

Algorithmique graphique

M. AMMI
ammi@ai.univ-paris8.fr

1

Historique

- 1944 : Le premier ordinateur électrique et programmable est construit aux États-Unis
- 1950 : MIT, premier écran (vectoriel) contrôlé par ordinateur
- 1962 : P. Bezier (Renault) met au point une méthode pour tracer des courbes ou des surfaces
- 1963 : Sketchpad (thèse de Ivan Sutherland), propose un premier modèle complet de système graphique interactif (électionner, pointer, dessiner, éditer); identifie les structures de données et algorithmes nécessaires. Avancée majeure dans le domaine du graphisme
- 1965 : J. Bresenham met au point un algorithme efficace pour le tracé de lignes
- 1969 : J. Warnock propose un algorithme de subdivision pour la suppression des faces cachées
- 1971 : H. Gouraud fait une thèse avec Evans à l'Université d'Utah où il met au point un algorithme de lissage d'ombres
- 1971 : R. Goldstein et R. Nagel posent les bases de la CSG (Constructive Solid Geometry)
- 1972 : R. Shoup (Xerox) met au point le premier frame-buffer 8 bits
- 1974 : E. Catmull (Utah) soutient sa thèse sur le placage de texture, le Z-buffer et le rendu de surfaces courbes
- 1974 : B. Phong (Utah) invente un nouvel algorithme de lissage d'ombres
- 1974 : Sutherland et Hodgman développent un algorithme de clipping de polygones
- 1977 : J. Bresenham met au point un algorithme efficace pour le tracé de cercles
- 1978 : Cyrus et Beck proposent un algorithme de clipping de segments

2

Historique

- 1979 : James H. Clark conçoit le premier processeur programmable dédié au graphisme 3D
- 1982 : création de Silicon Graphics, d'Adobe et d'AutoDesk
- 1984 : premiers travaux sur la radiosité à Cornell University
- 1985 : création d'ATI : conception de circuits intégrés graphiques
 - 1987 : création de la 1^{re} carte graphique
- 1987 : Marching Cubes : A High Resolution 3D Surface Construction Algorithm, par Lorensen et Cline (GE)
- 1992 : OpenGL 1.0
- 1993 : création de NVIDIA
- 1993 : Jurassic Park
- 1994 : standard VRML
- 1996 : Microsoft lance DirectX
- 1997 : Sun lance Java 3D
- 1998 : logiciel Alias Maya
- 2007 : Spécifications d'OpenGL 3.0

Pour plus de détails : <http://accad.osu.edu/~waynec/history/timeline.htm>

Domaines d'application

- Art & divertissement
 - Films d'animation, Effets spéciaux
 - Jeux (Temps réel, Interaction)

Domaines d'application

- Suivi de processus et Téléopération (online)

5

Domaines d'application

- Simulateurs (offline)
 - Conduite/Pilotage, Processus, Téléopération, Jeux!

6

Domaines d'application

- Visualisation de données
 - Médicales
 - Géophysiques
 - Biologiques

Domaines d'application

- Conception assistée par ordinateur
 - Immersive ou pas

Définitions

- Définition officielle de l'infographie (office de la langue française)
 - « Application de l'informatique à la création, au traitement, et à l'exploitation des images numériques »
- Infographie est un mot formé à partir de d'INFOmatique et GRAPHIque
 - Appellation déposée par la société Benson en 1974

9

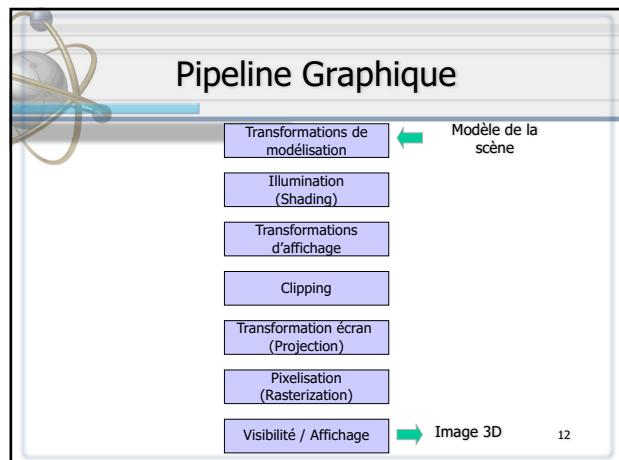
Définitions

- La géométrie est l'élément constitutif essentiel de toute image de synthèse
- Les objets présents dans une image de synthèse sont toujours définis à partir d'objets géométriques dont l'expression mathématique est connue.
- Ces objets ainsi que l'organisation spatiale de ces objets les uns par rapport aux autres forment un modèle géométrique complet de la scène virtuelle à représenter et constituent un pré-requis essentiel.
- L'image de synthèse obtenue est le résultat de l'interaction entre les différentes sources de lumière de la scène et les objets. Cette interaction s'évalue à partir des caractéristiques physiques des objets et des sources de lumières, et des propriétés géométriques locales en tout point de la scène (géométrie locale) et globales (chaque objet interagit avec une partie ou la totalité de la scène).

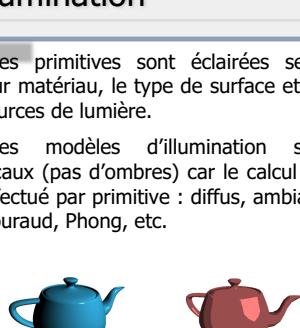
10

Pipeline Graphique

11



Transformations de modélisation

Illumination	
Transformations de modélisation	• Les primitives sont éclairées selon leur matériau, le type de surface et les sources de lumière.
Illumination (Shading)	• Les modèles d'illumination sont locaux (pas d'ombres) car le calcul est effectué par primitive : diffus, ambiant, Gouraud, Phong, etc.
Transformations d'affichage	
Clipping	
Transformation écran (Projection)	
Pixelisation (Rasterization)	
Visibilité / Affichage	

The diagram shows a vertical stack of seven colored rectangles representing stages in the rendering pipeline:

- Transformations de modélisation
- Illumination (Shading)
- Transformations d'affichage** (highlighted in orange)
- Clipping
- Transformation écran (Projection)
- Pixelisation (Rasterization)
- Visibilité / Affichage

To the right of the diagram, a 3D coordinate system labeled "World space" shows a cube. A camera is positioned above it, with a dashed cone representing its field of view. The intersection of the camera's frustum and the cube is highlighted in red. An arrow points from this intersection area to a second 3D coordinate system labeled "Eye space", which shows the resulting frustum and the cube's vertices. The "Eye space" diagram includes axes labeled u, v, and w.

The diagram illustrates the Clipping stage in the rendering pipeline. It shows the transition from Eye space to Normalized Device Coordinates (NDC).

Clipping

- Passage en coordonnées normalisées normalisées (NDC : normalized device coordinates)

Eye space → NDC

• Suppression des parties hors du volume de vision.

16

The diagram illustrates the 3D pipeline from NDC to Screen Space.

Transformations de modélisation

Transformation écran (Projection)

Transformations d'affichage

Clipping

Pixelisation (Rasterization)

Visibilité / Affichage

Les primitives 3D sont projetées sur l'espace image 2D (screen space)

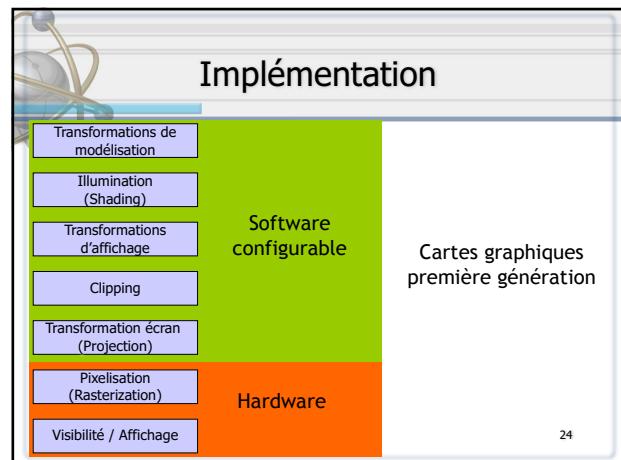
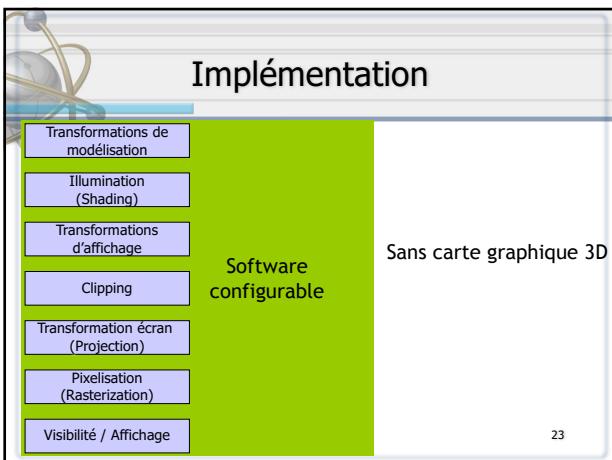
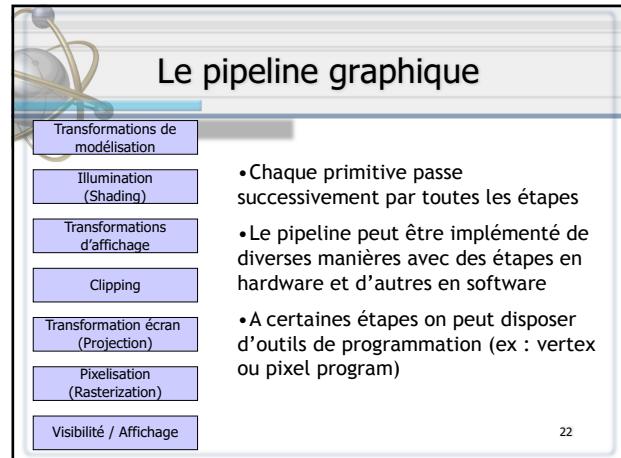
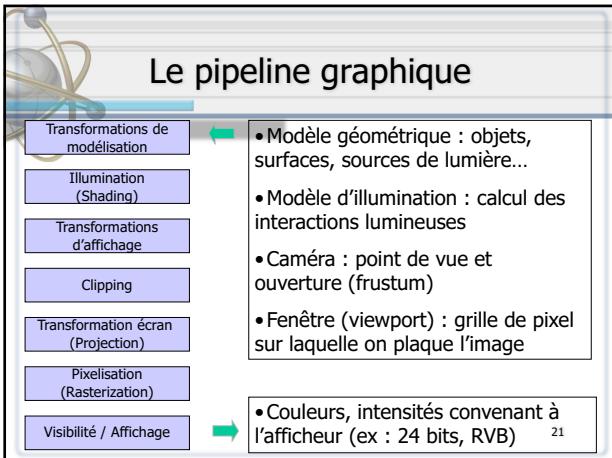
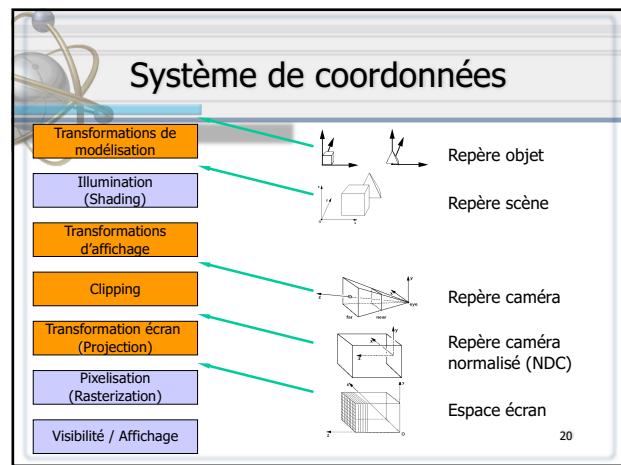
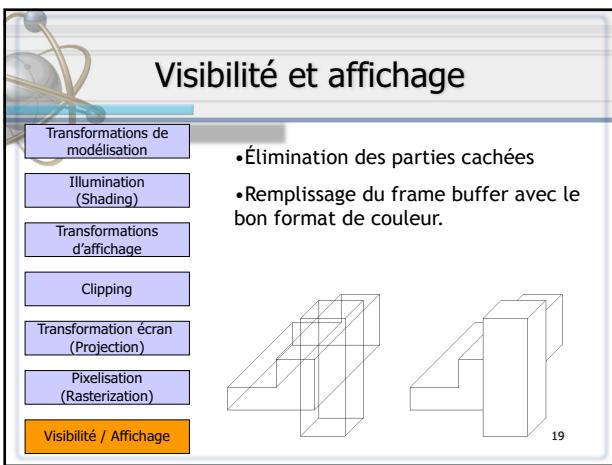
The diagram shows a 3D cube representing NDC (Normalized Device Coordinates) on the left, which is transformed by projection into a perspective view in Screen Space on the right. Below this, a 3D scene in Eye Space is shown, with a purple horse being rasterized into a grid of pixels in Screen Space on the far right.

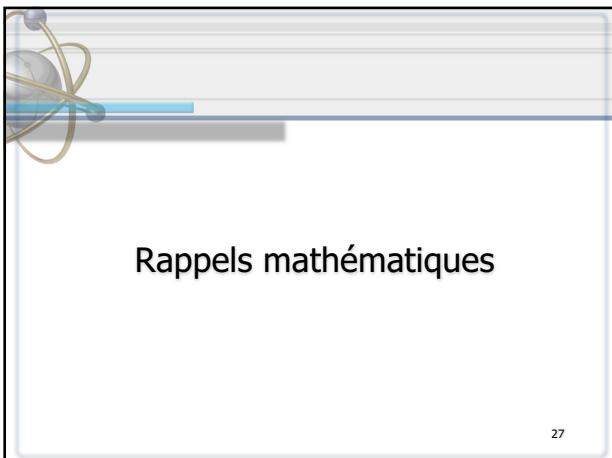
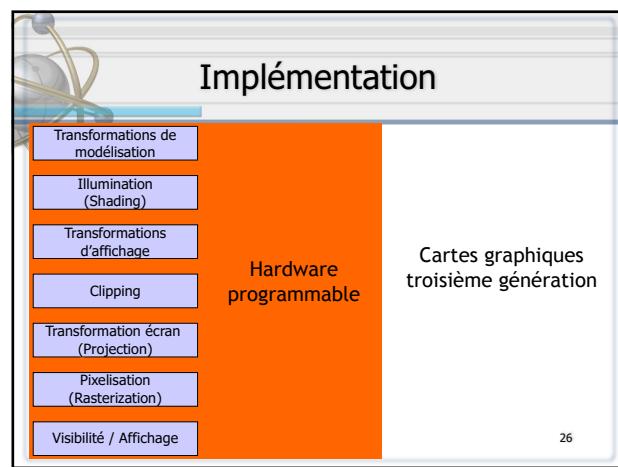
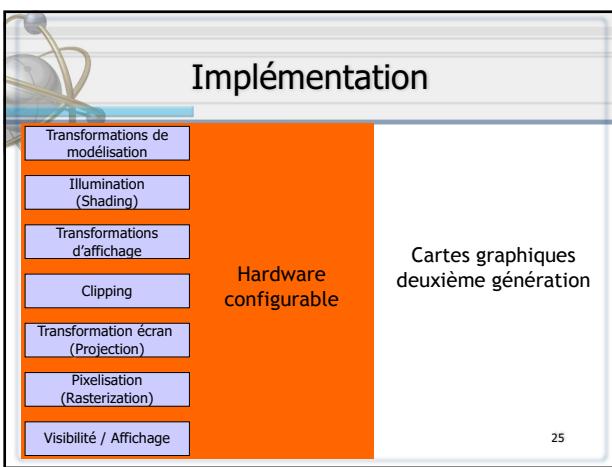
Rasterisation

- Transformations de modélisation
- Illumination (Shading)
- Transformations d'affichage
- Clipping
- Transformation écran (Projection)
- Pixelisation (Rasterisation)**
- Visibilité / Affichage

- Découpe des primitives 2D en pixels
- Interpole les valeurs connues aux sommets : couleur, profondeur, etc. pour chaque fragment affiché

A diagram illustrating rasterization. A triangle is defined by vertices at (10, 10), (50, 10), and (20, 50). It is filled with a diagonal hatching pattern. The triangle is overlaid on a 10x10 grid of small squares, representing pixels. The interior of the triangle is shaded with a green-to-blue gradient, showing the interpolation of values across the primitive.





Scalaire

- Un scalaire est une grandeur totalement définie par un nombre et une unité.
- Il a une valeur numérique mais pas d'orientation.
- Ex :
 - Masse
 - Distance
 - Température
 - Volume
 - Densité
 - Etc.
- Les scalaires obéissent aux lois de l'algèbre ordinaire

At the bottom right corner of the slide is the number 28.

Scalaire

- Opérations élémentaires:
 - Addition & multiplication
 - Propriétés
 - Commutativité $\alpha + \beta = \beta + \alpha$
 - $\alpha \cdot \beta = \beta \cdot \alpha$
 - Associativité $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
 - Distributivité $\alpha \cdot (\beta + \gamma) = (\alpha \cdot \beta) + (\alpha \cdot \gamma)$
- Identité
 - Addition : 0 $\alpha + 0 = 0 + \alpha = \alpha$
 - Multiplication : 1 $\alpha \cdot 1 = 1 \cdot \alpha = \alpha$

At the bottom right corner of the slide is the number 29.

Vecteur

- Un vecteur est une entité mathématique définie par n valeurs numériques extraites du même ensemble E (par exemple N, Z, R, C,...)
- Ces valeurs numériques décrivent le module et l'orientation du vecteur
- n est appellé la dimension du vecteur.
 - On dit que le vecteur est défini dans En.
 - En est un espace de dimension n.
 - Exemple : dans Z2 , dans R3.
- Ex :
 - Déplacement
 - Vitesse
 - Accélération
 - Force

At the bottom right corner of the slide is the number 30.

$$\begin{pmatrix} x \\ y \end{pmatrix}_{Z2} \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{R3}$$

Vecteurs unitaires

- Dans un repère les vecteurs sont décrit dans une base (unités de l'ensemble)
 - généralement unitaire
- Le vecteur est représenté par l'addition des vecteurs unitaires à chaque axe de coordonnées, en multipliant chacun par la projection (composante) respective du vecteur.

$$\vec{V} = x'\vec{i} + y'\vec{j} + z'\vec{k}$$

31

Vecteur

- Les vecteurs obéissent aux lois de l'algèbre vectorielle
- Opérations élémentaires
 - Produit scalaire
 - Produit vectoriel
 - Addition de vecteurs
 - Produit vecteur-scalaire
 - Normalisation

32

Produit scalaire

- Le produit scalaire de deux vecteurs est le produit du module du premier par la composante du second dans la direction du premier

- Produit scalaire en fonction
 - du module et de l'angle : $\vec{A} \cdot \vec{B} = |\vec{A}| |\vec{B}| \cos \theta$
 - des composantes : $\vec{A} \cdot \vec{B} = A_x B_x + A_y B_y + A_z B_z$

33

Produit scalaire

- Propriétés :
 - Commutativité $u \cdot v = v \cdot u$
 - Distributivité par l'addition $(u+v) \cdot w = u \cdot w + v \cdot w$
 - Distributivité par un scalaire $u \cdot (k \times v) = k \times (u \cdot v)$

34

Produit scalaire

- Angle entre deux vecteurs

$$\vec{A} \cdot \vec{B} = |\vec{A}| |\vec{B}| \cos \theta \Rightarrow \cos \theta = \frac{A_x B_x + A_y B_y + A_z B_z}{|\vec{A}| |\vec{B}|}$$
- Signe du produit scalaire
 - $\vec{A} \cdot \vec{B} > 0 \Rightarrow -90^\circ < \theta < 90^\circ$
 - $\vec{A} \cdot \vec{B} = 0 \Rightarrow \theta = \pm 90^\circ$
 - $\vec{A} \cdot \vec{B} < 0 \Rightarrow -180^\circ < \theta < -90^\circ \text{ ou } 90^\circ < \theta < 180^\circ$

35

Produit scalaire

- Application
 - Projection d'un vecteur sur un autre
 - Élimination des faces cachées
 - Calcul d'angle entre deux vecteurs
 - Calcul de la quantité de lumière perçue par une face
 - Ombrage
 - Etc.

36

Produit vectoriel

- Le module du produit vectoriel de deux vecteurs est le produit du module du premier par la composante du second qui est perpendiculaire au premier.

$$\|\vec{A} \times \vec{B}\| = \|\vec{A}\| \|\vec{B}\| \sin \theta = \|\vec{B}\| \|\vec{A}\| \sin \theta = \|\vec{A}\| \|\vec{B}\| \sin \theta$$

- Le produit vectoriel est un vecteur perpendiculaire à A et à B dont le sens est donné par la règle de la main droite.

$$\vec{A} \times \vec{B} = (\|\vec{A}\| \|\vec{B}\| \sin \theta) \vec{u}_n$$

- Le produit vectoriel est nul si les deux vecteurs sont parallèles et maximal s'ils sont perpendiculaires.

37

Produit vectoriel

Produit vectoriel en fonction des composantes:

$$\vec{A} \times \vec{B} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix} = +\vec{i} \begin{vmatrix} A_y & A_z \\ B_y & B_z \end{vmatrix} - \vec{j} \begin{vmatrix} A_x & A_z \\ B_x & B_z \end{vmatrix} + \vec{k} \begin{vmatrix} A_x & A_y \\ B_x & B_y \end{vmatrix}$$

$$\vec{A} \times \vec{B} = \vec{i}(A_y B_z - B_y A_z) - \vec{j}(A_x B_z - B_x A_z) + \vec{k}(A_x B_y - B_x A_y)$$

Exemple:

$$\vec{A} \times \vec{B} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 1 & 2 & -4 \\ 3 & -1 & 5 \end{vmatrix} = +\vec{i} \begin{vmatrix} 2 & -4 \\ -1 & 5 \end{vmatrix} - \vec{j} \begin{vmatrix} 1 & -4 \\ 3 & 5 \end{vmatrix} + \vec{k} \begin{vmatrix} 1 & 2 \\ 3 & -1 \end{vmatrix}$$

$$\vec{A} \times \vec{B} = \vec{i}(2 \times 5 - (-1) \times (-4)) - \vec{j}(1 \times 5 - 3 \times (-4)) + \vec{k}(1 \times (-1) - 3 \times 2)$$

38

Produit vectoriel

- Propriétés**
 - Anticommutativité $(u \times v) = -(v \times u)$
 - Distributivité sur l'addition $(u + v) \times w = u \times w + v \times w$
 - Distributivité par un scalaire $(u + v) \cdot k = u \cdot k + v \cdot k$
 - Non-associativité $(u \times v) \times w \neq u \times (v \times w)$

39

Produit vectoriel

Application

- Calcul de la normale à un plan

40

Addition de deux vecteurs

- L'addition de deux vecteurs obéit à la règle du parallélogramme.

- Algébriquement cela se met en oeuvre en additionnant les composantes individuellement.

$$\vec{A} = a_x \vec{i} + a_y \vec{j} + a_z \vec{k}$$

$$\vec{B} = b_x \vec{i} + b_y \vec{j} + b_z \vec{k}$$

$$\vec{A} + \vec{B} = (a_x + b_x) \vec{i} + (a_y + b_y) \vec{j} + (a_z + b_z) \vec{k}$$

41

Addition de deux vecteurs

Un vecteur A peut être décomposé en ses composantes rectangulaires A_x et A_y .

$$A_x = A \cos \theta_A$$

$$A_y = A \sin \theta_A$$

Il est possible d'ajouter des vecteurs en additionnant leurs composantes de ces vecteurs.

$A_x = A \cos \theta_A$	$A_y = A \sin \theta_A$
$B_x = B \cos \theta_B$	$B_y = B \sin \theta_B$
$R_x = A_x + B_x$	$R_y = A_y + B_y$
$R = \sqrt{R_x^2 + R_y^2}$	$\tan \theta_R = \frac{R_y}{R_x}$

42

Norme

- La norme d'un vecteur est sa taille.
- Cette taille est calculée par le théorème de Pythagore, puisque les composantes d'un vecteur forment toujours des triangles rectangles deux à deux.

$$\vec{v} = x' \vec{i} + y' \vec{j}$$

$$|\vec{v}| = \sqrt{(x')^2 + (y')^2}$$

For dimension n on applique le même principe, en faisant que les deux côtés du triangle rectangle soient les projections du vecteur dans un sous-espace de dimension $n-1$ et dans l'axe de la dimension manquante.

$$|\vec{v}| = \sqrt{(x')^2 + (y')^2 + (z')^2}$$

- La norme d'un vecteur AB est la distance de A à B

43

Normalisation d'un vecteur

- La normalisation fait qu'un vecteur devienne unitaire, c'est-à-dire, avec la norme 1.
- Pour normaliser un vecteur il suffit de diviser toutes ses composantes par sa norme.

$$\vec{v}_n = \frac{x' \vec{i} + y' \vec{j} + z' \vec{k}}{\sqrt{(x')^2 + (y')^2 + (z')^2}}$$

$$\vec{v}_n = \left(\frac{x'}{\sqrt{(x')^2 + (y')^2 + (z')^2}}, \frac{y'}{\sqrt{(x')^2 + (y')^2 + (z')^2}}, \frac{z'}{\sqrt{(x')^2 + (y')^2 + (z')^2}} \right)$$

$$\vec{v} = |\vec{v}| \cdot \vec{v}_n$$

44

Matrice

- On appelle matrice M un tableau à deux indices de $n*m$ valeurs numériques extraites du même ensemble E .

Exemple :

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} \end{pmatrix} \quad n = 4, m = 5.$$

- Usuellement n et m sont le nombre de lignes et le nombre de colonnes de la matrice.
- Si $n = m$ la matrice est dite carrée.

45

Matrice

- Opération possible
 - Addition
 - Matrice-matrice
 - Scalaire-matrice
 - Multiplication
 - Matrice-matrice
 - Matrice-vecteur
 - Scalaire-matrice
 - Inversion
 - Transposition

46

Produit matrice par vecteur

- Soient :
 - un vecteur \vec{v} de dimension n ($v_i, 1 \leq i \leq n$)
 - une matrice carrée M de dimension $n \times n$ ($m_{ij}, 1 \leq i \leq n, 1 \leq j \leq n$)
- Le vecteur \vec{w} produit de M par \vec{v} est : $\vec{w} = M \cdot \vec{v}$
- On calcule le produit de chaque ligne de la matrice par le vecteur colonne.

$$w_i = \sum_{k=1}^n m_{ik} \cdot v_k \quad 1 \leq i \leq n$$

47

Produit matrice par vecteur

- Exemple

$$(1 \ 3 \ 5) \times \begin{pmatrix} 2 & 4 \\ 6 & 8 \\ 10 & 12 \end{pmatrix} = (70 \ 88)$$

48

Produit matrice par matrice

- Soient deux matrices M_1 et M_2 de dimensions respectives :
 - $n \times m$ (m_{1ij} , $1 \leq i \leq n$, $1 \leq j \leq m$)
 - $m \times p$ (m_{2ij} , $1 \leq i \leq m$, $1 \leq j \leq p$)
- La matrice M produit de M_1 par M_2 est de dimension $n \times p$ est calculée par la formule suivante :

$$m_{ij} = \sum_{k=1}^m m_{1ik} \cdot m_{2kj} \quad (1 \leq i \leq n, 1 \leq j \leq p)$$

49

Produit matrice par matrice

- Exemple

$$\begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix} \times \begin{pmatrix} 2 & 4 \\ 6 & 8 \\ 10 \end{pmatrix} = \begin{pmatrix} 70 & 88 \\ 178 & 232 \end{pmatrix}$$

50

Espace vectoriel - Espace affine

- Un espace vectoriel** est l'espace où vivent les vecteurs (déplacements ou directions). Ses principales propriétés sont l'existence d'un vecteur nul et la stabilité de l'espace pour toute combinaison linéaire de vecteurs.
- Un espace affine** est l'espace dans lequel vivent les points à partir desquels on définit les objets géométriques usuels (droites, ...). Il se construit à partir d'un point de référence (l'origine) et d'un espace vectoriel (déplacements autorisés à partir de ce point).

51

Transformations

52

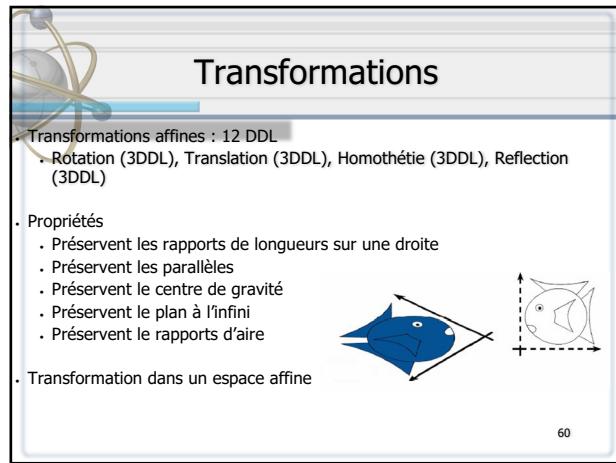
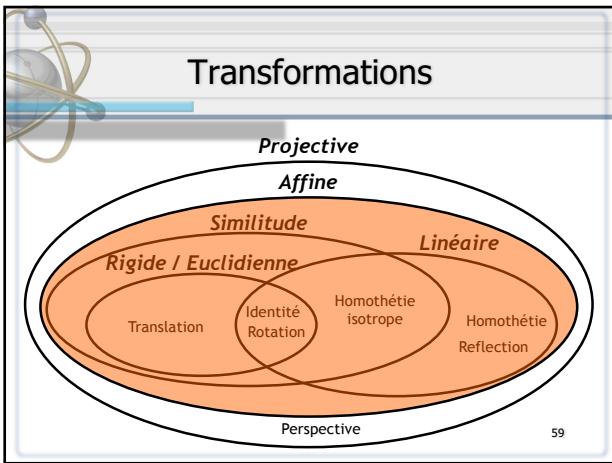
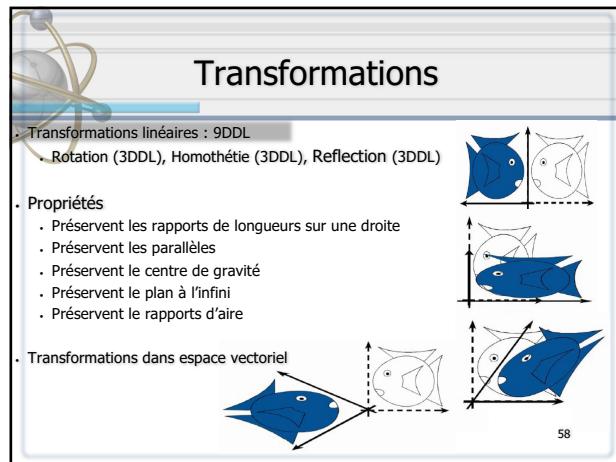
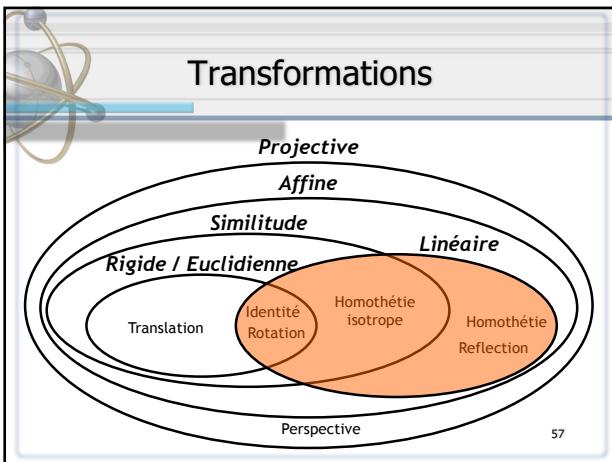
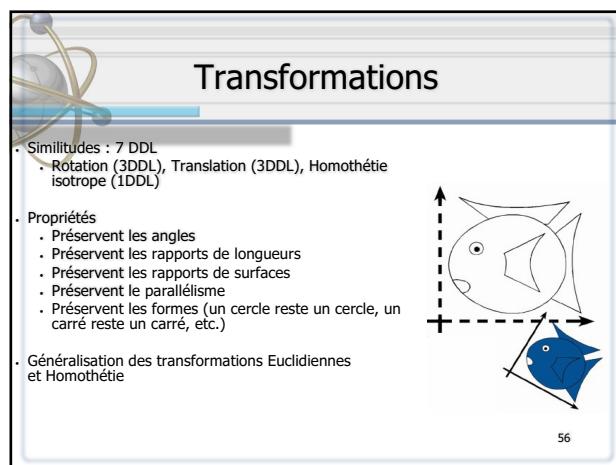
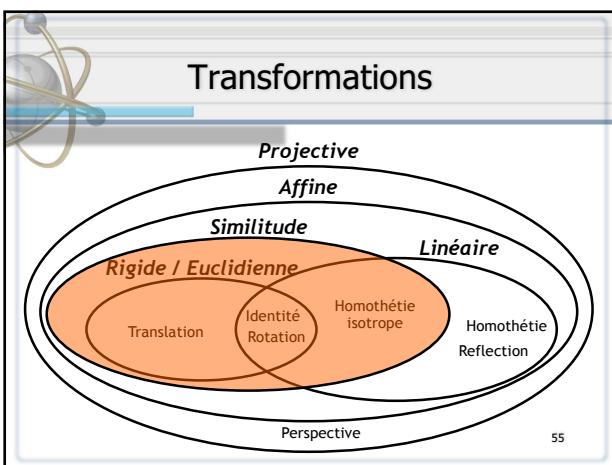
Transformations

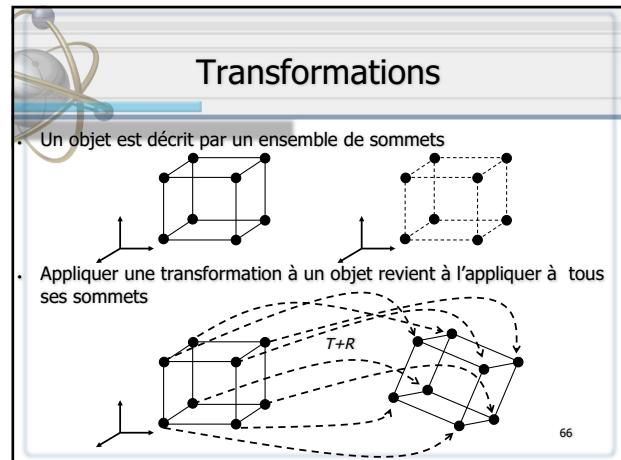
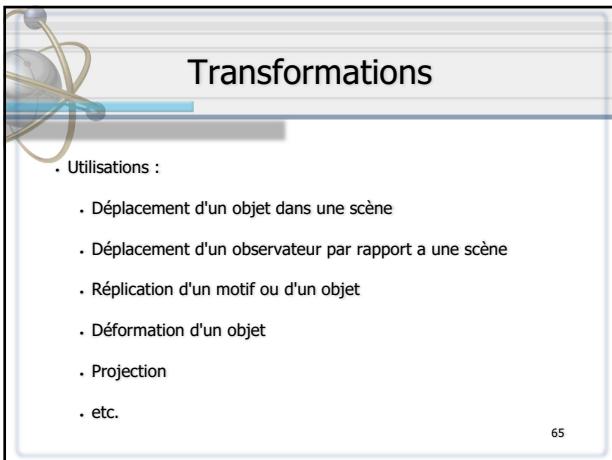
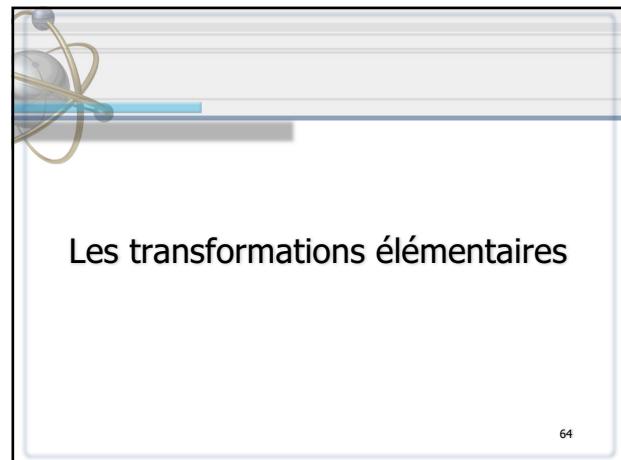
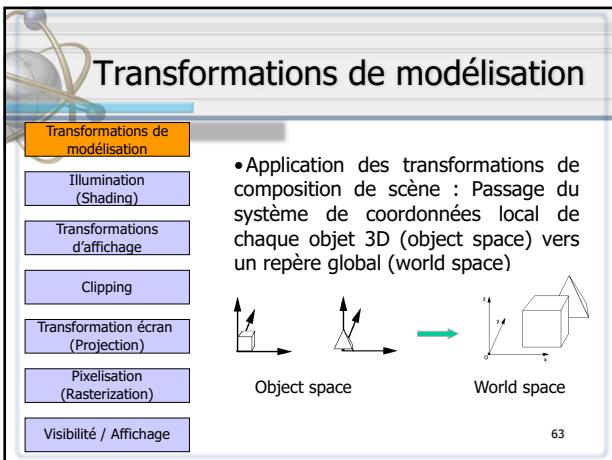
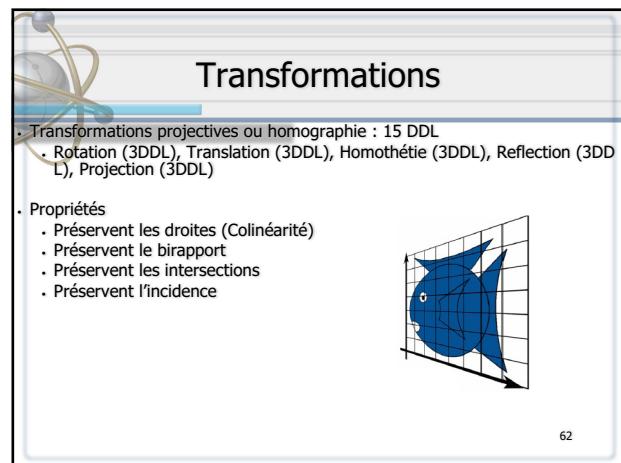
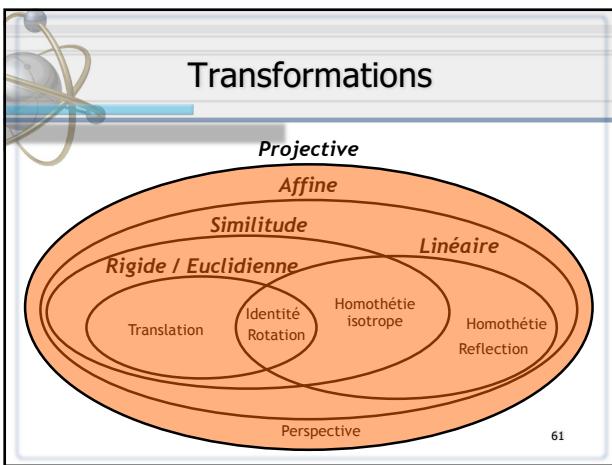
53

Transformations

- Transformations Euclidiennes rigides : 6 DDL
 - Rotation (3DDL), Translation (3DDL).
- Propriétés
 - Préserver les angles.
 - Préserver les distances.

54





Transformations

- On utilise la notation vectorielle
 - Les sommets sont représentés sous forme de vecteurs

$$p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

67

Transformations

- Translation :
 - Ajouter aux coordonnées du sommet la valeur de la translation

$$\begin{aligned} x'_i &= T_x + x_i \\ y'_i &= T_y + y_i \\ z'_i &= T_z + z_i \end{aligned}$$

- Notation matricielle
 - Addition du vecteur de translation

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} + \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

68

Transformations

- Rotation par rapport à l'origine autour de l'axe Z
 - Calcul à l'aide de l'algèbre vectorielle

$$\begin{aligned} x'_i &= x_i \cdot \cos \theta_z - y_i \cdot \sin \theta_z \\ y'_i &= x_i \cdot \sin \theta_z - y_i \cdot \cos \theta_z \\ z'_i &= z_i \end{aligned}$$

- Notation matricielle
 - Multiplication par la matrice de rotation

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

69

Transformations

- Homothétie
 - Multiplication par les facteurs d'échelles

$$\begin{aligned} x'_i &= S_x \cdot x_i \\ y'_i &= S_y \cdot y_i \\ z'_i &= S_z \cdot z_i \end{aligned}$$

- Notation matricielle
 - Multiplication par la matrice d'Homothétie

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

70

Transformations

- La notation matricielle
 - Permet une notation simple, concise
 - Mais pas vraiment unifiée
 - Addition ou bien multiplication en fonction de la transformation (translation, rotation...)
- On veut une notation unique
 - concaténer plusieurs transformations
 - permettre de noter aussi les combinaisons de transformations

71

Cochéances homogènes

- Les coordonnées homogènes sont utilisées en synthèse d'image afin d'unifier le traitement des transformations géométriques d'une scène et de les regrouper dans une seule matrice. En effet, si l'on utilise une matrice 2×2 pour les scènes bidimensionnelles et une matrice 3×3 pour les scènes tridimensionnelles, ces matrices ne peuvent exprimer que des rotations.
- Pour exprimer aussi les translations, les changements d'échelle et les projections, on va utiliser des matrices 4×4 pour les scènes tridimensionnelles.

- On rajoute également une 4ème coordonnée aux points manipulés w ($w=1$) : (x, y, z, w)

72

Coordonnées homogènes

- Ainsi, si (x,y,z) sont les coordonnées d'un point de la scène à transformer et M_H la matrice 4×4 de coordonnées homogènes, on effectuera la multiplication :

$$M_H * \begin{pmatrix} x & y & z & 1 \end{pmatrix}$$

- Ce qui donnera comme résultat $[X \ Y \ Z \ H]$.
- Les coordonnées du point transformé seront alors $(x' \ y' \ z') = (X/H, Y/H, Z/H)$.
- La matrice 4×4 des coordonnées homogènes peut être considérée comme étant composée de 4 sous-matrices, chacune d'elle étant associée à un type de transformation.

73

Manipulations géométriques

- Soit un point p
- Soit une transformation géométrique définie par la matrice M donnée en coordonnées homogènes.
- Le Point p' transformé de par la matrice M est : $p' = M \cdot p$
- Transformations :
 - Translation
 - Rotation
 - Changements d'échelle
 - Symétries
 - Projection
 - Affinités orthogonales
 - etc.

74

Translations

$$T = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} I_3 & T_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{pmatrix}$$

$$P' = T \cdot P = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} T_x + x \\ T_y + y \\ T_z + z \\ 1 \end{pmatrix} = T + P$$

75

Rotations (angles d'Euler)

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{3 \times 3} & 0_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{pmatrix}$$

$$R = R_x \cdot R_y \cdot R_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x & 0 \\ 0 & \sin \theta_x & \cos \theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_y & 0 & \sin \theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 & 0 \\ \sin \theta_z & \cos \theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$P = R \cdot P = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

76

Rotations (angles d'Euler)

- Règles pour la construction de la matrice de rotation d'angle θ
 - Ligne 1 associée à x , ligne 2 à y et ligne 3 à z
 - 1 sur la diagonale pour l'axe de rotation et la coordonnée homogène
 - $\cos(\theta)$ sur la diagonale pour les deux autres axes
 - $\sin(\theta)$ sur les diagonales supérieure et inférieure pour "compléter le carré"
 - Sur la ligne suivant celle de l'axe de rotation, le sinus est précédé d'un signe '-'

77

Homothétie isotrope

$$R = \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} S_{3 \times 3} & 0_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{pmatrix}$$

$$P' = S \cdot P = \begin{pmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S \cdot x \\ S \cdot y \\ S \cdot z \\ 1 \end{pmatrix} = S \cdot P$$

78

Homothétie : Affinités orthogonales

$R = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} S_{3x3} & 0_{3x1} \\ 0_{1x3} & 1 \end{pmatrix}$

$$P' = S.P = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_x.x \\ S_y.y \\ S_z.z \\ 1 \end{pmatrix} = S.P$$

Affinité d'axe x par rapport au plan yOz Affinité d'axe y par rapport au plan xOz Affinité d'axe z par rapport au plan xOy

Glissement

- Appelée aussi *shear* ou *cisaillement* : étirement suivant un axe
- La matrice d'un glissement parallèle à l'axe x de rapport k est :

$$\begin{pmatrix} 1 & k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La matrice d'un glissement parallèle à l'axe x, de rapport k et de ligne de base $y = y_{ref}$ est :

$$\begin{pmatrix} 1 & k & -k.y_{ref} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Composition de Transformations

81

Composition de Transformations

Exemple :

Multiplication de matrices : $p' = T.(S.p) = T.S.p$

$$T.S = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

82

Composition de Transformations

Non-commutatif

homothétie puis translation : $p' = T.(S.p) = T.S.p$

translation puis homothétie : $p' = S.(T.p) = S.T.p$

83

Composition de Transformations

$$T(3,1).S(2,2) = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S(2,2).T(3,1) = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

84

Composition de Transformations

Cas général

$$p' = M \cdot p \text{ et } p'' = M' \cdot p'$$

$$\Rightarrow$$

$$p'' = M' \cdot M \cdot p$$

ou

$$p'' = M'' \cdot p \text{ avec } M'' = M' \cdot M$$

85

Composition de Transformations

Ordre d'Application

$$p'' = M' \cdot M \cdot p \Rightarrow M \text{ puis } M'$$

On applique M à P, puis M' au résultat : l'ordre d'application des transformations se lit de droite à gauche, et non dans le sens de la lecture

86

Composition de Transformations

- Les transformations élémentaires sont définies par rapport à l'origine du repère de la scène
- Pour se placer d'un points quelconque , on doit :
 - Revenir à l'origine du repère (translation) : $T_{P \rightarrow O}$
 - Faire la(les) transformation(s) voulue(s)
 - Se remettre au point de départ (translation) : $T_{O \rightarrow P}$

87

Composition de Transformations

Exemple 1 :

On désire établir la transformation consistant à effectuer une rotation de Θ radians autour de l'axe colinéaire à z passant par le point P de coordonnées (T_x, T_y, T_z) .

88

Composition de Transformations

Cette transformation M est réalisée en amenant p à l'origine par une translation T de $-p$, puis en effectuant une rotation R d'angle Θ_z autour de l'axe Oz, et enfin en ramenant p à sa position initiale par une translation T' de p :

$$p' = (T(x, y, z) \cdot R(\theta_z) \cdot T(-x, -y, -z)) \cdot p = M \cdot p$$

89

Composition de Transformations

$$M = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 & 0 \\ \sin \theta_z & \cos \theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -T_x \\ 0 & 1 & 0 & -T_y \\ 0 & 0 & 1 & -T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 & -T_x \cos \theta_z + T_y \sin \theta_z \\ \sin \theta_z & \cos \theta_z & 0 & -T_x \sin \theta_z + T_y \cos \theta_z \\ 0 & 0 & 1 & -T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} \cos \theta_z & -\sin \theta_z & 0 & -T_x \cos \theta_z + T_y \sin \theta_z + T_x \\ \sin \theta_z & \cos \theta_z & 0 & -T_x \sin \theta_z + T_y \cos \theta_z + T_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

90

Composition de Transformations

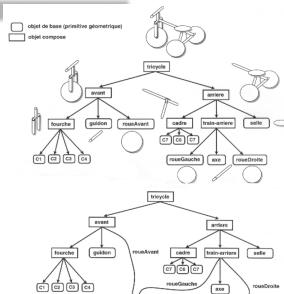
- Les compositions de transformations servent à
 - Décrire une hiérarchie de transformations : graphe de scène
 - Changement de repère

91

Modélisation hiérarchique

- Un modèle hiérarchique permet de décrire facilement des objets complexes composés d'objets simples
- La scène est organisée dans un arbre
- Les objets ne sont plus définis par leur transformation absolue par rapport au repère global, mais par leur transformation relative dans cet arbre :
 - Le repère associé à la racine est le repère de la scène
 - A chaque nœud est associé un repère
 - A chaque arc est associée une transformation géométrique qui positionne l'objet fils dans le repère de son père
- Un objet peut être inclus plusieurs fois dans la hiérarchie :
 - la structure de données est un Graphe Orienté Acyclique (DAG)

Modélisation hiérarchique



93

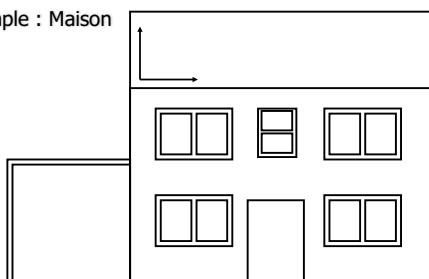
Modélisation hiérarchique

- Construction du graph
 - On commence par un processus descendant ("top-down") dans lequel on effectue une décomposition récursive du modèle géométrique en objets plus simples jusqu'à aboutir à des objets élémentaires (primitives géométriques)
 - On construit la chaîne cinématique (graph) depuis la racine pour aboutir aux nœuds

94

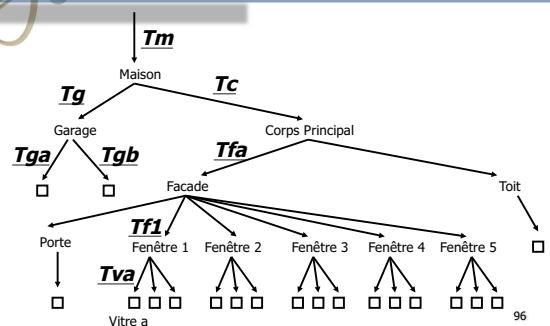
Modélisation hiérarchique

Exemple : Maison



95

Modélisation hiérarchique



96

Modélisation hiérarchique

Pile de transformations

- porte a du garage → Tm.Tg.Tga
- porte b du garage → Tm.Tg.Tgb
- vitre a → Tm.Tc.Tfa.Tf1.Tva
- etc . . .

97

Changement de repère

- Permet de transformer les coordonnées d'un point exprimées dans un premier repère en coordonnées exprimées dans deuxième repère
- Utile lorsque :
 - Les objets manipulés sont définis dans des repères locaux
 - Animation, modélisation, etc.
 - la modélisation des caméras.

98

Changement de repère

- Soient deux repères R_1 et R_2
- Soit M la matrice de passage du repère R_2 au repère R_1
- Soient $(x_2, y_2, z_2, 1)$ les coordonnées de p_2 dans R_2
- Soient $(x_1, y_1, z_1, 1)$ les coordonnées de p_1 dans R_1

99

Changement de repère

$$p_2 = M \cdot p_1$$

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}_{R_1} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}_{2 \rightarrow 1} \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix}_{R_2}$$

$$M = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}_{2 \rightarrow 1}$$

100

Changement de repère

101

Changement de repère

102

Changement de repère

- Passage du repère R1 au repère R2

$$p_1 = M'.p_2 \Rightarrow p_1 = M^{-1} \cdot p_2$$

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix}_{R_2} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}_{R_1}^{-1} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}_{R_1}$$

La projection

104

Transformation écran

- Transformations de modélisation
- Illumination (Shading)
- Transformations d'affichage
- Clipping
- Transformation écran (Projection)**
- Pixelisation (Rasterization)
- Visibilité / Affichage

Les primitives 3D sont projetées sur l'espace image 2D (screen space)

La projection

- La projection est une réduction du nombre de dimensions.
- L'infographie est concernée par les projections de 3-D vers 2-D.

106

La projection

- Les projections des objets sont formées par les intersections de lignes appelées projecteurs avec un plan appelé plan de vue ou plan de projection.
- Les projecteurs sont des lignes partant d'un point arbitrairement appelé centre de projection (CP), en traversant chaque point d'un objet.
- Si les projecteurs sont des lignes droites, et que le plan de projection est plat, la projection est une projection géométrique 2D.

107

Les projections

- Si le centre de projection (CP) est localisé à un point fini de l'espace tridimensionnel, le résultat est une projection en perspective.
- Si le CP est localisé à l'infini, tous les projecteurs sont parallèles et le résultat est une projection parallèle.

Projection perspective

projection parallèle 108

108

Les projections

- Deux familles de projections :
 - Projections parallèles.
 - Projections perspectives.

```

    graph TD
      A[Projections linéaires] --> B[Parallèle]
      A --> C[Perspective]
      B --> D[Orthographique]
      B --> E[Oblique]
      D --> F[Axes principaux]
      D --> G[Axonometric]
      F --> H[Top]
      F --> I[Front]
      F --> J[Side]
      E --> K[Cavalier]
      E --> L[Cabinet]
      E --> M[Autre]
      K --> N[1 point]
      L --> O[2 points]
      M --> P[3 points]
  
```

109

Projections parallèles

- Conditions :
 - La projection se fait dans le plan de projection suivant une direction de projection (DP).
 - Les lignes parallèles restent parallèles
- Propriétés
 - Les projections parallèles conservent les rapports des distances selon une direction donnée
 - Pas réaliste mais peut être utilisée pour des mesures exactes
- Il existe deux types de projections parallèles dépendant du fait que la direction de projection soit perpendiculaire au plan de projection :
 - Projections orthographiques
 - Projections obliques

110

Projections parallèles

- Projection orthographique
 - Projecteurs parallèles entre eux
 - Projecteurs perpendiculaires au plan image
- Projection Oblique
 - Projecteurs parallèles entre eux
 - Projecteurs NON perpendiculaires au plan image

111

Projection orthographique

- Vue de devant, d'en haut, et de côté

112

Projection orthographique

- La projection orthographique consiste à supprimer une dimension
 - La dimension supprimée est celle suivant laquelle on réalise la projection

113

Projection orthographique

- Forme matricielle :

$$p' = M_{PZ} \cdot p$$

$$\begin{pmatrix} x' \\ y' \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

114

Projection orthographique

- Vues de devant, d'en haut et de côté : la projection est faite sur un des plans perpendiculaires aux axes des coordonnées en mettant une des coordonnées du point à 0.

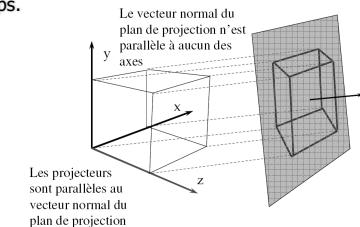
$$M_{PX} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M_{PY} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M_{PZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Elimination de la dimension suivant laquelle nous réalisons la projection

Projection orthographique

Projection orthographique axonométrique

- Le plan de projection n'est pas perpendiculaire aux axes des coordonnées, donc plusieurs facettes de l'objet sont montrées en même temps.



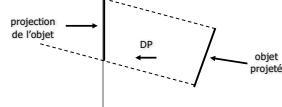
116

Projection oblique

- Direction de la projection n'est pas perpendiculaire au plan de projection
- DP coupe le plan de projection en faisant un angle oblique avec celui-ci.
- Une projection cavalière est obtenue quand l'angle est de 45°
- Pour une projection cabinet l'angle est de 63.4° .
- La matrice de projection est une combinaison de transformations d'étirement et de projection orthographique.

117

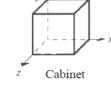
Projection oblique



$\phi = 45^\circ$



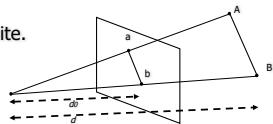
$\phi = 63.4^\circ$



118

Projection perspective

- Réaliste : La taille de la projection d'un objet varie inversement avec sa distance de CP
- $|ab| = do \cdot 1/d \cdot |AB|$
- Ne maintient pas la forme et les mesures exactes
- Les projections perspectives ne conservent pas le parallélisme des droites non parallèles au plan de projection
- La projection de n'importe quel ensemble de lignes parallèles non parallèles au plan de projection convergent vers un même point
 - Le point de fuite.

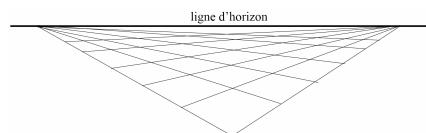


119

Projection perspective

Point de fuite

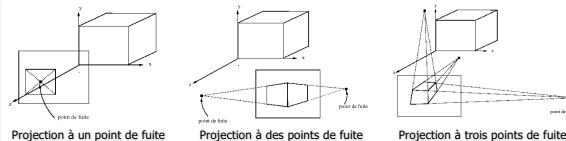
- Si une droite D coupe le plan de projection, il existe un point F, appelé point de fuite appartenant à la projection de toute droite parallèle à D.



120

Projection perspective

- On différencie les projections en perspective par le nombre de points de fuite pour les directions des axes du repère.



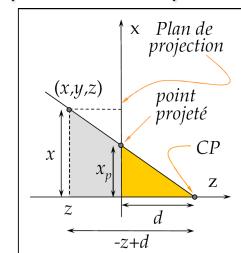
121

Projection perspective

- Calcul d'une perspective de point de fuite unique

$$\begin{aligned} \frac{x_p}{d} &= \frac{x}{-z+d} \Rightarrow \\ x_p &= \frac{x d}{-z+d} \Rightarrow \\ x_p &= \frac{x}{\frac{-z}{d} + 1} = \frac{x}{-zr+1} \end{aligned}$$

où $r = 1/d$



122

Projection perspective

- Calcul d'une perspective de point de fuite unique

$$\begin{aligned} \frac{x_p}{f} &= \frac{x}{-z+f} \Rightarrow \\ x_p &= \frac{x.f}{-z+f} \Rightarrow \\ x_p &= \frac{x}{\frac{-z}{f} + 1} = \frac{x}{-z+1} \end{aligned}$$

Centre de projection

123

Projection perspective

- Calcul d'une perspective de point de fuite unique

$$\begin{aligned} \frac{y_p}{f} &= \frac{y}{-z+f} \Rightarrow \\ y_p &= \frac{y.f}{-z+f} \Rightarrow \\ y_p &= \frac{y}{\frac{-z}{f} + 1} = \frac{y}{-z+1} \end{aligned}$$

Centre de projection

124

Projection perspective

- Représentation en forme matricielle :

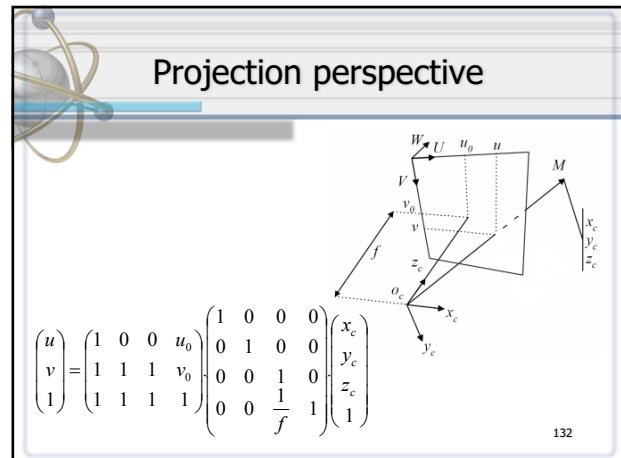
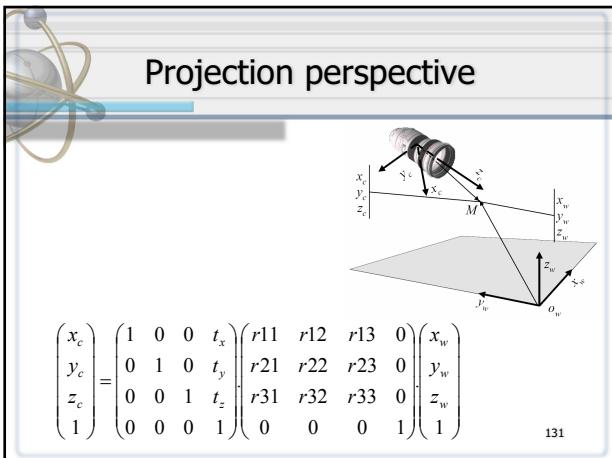
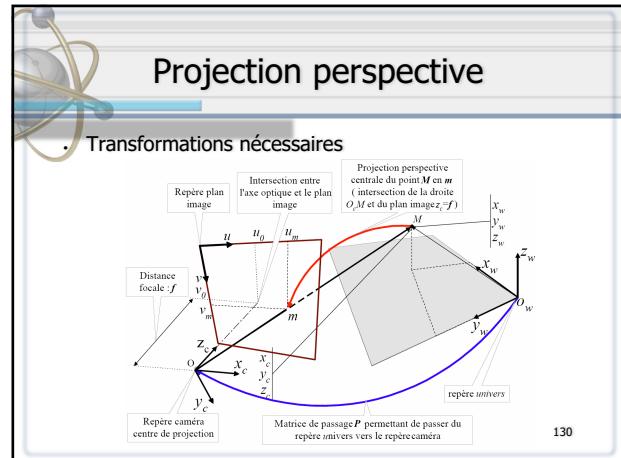
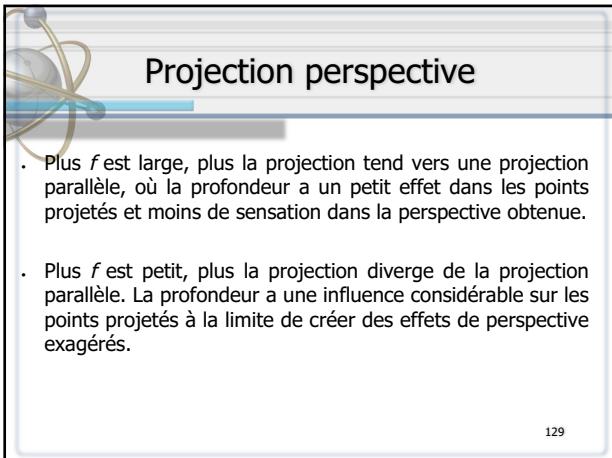
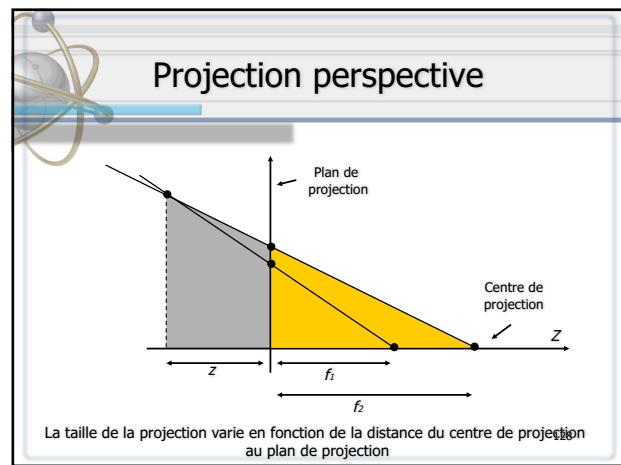
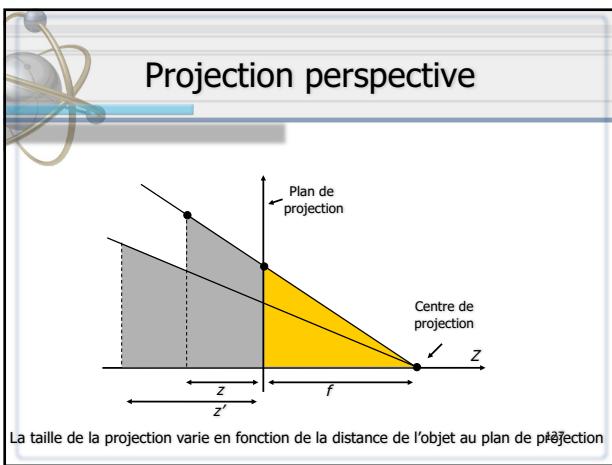
$$\begin{aligned} x_p &= \frac{x}{-\frac{z}{f} + 1}, y_p = \frac{y}{-\frac{z}{f} + 1} \\ \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{f} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \end{aligned}$$

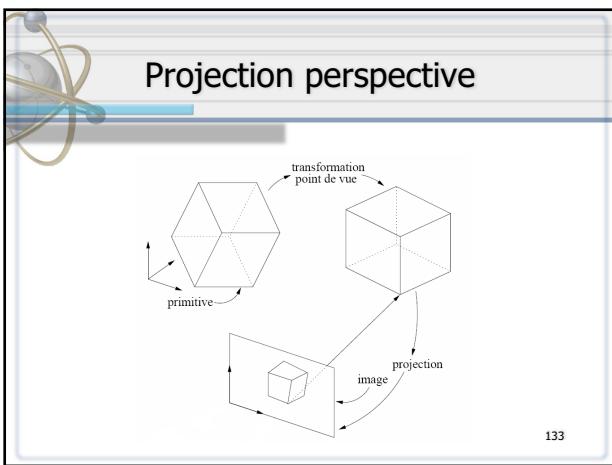
125

Projection perspective

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -\frac{z}{f} + 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x}{-\frac{z}{f} + 1} \\ \frac{y}{-\frac{z}{f} + 1} \\ \frac{z}{-\frac{z}{f} + 1} \\ 1 \end{pmatrix}$$

126





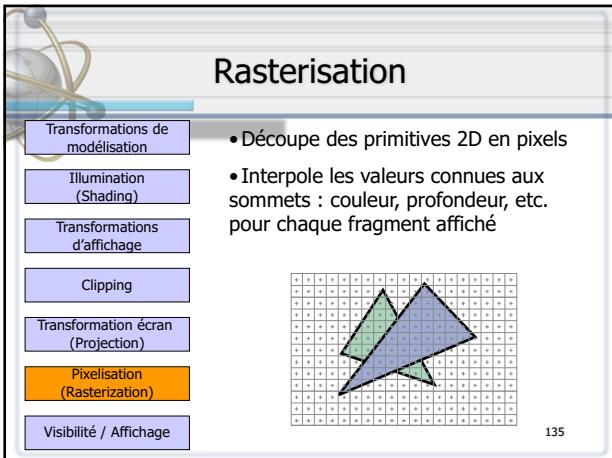
Projection perspective

Passage du repère monde au plan image

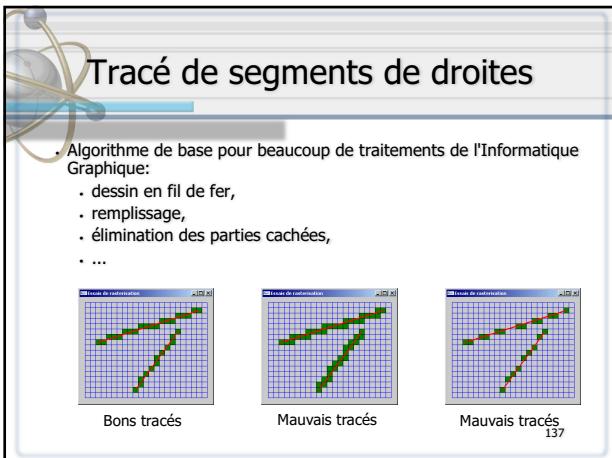
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & u_0 \\ 1 & 1 & 1 & v_0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & f \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 & 0 \\ 1 & f & v_0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}$$

134



- ## Rasterisation
- Traçage :
 - Segments de droites
 - Cercles
 - Ellipses
 - Remplissage des primitives projetées
- 136



- ## Tracé de segments de droites
- Trois impératifs:
- Tout point du segment discret est traversé par le segment continu.
 - Tout point du segment discret touche au moins un autre point soit par l'un de ses cotés (4-connexité), soit par un de ses sommets (8-connexité).
 - On trace le moins de points possibles.
-
- 4/8 Connexité
8-Connexité
- 138

Tracé de segments par l'équation cartésienne

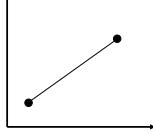
- Le tracé d'un segment entre deux points (x_1, y_1) et (x_2, y_2) de R^2 .
- On pose comme hypothèse simplificatrice :

$x_2 > x_1, y_2 > y_1$ et $(x_2 - x_1) \geq (y_2 - y_1)$

- Tous les autres cas peuvent s'y rapporter.
- On utilise l'équation cartésienne de la droite : $y = ax + b$

avec

$$a = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 + ax_1$$


139

Tracé de segments par l'équation cartésienne

```
DroiteSimple(int x1, int x2, int y1, int y2)
{
    a = y2 - y1 / x2 - x1 ;
    b = y1 - a . x1 ;
    x <- x1 ;
    while (x < x1)
    {
        x <- x1+1 ;
        AfficherPixel(x, (int) a.x+b) ;
    }
}
```

140

Tracé de segments par l'équation cartésienne

- incrémentation suivant l'axe x

```
DroiteSimple(int x1, int x2, int y1, int y2)
{
    a = y2 - y1 / x2 - x1 ; // a = 1.25
    b = y1 - a . x1 ; // b = -0.25
    x <- x1 ;
    while (x < x2) // x : 1 -> 5
    {
        x <- x1+1 ;
        AfficherPixel(x, (int) a.x+b) ;
    }
}
```

Tracé : (1,1) à (5,6)

141

Tracé de segments par l'équation cartésienne

- incrémentation suivant l'axe y

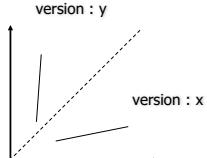
```
DroiteSimple(int x1, int x2, int y1, int y2)
{
    a = y2 - y1 / x2 - x1 ; // a = 1.25
    b = y1 - a . x1 ; // b = -0.25
    x <- x1 ;
    while (y < y2) // y : 1 -> 6
    {
        y <- y+1 ;
        AfficherPixel((int) (y-b)/a, y) ;
    }
}
```

Tracé : (1,1) à (5,6)

142

Tracé de segments par l'équation cartésienne

- Il faut switcher entre les deux versions en fonction de la pente de la droite
- Pente < 1 : version x
- Pente > 1 : version y



143

Tracé de segments par l'équation cartésienne

- Caractéristiques:
- Simplicité algorithmique
- Lenteur due à l'utilisation de réels, d'une division et de multiplications et d'opérations d'arrondissement (cast)

144

Algorithmes de Bresenham

Jack Bresenham
1962

145

Algorithmes de Bresenham

d1-d2 > 0 ?

d1-d2 > 0

146

Algorithmes de Bresenham

d1-d2 = 0

147

Algorithmes de Bresenham

d1-d2 < 0

148

Algorithmes de Bresenham

149

Algorithmes de Bresenham

```
DroiteBresenham(int x0, int x1, int y0, int y1)
{
    dx = / x1 - x0;
    dy = y2 - y1;
    x <= x1 ;
    y <= y1 ;
    e = 2 . dy - dx ;
    IncH = 2 . dy ;
    IncD = 2 . (dy - dx) ;
    while (x < x1)
    {
        AfficherPixel(x, y) ;
        x = x + 1;
        if (e > 0)
        {
            y = y + 1;
            e = e + IncD ;
        }
        else
            e = e + IncH ;
    }
}
```

150

Algorithmes de Bresenham

Principe :

- Paramètre de décision : e (distance pixel -> droite)
- Si $e < 0.5$ pixel => activer le pixel E
- Si $e > 0.5$ pixel => activer le pixel NE

Pixel NE

Pixel E

e

Pixel NE

Pixel E

151

Algorithmes de Bresenham

$e = 0$

$x = x_1$

$y = y_1$

152

Algorithmes de Bresenham

$e = 0$

$x = x_1$

$y = y_1$

153

Algorithmes de Bresenham

$e = 0$

$x = x_1$

$y = y_1$

154

Algorithmes de Bresenham

$e = dy/dx$

$x = x_1$

$y = y_1$

155

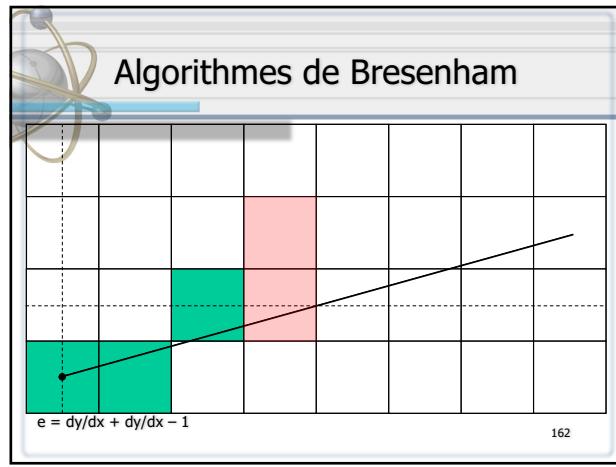
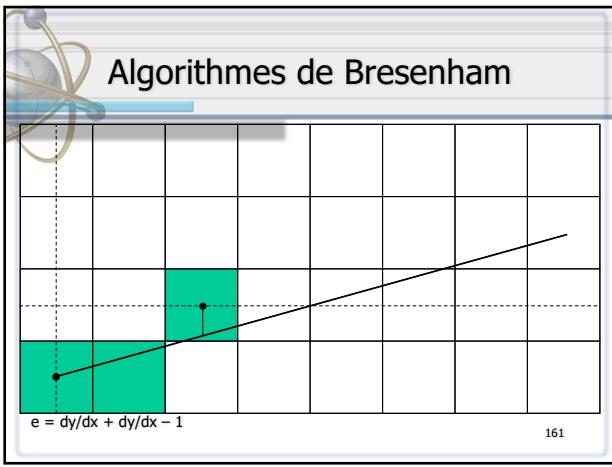
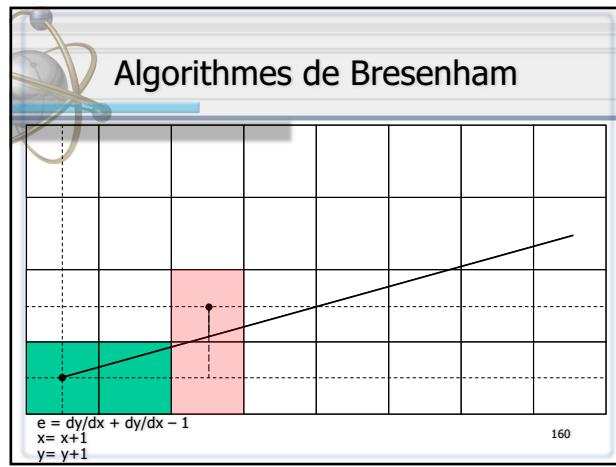
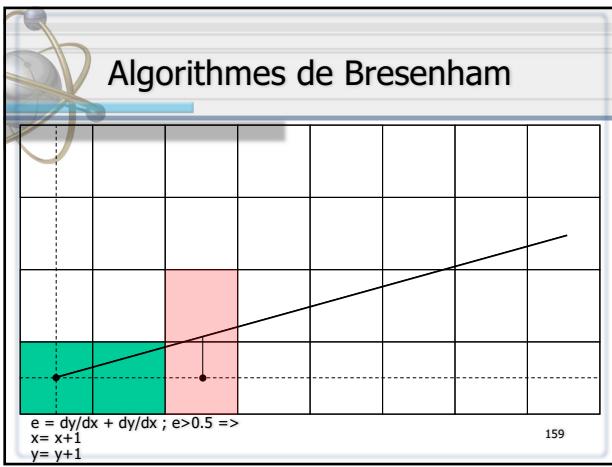
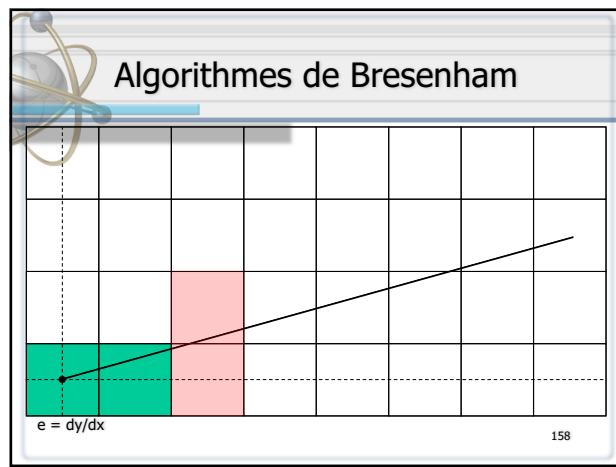
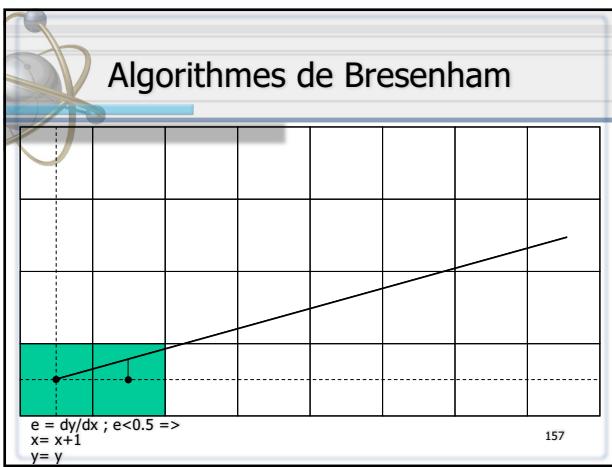
Algorithmes de Bresenham

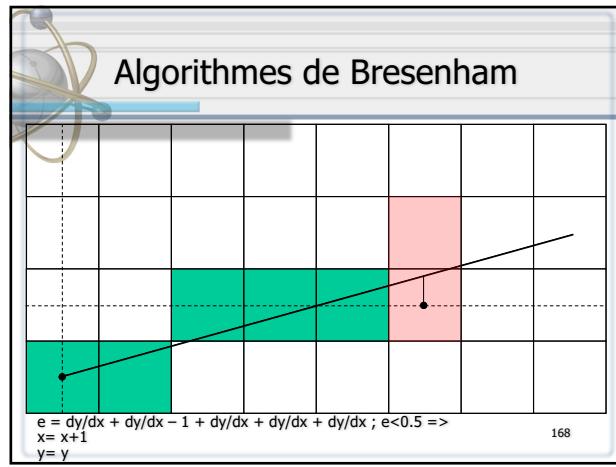
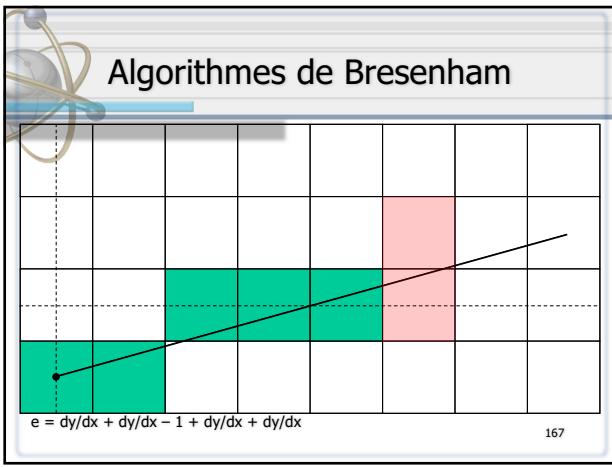
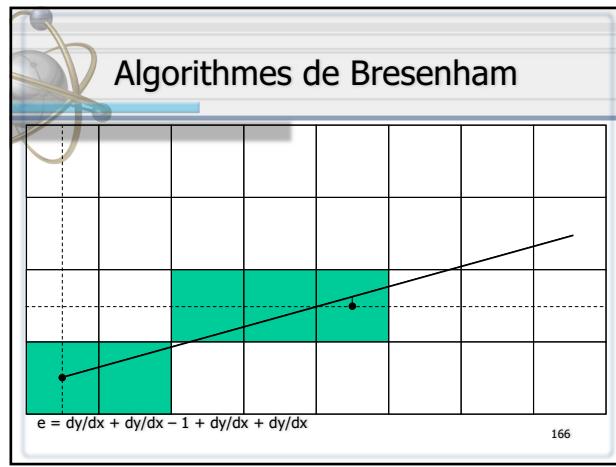
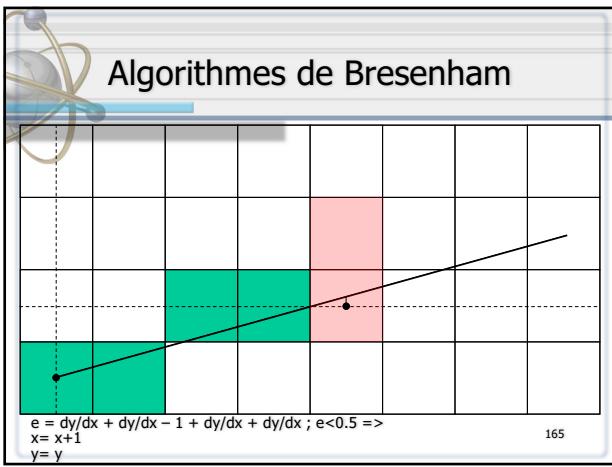
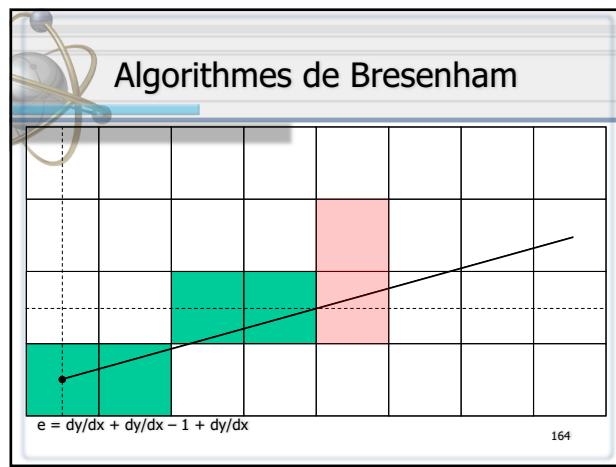
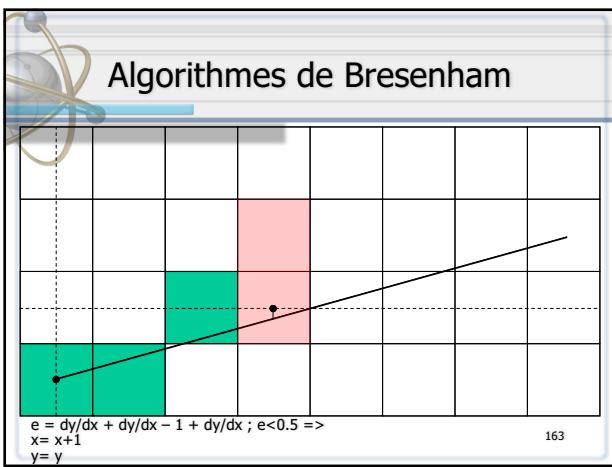
$e = dy/dx ; e < 0.5 =>$

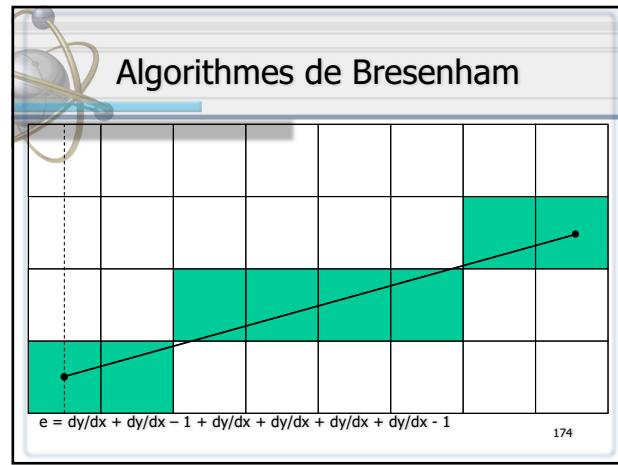
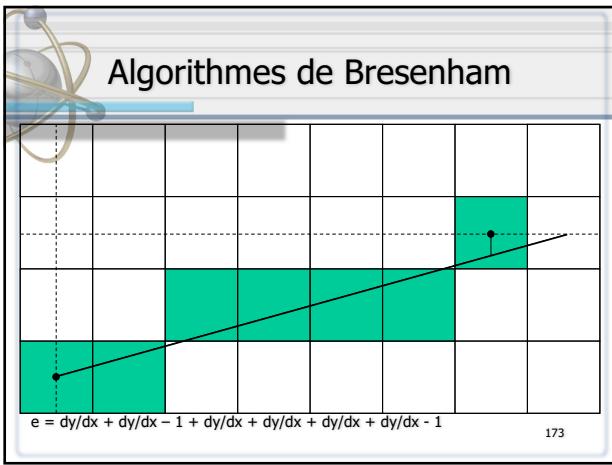
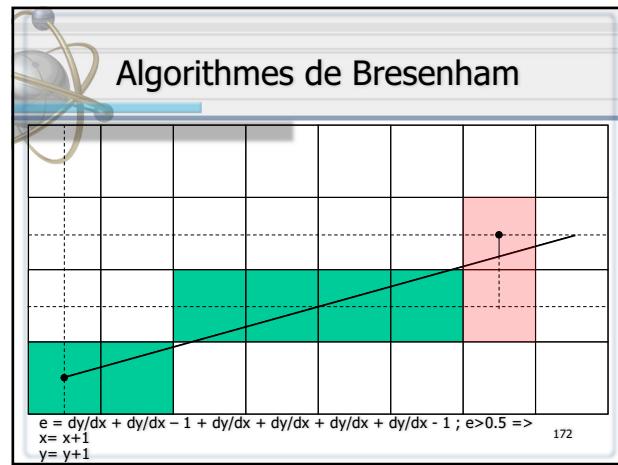
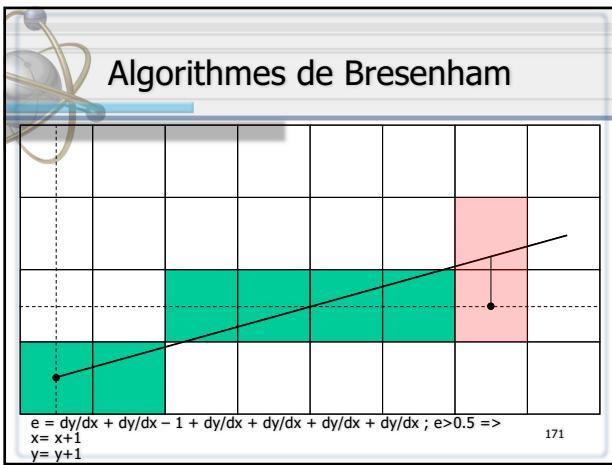
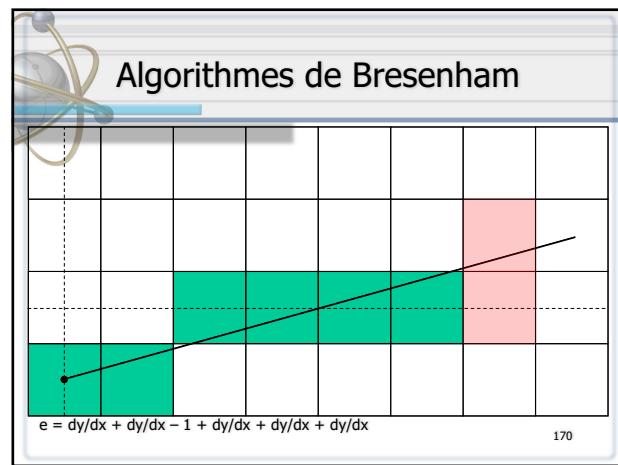
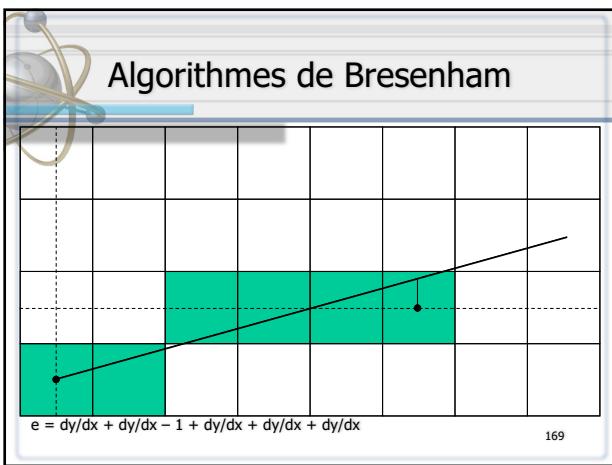
$x = x+1$

$y = y$

156









Algorithmes de Bresenham

- Algorithme de base

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    a = dy / dx ;
    x <= xi ;
    y <= yi ;
    e = 0 ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        e = e + a ;
        x = x + 1;
        if (e > 0.5)
        {
            y = y + 1;
            e = e - 1;
        }
    }
}
```

175



Algorithmes de Bresenham

- supprimer la division par dx

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    x <= xi ;
    y <= yi ;
    e = 0 ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        e = e + dy ;
        x = x + 1;
        if (e > dx/2)
        {
            y = y + 1;
            e = e - dx;
        }
    }
}
```

176



Algorithmes de Bresenham

- division par 2 à supprimer

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    x <= xi ;
    y <= yi ;
    e = 0 ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        e = e + 2 . dy ;
        x = x + 1;
        if (e > dx)
        {
            y = y + 1;
            e = e - 2 . dx;
        }
    }
}
```

177



Algorithmes de Bresenham

- optimisations pour rendre l'algorithme plus rapide : tester le signe de e au lieu de le comparer à un autre nombre

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    x <= xi ;
    y <= yi ;
    e = dx ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        e = e + 2 . dy ;
        x = x + 1;
        if (e > 0)
        {
            y = y + 1;
            e = e - 2 . dx;
        }
    }
}
```

178



Algorithmes de Bresenham

- modifier e qu'une seule fois lors de chaque itération

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    x <= xi ;
    y <= yi ;
    e = 2 . dy - dx ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        x = x + 1;
        if (e > 0)
        {
            y = y + 1;
            e = e + 2 . (dy - d x) ;
        }
        else
            e = e + 2 . dy ;
    }
}
```

179



Algorithmes de Bresenham

- calcul d'avance les deux incréments possibles pour e : déplacement horizontal ou diagonal

```
DoiteBresenham(int xi, int xf, int yi, int yf)
{
    dx = / xf - xi
    dy = yf - yi
    x <= xi ;
    y <= yi ;
    e = 2 . dy - dx ;
    IncH = 2 . dy ;
    IncD = 2 . (dy - d x) ;
    while (x < xf)
    {
        AfficherPixel(x, y) ;
        x = x + 1;
        if (e > 0)
        {
            y = y + 1;
            e = e + IncD ;
        }
        else
            e = e + IncH ;
    }
}
```

180

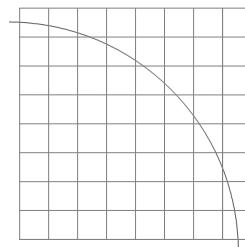
Algorithmes de Bresenham

. Caractéristiques :

- . Plus grande complexité algorithmique.
- . Rapidité due à
 - . Utilisation exclusive d'entiers courts (valeurs maximales de l'ordre de la résolution de l'écran -> petites valeurs)
 - . Opérations arithmétiques simples sur ces entiers (additions, soustractions et comparaisons).

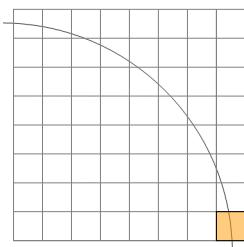
181

Tracé de cercles



182

Tracé de cercles



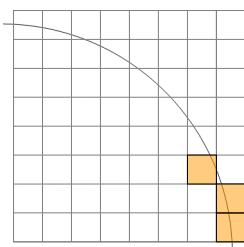
183

Tracé de cercles



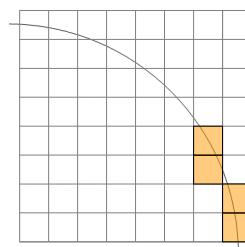
184

Tracé de cercles

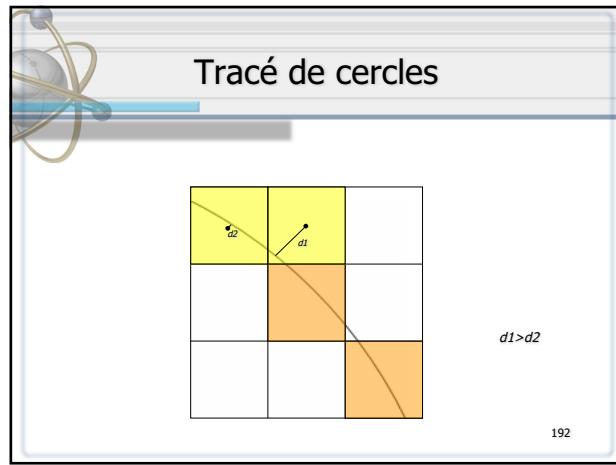
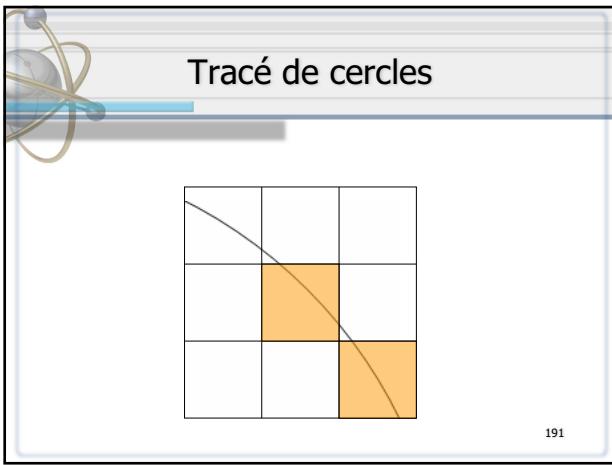
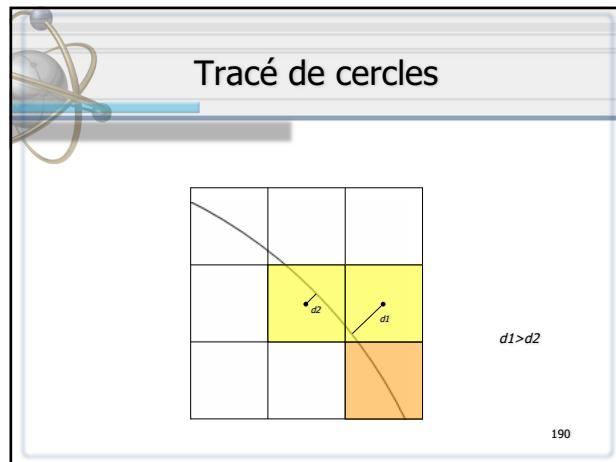
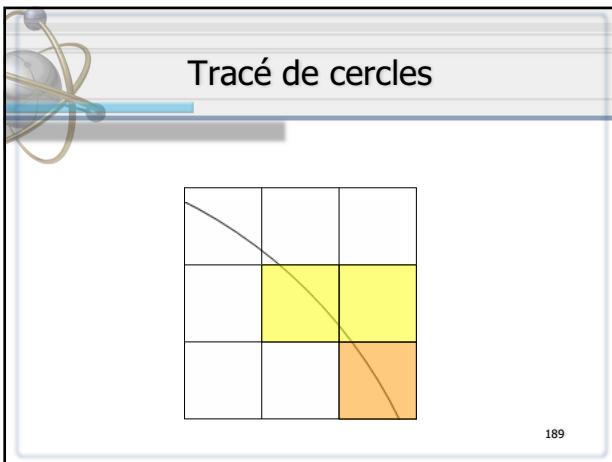
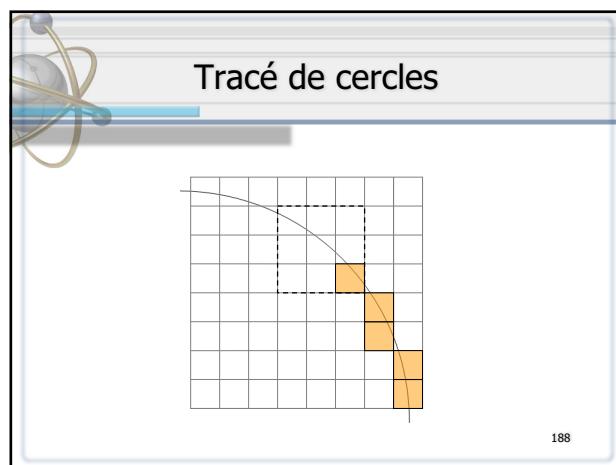
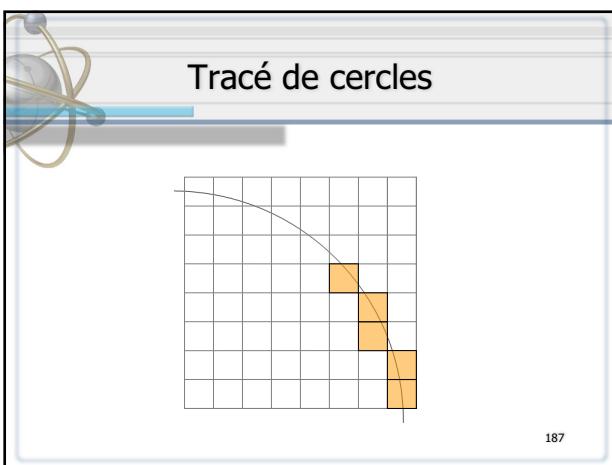


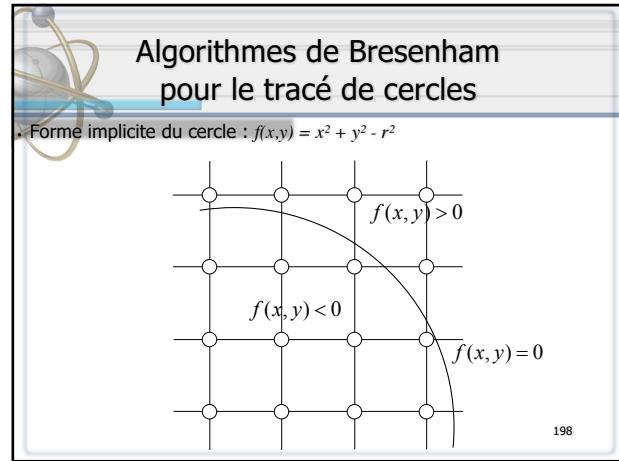
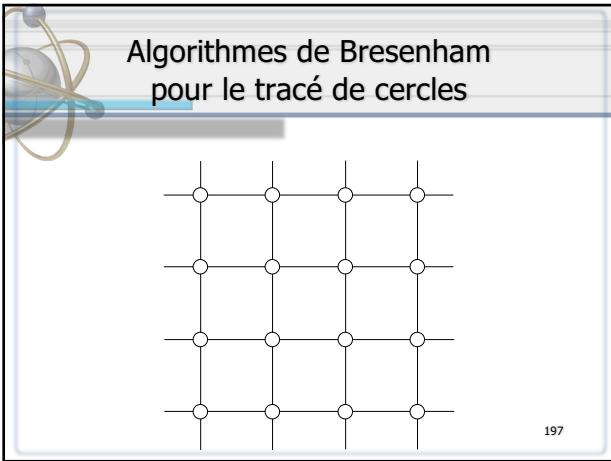
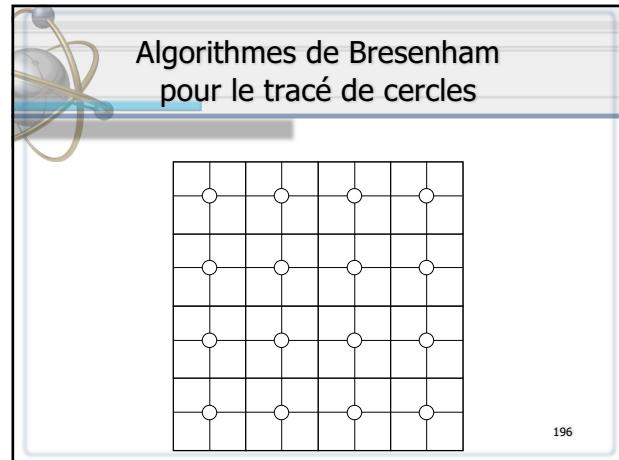
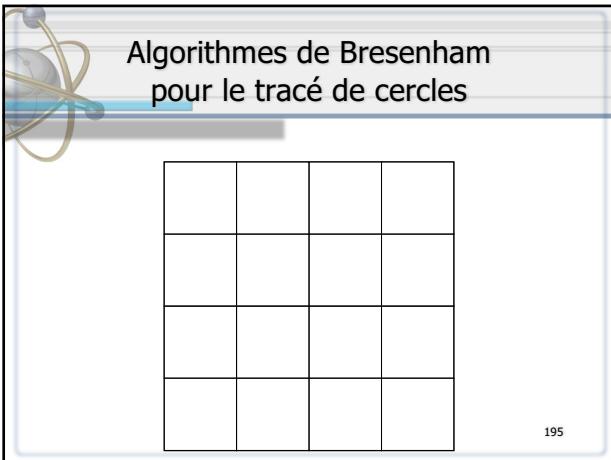
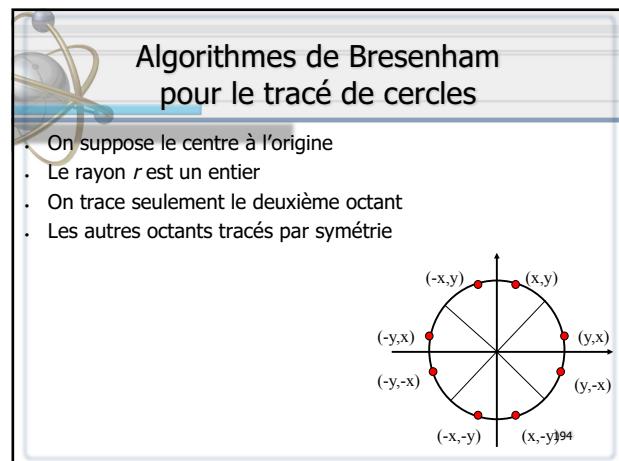
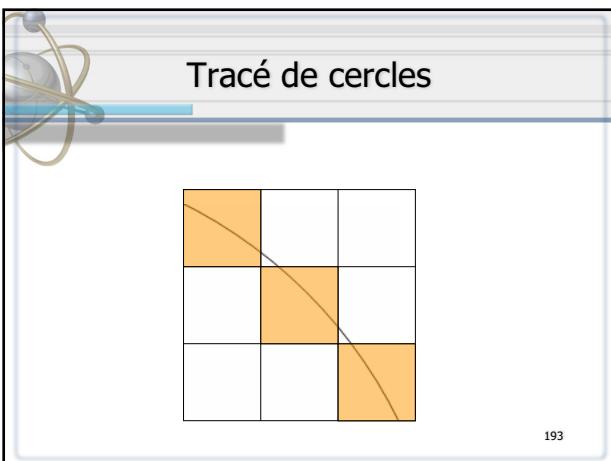
185

Tracé de cercles



186





Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

199

Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

E

M

Q

SE

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

200

Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

201

Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

E

M

Q

SE

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

202

Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

203

Algorithmes de Bresenham pour le tracé de cercles

On calcule la valeur de $f(x,y)$ au point M , noté e_i :

- $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$

(x_i, y_i)

$(x_i + 1, y_i - 1/2)$

E

M

Q

SE

$e_i \geq 0: M$ est au-dessus de la droite
· Q est sous M on va à SE

$e_i < 0: M$ est sous la droite
· Q est au-dessus de M on va à E

204

Algorithmes de Bresenham pour le tracé de cercles

- On calcule la valeur de $f(x,y)$ au point M , noté e_i :
 - $e_i = f(x_i + 1, y_i - 1/2) = (x_i + 1)^2 - (y_i - 1/2)^2 - r^2$
 - $e_i \geq 0 : M$ est au-dessus de la droite
• Q est sous M on va à SE
 - $e_i < 0 : M$ est sous la droite
• Q est au-dessus de M on va à E

Algorithmes de Bresenham pour le tracé de cercles

- Le calcul de l'erreur se fait manière incrémentale
 - Pour chaque pixel activé (en cours de trainement) nous calculons le paramètre de décision du pixel suivant
 - $e_i = f(x_i + 1, y_i - 1/2)$
 - Si $e_i > 0$ (M à l'extérieur) alors le pixel à afficher est SE
 - Le prochain point milieu sera $(x_i + 2, y_i - 3/2)$
 - $e_{i+1} = f(x_i + 2, y_i - 3/2) = e_i + 2 \cdot x_i - 2 \cdot y_i + 5$
 - Si $e_i < 0$ (M à l'intérieur) alors le pixel à afficher est E
 - Le prochain point milieu sera $(x_i + 2, y_i - 1/2)$
 - $e_{i+1} = f(x_i + 2, y_i - 1/2) = e_i + 2 \cdot x_i + 3$

Algorithmes de Bresenham pour le tracé de cercles

```
CerclePointMilieu(int r) // le centre du cercle est à l'origine
{
    int x = 0;
    int y = r;
    double e = 5.0/4.0 - r; // point de départ(0,r) => point intermédiaire (1,r-1/2)
    AfficherPixel(x, y);
    While (y > x) { // octant 2
        if(e < 0) // pixel suivant E
            e = e + 2 . x + 3 ;
        else { // pixel suivant SE
            e = e + 2 . x - 2 . y + 5 ;
            y = y - 1 ;
        }
        AfficherPixel(x, y);
        x = x + 1 ;
    }
}
```

207

Algorithmes de Bresenham pour le tracé de cercles

- La généralisation aux huit octants se fait en remplaçant la procédure `AfficherPixel()` de l'algorithme par la procédure suivante :

```
AfficherPixelCercle(x, y)
{
    AfficherPixel(x, y);
    AfficherPixel(x, -y);
    AfficherPixel(-x, y);
    AfficherPixel(-x, -y);
    AfficherPixel(y, x);
    AfficherPixel(y, -x);
    AfficherPixel(-y, x);
    AfficherPixel(-y, -x);
}
```

208

Optimisation

- Au départ
 - $e_0 = 5.0/4.0 - r = 1.25 - r$
- Si $e_i < 0$ alors on va à « E »
 - $e_{i+1} = e_i + 2 \cdot x_i + 3$;
- Sinon $e_i > 0$ alors on va à « SE »
 - $e_{i+1} = e_i + 2 \cdot x_i - 2 \cdot y_i + 5$;

209

Algorithmes de Bresenham pour le tracé de cercles

- Si on pose $h_i = e_i - 0.25$, on peut remplacer linitialisation par :

$$h_0 = 1 - r$$
- On devrait donc aussi modifier la comparaison « `if(e < 0)` » par :

$$\text{if } (h_i < -0.25) \{$$
- Cependant, on remarque que h_i sera toujours un entier donc :

$$h_i < -0.25 \Rightarrow h_i < 0$$
- On peut donc seulement changer linitialisation et remplacer e_i par h_i ailleurs

210

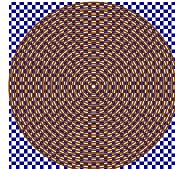
Algorithmes de Bresenham pour le tracé de cercles

```
CerclePointMilieu(int r)           // le centre du cercle est à l'origine
{
    int x = 0;
    int y = r;
    double h = 1 - r;              // point de départ(0,r) => point intermédiaire (1,r-1/2)
    AfficherPixelCercle(x, y);
    While (y > x) {               // octant 2
        if(h < 0)                 // pixel suivant E
            h = h + 2 . x + 3 ;
        else{                      // pixel suivant SE
            h = h + 2 . x - 2 . y + 5 ;
            y = y - 1 ;
        }
        AfficherPixelCercle(x, y);
        x = x + 1 ;
    }
}
```

211

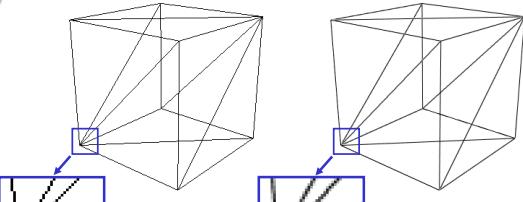
Anticrenelage (antialiasing)

Les algorithmes de tracé présentés précédemment ont un problème commun de précision.
La discréétisation implique une perte d'informations qui se traduit par des contours en forme de marches d'escaliers.
Cette perte d'informations est bien connu en traitement du signal (analyse de Fourier) et le terme aliasing qui en est issu est utilisé en graphisme pour caractériser ces artefacts dans les images produites : Moirées
L'antialiasing consiste alors à corriger (diminuer) ces effets.



212

Anticrenelage (antialiasing)



Sans correction Avec correction

213

Anticrenelage (antialiasing)

- Les deux principales solutions :
 - Sur-échantillonnage de l'image
 - Algorithmes de tracé de segments corrigés

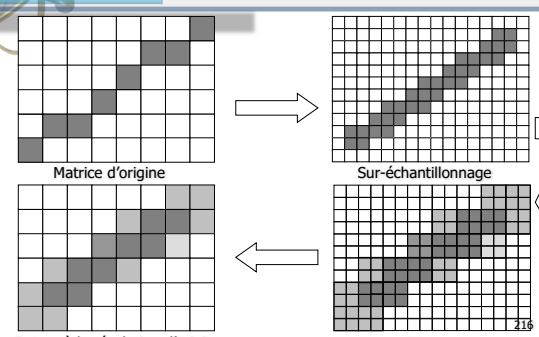
214

Sur-échantillonnage de l'image

- Cette méthode consiste à calculer, dans un premier temps, une image virtuelle à une résolution plus importante que celle de l'image désirée (en mémoire mais non à l'écran), en pratique X3, X5 ou X7, puis de réduire cette image par filtrage :
- L'intensité de chaque pixel de l'image finale est déterminée par moyennage pondéré ou par filtrage (filtre passe-bas par convolution) de plusieurs pixels de l'image sur-échantillonnée.

215

Sur-échantillonnage de l'image



Matrice d'origine Sur-échantillonnage Filtrage Retour à la résolution d'origine

216

Sur-échantillonnage de l'image

Paramètres

- Echantillonage
 - Type
 - Déterministe
 - Régulier
 - Adaptatif
 - Aléatoire
 - Uniforme
 - Disque de Poisson
 - Jittering
 - Résolution
 - Filtre
 - Forme du filtre
 - Taille du filtre

217

Sur-échantillonnage de l'image

Echantillonage Déterministe

Suréchantillonnage

- Régulier
 - images avec des résolutions beaucoup plus grandes celles nécessaire,
 - le temps de calcul augmente (au moins d'un facteur n^2 si n est le facteur du suréchantillonnage).
 - le suréchantillonnage est loin d'être nécessaire partout

Adaptatif : raffinement progressif de certaines zones particulières du pixel

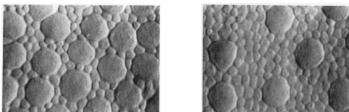
- Critères
 - on se trouve sur un contour.
 - passage d'un objet à un autre.
 - auto-occlusion d'un objet complexe.
 - phénomène d'ombrage et de lumière
 - un des phénomènes précédents derrière un objet transparent.
 - etc.

218

Sur-échantillonnage de l'image

Echantillonage aléatoire

- l'incohérence d'un bruit est beaucoup moins choquante que la cohérence de l'aliasing
 - l'attention est d'abord attirée par les parties cohérentes d'une scène
 - les photo-récepteurs de l'œil humain ne sont pas distribués de manière uniforme : c'est l'une des raisons pour lesquelles l'œil humain ne fait pas d'aliasing.



• l'échantillonage aléatoire

- permet de transformer l'aliasing en bruit tout en perturbant relativement peu les parties basse fréquence de l'image.

219

Sur-échantillonnage de l'image

Uniforme

- Le tirage de chaque point est complètement indépendant des autres
 - Possibilité de tirage proche

Disque de poisson : inspiré de la répartition des photorécepteurs

- Chaque point est placé de façon à ce qu'il soit suffisamment loin de tous les points déjà acceptés.

Jittering

- consiste à partir d'une grille régulière et à la perturber avec un bruit aléatoire

○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

Déterministe : Régulier

○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

Uniforme

○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

Disque de poisson

○	○	○	○
○	○	○	○
○	○	○	○
○	○	○	○

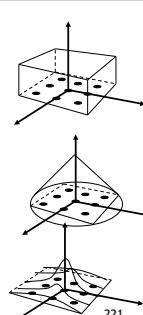
Jittering

220

Sur-échantillonnage de l'image

Forme du Filtres

- Boîte
 - Moyenne des pixels recouverts
- Cône
 - Moyenne pondérées en fonction de la distance du centre
- Fonction Gaussienne
 - Moyenne pondérées suivant la fonction gaussienne



221

Filtre Gaussien

Fonction Gaussienne avec différentes tailles

1	2	1
2	4	2
1	2	1

3x3

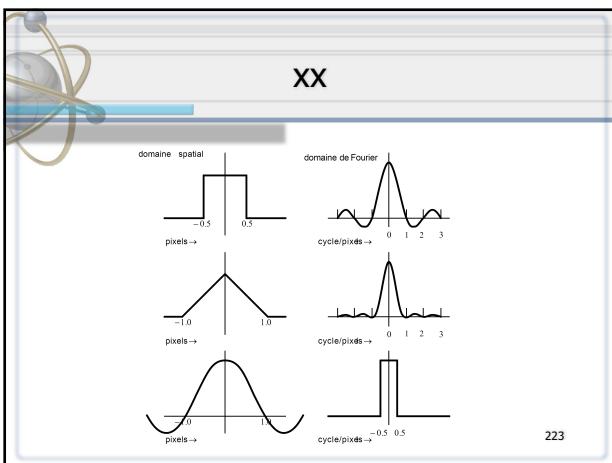
1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

5x5

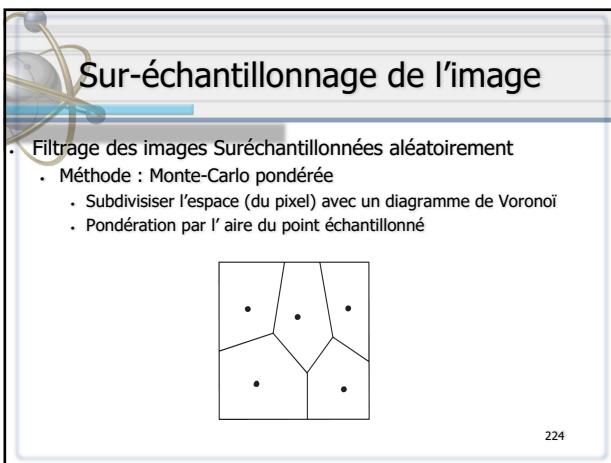
1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

7x7

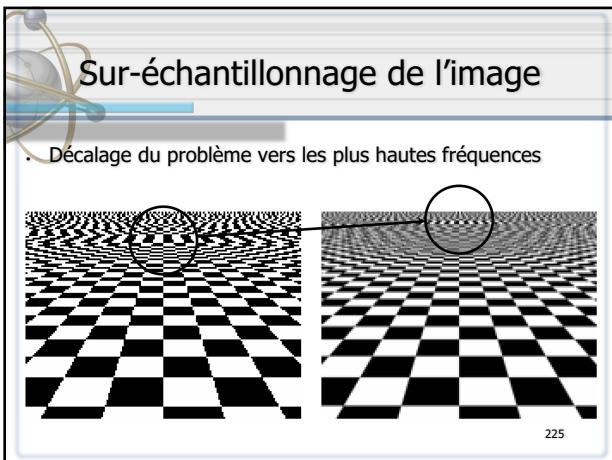
222



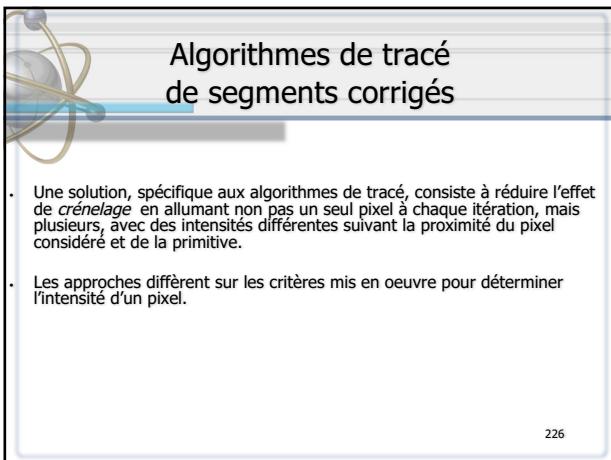
223



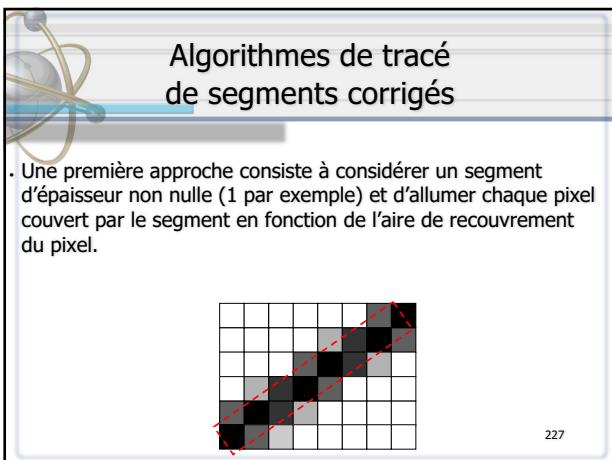
224



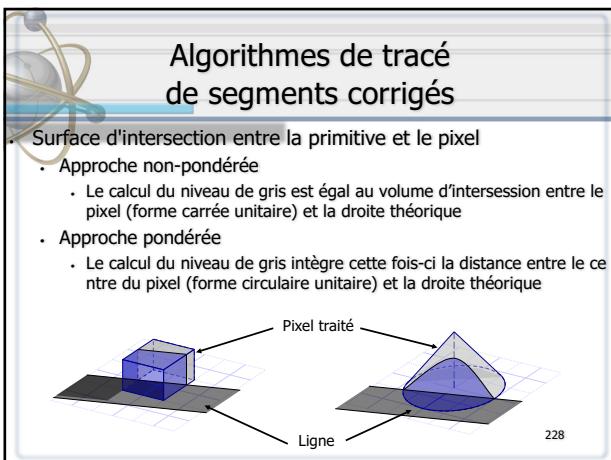
225



226



227

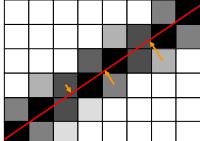


228

Algorithmes de tracé de segments corrigés

Algorithmie Gupta-Sprout

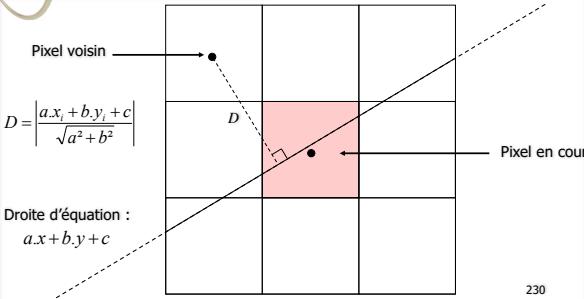
- Cette approche considère uniquement la distance entre les pixels et le segment de droite
 - À chaque itération de l'algorithme de Bresenham on active le pixel en cours ainsi que le voisinage : généralement 3x3
 - Le niveau de gris est proportionnel à la distance entre le pixel activé et la droite (y compris le pixel calculé par l'algorithme de Bresenham)



229

Algorithmes de tracé de segments corrigés

Voisinage 3x3



Droite d'équation :

$$a.x + b.y + c$$

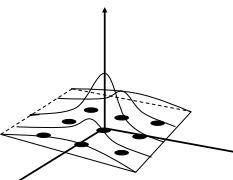
Pixel voisin

Pixel en cours

230

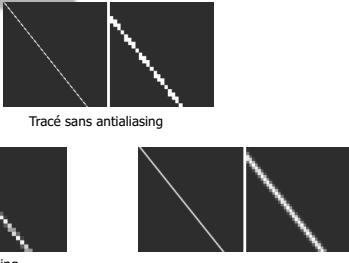
Algorithmes de tracé de segments corrigés

- Il est facile de modifier l'algorithme de trace de segments vu précédemment de manière à prendre en compte cet aspect.
- La question qui se pose ensuite est comment passer d'une distance à une intensité.
 - Classiquement, on applique une fonction de filtrage Gaussienne



231

Anticrénelage (résultats)



Tracé sans antialiasing

Tracé avec antialiasing
Méthode basée sur le moyenage (Gaussian) d'une image de plus grande résolution

Tracé avec antialiasing
Méthode basée sur les distances des pixels au segment

232

Le découpage (clipping)

- Clipping écran
 - Traitement permettant de réduire le dessin d'un objet graphique à une région de l'écran
- Cette région est classiquement un rectangle mais peut être de toute autre forme
 - Carré : Écran
 - Trapèze : Pare-brise/Rétroviseur
 - Circulaire : Lunette/Jumelle
 - ...
- Traitement de base de l'Informatique Graphique :
 - Économiser des opérations inutiles
- Extension
 - en 3D : éliminer les calculs en dehors de l'espace visible

233

Le découpage (clipping)

Trois approches :

- AVANT LA CONVERSION**
 - calcul analytique des intersections avec la frontière de la région de découpage
 - abordable pour des formes simples (p.e. algorithme Cohen-Sutherland pour segment découpé par région rectangulaire)
- PENDANT LA CONVERSION**
 - on construit un masque de la région puis on vérifie que chaque pixel devant être « allumé » y figure
 - permet de représenter des régions de découpage arbitrairement complètes
- APRÈS LA CONVERSION**
 - on convertit tout l'objet virtuel dans un espace mémoire temporaire puis on copie le sous-ensemble d'intérêt (clôture)
 - efficace si on fait face à un objet complexe coûteux à convertir et qu'on déplace la région de découpage de façon interactive

234

Le découpage (clipping)

Familles d'algorithmes

- Clipping Point / Fenêtre
- Clipping Segment / Fenêtre
 - Cohen-Sutherland
 - Liang-Barsky
- Clipping Polygon / Fenêtre
 - Sutherland-Hodgeman

235

Le découpage (Clipping)

Clipping de points : Point / Fenêtre

- Test :

$$x_{\min} < x < x_{\max}$$

$$\&$$

$$y_{\min} < y < y_{\max}$$

236

Le découpage (Clipping)

Clipping de segments : Segment / Fenêtre

- Déterminer les portions de segments à l'intérieur de la fenêtre

Le découpage (Clipping)

Algorithme de Cohen et Sutherland

- Traiter les paires de points relativement aux codes des zones

Ivan Sutherland

Le découpage (Clipping)

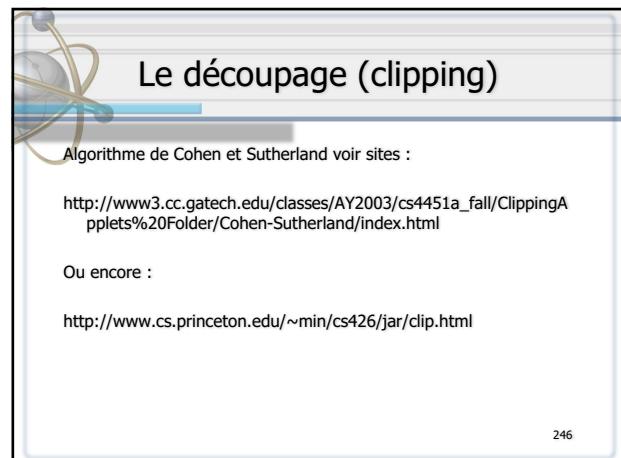
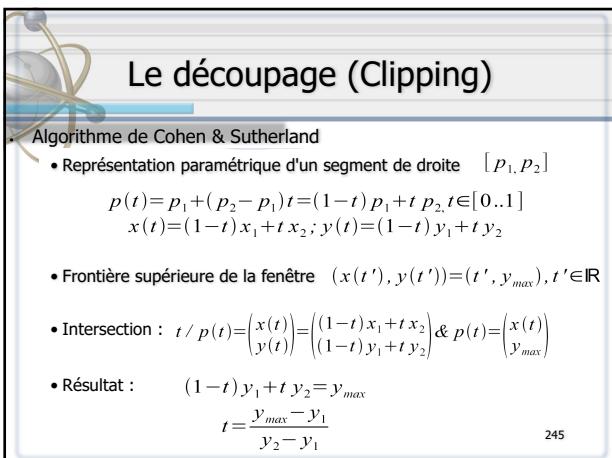
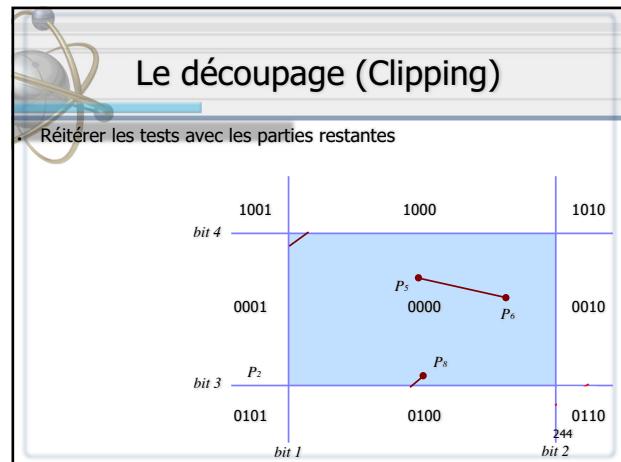
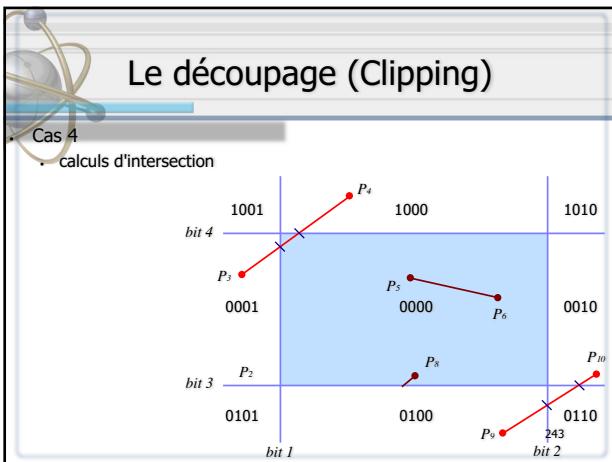
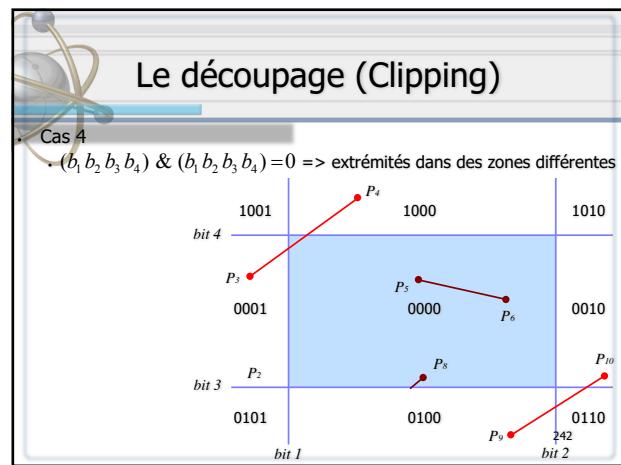
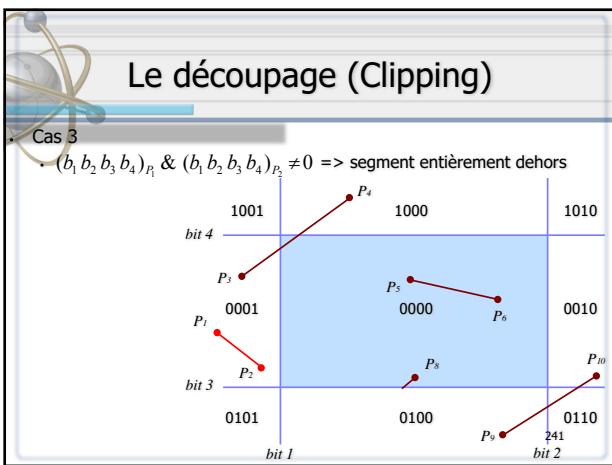
Cas 1

- $(b_1 b_2 b_3 b_4)_{P_3} = (b_1 b_2 b_3 b_4)_{P_6} = 0 \Rightarrow$ segment dans la fenêtre

Le découpage (Clipping)

Cas 2

- $(b_1 b_2 b_3 b_4)_{P_3} \neq 0 \& (b_1 b_2 b_3 b_4)_{P_6} = 0 \Rightarrow$ segment partiellement dedans



Le découpage (clipping)

Clipping 3D

- Extension de l'algorithme de Cohen & Sutherland
- 6 Bits pour le outcode
 - b4 : $z > z_{\max}$
 - b5 : $z < z_{\min}$
- Nécessite d'autres précalculs

247

Le découpage (clipping)

Clipping de segments : Algorithme de Liang et Barsky

- Intersection de la droite ($p_1 p_2$) avec les frontières dans l'ordre :
 - Bas puis Gauche puis Haut puis Droite
- Tri des paramètres et comparaison avec l'ordre attendu
 - si ($t_1 < t_2 < t_3 < t_4$) alors segment à conserver
 - sinon segment rejeté

248

Le découpage (Clipping)

Clipping de polygones

- Déterminer les parties de polygones à l'intérieur de la fenêtre

249

Le découpage (Clipping)

Algorithme de Sutherland et Hodgeman

- Tester relativement à chaque frontière de la fenêtre

250

Le découpage (Clipping)

Algorithme de Sutherland et Hodgeman

- Tester relativement à chaque frontière de la fenêtre

251

Le découpage (Clipping)

Algorithme de Sutherland et Hodgeman

- Tester relativement à chaque frontière de la fenêtre

252

Le découpage (Clipping)

- Algorithme de Sutherland et Hodgeman
 - Tester relativement à chaque frontière de la fenêtre

253

Le découpage (Clipping)

Algorithme de Sutherland et Hodgeman

- Pour chaque frontière et chaque polygone :
 - tester l'appartenance sur les deux extrémités des segments
 - si cas 2 ou cas 4 calculer l'intersection avec la frontière
 - insérer les nouveaux points calculés
 - supprimer les points en dehors de la fenêtre

254

Le découpage (clipping)

- Algorithme de Sutherland et Hodgeman voir site

http://www3.cc.gatech.edu/classes/AY2003/cs4451a_fall/Clipping_Applets%20Folder/Cohen-Sutherland/index.html

255

Élimination des parties cachées

- Déterminer les lignes, arêtes, surfaces et volumes visibles par un observateur
 - Cohérence visuelle de la scène
 - Réduire le nombre de primitives traitées par le pipeline graphique
- Algorithme de base
 - Équivalent au tri => $O(n \log n)$
- Deux familles d'algorithme :
 - Algorithmes objet : dans l'espace scène
 - Algorithmes image : dans l'espace image

256

Élimination des parties cachées

- Dans l'espace scène : le masquage se fait sur le modèle de données physiques => Calcul en coordonnées du monde. le masquage des parties cachées est valable quelle que soit l'échelle du dessin.
- Caractéristiques :
 - Test objet par objet ($O(n^2)$)
 - Précision dépend de la résolution des objets
 - Techniques relativement complexes à mettre en œuvre
- Méthodes :
 - Élimination de faces arrières (Backface Culling)
 - Tri des primitives selon leur éloignement
 - Algorithme du peintre
 - Algorithme de tri par arbre binaire (BSP Trees)

257

Élimination des parties cachées

- Dans l'espace image : le masquage se fait sur les pixels écran => Calcul en coordonnées fenêtre
- Caractéristiques :
 - Test de visibilité en chaque pixel ($O(n \times p)$)
 - Précision dépend de la résolution de l'espace image
 - Recalculer à chaque changement caméra/scène
 - Techniques relativement simples
- Méthodes :
 - Subdivision récursive de l'image (Warnock)
 - Test de profondeur
 - Ligne de balayage :
 - Scan-line-Watkins
 - Scan-line-Zbuffer
 - Tampon de profondeur - Zbuffer
 - Lancer de rayons (Whitted)

258

Algorithmes objet

259

Élimination de faces arrières (Backface Culling)

- Backface Culling : garder les parties de la surface pour lesquelles la normale pointe dans la direction opposée au point d'observation.
- Ces parties, vues du point d'observation, sont occultées par l'objet lui-même.

- Conditions : La suppression des faces arrières suffit à éliminer les parties cachées :
 - si l'objet est seul dans la scène
 - il ne faut pas qu'un objet puisse en masquer un autre
 - si l'objet est convexe
 - dans un objet concave, des faces avant peuvent être masquées par d'autres

260

Élimination de faces arrières (Backface Culling)

- Calcul : produit scalaire
 - (Eil-Face) . Normale
 - < 0 : on garde le polygone
 - > 0 : on l'élimine
- Économise 50 % du temps de calcul
 - En moyenne
- Faible coût par polygone
- Étape préliminaire pour les autres algorithmes
- Suffisant pour un seul objet convexe

261

Algorithme du Peintre

- Dit aussi « Algorithme de Tri par la Profondeur » (Newell, Newell & Sancha 1972)
- Peindre les facettes polygonales dans la mémoire vidéo suivant un ordre de distance décroissante au point d'observation.

262

Algorithme du Peintre

- Principe :
 - Tracé de polygones bien orientés, du plus éloigné au plus proche
 - Tri des polygones
- Cas trivial (non chevauchement)

263

Algorithme du Peintre

- Principe :
 - Tracé des polygones « front » du plus éloigné au plus proche
 - Cas ambigu (chevauchement) : Découpage de polygones

264

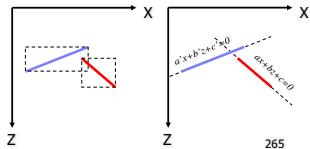
Algorithme du Peintre

- Algorithme complet

- Trier les polygones en fonction de leur plus grande coordonnée en z
 - z distance à la caméra suivant la direction de visée
- Si deux polygones ont des étendues en z qui se recouvrent :
 - On test :
 - Si les boîtes englobantes de leurs projections sont séparées
 - Si leurs projections ne sont pas séparées : calcul d'intersection de segment de droite
 - Si rien ne marche, on coupe l'un des polygones par le plan de l'autre
 - Calcul d'intersection de plans

- Caractéristiques

- Le plus intuitif des algorithmes
- Coût en mémoire :
 - Affichage direct à l'écran : $O(p)$
 - Il faut trier les polygones : $O(n \log n)$
- Temps de calcul :
 - On affiche toute la scène
 - Efficace surtout sur des petites scènes



265

Partition binaire de l'espace par BSP Trees

- BSP-tree (Binary Space Partition) : arbre binaire utilisé pour trier des primitives dans l'espace

- Principe (Schumaker 1969, Fluch 1980) :

- Choix d'une arête de référence
 - Partage de l'espace en 2 demi-espaces (in et out)
 - Répartition des objets (& primitives) selon le demi-espace occupé
 - Les objets (& primitives) à cheval sont découpés selon le plan de référence
- Réitération dans les deux demi-espaces résultants

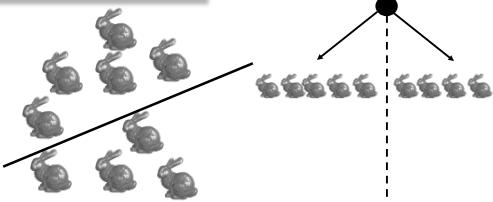
266

Partition binaire de l'espace par BSP Trees



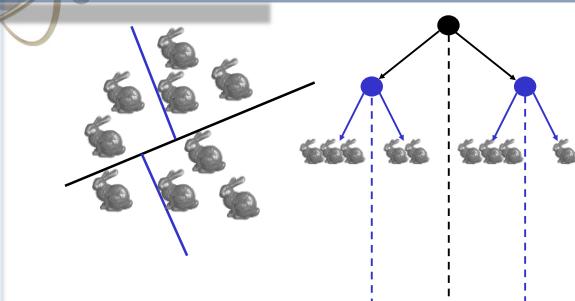
267

Partition binaire de l'espace par BSP Trees



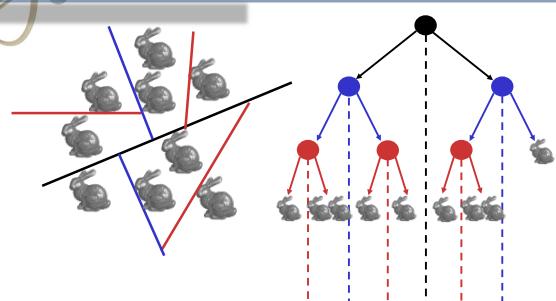
268

Partition binaire de l'espace par BSP Trees

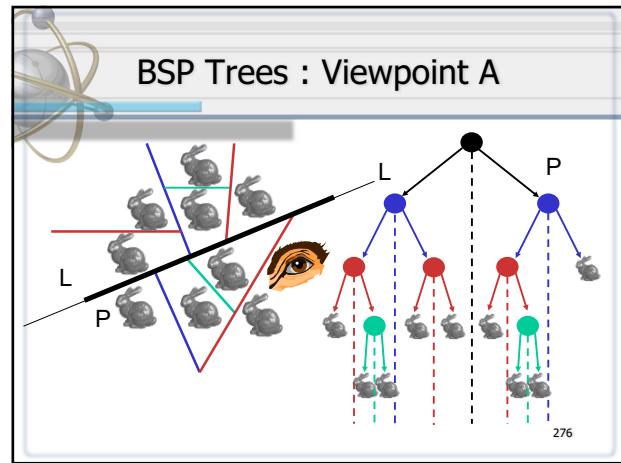
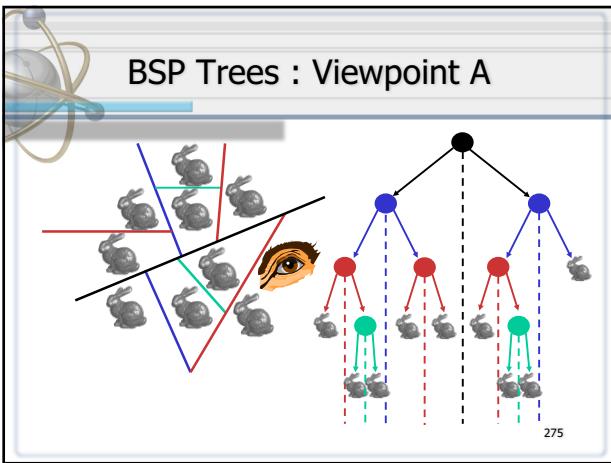
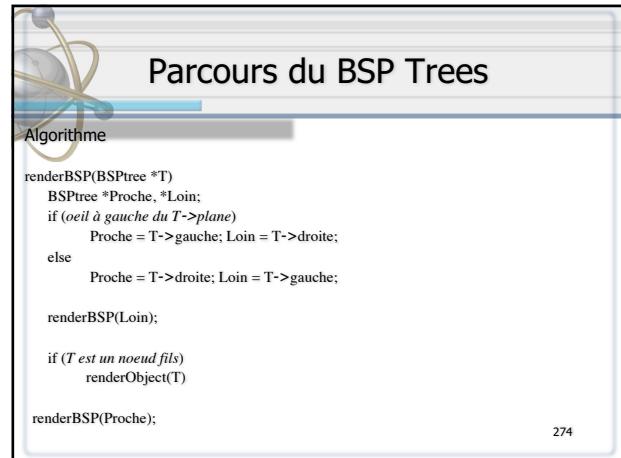
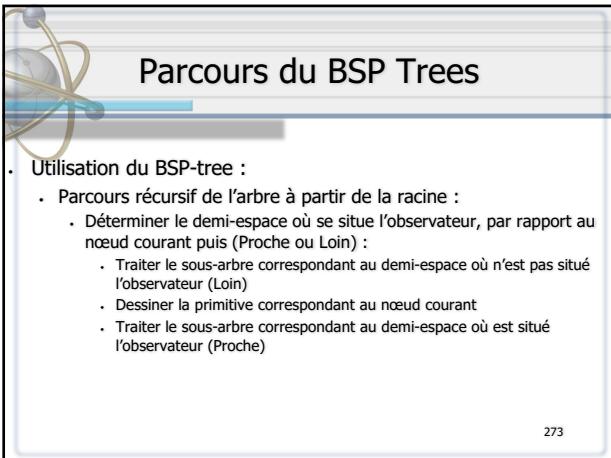
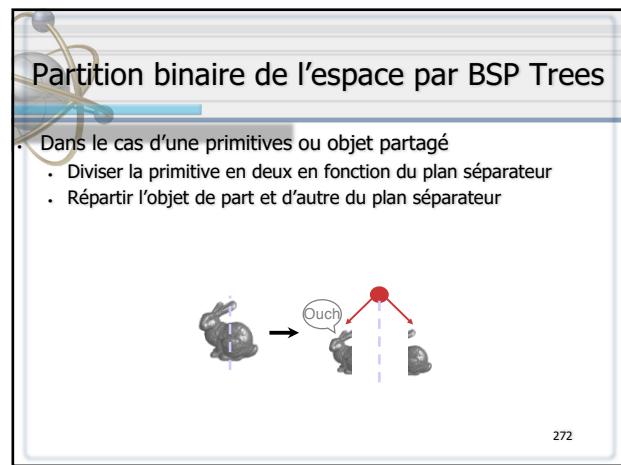
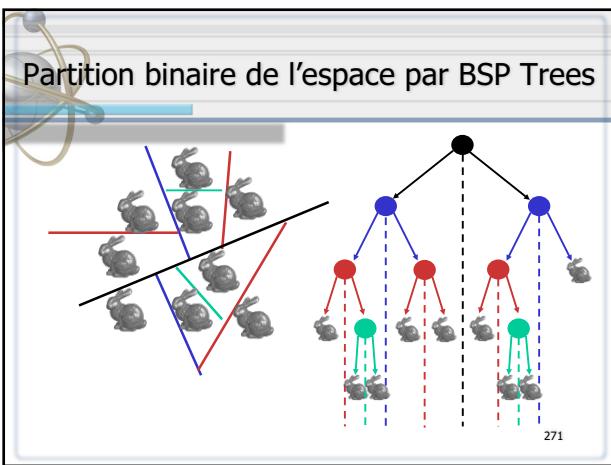


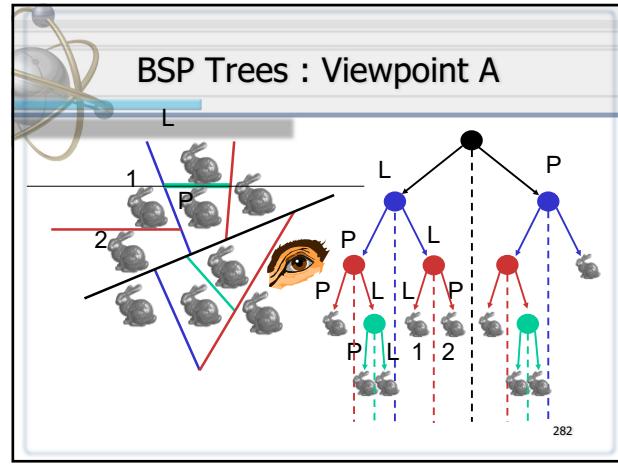
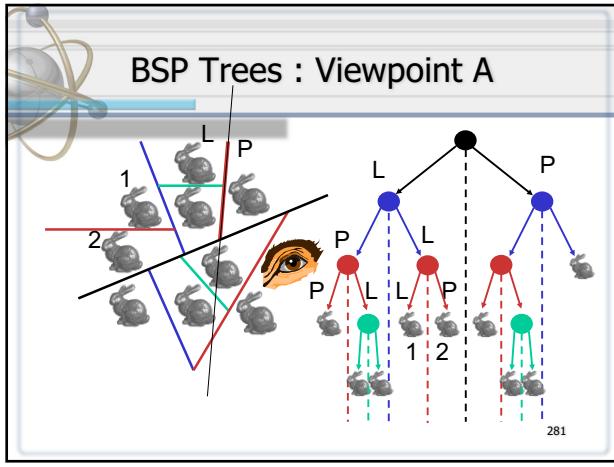
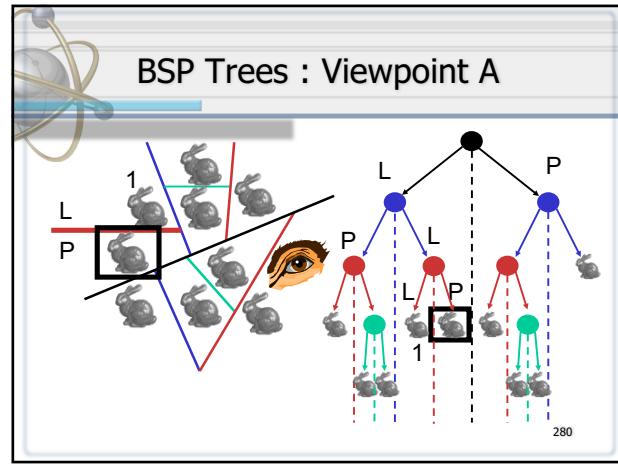
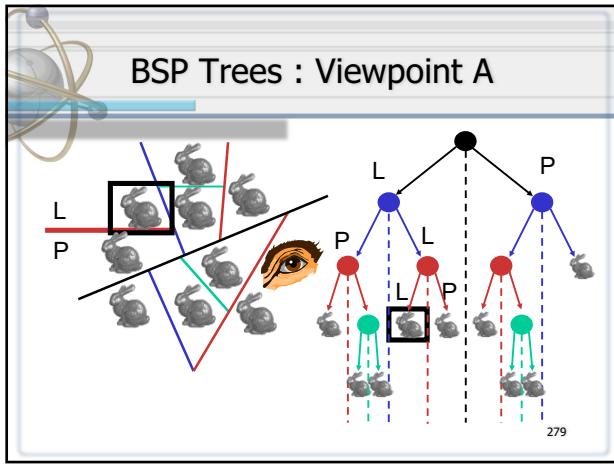
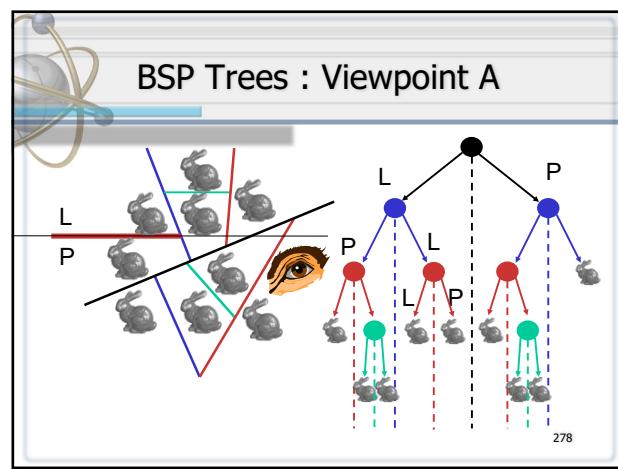
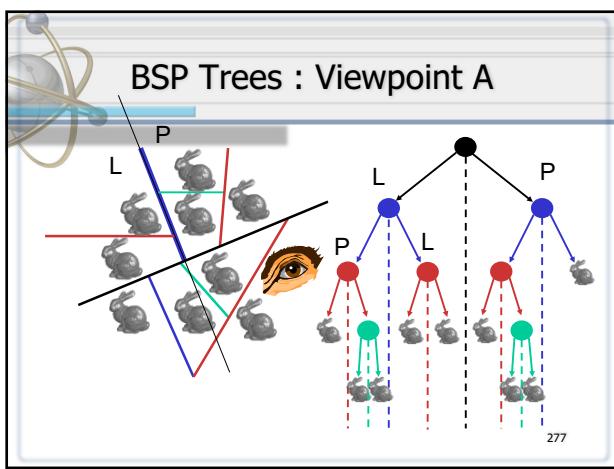
269

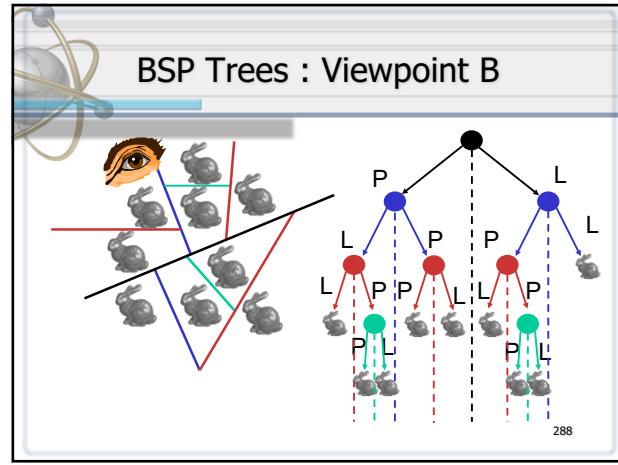
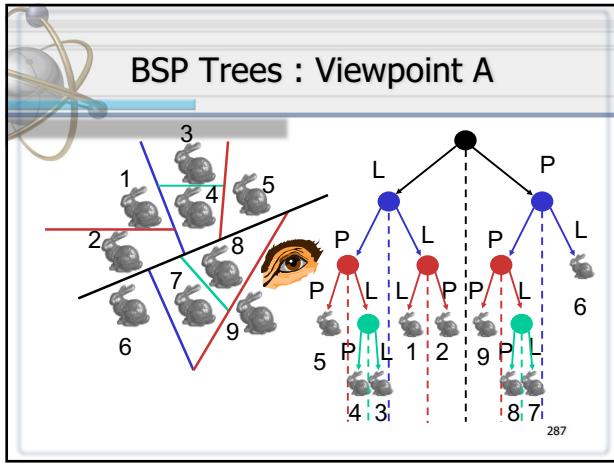
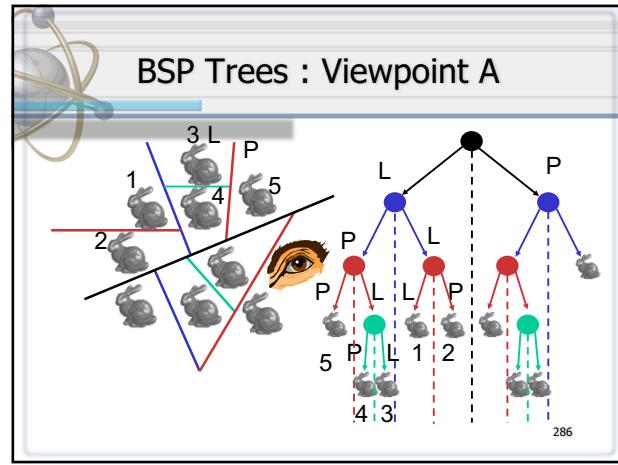
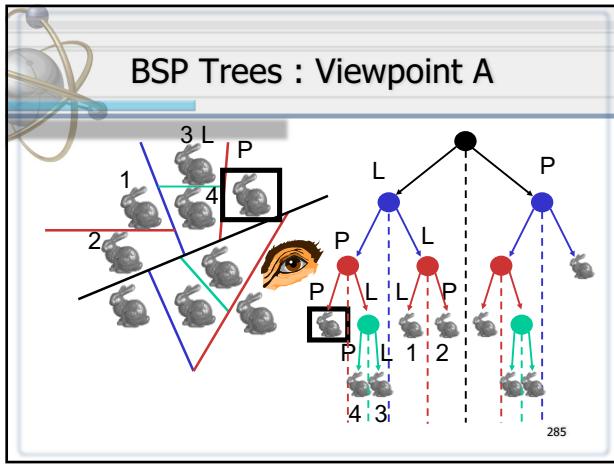
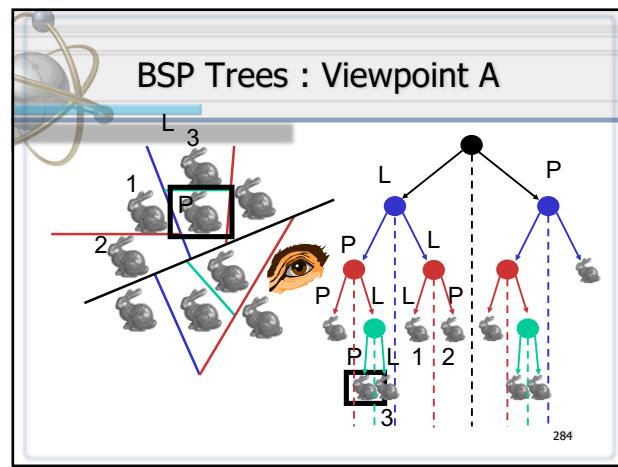
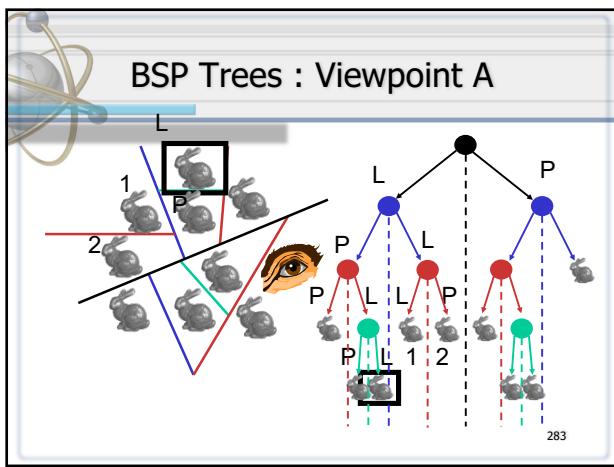
Partition binaire de l'espace par BSP Trees

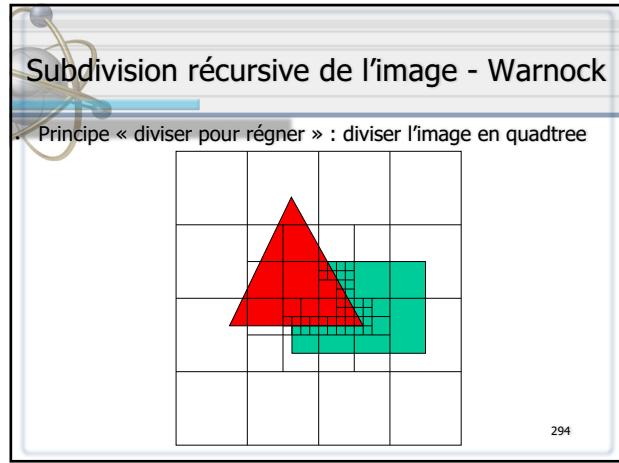
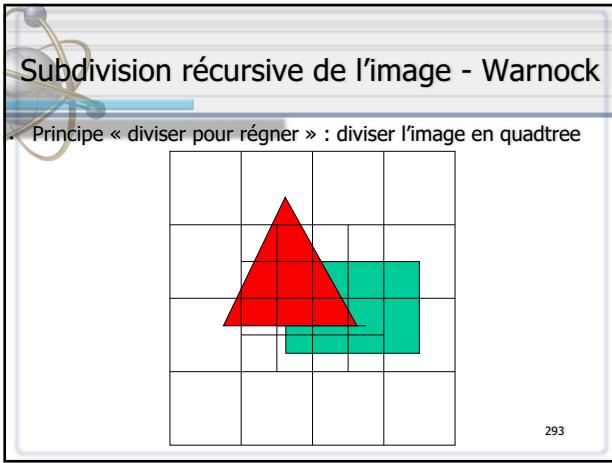
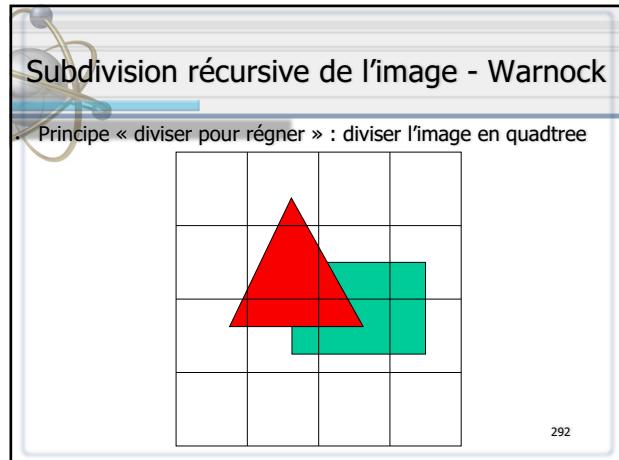
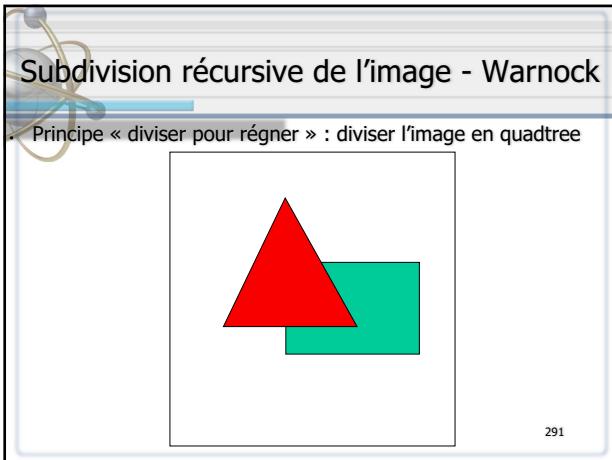
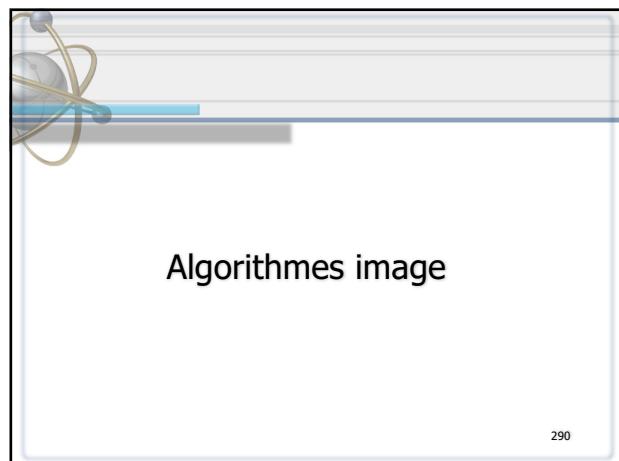
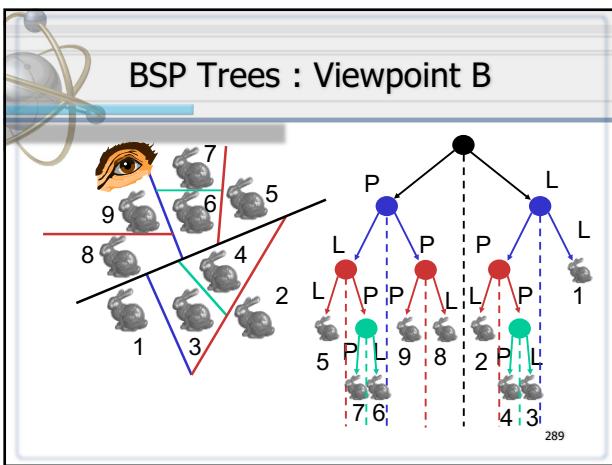


270



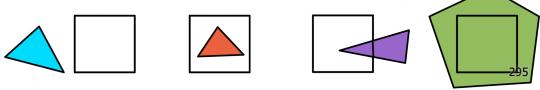






Subdivision récursive de l'image - Warnock

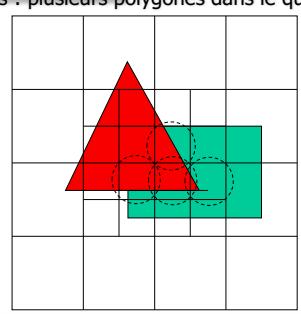
- Pour chaque quadrant :
 - Déterminer la liste des polygones potentiellement visibles
 - Tracer le quadrant de l'image directement dans les cas simples :
 - Rien dans le quadrant
 - Couleur du fond
 - Un seul polygone : contenu facile à dessiner avec la couleur du polygone
 - Le polygone couvrant le quadrant
 - Partiellement
 - Totalement



295

Subdivision récursive de l'image - Warnock

- Cas complexes : plusieurs polygones dans le quadrant



296

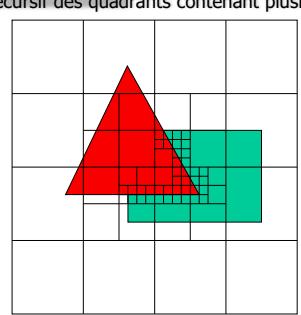
Subdivision récursive de l'image - Warnock

- Principe « diviser pour régner » : diviser l'image en quadtree
 - Cas complexes : plusieurs polygones dans le quadrant
 - Trier les polygones selon la profondeur
 - Si un polygone masque tous les autres :
 - Couvre tout le quadrant et est avant les autres (sans test complexe)
 - Tracer avec la couleur du polygone
 - Sinon subdiviser et itérer le processus
 - Arrêt quand le quadrant est plus petit qu'un pixel
 - Tracer avec la couleur du polygone en avant des autres

297

Subdivision récursive de l'image - Warnock

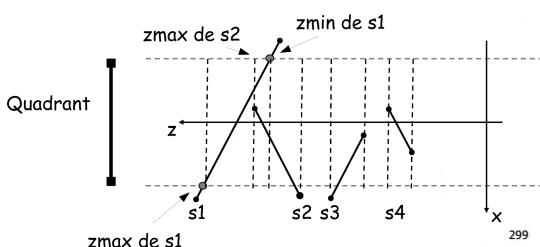
- Découpage récursif des quadrants contenant plusieurs polygones



298

Subdivision récursive de l'image - Warnock

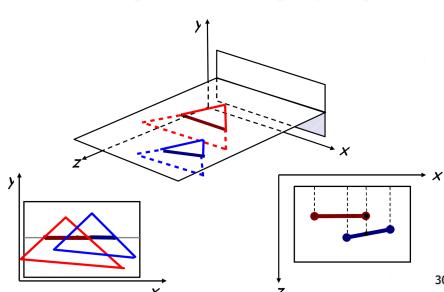
- Problème : plusieurs polygones dans le quadrant
 - s_1 masque tous les autres polygones, mais...
 - Subdivision (masquage non détecté par les tests simples de profondeur)



299

Algorithme de balayage (scan-line)-Watkins

- Principe : éliminer les lignes cachées, ligne par ligne

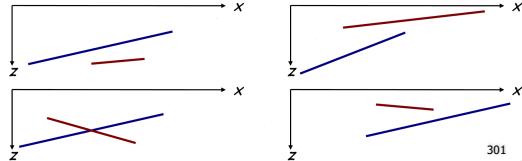


300

Algorithme de balayage (scan-line)-Watkins

- Pour une ligne de balayage donnée :
 - Obtention d'un segment par polygone
 - Détermination de la visibilité des segments
 - Affichage des segments visibles

- Differentes cas rencontrés



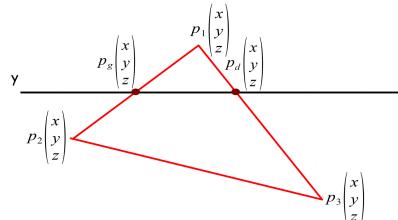
301

Algorithme de balayage (scan-line)-Watkins

- Obtention d'un segment par polygone

- Intersection plan-segment

- Pour un y fixé : on calcule les coordonnées x & z à partir de l'équation du segment traité de la primitive

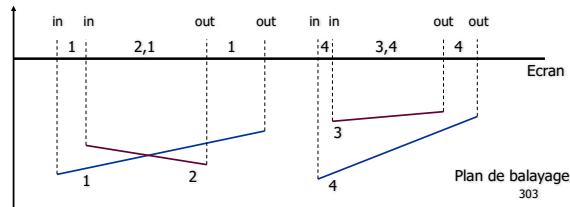


302

Algorithme de balayage (scan-line)-Watkins

- Détermination de la visibilité

- Projection sur la ligne de balayage de l'écran
 - Déterminations des intervalles associés aux segments projetés [in, out]

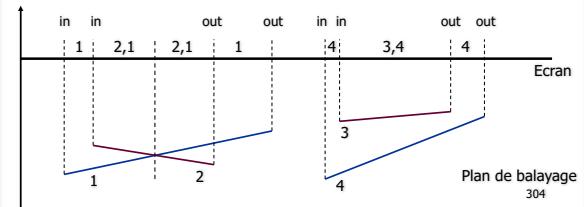


303

Algorithme de balayage (scan-line)-Watkins

- Détermination de la visibilité

- Superposition d'intervalles -> plusieurs segments
 - Calcul d'intersection : intersection entre segments de droite
 - Si intersection -> ajout d'un point sur la ligne de balayage

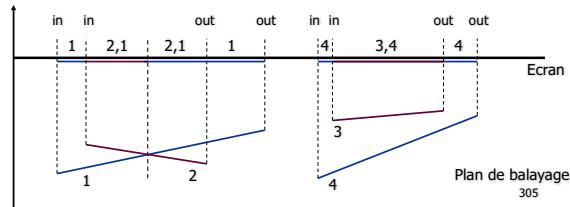


304

Algorithme de balayage (scan-line)-Watkins

- Affichage des segments visibles

- Test de profondeur
- Affichage de la couleur du segment traité

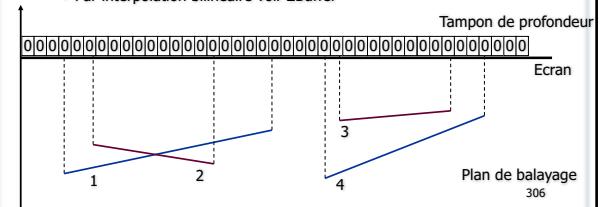


305

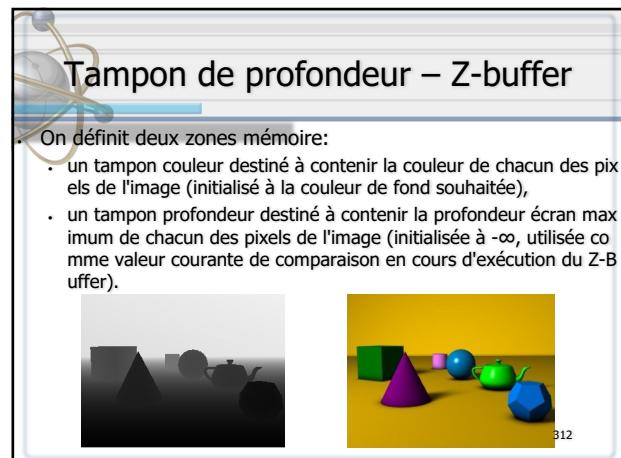
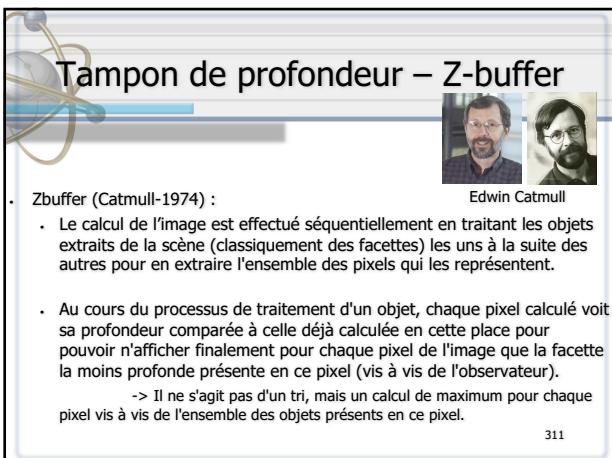
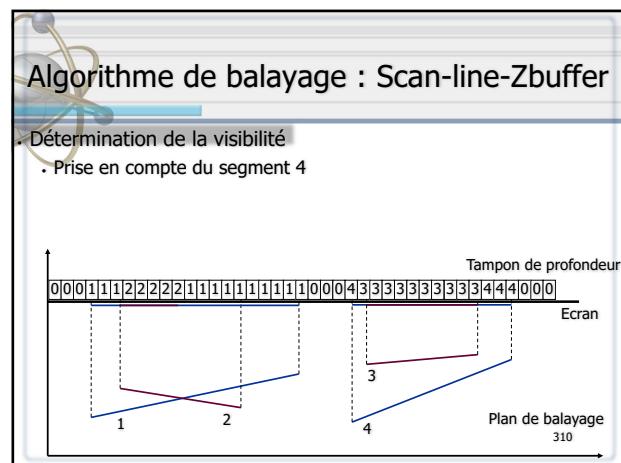
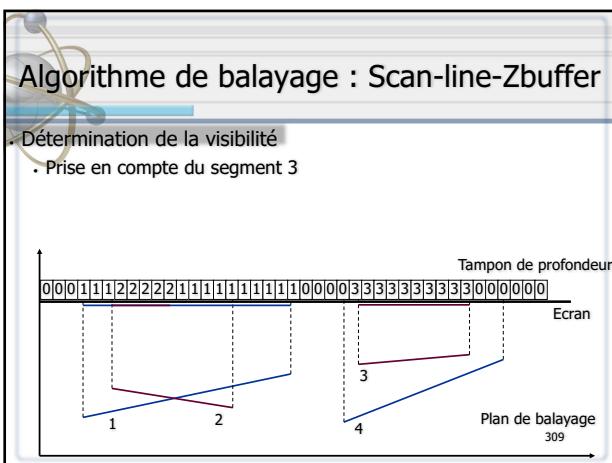
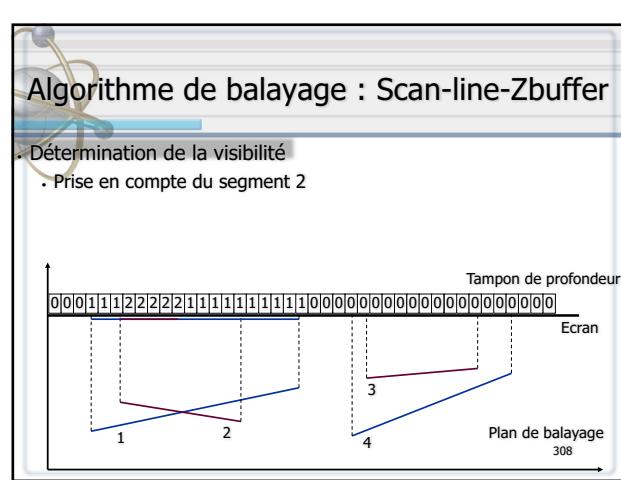
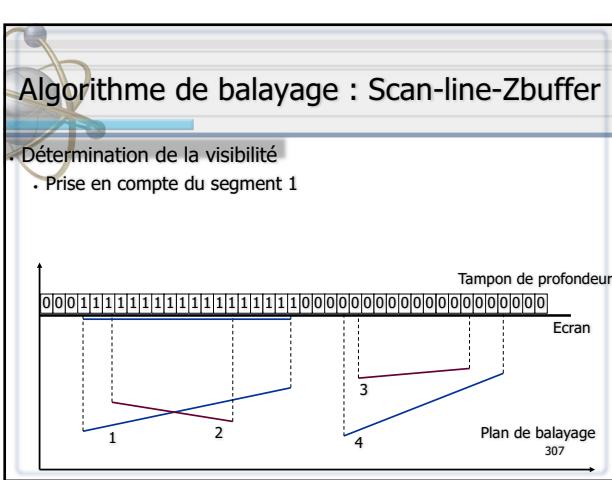
Algorithme de balayage : Scan-line-Zbuffer

- Détermination de la visibilité

- Tampon de profondeur + mémoire d'image pour la ligne de balayage
 - Pour chaque élément traité on calcule sa profondeur
 - Par interpolation bilinéaire voir ZBuffer



306



Tampon de profondeur – Z-buffer

- Initialiser le Zbuffer à $-\infty$
- Initialiser le buffer de couleur avec couleur du fond

313

Tampon de profondeur – Z-buffer

Z-buffer	Buffer couleur
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	
$\infty \infty \infty \infty \infty \infty \infty$	

314

Tampon de profondeur – Z-buffer

- Pour chaque polygone
 - Pour chaque sommet
 - P_1 = projection (sommet 1) sur l'écran
 - P_2 = projection (sommet 2) sur l'écran
 - P_3 = projection (sommet 3) sur l'écran
 - Pour chaque pixel (x,y) du polygone projeté
 - Déterminer la profondeur z du point correspondant du polygone par interpolation bilinéaire
 - Si $z < \text{Zbuffer}(x,y)$ alors
 - $\text{Zbuffer}(x,y) = z$
 - $\text{FrameBuffer}(x,y) = \text{couleur du point}$
 - Application du modèle d'ombrage

315

Tampon de profondeur – Z-buffer

Z-buffer	Color Buffer
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	

316

Tampon de profondeur – Z-buffer

Z-buffer	Color Buffer
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	
5 5 5 5 5 5 5	

317

Tampon de profondeur – Z-buffer

Calcul de la profondeur par interpolation

$$z_a = z_1 + (z_3 - z_1) \frac{(y_1 - y_s)}{(y_1 - y_3)}$$

$$z_b = z_1 + (z_2 - z_1) \frac{(y_1 - y_s)}{(y_1 - y_2)}$$

$$z_p = z_a + (z_b - z_a) \frac{(x_a - x_p)}{(x_a - x_b)}$$

318

Tampon de profondeur – Z-Buffer

Algorithme

```

for all i,j {
    Depth[i,j] = MAX_DEPTH
    Image[i,j] = BACKGROUND_COLOUR
}
for all polygons P {
    for all pixels in P {
        if (Z_pixel < Depth[i,j]) {
            Image[i,j] = C_pixel
            Depth[i,j] = Z_pixel
        }
    }
}

```

319

Tampon de profondeur - Zbuffer

- Avantages**
 - Simplicité de mise en œuvre
 - Pas de tri préalable
 - Rapidité (algorithme câblé, parallélisable)
- Inconvénients**
 - Traitement de tous les polygones, même cachés
 - Taille mémoire nécessaire (de moins en moins)
 - Ne traite pas les effets particuliers
 - Inter-reflets, transparence, réfraction

320

Coûts comparés

Catégorie	Peintre	Warnock	Scan-Line	Z-Buffer
100 pixels	~10	~10	~10	~10
2500 pixels	~10	~10	~10	~100
60000 pixels	~550	~350	~150	~100

321

Choix de l'algorithme

- L'algorithme idéal dépend :
 - De la complexité de la scène
 - Du matériel disponible
 - De ce qu'on veut faire en plus de l'affichage
- Z-Buffer en général :
 - Fourni gratuitement dans la librairie graphique
 - Pas de pré-traitement
 - Quelques effets de bord si on le connaît mal
- Scan-line pour les hackers :
 - Quand on ne peut pas utiliser la carte graphique
 - Faible coût mémoire
 - Lié à la fonction d'affichage, peut être très efficace³²²

Remplissage

- Remplissage : Appelé également Rastérisation, cette étape consiste à remplir les polygones projetés (2D) pour donner un aspect plus réaliste aux objets.
- Le remplissage se fait suivant le modèle d'Ombrage utilisé par le pipeline graphique (Gouraud, Phong, Lambert, etc.)
- Méthodes :
 - Test de l'appartenance d'un point à polygone
 - Test d'intersection
 - Algorithme de balayage de lignes
 - Remplissage de régions

323

Remplissage

Test d'appartenance d'un point à polygone

- Polygone convexe
 - Détermination des normales « extérieures »
 $\vec{N} = \overrightarrow{P_1 P_2} \times \overrightarrow{P_1 P_5}$
 $\vec{n}_{i,i+1} = \overrightarrow{P_i P_{i+1}} \times \vec{N}$
- Positionnement par rapport aux demi-plans
 - p dans le polygone si
 $\forall i \in [1, 4], \vec{n}_{i,i+1} \cdot \overrightarrow{P_i p} \leq 0$
 - p hors du polygone si
 $\exists i \in [1, 4], \vec{n}_{i,i+1} \cdot \overrightarrow{P_i p} > 0$

324

Remplissage

Test d'appartenance d'un point à polygone

- Test d'angles

$$\sum \alpha_i = 0$$

$$\sum \alpha_i = 2\pi$$

Remplissage

Test d'appartenance d'un point à un polygone

- Test du nombre de points d'intersection

4 points d'intersection

3 points d'intersection

- Attention aux cas limites :
 - points extrémités de segments et segments tangents

Remplissage

Identification d'un polygone

- Polygone convexe
 - Si $p_1x p_2 > 0$ alors ccw

$$p1 \begin{pmatrix} x \\ y \end{pmatrix} \times p2 \begin{pmatrix} x \\ y \end{pmatrix} = p1_x \cdot p2_y - p1_y \cdot p2_x$$

Sommets	Vecteurs	Produits vectoriels
S1	$S4S1 \wedge S1S2$	$[0 \ 3] \times [2 \ 1] = +6$
S2	$S1S2 \wedge S2S3$	$[2 \ 1] \times [0 \ 2] = +4$
S3	$S2S3 \wedge S3S4$	$[0 \ 2] \times [-2 \ 0] = +4$
S4	$S3S4 \wedge S4S1$	$[-2 \ 0] \times [0 \ -3] = +6$

327

Remplissage

Identification d'un polygone

- Polygone concave
 - Changement de sens lors du parcours
 - S3 ré-entrant

Sommets	Vecteurs	Produits vectoriels
S1	$S5S1 \wedge S1S2$	$[0 \ 3] \times [2 \ 1] = +6$
S2	$S1S2 \wedge S2S3$	$[2 \ 1] \times [0 \ 2] = +4$
S3	$S2S3 \wedge S3S4$	$[-1 \ 1] \times [1 \ 1] = -2$
S4	$S3S4 \wedge S4S5$	$[1 \ 1] \times [-2 \ 0] = +2$
S5	$S4S5 \wedge S5S1$	$[-2 \ 0] \times [0 \ -3] = +6$

328

Remplissage

- Algorithme de balayage de lignes
 - Cet algorithme consiste à balayer les lignes horizontales appartenant au rectangle englobant le polygone à traiter, et à afficher les pixels intérieurs au polygone sur chaque ligne.
 - La recherche des intersections des lignes de balayage avec le polygone se fait en calculant l'approximation des segments à l'aide d'un algorithme de traçage et en mémorisant les points écrans correspondants.

329

Remplissage

- Pour chaque côté impliqué, et pour chaque ligne y, traitée le calcul d'intersection nécessite les valeurs suivantes :

- y_{max}
- x_{min}
- dx/dy (1/pente)

Segment : $y \rightarrow [y_{max} \ | \ x_{min} \ | \ dx/dy]$

A-B : $7 \rightarrow [10 \ | \ 1 \ | \ 3/6]$

330

Remplissage

The diagram shows a triangular region defined by vertices A, B, and C on the grid. The region is filled with several horizontal red bands. The vertices are labeled A (top), B (bottom left), and C (bottom right). The base BC has a length of 6 units, and the height AB is 9 units. The area is divided into three main sections by the bands:

- Section A:** The top section from y=0 to y=7. It contains two bands: one from x=1 to x=8 and another from x=7 to x=11. The total width of this section is 8 units.
- Section E:** The middle section from y=7 to y=11. It contains one band from x=1 to x=11. The total width of this section is 11 units.
- Section D:** The bottom section from y=11 to y=18. It contains one band from x=1 to x=11. The total width of this section is 11 units.

Below the triangle, there is a vertical stack of boxes representing the areas of the sections. The stack consists of 9 empty boxes at the bottom, followed by three groups of boxes:

- Group 1:** Contains 10 boxes. The first 8 boxes represent the width of section A (8 units), and the next 2 boxes represent the width of section E (11 units).
- Group 2:** Contains 10 boxes. The first 7 boxes represent the width of section E (11 units), and the next 3 boxes represent the width of section D (11 units).
- Group 3:** Contains 6 boxes. All 6 boxes represent the width of section D (11 units).

A decorative graphic in the top-left corner features a blue ribbon banner with a subtle shadow effect. To its left is a stylized atomic model consisting of three spheres connected by thin lines, set against a light gray background.

Remplissage

- Une fois la liste établie
 - Remplissage selon une règle de parité : incrémentation de la parité à chaque traversée de frontière et tracé si impair.
 - Après tri en x des n intersections P_i ($1 \leq i \leq n$), on tracera des segments entre chaque couple de coordonnées (P_{2j+1}, P_{2i}) avec $1 \leq j \leq n$.

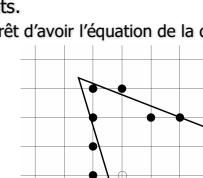
333

The diagram shows a polygon labeled A-B-C-D-E filled with dots. The vertices A, B, C, D, and E are marked. The interior of the polygon is filled with dots, some of which are red and some are green. The red dots are located at vertices A, B, C, D, and E, as well as at several points along the edges. The green dots are located in the central region of the polygon. The polygon is drawn on a grid where the x-axis is horizontal and the y-axis is vertical.



Remplissage

- Gestion des conflits frontaliers
 - En raison de l'approximation de l'algorithme de traçage, les points d'intersections peuvent être à l'extérieur du polygone.
 - On prend les points intérieurs au polygone pour éviter les chevauchements.
 - D'où l'intérêt d'avoir l'équation de la droite traitée : $y_{\max}, x_{\min}, dx/dy$



335

Remplissage

Remplissage de régions (méthode du germe)

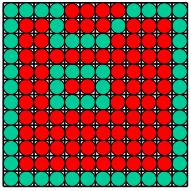
- Cet algorithme traite les régions définies par une frontière.
- À partir d'un pixel intérieur à la région, on propage récursivement la couleur de remplissage au voisinage de ce pixel jusqu'à atteindre la frontière.

Remplissage

- A chaque itération:
 - On remplit tous les pixels pi jusqu'à la couleur limite à droite et à gauche du germe courant.
 - On recherche parmi les pixels au dessus et en dessous des Pi ceux qui sont le plus à gauche d'une suite horizontale maximale à remplir.
 - Ces pixels sont empilés comme germes des itérations suivantes.

337

Remplissage



338

Apparence (Visibilité)

- Les calculs d'ombrage (shading)
- Le placage de texture (plaquage, mappage)

339

Les calculs d'éclaircissement et d'ombrage

340

Définitions

- Illumination**
 - Transport du flux lumineux direct ou indirect depuis les sources lumineuses : modèle local vs. global
- Éclaircissement**
 - Calcul de l'intensité lumineuse en un point de la scène : modèle d'interaction source lumineuse point éclairé
- Ombrage (shading)**
 - Utilisation du modèle éclaircissement pour obtenir la couleur d'un pixel

341

Éclaircissement et Ombrage

- Dépend de :
 - Position du point dans l'espace,
 - Orientation du point (élément de surface),
 - Caractéristiques de la surface (diffusion, réflexion, transparence...)
 - Sources de lumière (directionnelles, ponctuelles...).
 - Position et orientation de la "caméra" ou du point de vue.

342

Sources de lumières

- la lumière **ambiente** : c'est une lumière qui éclaire toute la scène uniformément et qui est uniquement caractérisée par son intensité.
- les sources **ponctuelles** : ce sont des sources de lumière, supposées placées en un point précis et qui rayonnent la lumière radialement; elles sont donc caractérisées par leur intensité et leur position. Elles peuvent être **Isotropique** ou **Anisotropique**.
 - Lorsque la source prend du volume elle devient « **source étendue** »
- les sources **directionnelles** : ce sont des sources de lumière, supposées à l'infini et qui éclairent la scène avec des rayons parallèles à une direction donnée; elles sont donc caractérisées par leur intensité et leur direction.
- les sources de type **Projecteur** ou **spot** : ce sont des sources de lumière caractérisées par leur position, leur direction et un facteur de concentration.

Modèles d'éclairage

- La lumière émise
- La lumière ambiante
- La réflexion diffuse
- La réflexion spéculaire
- Brillance

344

Modèles d'éclairage

- Lumière émise** : Les objets ne sont pas intrinsèquement émetteurs de lumière et n'éclairent donc pas les autres objets mais possèdent ce niveau minimum d'éclairage.

345

Modèles d'éclairage

- Lumière ambiante** : correspond au modèle le plus simple. On considère qu'il existe une source lumineuse présente partout et qui éclaire de manière égale dans toutes les directions.
- Ce modèle de lumière correspond au niveau minimum d'éclairage qui sera appliqué sur les objets.
- En terme physique cela correspond un peu au soleil réfléchi par tout l'environnement qui donne une sorte de lumière présente partout.

346

Modèles d'éclairage

- On définit l'intensité de cette lumière sur une surface comme suit :
$$I_p = p_a I_a$$
- Cette intensité lumineuse est constante sur toute la surface.
 - I_a désigne l'intensité de la lumière
 - p_a est le coefficient de réflexion de la lumière ambiante par la surface
 - $0 \leq p_a \leq 1$
 - I_p correspond à l'intensité de la lumière résultant de la réflexion sur la surface.

347

Modèles d'éclairage

On augmente p_a

- Propriétés :
- On ne voit pas la 3D;
- Modélise simplement l'interréflexion entre toutes les surfaces d'une scène;
- Evite qu'un objet dans l'ombre soit complètement noir

348

Modèles d'éclairement

- La réflexion diffuse :** Dans ce modèle, l'intensité en un point d'une surface dépend de l'angle formé entre le rayon de lumière qui touche le point de la surface et la normale à la surface.
- Plus l'angle formé entre le rayon de lumière et la normale au plan est faible, plus l'intensité lumineuse réfléchie visible par l'observateur est forte.
- Le principe physique qui se cache derrière ce modèle est simple. Notre source lumineuse émet une certaine énergie au mètre carré. Suivant l'incidence des rayons lumineux, cette énergie sera répartie sur une plus ou moins importante surface de l'objet. Si les rayons et la surface sont perpendiculaires, alors l'énergie lumineuse sera répartie sur la plus petite surface possible et donc l'énergie par unité de surface sera maximale.

349

Modèles d'éclairement

- Si on considère que la surface est une surface Lambertienne (surface mate), alors, on peut considérer que la lumière qui arrive sur la surface est réfléchie dans toutes les directions de manière égale. Dans ce cas, la position de l'observateur ne compte pas. La lumière émise en direction de l'observateur dépend donc de :
 - L'intensité de la source lumineuse I_l
 - L'angle θ formé par le rayon de lumière et la normale au plan
 - Coefficient de réflexion pd de la lumière diffusée par la surface
 - $0 \leq pd \leq 1$
- On obtient la formule :

$$I_p = pd \cdot I_l \cdot \cos(\theta)$$

350

Modèles d'éclairement

On augmente pd , $pa = 0$

351

Modèles d'éclairement

diffuse + ambiant

352

Modèles d'éclairement

- La réflexion spéculaire (Modèle de Phong [1973]) :**
 - Le modèle de réflexion spéculaire se différencie du modèle de diffusion en faisant intervenir le point d'observation. Dans ce modèle les rayons de lumière sont réfléchis par symétrie par rapport à la normale à la surface. Ce modèle correspond aux propriétés de "miroir" des objets.

On observe bien l'aspect brillant des objets et les fortes variations en fonction du point de vue.

353

Modèles d'éclairement

- Il faut calculer le rayon réfléchi sur la face. Ensuite, l'intensité de la lumière observée dépend de :
 - θ' qui correspond à l'angle entre le rayon réfléchi et le point d'observation
 - I_l l'intensité de la source de lumière
 - ps : coefficient de réflexion de la lumière spéculaire par la surface
 - $0 \leq ps \leq 1$
- Si on se prend le rayon réfléchi dans l'oeil alors on a l'intensité maximum. Autour de cela, on peut quand même voir quelque chose d'un peu atténué. Cela veut dire que la surface ne réfléchit pas directement la lumière mais qu'il y a une certaine "diffusion" autour du rayon réfléchi. La fonction cosinus joue bien son rôle ici mais comme on veut pouvoir régler la diffusion autour du rayon réfléchi, on introduit le coefficient n . On obtient la formule :

$$Is = ps * I_l * \cos(\theta')^n$$

- n : rugosité
 - ∞ (1024) pour un miroir
 - 1 pour une surface très rugueuse.

354

Modèles d'éclairement

- La rugosité correspond à la brillance (*shininess*) : cette valeur détermine la taille et l'intensité de la tâche de réflexion spéculaire.
- Plus la valeur est grande, et plus la taille est petite et l'intensité importante.

Rayon de réflexion
Point de vue
Normale
Rayon de lumière
theta'
Rayon de réflexion
Point de vue
Normale
Rayon de lumière
theta'

Modèles d'éclairement

La réflexion spéculaire (Modèle Blinn-Phong [1977])

- Évaluation du rayon de réfléchi relativement coûteuse
 - Version simplifiée/accélérée de l'éclairage Phong
- Évaluation du vecteur H : vecteur séparateur point de vue-rayon de lumière

$$Is = ps * Il * \cos(\theta'')^n$$

$$H = \frac{V + L}{2}$$

V : direction de vision
L : source d'éclairage

$$Is = ps * Il * \cos(\theta'')^n$$

Point de vue
theta''
Vecteur H
Normale
Rayon de lumière

Modèles d'éclairement

Blinn-Phong Phong

357

Modèles d'éclairement

diffuse + ambiante + spéculaire

ps n

358

Modèles d'éclairement

- Modèle complet** : on ajoute tout, on pondère avec un coefficient d'atténuation **Fd**

$$I(P) = pa . Ia + Fd . (pd . Il . \cos(\theta_1) + ps . Il . \cos(\theta_2)^n)$$

Incident light
Ideal dif fuse
Directional dif fuse
Ideal specular

359

Modèles d'éclairement

$$I(P) = pa . Ia + Fd . (pd . Il . \cos(\theta_1) + ps . Il . \cos(\theta_2)^n)$$

360

Modèles d'éclairage

$I(P) = pa \cdot I_a + Fd \cdot (pd \cdot I_l \cdot \cos(\theta_1) + ps \cdot I_l \cdot \cos(\theta_2)^n)$

361

Modèles d'éclairage

$I(P) = pa \cdot I_a + Fd \cdot (pd \cdot I_l \cdot \cos(\theta_1) + ps \cdot I_l \cdot \cos(\theta_2)^n)$

362

Modèles d'éclairage

$I(P) = pa \cdot I_a + Fd \cdot (pd \cdot I_l \cdot \cos(\theta_1) + ps \cdot I_l \cdot \cos(\theta_2)^n)$

363

Modèles d'éclairage

- Modèle coloré**
 - une intensité par composante de couleur.
- Plusieurs sources lumineuses**
 - somme des intensités.
- Transparence**
 - manière de combiner couleur de fond et couleur de l'objet.
 - Un paramètre de transparence t .
$$I = t \cdot I(P) + (1-t) \cdot I(\text{derrière } P)$$
- Halo**
 - la couleur dépend de l'épaisseur traversée.

364

Modèles d'ombrage

```

graph TD
    ME[MODELE D'ECLAIREMENT] --> L[LOCAL]
    ME --> G[GLOBAL]
    L --> GOURAUD[GOURAUD]
    L --> PHONG[PHONG]
    G --> RT[RAY-TRACING]
    G --> RS[RADIOSITE]
  
```

365

Modèles d'ombrage

- Les modèles locaux**
 - La luminance à la surface d'un objet est calculée à partir
 - Des paramètres de l'objet
 - Des paramètres des sources de lumière
 - Un objet est considéré comme isolé
- Les modèles globaux**
 - La luminance à la surface d'un objet est calculée à partir
 - Des paramètres de tous les objets de la scène
 - Des paramètres des sources de lumière

366

Les modèles locaux

367

Ombrage de Lambert

Ombrage de Lambert (plat) :

- La méthode d'ombrage la plus simple pour les facettes polygonales est l'ombrage plat (ou constant).
- L'idée est de calculer une seule valeur d'illumination pour l'ensemble de la facette.
- Par exemple au point milieu de la facette en prenant pour normale à la surface celle du plan contenant la facette.

Ombrage plat de la sphère pour : 16 16 facettes, 32 32 et 64 64 368

Ombrage de Lambert

$$I_p = pd \cdot Il \cdot \cos(\theta)$$

- Composantes :
 - Il** : intensité de la source lumineuse
 - theta** : angle formé par le rayon de lumière et la normale au plan
 - pd** : coefficient de réflexion de la lumière diffus par la surface
 - matériau diffus**
- Nous répétons le même schéma de calcul pour les autres composantes d'éclairage : spéculaire, ambiant, etc.

369

Ombrage de Lambert

Problème :

- Il y a des discontinuités le long des facettes;
- L'œil exagère les changements d'intensité et les changements de pente de l'intensité => effet Mach Banding;

Solution => Interpolation.

370

Ombrage de Gouraud

Henri Gouraud

- Ombrage de Gouraud :**
- La méthode développée par Gouraud [1971] élimine les discontinuités d'intensité sur une facette polygonale par interpolation des valeurs d'intensité aux sommets des facette.
- Cette méthode est largement utilisée et se retrouve dans la majorité des matériaux graphiques existants (librairies, cartes graphiques).

371

Ombrage de Gouraud

- Cette méthode requiert la connaissance de la normale à la surface aux sommets des facettes polygonales.

372

Ombrage de Gouraud

- Cette méthode requiert la connaissance de la normale à la surface aux sommets des facettes polygonales.
- Ces normales peuvent être soit données soit déterminées en calculant la moyenne des normales des facettes partageant un sommet P

$$N_s = \frac{\sum N_i}{|\sum N_i|}$$

N_s : normale du sommet traité
N_i : normales des faces incidentes

373

Ombrage de Gouraud

- Lorsque les normales sont connues, les intensités (diffuses) aux sommets des facettes polygonales sont calculées.

$$I_p = pd \cdot Il \cdot \cos(\theta)$$

N_{s1}, *I_{s1}*, *N_{s2}*, *I_{s2}*, *N_{s3}*, *I_{s3}*

374

Ombrage de Gouraud

- On réalise interpolation bilinéaire des intensités lumineuses sur toute la face
 - Interpolation suivant les arêtes

$$I_{gauche} = I_{s1} \frac{y - y_2}{y_1 - y_2} + I_{s2} \frac{y_1 - y}{y_1 - y_2}$$

$$I_{droite} = I_{s1} \frac{y - y_3}{y_1 - y_3} + I_{s2} \frac{y_1 - y}{y_1 - y_3}$$

375

Ombrage de Gouraud

- On réalise interpolation bilinéaire des intensités lumineuses sur toute la face
 - Interpolation suivant les arêtes
 - Interpolation horizontale à l'intérieur de la face
 - L'interpolation s'effectue à l'aide de l'algorithme de balayage de ligne utilisée pour le remplissage de polygone et le z-buffer : Scan-line

$$I = I_{gauche} \frac{x - x_2}{x_1 - x_2} + I_{droite} \frac{x_1 - x}{x_1 - x_2}$$

376

Ombrage de Gouraud

```

InterpolationGouraud(polygone)
{
  POUR chaque polygone FAIRE
    calcul de la normale au polygone

  POUR chaque sommet (S1, S2, etc.) du polygone FAIRE
    calcul de la normale : moyenne des normales des faces incidentes

  POUR chaque sommet (S1, S2, etc.) du polygone FAIRE
    calcul l'intensité lumineuse d'un modèle d'éclairage (diffus, spéculaire, etc.)

  POUR chaque point S FAIRE
    SI S appartient à une arête [S1,S2] du polygone ALORS
      calcul de l'intensité lumineuse Igauche || Idroite par interpolation linéaire entre [Is1, Is2]
    Sinon
      calcul de l'intensité lumineuse I par interpolation linéaire horizontale entre[Igauche, Idroite]
}
  
```

377

Ombrage de Gouraud

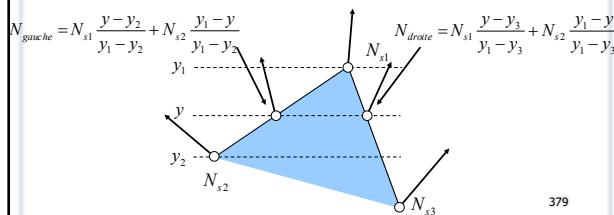
- Caractéristiques :
 - Technique efficace en terme de temps de calcul pour l'obtention d'un réalisme moyen.
 - Implantation facile dans le cadre de la programmation de l'algorithme du Z-Buffer.
 - L'utilisation d'une interpolation entre des valeurs numériques n'est qu'une approximation qui peut conduire à des défauts de visualisation.

378

Ombrage de Phong

Ombrage de Phong [1973] :

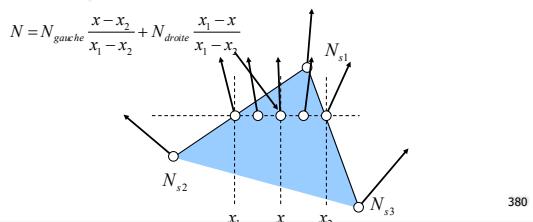
- Consiste à déterminer la normale en un point d'une facette polygonale par interpolation des normales aux sommets de cette facette.
- Interpolation suivant les arêtes



Ombrage de Phong

Ombrage de Phong [1973] :

- Consiste à déterminer la normale en un point d'une facette polygonale par interpolation des normales aux sommets de cette facette.
- Interpolation suivant les arêtes
- Interpolation horizontale à l'intérieur de la face



Ombrage de Phong

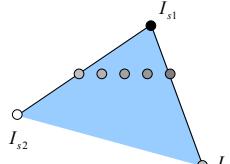
Utilisation des modèles d'illumination pour calculer les intensités lumineuses

- Modèle diffus

Diagram showing a light ray hitting a point on a triangle. The angle between the normal and the ray is \$\theta\$. The formula for diffuse lighting is:

$$I_p = pd \cdot Il \cdot \cos(\theta)$$

pd : matériau diffus



Ombrage de Phong

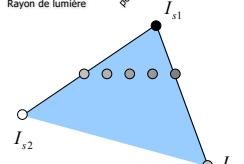
Utilisation des modèles d'illumination pour calculer les intensités lumineuses

- Modèle spéculaire

Diagram showing a light ray hitting a point on a triangle. The angle between the normal and the ray is \$\theta\$. The formula for specular lighting is:

$$I_s = ps * Il * \cos(\theta'')^n$$

ps : matériau spéculaire



Ombrage de Phong

Ombrage de Phong [1973] :

- L'intérêt de cette approche par rapport à l'ombrage de Gouraud réside principalement dans sa capacité à traiter les réflexions spéculaires.
- Gouraud ne permet pas de prendre en compte les réflexions spéculaires lorsque celles-ci sont localisées au centre d'une facette.

383

Modèles d'ombrage



Modèles d'ombrage

- Ombrage de Phong [1973] :**
 - L'ombrage de Phong est d'une manière générale meilleur que celui de Gouraud car l'interpolation est effectuée sur les normales et non les intensités.
 - Cela même pour un modèle d'éclairage sans réflexion spéculaire.
 - Augmente le coût du rendu.

385

Modèles d'ombrage

Ombrage plat Ombrage de Gouraud Ombrage de Phong

386

Modèles d'ombrage

```

InterpolationPhong(polygone)
{
    POUR chaque polygone FAIRE
        calcul de la normale au polygone

    POUR chaque sommet (S1, S2, etc.) du polygone FAIRE
        calcul de la normale : moyenne des normales des faces incidentes

    POUR chaque point S du polygone FAIRE
        SI S appartient à une arête [S1,S2] du polygone ALORS
            calcul de la normale N_gauche || N_droite par interpolation linéaire entre [N_s1, N_s2]
        Sinon
            calcul de l'intensité lumineuse N par interpolation linéaire horizontale entre[N_gauche, N_droite]

    POUR chaque point S du polygone FAIRE
        calcul l'intensité lumineuse d'un modèle d'éclairage (diffus, spéculaire, etc.)
}

```

387

Modèles d'ombrage

- Rajouter des couleurs
 - Trois composantes

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r} p_{a_r} \\ I_{a_v} p_{a_v} \\ I_{a_b} p_{a_b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l_r} p_{d_r} \\ I_{l_v} p_{d_v} \\ I_{l_b} p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r} p_{s_r} \\ I_{l_v} p_{s_v} \\ I_{l_b} p_{s_b} \end{bmatrix}$$

388

Modèles d'ombrage

- Rajouter des couleurs
 - Trois composantes

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r} p_{a_r} \\ I_{a_v} p_{a_v} \\ I_{a_b} p_{a_b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l_r} p_{d_r} \\ I_{l_v} p_{d_v} \\ I_{l_b} p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r} p_{s_r} \\ I_{l_v} p_{s_v} \\ I_{l_b} p_{s_b} \end{bmatrix}$$

Rouge
Vert
Bleu

389

Modèles d'ombrage

- Rajouter des couleurs
 - Trois composantes

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r} p_{a_r} \\ I_{a_v} p_{a_v} \\ I_{a_b} p_{a_b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l_r} p_{d_r} \\ I_{l_v} p_{d_v} \\ I_{l_b} p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r} p_{s_r} \\ I_{l_v} p_{s_v} \\ I_{l_b} p_{s_b} \end{bmatrix}$$

Matière ambiant Matière diffus Matière spéculaire

390

Modèles d'ombrage

- Rajouter des couleurs
 - Trois composantes

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r}p_{a_r} \\ I_{a_v}p_{a_v} \\ I_{a_b}p_{a_b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l_r}p_{d_r} \\ I_{l_v}p_{d_v} \\ I_{l_b}p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r}p_{s_r} \\ I_{l_v}p_{s_v} \\ I_{l_b}p_{s_b} \end{bmatrix}$$

Source ambiante Source diffuse Source spéculaire

391

Modèles d'ombrage

- Rajouter des couleurs
 - Trois composantes

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r}p_{a_r} \\ I_{a_v}p_{a_v} \\ I_{a_b}p_{a_b} \end{bmatrix} + (N \cdot L) \begin{bmatrix} I_{l_r}p_{d_r} \\ I_{l_v}p_{d_v} \\ I_{l_b}p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r}p_{s_r} \\ I_{l_v}p_{s_v} \\ I_{l_b}p_{s_b} \end{bmatrix}$$

Couleur finale

392

Modèles d'ombrage

- Plusieurs sources d'éclairage

$$\begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} = \begin{bmatrix} I_{a_r}p_{a_r} \\ I_{a_v}p_{a_v} \\ I_{a_b}p_{a_b} \end{bmatrix} + \sum_{ll=lights} \left((N \cdot L) \begin{bmatrix} I_{l_r}p_{d_r} \\ I_{l_v}p_{d_v} \\ I_{l_b}p_{d_b} \end{bmatrix} + (N \cdot H)^n \begin{bmatrix} I_{l_r}p_{s_r} \\ I_{l_v}p_{s_v} \\ I_{l_b}p_{s_b} \end{bmatrix} \right)$$

393