

- Nous pouvons donc écrire l'équation paramétrique de la droite entre S et P:

$$\begin{cases} x = x_P + kL_x \\ y = y_P + kL_y \\ z = z_P + kL_z \end{cases} \text{ avec le vecteur directeur } \vec{L} \begin{vmatrix} x_S - x_P \\ y_S - y_P \\ z_S - z_P \end{vmatrix}$$

- Il s'agit alors de tester l'intersection entre cette droite et tous les objets de notre espace (exactement comme pour la droite de vision). Attention, seules les valeurs de k telles que $0 < k < 1$ sont interressantes : au delà de 1, on se situe après le spot, et avant 0, on est derrière le point P.

- Si il n'y a pas d'ombre, on utilise la couleur de l'objet. Dans le cas contraire, on met du noir ou une couleur très proche.

Luminosité

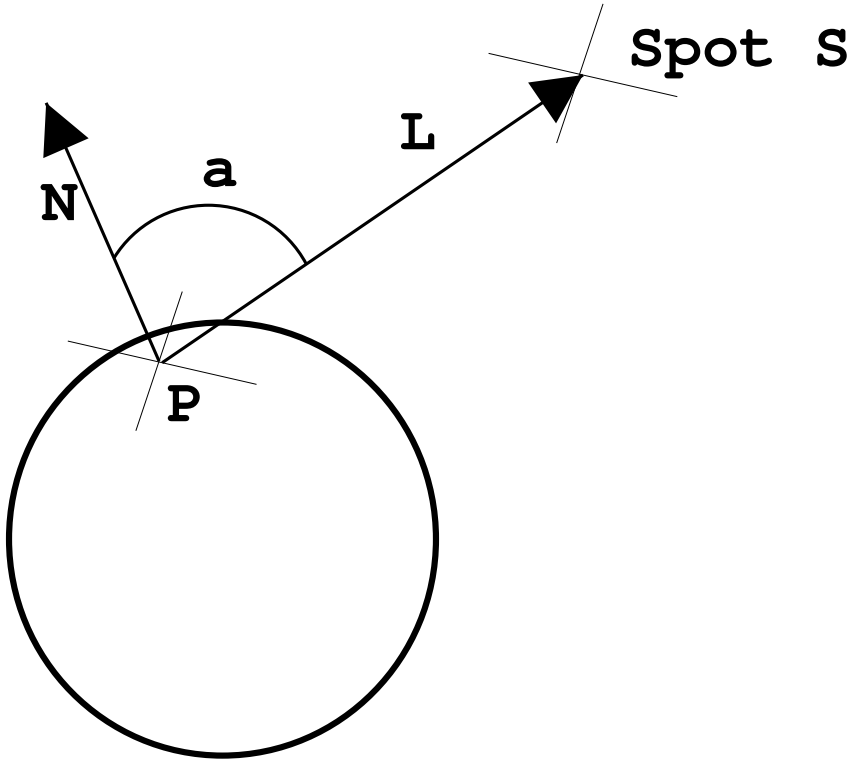
- Il s'agit de faire ressortir visuellement les parties les plus éclairées de l'objet: plus un point est "face" au spot, plus l'intensité de la couleur sera grande.

- Cette technique nécessite de connaitre un vecteur $\vec{N}$ *normal* à l'objet au point d'intersection P (un vecteur perpendiculaire au plan tangeant à l'objet au point d'intersection).

- On ré-utilise le vecteur $\vec{L}$ entre le point d'intersection et le spot.

- Plus les vecteurs sont "proches", plus le point sera éclairé. Cela se traduit par l'angle a entre les vecteurs $\vec{N}$ et $\vec{L}$. Plus l'angle a est petit, plus le point est lumineux.

Luminosité...



Calculs de la luminosité

- Ce calcul se fait grâce au produit scalaire des deux vecteurs:
  $\vec{N}.\vec{L} = ||\vec{N}||||\vec{L}||cos(a) = N_xL_x + N_yL_y + N_zL_z$.
  Bien sûr $||\vec{N}|| = \sqrt{N_x^2 + N_y^2 + N_z^2}$.

- $cos(a)$ est donc trouvé. Plus la valeur du cosinus est proche 1, plus l'angle a est proche de 0. Lorsque le cosinus est négatif, le point ne "regarde" pas la lumière et sera donc noir (comme s'il était dans l'ombre).

- Pour les valeurs entre 0 et 1, le cosinus va servir de coefficient multiplicateur des trois composantes RVB de la couleur de l'objet. Par ex :
  $new\_rouge = rouge \times cos(a)$

Les normales ...

- Le vecteur normal au point d'intersection est tout d'abord déterminé dans la position simple de l'objet, puis ensuite reconvertit en ses vraies coordonnées, grâce aux rotations et à la translation.

- On appelle P le point d'intersection dans la position simple de l'objet.

- Pour la sphère, le vecteur $\left|\begin{array}{l} x_P \\ y_P \\ z_P \end{array}\right.$ convient comme vecteur normal $\vec{N}$.

- Pour le plan, vu que notre objet simple est le plan horizontal $z = 0$, le vecteur $\left|\begin{array}{l} 0 \\ 0 \\ 100 \end{array}\right.$ convient, quel que soit le point d'intersection.

... de nos objets

- Pour le cylindre : $\left| \begin{array}{l} x_P \\ y_P \\ 0 \end{array} \right.$

- Pour le cône d'équation $x^2 + y^2 - cte \times z^2 = 0$:
$\left| \begin{array}{l} x_P \\ y_P \\ -cte \times z_P \end{array} \right.$

- *Attention :* Tout comme pour le point P, vous allez calculer les coordonnées de $\vec{N}$ en position simple de l'objet.
N'OUBLIEZ PAS de convertir ces coordonnées en position réelle de l'objet. Dans le cas contraire, $\vec{L}$ et $\vec{N}.\vec{L}$ seront complètement faux.

Brillance

- La brillance de l'objet va intervenir en même temps que la luminosité.

- Le principe est le suivant : plus une surface est brillante, plus la couleur finale dépendra de celle du spot (un spot de couleur ne traumatise personne j'espère).

- Le concept de brillance sera concrètement exprimé par une constante $brillance$ entre 0 et 1 propre à chaque objet. Une des formules possible est :
$new\_rouge2 = new\_rouge + brillance \times rouge_{spot} \times cos(a)$

- Le $cos(a)$ est toujours présent : une face à l'ombre qui brille, ça fait désordre !

Formules: trouvez la votre !

- *Attention:* pour de nombreux calculs de couleurs, luminosité, transparence, réflexion, ... les formules sont loin d'être uniques. Si d'autres proportions, d'autres façon de mélanger les couleurs vous semblent plus réalistes, n'hésitez pas à les employer !!

- Par exemple : dans la formule de luminosité, on ne tient pas compte de la puissance de la lumière. Si la lumière est plus faible, l'objet donnera l'impression d'être moins éclairé.

- On va donc diminuer notre couleur proportionnellement à l'intensité de la couleur du spot:
$new\_rouge = rouge \times cos(a) \times \frac{rouge_{spot}}{rouge_{MAX}}$
avec $rouge_{MAX}$ la plus grande valeur possible pour la composante rouge.

Multi-spot

- Comment faire lorsque l'on a plusieurs spots ?
  Là encore, vous pouvez choisir votre formule.
  Le plus simple est d'ajouter les composantes des différentes couleur obtenues pour chaque spot.
  $new\_rouge_{final} = new\_rouge_{spot1} + new\_rouge_{spot2} + ...$

- Afin de rester dans des niveaux de couleurs raisonnables, vous pouvez ensuite diviser le tout par le nombre de spots.
  $new\_rouge_{final} = (new\_rouge_{spot1} + new\_rouge_{spot2} + ...)/nb\_spots$

- Autre solution: intégrez à vos formules l'intensité des spots pour que la couleur finale ne soit pas complètement saturée (généralement ça donne du blanc).

Objets limités

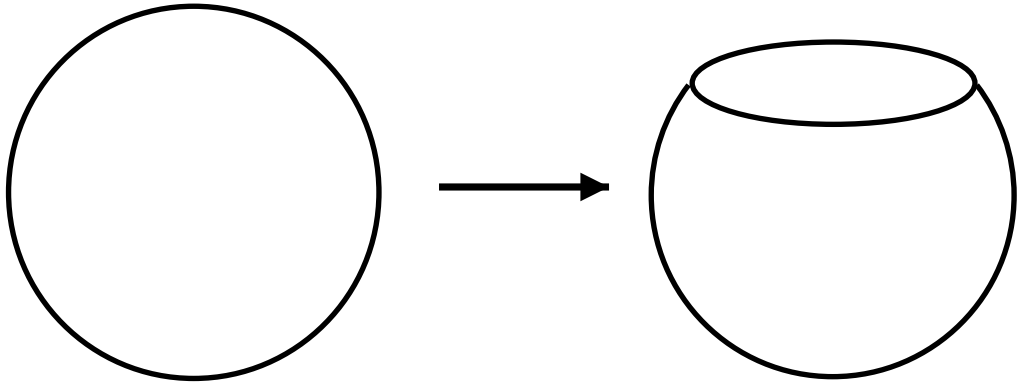
- Avec les objets limités, on va pouvoir créer:
  - des morceaux de sphère
  - des morceaux de cylindre
  - des morceaux de cône
  - des parallélogrammes
  - des triangles
  - des disques

- Les méthodes exposées ici sont bien sûr modifiables à votre guise.

- Les limites d'un objet seront définies dans la position simple.

On y va : la sphère

- On souhaite enlever la partie supérieure d'une sphère, à la moitié du rayon R.



- Dans la position simple, il suffit de dire que les points au dessus du plan $z = \frac{R}{2}$ n'appartiennent pas à la sphère.

L'algo

- Concrètement:
  - Calcul de l'intersection avec l'objet en position simple.
  - Calcul du point P d'intersection toujours en position simple.
  - Avons-nous $z_P > \frac{R}{2}$? Si oui, la fonction sphère répond "pas d'intersection". Sinon, rien n'est changé.
  - $\vec{N}$ est calculé, les coordonnées de P et $\vec{N}$ sont transformées par les rotations puis translatées, comme d'habitude.

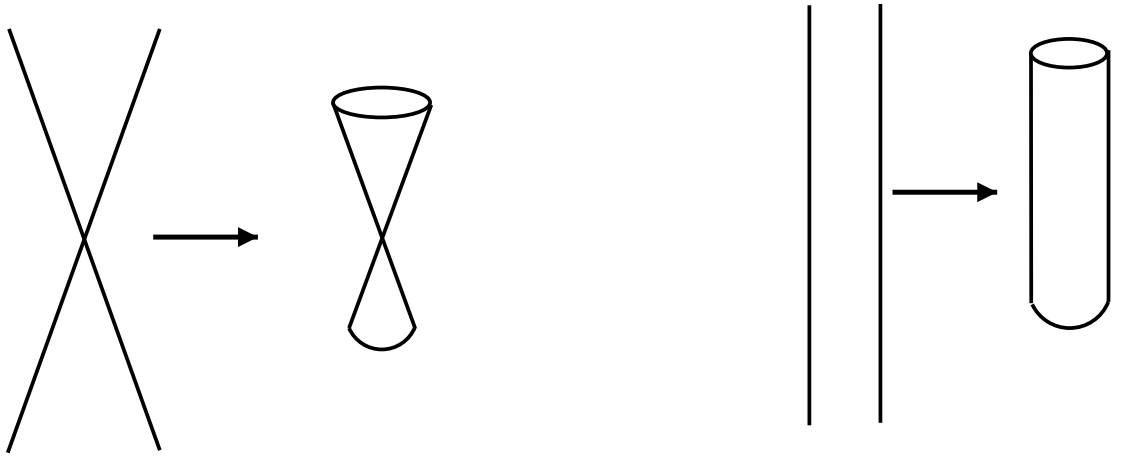
- Les données relatives à la sphère s'enrichissent :
  - origine - rayon - couleur - les 3 rotations - coefficient brillance
  - limite haute - limite basse - bientôt transparence, reflexion, texture, fonctions de perturbation.

Limite cylindre et cône

- Le cylindre et le cône dans leur position simple sont disposés le long de l'axe des z.

- On utilise le même test que pour la sphère : $limite_{basse} < z_P < limite_{haute}$

- Nos deux objets s'enrichissent de 2 caractéristiques supplémentaires.



Le plan limité

- Chacun des cas suivant implique l'ajout de données dans la définition du plan.

- Le disque : on teste $(x_P^2 + y_P^2) < R^2$ (Rappel : on est dans le plan $z = 0$).

- Le parallélogramme défini par $\vec{V_1}$ et $\vec{V_2}$ dans le plan $z = 0$: on veut avoir $P = k_1\vec{V_1} + k_2\vec{V_2}$ avec k_1 et k_2 compris entre 0 et 1.

- On obtient rapidement : $k_2 = \frac{V_{1x}y_P - x_P V_{1y}}{V_{2y}V_{1x} - V_{1y}V_{2x}}$ et $k_1 = \frac{x_P - k_2 V_{2x}}{V_{1x}}$. Si k_1 et k_2 ne remplissent pas la condition, la fonction plan repond "pas d'intersection".

- Le triangle : avec exactement le même calcul, on teste, EN PLUS du parallélogramme, si $k_1 + k_2 < 1$.

Réflexion

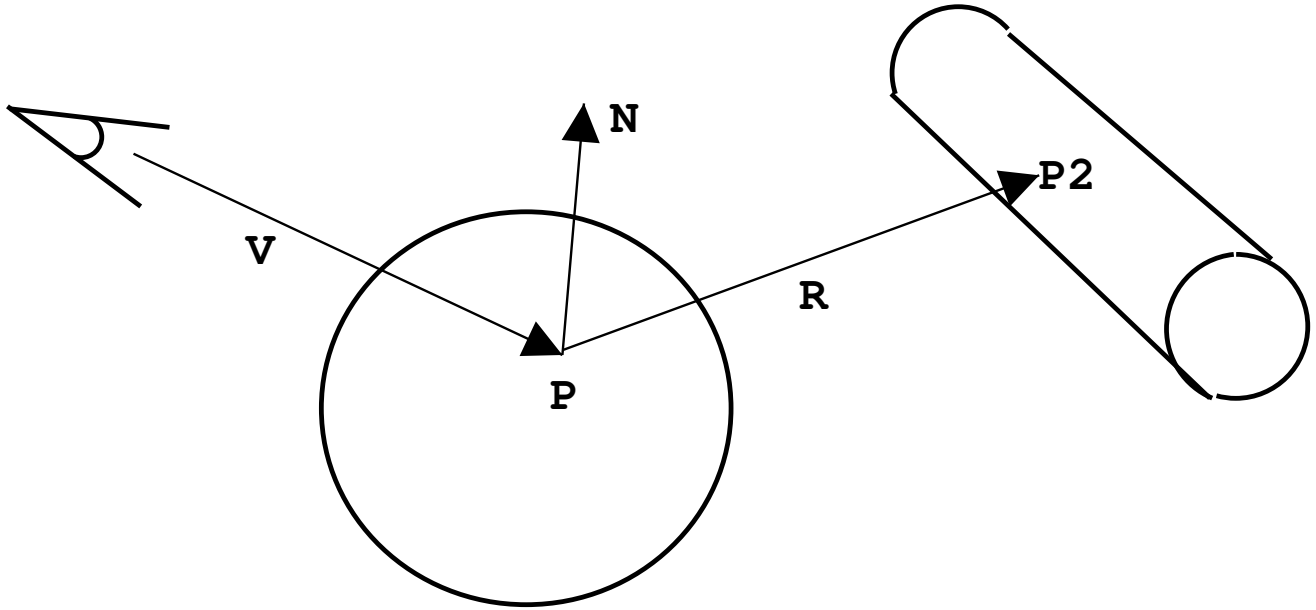
- Il s'agit ici d'obtenir un effet "miroir".

- Cet effet est quantifié par une constante entre 0 et 1. Plus la constante est proche de 1, plus l'effet miroir est important.

- Nous allons pour cela calculer un "rayon réfléchi" pour notre rayon de vision :
$\vec{R} = -2\vec{N} \times \vec{V}.\vec{N} + \vec{V}$
ATTENTION !!! Cette formule fonctionne pour des vecteurs UNITAIRES (diviser les coordonnées du vecteur par sa norme).

- Le point d'intersection P et ce vecteur $\vec{R}$ nous donnent un nouveau rayon pour qui on va recursivement calculer les intersections avec les objets et la couleur que l'on verrait si notre oeil était en P.

Miroir mon beau miroir...



- Nous avons donc desormais 2 couleurs pour notre point P : celle provenant de l'objet lui-même (avec luminosité), et celle du rayon réfléchi. On les combine de la façon suivante :
$rouge = rouge_{reflechi} \times reflexion + rouge_{objet} \times (1 - reflexion)$

Transparence

- Il s'agit ici d'obtenir un effet transparent (vous en doutiez ?)

- Comme pour la réflexion, une constante *transparence* entre 0 et 1 caractérisera cette propriété pour nos objets.

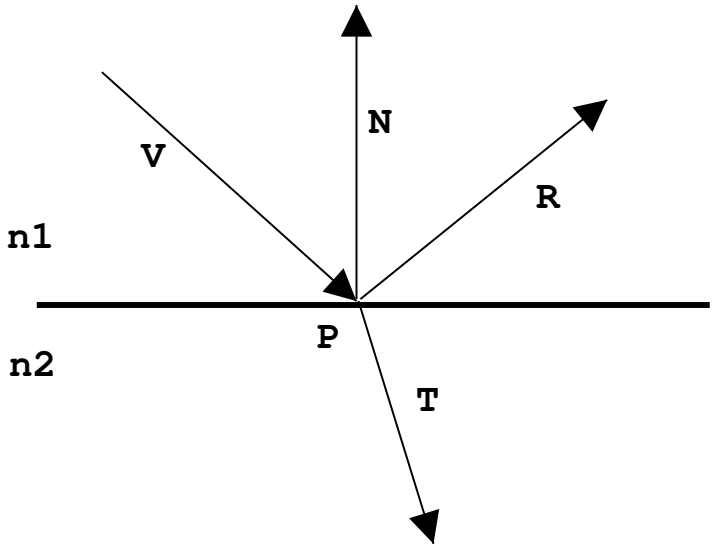
- Ici aussi, on va passer par le calcul d'un "rayon transmis" :
  $\vec{T} = n \times \vec{V} + \left(n \times \vec{V}.\vec{N} - \sqrt{(1 + n^2((\vec{V}.\vec{N})^2 - 1))} \right) \times \vec{N}$
  $n = \frac{n_2}{n_1}$ où $n1$ et $n2$ représentent les indices optiques des différents milieux, $\vec{V}$ et $\vec{N}$ étant dans le milieu d'indice n_1.

- Nos vecteurs devront être unitaires. Dans le cas où la partie sous la racine carrée serait négative, on la considèrera comme nulle.

- L'indice optique de l'air vaut 1. On a déjà de bons résultats avec des objets ayant des indices proches : 0.9 ou 1.1.

Ajax vitres

- De nouveau, récursivement, on calcule la couleur "vue" depuis le point P dans la direction $\vec{T}$.



- La combinaison de couleurs originale/transmise se fait sur le même principe que la réflexion :
$rouge = rouge_{transmis} \times transparence + rouge_{objet} \times (1 - transparence)$

Modification des ombres

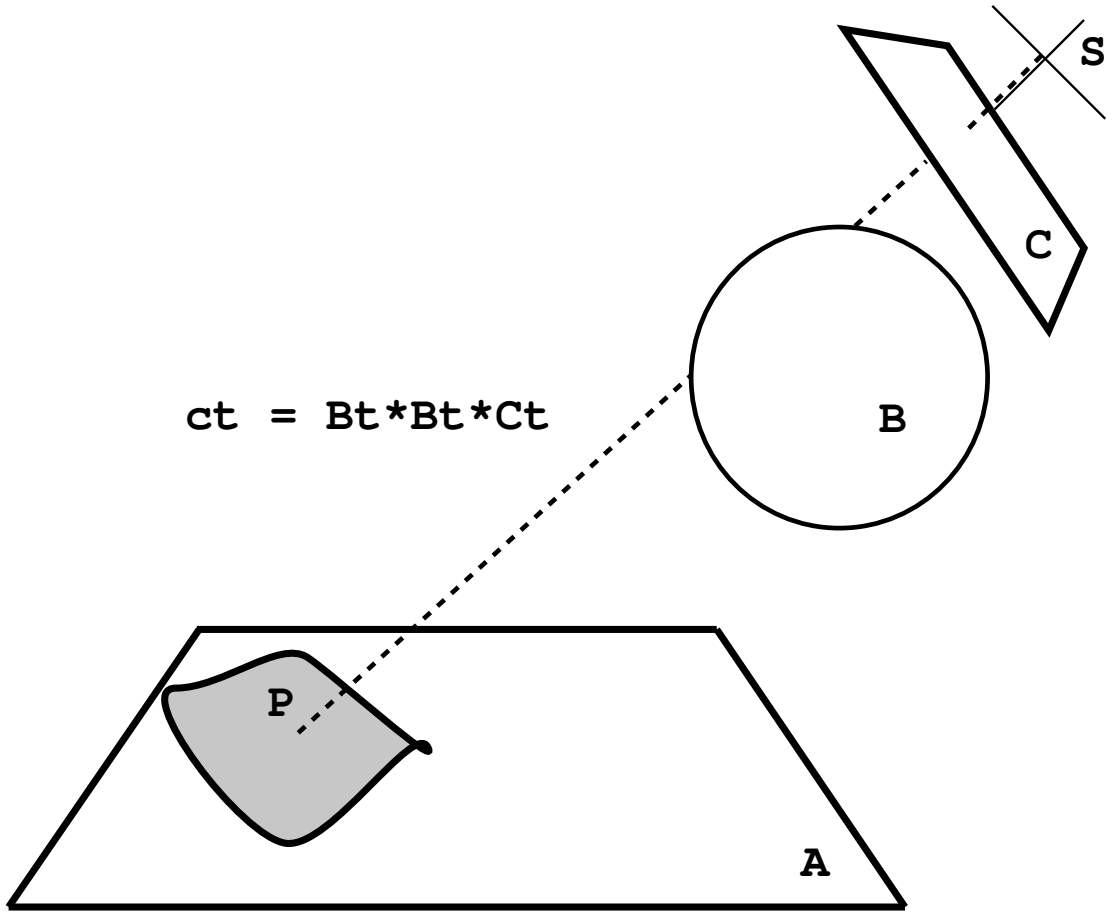
- Les ombres sur les objets vont devenir un petit peu plus complexes à traiter (mais pas *beaucoup* plus complexes).

- Le cas est le suivant : entre mon objet A et mon spot S, il y a un objet B. Seul hic, B est transparent, ou presque. La logique voudrait donc que mon objet A soit presque autant éclairé qui s'il n'y avait pas B.

- Pour solutionner ce problème, on va calculer un coefficient de transparence c_t global à tous les objets entre A et S. Il s'agira du produit de tous les coefficients des objets : $c_t = 1 \times B_t \times C_t \times ...$.

- S'il n'y a pas d'objets ou si tous les objets sont complètement transparents, $c_t = 1$. Pour des objets moyennement transparents, c_t se rapproche de 0. Enfin, dès qu'un objet est totalement opaque, son coefficient est nul et donc $c_t = 0$, quels que soient les autres objets.

Ombres et transparence



Synthèse de la couleur d'un objet

- Nous allons finalement intégrer d'un seul coup
  - la luminosité
  - la brillance
  - les ombres

- Les deux premièrs effets ont déjà été regroupés dans la même formule :
  $new\_rouge2 = rouge \times cos(a) \times \frac{rouge_{spot}}{rouge_{MAX}} + brillance \times rouge_{spot} \times cos(a)$

- Notre coefficient de transparence global c_t va servir à modifier la quantité de lumière du spot qui parvient sur l'objet. On va le multiplier à $rouge_{spot}$:
  $new\_rouge2 = (\frac{rouge}{rouge_{MAX}} + brillance) \times rouge_{spot} \times cos(a) \times c_t$

Lumière directe

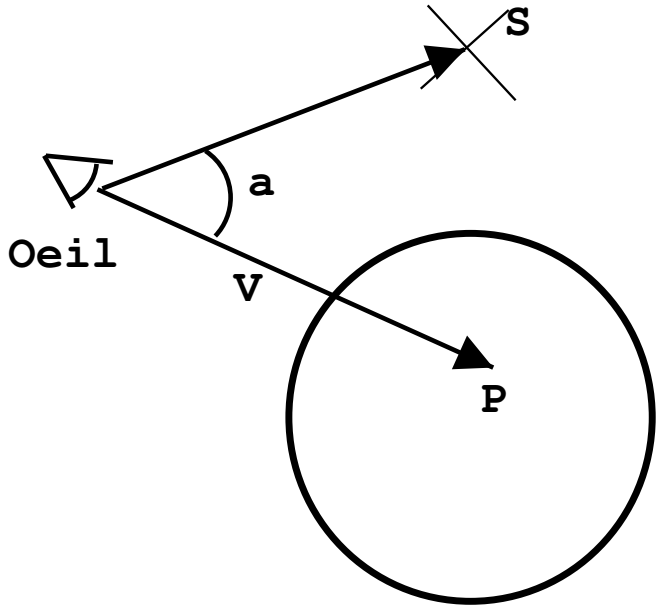
- Regardez un spot de face, vous êtes ébloui. Notre modèle actuel de Raytracer ne permet pas de gérer cet effet.

- Décomposons le phénomène :
  - plus l'on regarde un point proche du spot, plus la couleur perçue est vive et proche de celle du spot.
  - plus l'on éloigne le regard de la direction du spot, moins la couleur finale dépend de celle du spot.

- Nous allons chercher l'angle entre notre vecteur de vision $\vec{V}$ et le vecteur $O\vec{eil}S$. Plus cet angle est petit, plus on est ébloui. Le cos de l'angle servira donc pour faire intervenir la lumière du spot sur l'oeil.

- Cette lumière directe peut se comprendre comme la luminosité crée par un spot sur notre oeil.

Nous avons les moyens...



- Surtout ne pas oublier de faire intervenir ici aussi (comme pour la luminosité) un coefficient de transparence c_t des objets qui pourraient se trouver entre l'oeil et le spot. Etre ébloui par un spot caché, c'est pas super !

- Le plus simple etant bien sûr de réutiliser la même fonction que pour la luminosité.

Après réflexion et transparence

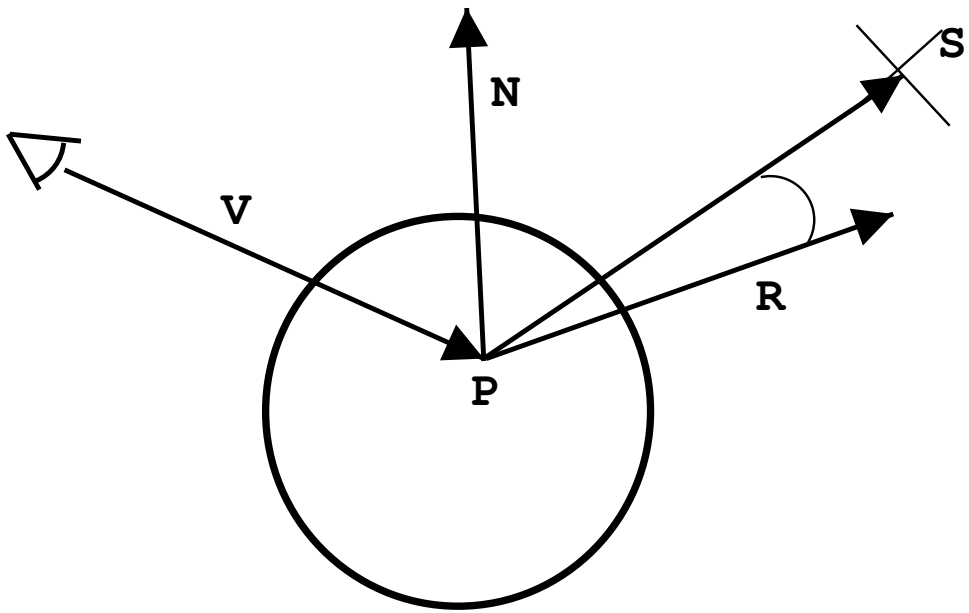
- Que se passe-t-il lorsque vous regardez une source lumineuse dans un miroir ? Vous avez un éclat important.

- De même, lorsqu'un spot se trouve derrière un objet relativement transparent, vous avez au travers de l'objet un certain éclat lumineux.

- Comment prendre en compte ce phénomène ?
  En faisant confiance à son code!

- En effet : pour faire intervenir la réflexion (ou la transparence), on s'est demandé "Quelle couleur voit-on si l'oeil est en P et regarde suivant $\vec{R}$?"
  Le code pour le calcul de la couleur vue par l'oeil a été réutilisé récursivement.
  Si celui-ci intègre l'effet "lumière directe", le point P regardant suivant $\vec{R}$ sera ébloui, et transmettra cet effet à notre oeil.

Direct reflet



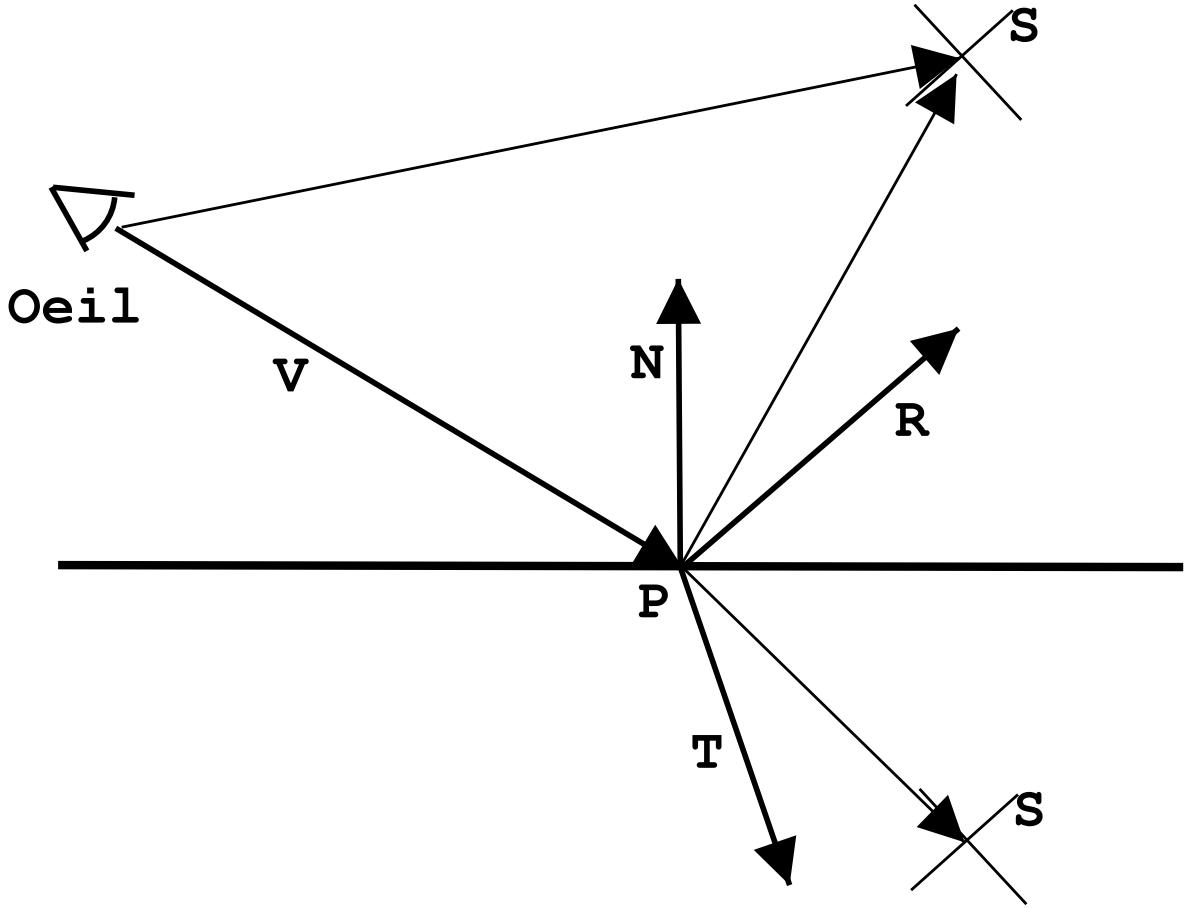
Synthèse de la couleur du pixel

- Nous pouvons maintenant rassembler tous les éléments qui interviennent dans l'élaboration de la couleur du pixel.

- Nous avons donc les quatre grandes catégories suivantes :
  - La couleur de l'objet modifiée par la luminosité, la brillance et les ombres, en prenant en compte les effets de transparence
  - La couleur issue de la réflexion
  - La couleur issue de la transparence
  - La lumière directe

- Ces quatres composantes peuvent simplement être additionnées car il est courant qu'une seule prévale sur les autres. A vous de régler vos couleurs et intensités de spots pour ne pas obtenir trop d'effets de saturation.

La totale



Objets composés

- Nous voulons réaliser un cube, et pouvoir ensuite manipuler ce cube de façon simple.

- A la base, un cube est constitué de 6 faces, soit donc 6 parallélogrammes (carrés). Pour obtenir ces 6 faces correctement positionnées, il faut pour 5 d'entre elles effectuer des rotations et une translation (90^o sur l'axe des x nous donne la face de gauche de notre cube par exemple..).

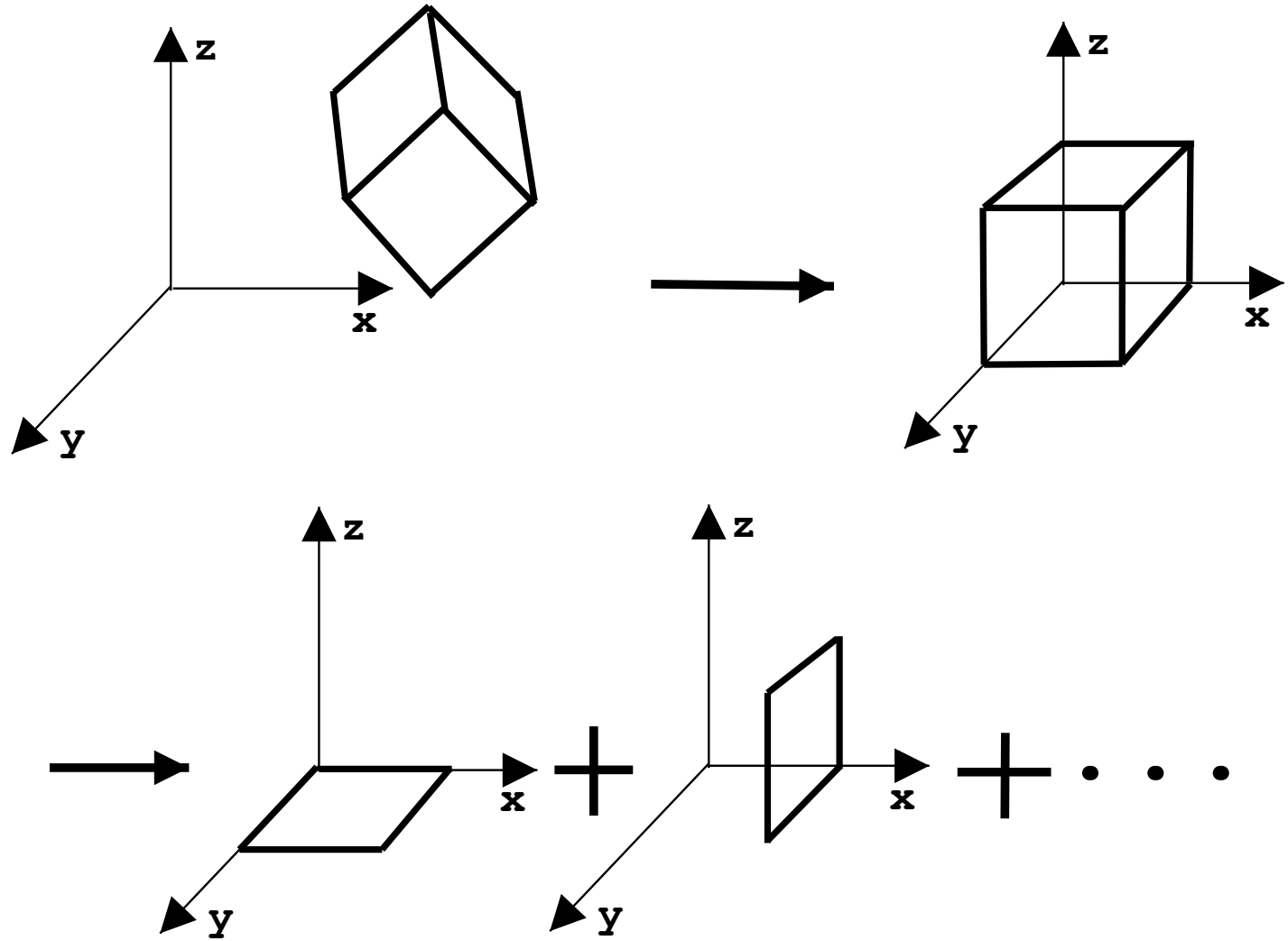
- Imaginons que nous voulons considérer le cube comme une entité unique, et lui appliquer dans son ensemble une translation et des rotations. Ce principe tend à nous masquer les 6 bouts de plan qui composent notre objet. En revanche, décrire notre scène 3D en parlant simplement d'un cube dans l'espace avec sa position et ses caractéristiques, est nettement plus interressant.

Bouts regroupés

- Les 6 bouts de plans vont être regroupés dans un objet de type cube. Ces 6 bouts, étant des objets eux mêmes, ont leurs propres translations et rotations. Ces transformations sont celles qui permettent de créer l'objet cube dans sa position simple: un bout qui ne bouge pas pour le bas du cube, un deuxième bout identique mais élevé pour faire le haut du cube, ...

- L'objet cube aura lui-même ses propres transformations : elles servent à rammener le cube de sa position réelle à sa position simple. Une fois en position simple pour le cube, pour chacun des 6 bouts de plan, il faut se remettre dans la position simple de ce bout de plan. On peut alors faire intervenir la fonction d'intersection avec un plan.

Cube décomposé



Perturbations

- Pour nos objets, nous avons un ensemble de caractéristiques physiques fixes : couleur, brillance, transparence ... et bien d'autres.

- La forme de nos objets conçus mathématiquement est sans défaut.

- Si on créait des objets un peu moins parfaits ?

- Il suffit pour cela d'avoir de simples fonctions qui vont perturber la jolie unicité des propriétés. D'après les coordonnées d'un point sur l'objet (notre point P par exemple), ces fonctions vont sensiblement modifier la normale $\vec{N}$ en ce point (et donner une surface d'aspect moins lisse), la transparence, la couleur ...

- Pour minimiser les calculs dans nos fonctions, on les applique dans la position simple de l'objet, donc avec les coordonnées simple de P, $\vec{N}$, etc..

Effet vagues

- Voyons par exemple comment obtenir un effet “vagues”.

- Nous allons simplement modifier légèrement la normale d’un point d’un plan. Les fonctionnalités brillance et luminosité créeront l’effet visuel.

- Des vagues parallèles à l’axe des x :
$N_y = N_y + cos(\frac{y_P}{10}) \times \frac{||\vec{N}||}{10}$, N_x et N_z restent inchangés.

- Les fonctions de perturbation feront elles-mêmes partie des caractéristiques d’un objet.

- C’est à vous de faire travailler votre imagination pour trouver des fonctions produisant des effets visuels incomparables!! Certaines perturbations combinées avec les textures adéquates donnent des effets de bois, granit, velour (...) très réalistes.

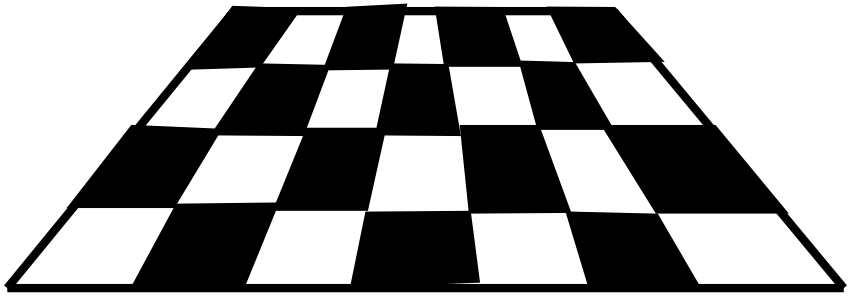
Couleurs et textures

- Si nos fonctions de perturbation peuvent modifier une normale d'un objet en un point donné, elles peuvent bien sûr modifier la couleur.

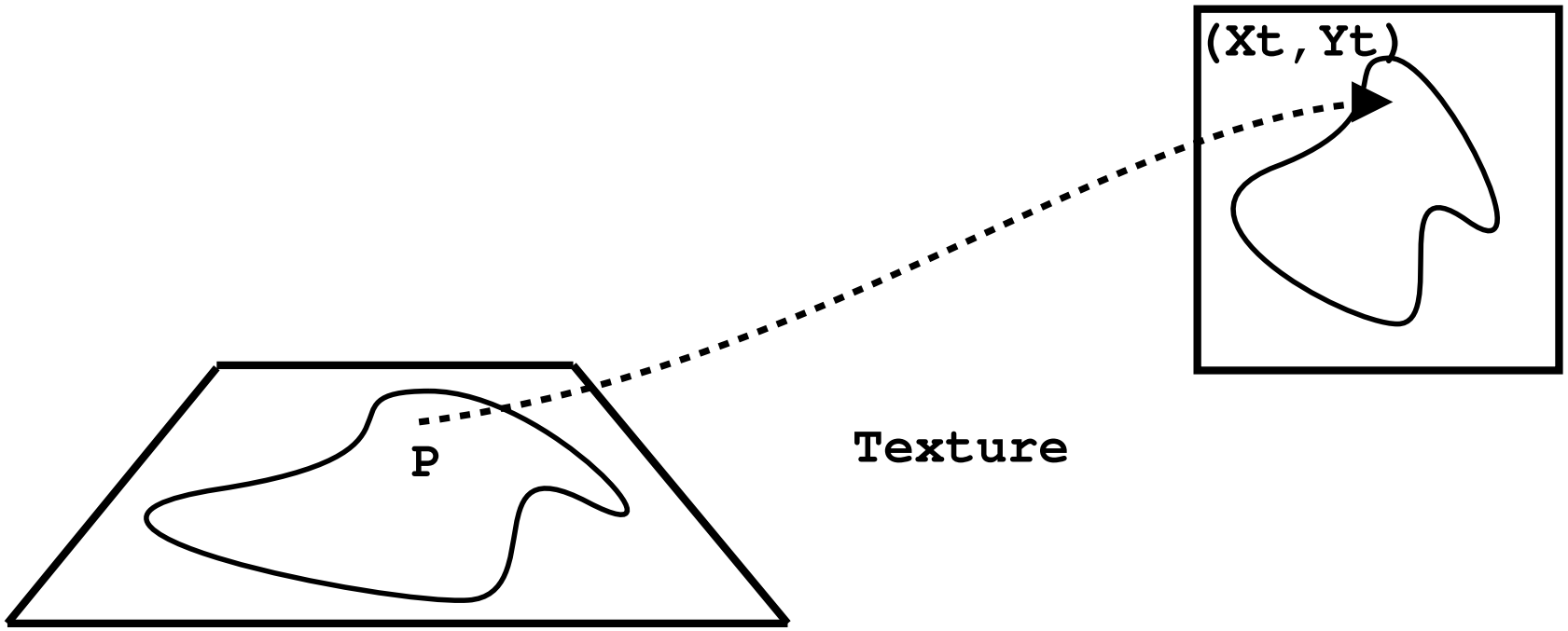
- Créons un "damier" sur un plan :
on teste si $x_P\%100 > 50$ et si $y_P\%100 > 50$, si ces deux conditions sont vraies ou fausses en même temps, on choisit une couleur, sinon, une autre.

- Notre fonction de perturbation couleur va aussi pouvoir effectuer des calculs simples (règle de 3...) pour obtenir à partir de (x_P, y_P, z_P) des coordonnées (X_t, Y_t) dans une texture, et renvoyer la couleur du pixel correspondant dans la texture.
On a texturé notre objet, en gérant les problèmes de zoom et de textures "qui bougent".

Textures



Damier



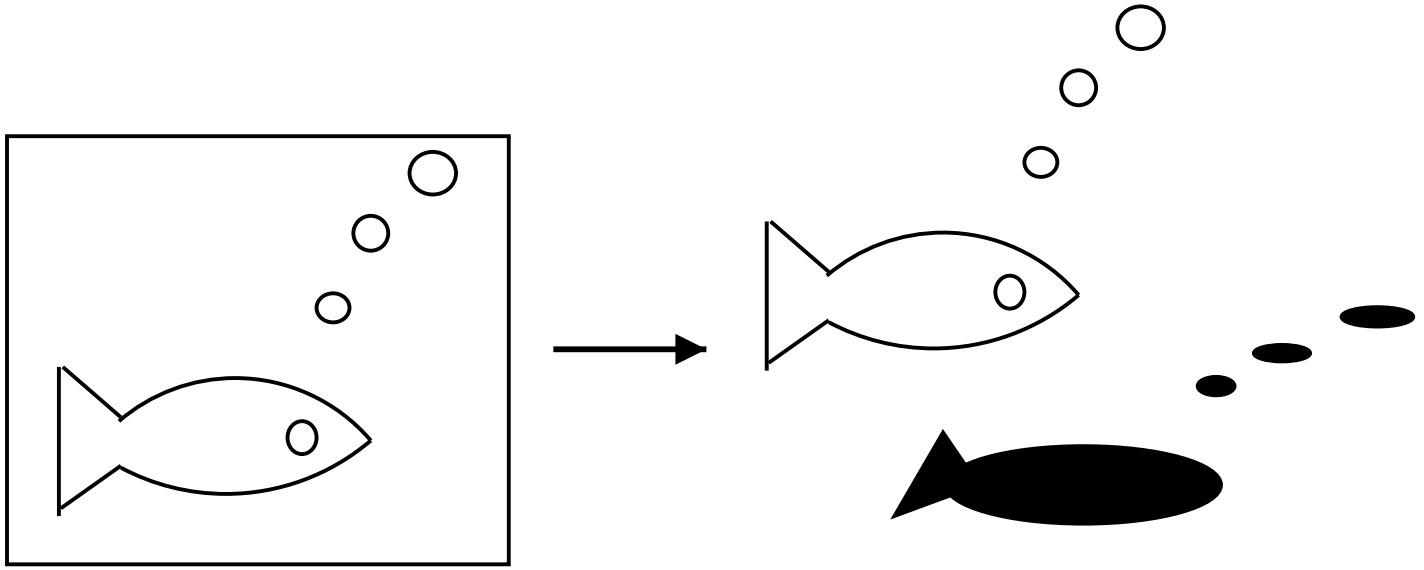
Encore des perturbations

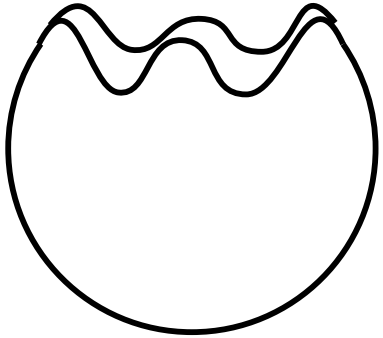
- Le système des perturbations peut s'appliquer à pratiquement tous les paramètres qui interviennent dans les calculs post-intersection. Il est ainsi possible de modifier la transparence ou la reflexion en fonction d'une formule.

- Une texture peut à la fois servir de perturbation de couleur et de perturbation de transparence. Un dessin initialement sur fond blanc peut se retrouver texturé sur un morceau de plan, mais en plus permettre la découpe du bout de plan là où se trouve la couleur blanche.

- Plus délicat mais toujours réalisable, une perturbation de la limite d'un objet, en fonction du point d'intersection.

Perturbations ++





Lumière d'ambiance

- Jusqu'à présent, les parties d'objet non exposées à la lumière, donc dans l'ombre, ont été rendues à l'écran en noir. Ce schéma manque quelque peu de réalisme. Les objets éclairés ré-émettent eux-mêmes un peu de lumière (lumière diffuse).

- Il serait malheureusement impensable de considérer tous les points d'objets éclairés comme à leur tour des spots de plus faible intensité.

- On définit alors une lumière d'ambiance, qui sera rajoutée à tous nos points. Ainsi, un objet complètement à l'ombre sera quand-même visible.

- Comme toujours, libre à vous d'intégrer ce facteur à votre sauce. Et pourquoi pas, faire des fonctions de perturbation de la lumière d'ambiance...
  Un possibilité simple consiste à ne pas baisser le $cos(a)$ de la luminosité en dessous de 0.10 par exemple.

Lumière parallèle

- Nous avons jusqu'ici utilisé des sources de lumière ponctuelles (nos spots). La lumière est émise dans toutes les directions.

- Cet aspect ponctuel n'est pas vraiment conforme à ce que l'on peut observer dans la réalité. Il est possible de créer une source de lumière parallèle. Elle n'est pas située à des coordonnées précises, mais elle dispose d'un vecteur indicant son orientation.

- Concrètement, le vecteur $\vec{L}$ (entre le spot et le point P) sera pour une lumière parallèle l'opposé du vecteur d'orientation de cette lumière.

- *Attention:* dans ce cas particulier, l'ombre devra tenir compte de TOUS les objets se trouvant devant le point P (donc pour tous $k > 0$) là où pour un spot normal, seuls les objets entre P et S étaient interessants ($0 < k < 1$).

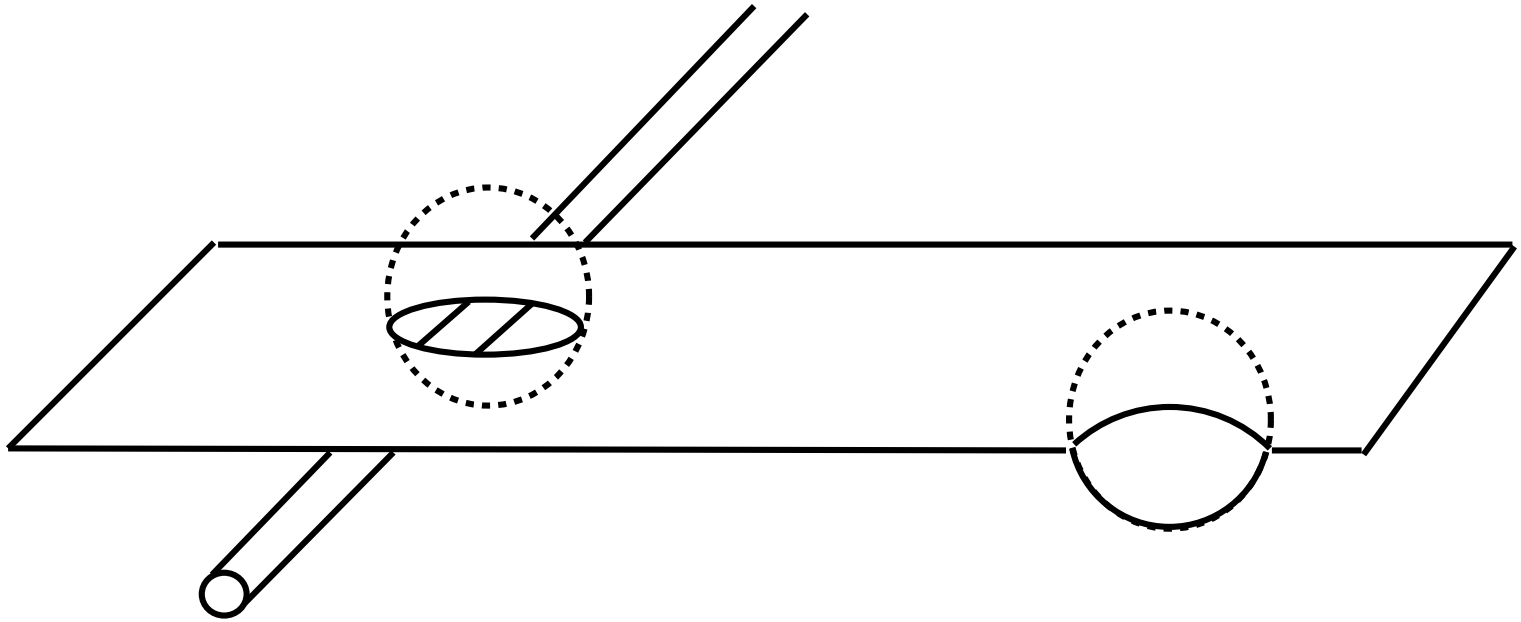
Objets négatifs

- Il s'agit d'*enlever* un objet à un autre. C'est un système de découpage d'une forme par une autre forme. L'objet négatif est généralement fermé.
  On peut distinguer deux méthodes pour réaliser cet effet:

- La première consiste à répondre "objet non touché" si le point d'intersection se trouve à l'interieur de l'objet négatif. Cela a pour effet de laisser un trou dans la forme originale.

- La seconde réalise une empreinte de l'objet négatif, simulant une déformation de la forme de départ. C'est ici une partie de l'objet négatif qui va servir de surface modifiée sur l'objet initial.

Objets - -



Technique négative envisageable

- Il y a bien sûr plusieurs techniques pour réaliser ce dispositif d'objets négatif. Celle présentée ici tente d'être au plus près de l'architecture du raytracer vue jusqu'à présent.
  Il s'agit de la première des deux solutions envisagées.

- L'objet négatif est dans un premier temps considéré comme un objet normal. On réalise les calculs d'intersection de manière classique.

- Il va cependant falloir conserver la totalité des intersections d'un rayon. En effet, si notre intersection la plus proche est un objet négatif, on ne peut pas la conserver. Il faut classer les intersections, et sélectionner la première qui correspond à un objet classique se trouvant hors tout objet négatif.

Goodies pour le projet *Pauvre Raymond*

- Un fichier de configuration où l'on trouve la description de notre scène, les objets et leurs caractéristiques.

- Un fichier de configuration complexe permettant toutes les fantaisies, un mini-language pour integrer des fonctions de perturbation originales ou des équations d'objets chiadés.

- Pourquoi pas un format compatible avec les modeleurs ou les raytracers du marché ?

Des objets exotiques ?

- Le paraboloïde

  Cet objet est de la même famille que la sphère, le cylindre et le cône. Son équation en position simple, suivant l'axe des z est la suivante :

  $x^2 + y^2 - cte \times z = 0$ et la normale : $\left| \begin{array}{l} x_P \\ y_P \\ -cte \end{array} \right.$

- L'hyperboloïde

  Mêmes remarques que pour l'objet précédent.

  $x^2 + y^2 - cte \times z^2 - 1 = 0$ et la normale : $\left| \begin{array}{l} x_P \\ y_P \\ -cte \times z_P \end{array} \right.$

Ou d'un univers parallèle ?

- Les nappes

  Il s'agit d'équations mathématiques de la forme $z = f(x, y)$.
  Vous pouvez les considérer comme des objets à part entière,
  ou bien comme des fonctions de perturbation d'un plan.
  Attention, le calcul de la normale nécessite de solides
  connaissances mathématiques. Exemple :

  $$z = cte_1 \times \frac{cos(\frac{\sqrt{x^2+y^2}}{cte_2})}{1+\sqrt{x^2+y^2}}$$

- Le torre

  Un torre, c'est une bouée, une chambre à air, comme vous voulez.
  Voici son équation paramétrique :

  $$\left| \begin{array}{l} x = (a + Rcos(u))cos(v) \\ y = (a + Rcos(u))sin(v) \\ z = Rsin(u) \end{array} \right.$$

  a est le rayon central du torre, R est le rayon de la coupe
  transversale, u et v sont les paramètres variables (comme
  k dans nos équations de droite). Là c'est nettement plus costaud.

Peut-être de plus loin encore...

- Le ruban de Moebius

  C'est mon préféré:

  $$\left| \begin{array}{l} x = (a + u \times sin(\frac{v}{2}))cos(v) \\ y = (a + u \times sin(\frac{v}{2}))sin(v) \\ z = u \times cos(\frac{v}{2}) \end{array} \right.$$

  A vômir, non ?