Contributeurs

 **alain marty**

Randonnée
Infographique
Chemins de traverse

Introduction

Les logiciels de dessin assisté par ordinateur 2d et 3d fleurissent et l'on peut se demander s'il est encore utile aujourd'hui d'étudier la géométrie, et si oui, quelle géométrie. Les logiciels de DAO forment une population en perpétuelle transformation, ils ont un comportement instable, parfois même mystérieux; ils sont très doués pour le show-bizz, beaucoup moins pour l'enseignement. Pour en maitriser un, il faut en étudier dix, il faut dégager l'essentiel de l'accessoire, trouver les points communs, et découvrir les invariants. Vient le moment inévitable des questions fondamentales, et le début d'une longue randonnée dans le monde de l'Infographie, peuplé de tous les êtres mathématiques possibles et imaginables. Et dans ce monde règne la géométrie !

Dans l'exposé d'aujourd'hui, afin d'illustrer le caractère central de la réflexion géométrique tous azimuts, je vous propose de suivre les quelques chemins de traverse suivants parcourus à l'occasion de l'écriture d'un logiciel minimaliste appelé • µ3d • :

1) une réflexion sur la génération des formes,
2) une réflexion sur la composition des formes,
3) une réflexion sur la perspective.

Sur la génération des formes

Comment décrire, représenter, manipuler et combiner de façon unifiée des segments, des polygones, des coniques, des quadriques, des prismes, des surfaces de révolution, des tubes constants ou modulés, des surfaces minimales (ou au moins à Laplacien nul), des surfaces interpolant "doucement" des courbes gabarits, des bananes, des hélices, des coquilles Saint Jacques, le pelage d'un ours (peau et poils), une sphère dans l'espace R4 ?
Les courbes de Bézier constituent une réponse à ce problème, réponse certainement partielle, mais étonnamment simple et riche: il suffit de quelques points pour les définir, elles sont contenues dans leur polygone de contrôle et elles sont invariantes dans les transformations affines; leur expression algébrique est élémentaire (polynômes de Bernstein, triangle de Pascal); l'algorithme de de Casteljau en ramène le calcul et la représentation à celle d'une suite récursive d'interpolations linéaires et de tracés de segments; et la composition de plusieurs courbes de Bézier produit de même des formes (surfaces, volumes, hypersurfaces,...) qui bénéficient de ces propriétés "linéaires par récursion". Enfin il suffit de travailler dans l'espace R4 pour représenter avec ces courbes les formes rationnelles (quotient de polynômes) comme les coniques et les quadriques. L'étude de l'interpolation N_linéaire de 2N points dans Rn conduit à la génération automatique d'une famille de formes, et notamment aux courbes et aux surfaces de Bézier.

1) Interpolation linéaire de 2 points:

L'idée de base est simple: un point P peut toujours être vu comme interpolation linéaire de deux autres points quelconques. On définit ainsi un segment de droite C(u), qui peut s'exprimer analytiquement de la façon suivante:

$P = (1-u).P0 + u.P1$, avec P0 et P1 dans Rn, et u dans [0,1].

2) Interpolation bilinéaire de 4 points:

Les points précédents P0 et P1 peuvent être vus chacun comme interpolation linéaire de deux autres points quelconques; avec:

$P0 = (1-v).P00 + v.P10$, $P1 = (1-v).P01 + v.P11$, et v dans [0,1]

on obtient:

$P = (1-u)(1-v).P00 + u(1-v).P10 + (1-u)v.P01 + uv.P11$

On définit ainsi une surface quadrilatère gauche S(uv), un PH. Mais on peut définir également une courbe quadrique en égalant u et v:

$P = (1-u)2.P00 + u(1-u).(P10 + P01) + u2.P11$

Enfin, en égalant les points P10 et P01, et en changeant la notation des points, on aboutit à l'expression:

$P = (1-u)2.P0 + 2u(1-u).P1 + u2.P2$

dans laquelle on reconnait l'expression classique d'une courbe de Bézier.

3) Interpolation trilinéaire de 8 points:

Les points précédents P00, P10, P01 et P11 sont vus chacun comme interpolation linéaire de deux autres points quelconques; avec

$P00 = (1-w).P000 + w.P101$, $P10 = (1-w).P100 + w.P101$,
$P01 = (1-w).P010 + w.P111$, $P11 = (1-w).P110 + w.P111$, et w = [0,1]

on obtient:

```
    P = (1-u)(1-v)(1-w).P000 + u(1-v)(1-w).P100 + (1-u)v(1-w).P010
          + (1-u)(1-v)w.P001 + u v(1-w).P110 + u(1-v)w.P011 + (1-u)vw.P101 + uvw.P111
```

On définit ainsi 1 volume cuboïde V(u,v,w), dans lequel le point courant est exprimé en fonction de 8 points quelconques de l'espace. Définir un cube devient évident, il n'est plus nécessaire de passer par une structure complexe à base de sommets (8), d'arêtes (12), de faces (6). Un cube est l'ensemble des points satisfaisant à u,v,w = [0,1]; une face est le sous-ensemble des points satisfaisant par exemple à w = 0; une arête le sous-ensemble w = v = 0. Il faut bien sur que le moteur de rendu accepte ces données et sache en extraire les informations nécessaires à la représentation souhaitée (cube plein, cube mille-feuilles, cube semi de points,...). Comme nous l'avons fait au § précédent, on peut également définir 3 surfaces, par exemple en égalant u et v on obtient:

```
    P =     (1-u)2      ((1-w).P000  + w.P001)
        + u(1-u)    ((1-w).(P100 + P010) + w.(P011 + P101))
        + u2        ((1-w).P110  + w.P111)
```

Avec: P000=P0, P001=Q0, P100=P010=P1, P011=P101=Q1, P110=P2, P111=Q2
on obtient:

```
    P = (1-u)2((1-w).P0+w.Q0 )+2u(1-u)((1-w).P1+w.Q1)+u2((1-w).P2 + w.Q2)
```

qui est une surface interpolant linéairement 2 quadriques.

En égalant u, v et w, et en fusionnant quelques points, on obtient enfin une courbe cubique C (u):

```
    P = (1-u)3P0 + 3(1-u)2uP1 + 3(1-u)u2P2 + u3P3
```

4) Interpolation N_linéaire de 2N points:
Au delà de 3, une interpolation 4_linéaire de 24 points permettrait de définir:
1 hypervolume cuboïde V4(u,v,w,x),
3 volumes réglés sur 4 quadriques V(u,v,w), V(v,w,x), V(u,w,x),
6 surfaces réglées sur 2 cubiques S(u,v), S(u,w), ..., S(w,x)
1 courbe polynomiale de degré 4.

Et ainsi de suite... On possède ainsi un outil conceptuel pour engendrer toute une famille de formes pouvant devenir complexe tout en conservant des propriétés fondamentalement linéaires. On note la correspondance entre l'interpolation N_linéaire de 2N points et les courbes de Bézier définies par un polynome d'ordre N sur N+1 points obtenus par la fusion des points de départ. Les propriétés de ces courbes sont nombreuses et bien connues:

1) invariance dans une transformation affine: la courbe construite sur les points transformés est la transformée de la courbe construite sur les points initiaux. On peut également définir des courbes de Bézier rationnelles, qui seront invariantes dans une transformation projective. En les considérant dans R4, on est ramené au problème de la courbe de Bézier sous sa forme classique.

2) enveloppe convexe: la courbe est entièrement contenue dans la surface construite sur le polygone de contrôle; facilité de calcul pour les intersections, mais également similitude entre le polygone et la courbe engendrée: le polygone “guide” bien la courbe.

3) calcul: l'évaluation d'une courbe en un point peut être ramenée à une suite récursive d'interpolations linéaires (algorithme de de Casteljau); voici par exemple une expression de l'algorithme qui pourrait être utilisé pour tracer une courbe de Bézier:

```
tracer_bezier ( POINT3D *P, INT NIVEAU )
{   si ( polygone_assez_plat (P) ou NIVEAU == 0)
        tracer_polygone ( P )
    sinon
    {   Pg = polygone_gauche( P )
        Pd = polygone_droite( P )
        tracer_bezier ( Pg, NIVEAU - 1 )
        tracer_bezier ( Pd, NIVEAU - 1 )
    }
}
```

4) les dérivées et primitives d'une courbe de Bézier sont des courbes de Bézier. Par exemple, pour une cubique:

```
    P  = (1-u)3.P0 + 3(1-u)2u.P1 + 3(1-u)u2.P2 + u3.P3
```

la dérivée première est une quadrique:

```
    P’ = (1-u)2.Q0 + 2u(1-u).Q1 + u2.Q2,
```

et la dérivée seconde est une droite:

```
    P” = (1-u).R0 + u.Q1
```

en posant   Q0 = 3(P1 - P0), Q1 = 3(P2 - P1), Q2 = 3(P3 - P2)

R0 = 6(Q1 - Q0), R1 = 6(Q2 - Q1)

On peut noter également que les formes étudiées sont dans Rn, avec une valeur de n quelconque; on peut en particulier travailler dans R4 (x,y,z,w), et revenir dans R3 par une transformation projective sur l'origine à travers le plan w = 1; les formes se présentent alors sous la forme de quotients de polynomes (fonctions rationnelles), ce qui permet de définir notamment les coniques. Par exemple, la quadrique dans R4:

P = (1-u)2.P0 + 2u(1-u).P1 + u2.P2,
avec P0 = {1,0,0,1}, P1 = {1,1,0,1}, P2 = {0,2,0,2},

s'écrit en fait:

x = 1-u2, y = 2u, z = 0, w = 1+u2,

et le passage dans R3 conduit à l'expression exacte d'un arc de cercle:

x = (1-u2)/(1+u2) = cos (a),
y = 2u/(1+u2) = sin (a), avec u = tan(a/2).

Enfin on peut également noter que la simple cubique dans R3:

P = (1-u)3.P0 + 3(1-u)2u.P1 + 3(1-u)u2.P2 + u3.P3
avec P0 = {1,0,0}, P1 = {1,F,0}, P2 = {F,1,0}, P3 = {0,1,0}, et F= 4/3(√2-1)

est une très bonne approximation (0.001) du cercle unité sur [0,π/2], comme le montre la figure en annexe, planche 2. Ainsi, dans un premier temps n'est-il pas nécessaire de recourir aux courbes rationnelles pour composer par exemple une courbe et un quasi-arc de cercle et obtenir une surface de révolution.

Sur la composition des formes

On peut très bien imaginer un logiciel minimaliste de modélisation de formes qui ne disposerait (pour l'essentiel) que de 2 outils élémentaires: l'outil sélection_3D, et l'outil polygone_3D. Chaque polygone pouvant être qualifié plus ou moins lisse (de la représentation polygonale à la représentation courbe parfaite). Penser à une enfilade de colonnes cylindriques et au problème de la définition nécessaire et suffisante pour la première colonne, puis pour celle qui se trouve très loin et réduite à un point: dans les deux cas, 4 points suffisent à définir l'arc de cercle et la profondeur de récursion pour le tracé dépendra de la taille (3 segments pour la plus éloignée, une centaine peut-être pour la première). Au delà, il est bien sur nécessaire de disposer d'outils de composition, et on peut imaginer les touches du clavier numérique “ +, -, *, / “ comme des opérateurs combinant les formes sélectionnées pour produire d'autres formes plus complexes: par exemple deux arcs de cercle sélectionnés produisent un tore; le tore et un segment sélectionnés produisent un volume, qui sera visualisé sous la forme d'un tube épais si seules sont représentées les faces limites, ou sous la forme d'un tube poilu si on décide un autre mode de représentation. L'idée est que les opérateurs de composition s'appliquent à des formes de dimension quelconque pour produire d'autres formes, et on pense à un langage structuré, un groupe de formes avec ses opérateurs commutatifs ou pas, etc...

L'opérateur * est l'outil essentiel de composition, permettant de fusionner des formes primitives en une famille de formes plus complexes dans laquelle pourra choisir l'utilisateur (la barre espace du clavier peut servir de permutateur). Quelques exemples:

1) Le produit de N polygones de M points génère un polyèdre interpolant de N*M points, dont le lissage conduit à une surface constituée de carreaux de S3Bezier raccordés suivant les conditions de raccordement des polygones primitifs. Soient par exemple 4 C2Bezier:

Pi = (1-u)2.Pi0 + 2(1-u)u.Pi1 + u2.Pi2 avec i = [0,3]

L'expression du point courant de la surface est simplement:

P = (1-v)3.P0 + 3(1-v)2v.P1 + 3(1-v)v2.P2 + v3.P3

2) Le produit de 4 polygones (donc courbes) C1, C2, C3, C4 formant un contour fermé conduit à une double interpolation bilinéaire pondérée (carreau de coons). En voici l'expression:

P = (1-u).C1(v) + u.C2(v) + (1-v).C3(u) + v.C4(u)
- ((1-u)(1-v).C1(0) + u (1-v).C2(0) + (1-u) v.C1(1) + uv.C2(1))

3) Le produit de deux courbes peut également conduire à un tubage: chaque point de la surface s'exprime de la façon suivante, en appelant section la première courbe et chemin la seconde:

P = chemin(u) + TNB(u) • section(v),

ou bien encore:

P = A(u) • section(v),
```
avec    A(u) =  |   TNB(u)       chemin(u)   |
                |   0       1       |
```

dans laquelle A(u) est une matrice 4x4, TNB est une matrice 3x3 construite sur les vecteurs tangente, normale et binormale (le trièdre de Serret-Frenet); ces vecteurs sont exprimables facilement à partir des dérivées première et seconde du secteur de C3Bezier, P' et P”, par les relations:

T = P' / |P'|,
B = P' x P" / |P' x P"|, où x est le produit vectoriel.
N = B x T

Par exemple, le tubage d'une courbe le long d'un arc de (quasi) cercle engendre une surface de révolution (donc peut engendrer une portion de sphère, de tore, de cylindre, d'hyperboloïde de révolution, etc). La forme :
P = A(u) • section(v),

conduit au cas d'une transformation affine générale, dans laquelle la matrice A(u) est construite sur une composition plus complexe de fonctions, par exemple en concaténant à la matrice TNB une homothétie fonction de u ou de v. On peut ainsi imaginer des tubages modulés tout le long de la courbe chemin suivant les trois vecteurs tangente, normale et binormale. Pour illustrer ce qui précède, voici en langage C++ l'expression de l'équation de la surface tubage évolutif présentée en annexe:

```
VEC TUBE_EVOLUTIF::equation (FLOAT u, FLOAT v)
{   VEC section     = c1->equation( v ),    // c1, c2 et c2 sont les
        chemin  = c2->equation( u ),    // courbes en entrée
        modulante   = c3->equation( u ),
        t, n, b;                // vecteurs du trièdre TNB
    c2->triedre(u, &t, &n, &b );        // calculés ici
    return  ( chemin    + section.x * n * modulante.x
            + section.y * b * modulante.y
            + section.z * t * modulante.z );
}
```

Dans l'exemple, la courbe section est un arc de cercle de 270°, la courbe chemin est une C4Bezier, la courbe modulante est sinusoïdale en x, y et z, ce qui permet d'appliquer à la courbe section une homothétie sur la normale et la binormale, et même sur la tangente. Ainsi, sur la base de compositions purement linéaires, peut-on aisément ouvrir le champ à la génération de formes complexes, qui pourront elles-même se comporter comme opérandes pour de nouvelles compositions.

Sur la perspective
En travaillant dans l'espace R4, la translation, la rotation, et l'homothétie et même la perspective, sont des transformations exprimables sous forme de matrices 4x4, elles sont donc linéaires, peuvent être concaténées, et peuvent être inversées. En particulier, on peut voir la perspective inverse comme une opération qui consiste à passer de l'espace vu (virtuel ou réel?) dans lequel les formes ne sont pas invariantes dans les transformations, à un espace construit (réel ou virtuel?) dans lequel elles deviennent invariantes. On peut imaginer une norme invariante dans la transformation perspective:
ds2 = dx2 + dy2 + dz2 -(H2-F2)dw2,

avec F = focale, et H = hauteur de l'observateur, qui sont les paramètres de la construction perspective classique. On pense aux transformations de Lorentz, qui conservent la norme:
ds2 = dx2 + dy2 + dz2 - c2dt2.

Et l'on se demande s'il ne faudrait pas mettre la relativité en perspective...

Un pemier petit rappel: on établit une correspondance entre l'espace R3 et l'espace R4, en associant à un point de R3 P(x,y,z) un point de R4 P(X,Y,Z,W), tel que:
P(X/W,Y/W,Z/W) = P(x,y,z,1).

Les transformations étudiées seront appliquées au point de R4.

Un rappel sur la translation, la rotation et l'homothétie:

1) rotation: une rotation d'angle a par rapport à l'axe Oz transforme un point P(x,y,z,w) en un point P'(x',y',z',w'), suivant l'expression suivante:

P' = R(a) • P,

```
avec    R(a) =  | cos(a) -sin(a)    0   0   |
        | sin(a)  cos(a)    0   0   |
        | 0 0   1   0   |
        | 0 0   0   1   |,  z et w inchangés.
```

2) homothétie: l'expression matricielle d'une homothétie de coefficients hx, hy, hz s'écrit:

```
    R =     | hx    0   0   0   |
        | 0     hy  0   0   |
```

```
        | 0 0   hz  0   |
        | 0 0   0   1   |,  w est inchangé
```

3) translation: l'expression matricielle d'une translation de coefficients tx, ty, tz s'écrit:

```
T =     | 1     0   0   tx      |
        | 0     1   0   ty      |
        | 0 0   1   tz  |
        | 0 0   0   1   |,  w est inchangé
```

4) perspective conique: on a l'habitude de considérer la transformation perspective comme un passage de R3 dans R2, parcequ'on s'intéresse avant tout à l'image projetée de la scène; mais traiter la transformation perspective comme une transformation de R3 dans R3 présente le grand avantage de ne pas perdre l'information de la troisième dimension (z), et passer dans R4 en fait une transformation linéaire. Pour écrire l'expression de cette transformation, en marge des approches habituelles de la littérature infographique, nous proposons une relecture de la construction classique de la perspective “Renaissance”, avec F la distance de l'observateur à la fenêtre, et H représentant la hauteur de la ligne d'horizon (la demi-hauteur de la fenêtre). Avec ces notations, quelques considérations géométriques élémentaires (triangles semblables) conduisent aux expressions suivantes dans R3 de x', y' et z' en fonction de x, y et z:

```
x' = x.F/z      // x' et y' représentent le point
y' = y.F/z      // projeté sur l'écran
z' = H(1 - F/z)         // z' conserve l'information z
```

où l'on retrouve bien z' = 0 pour z = F et z' = H pour z = infini.

On en déduit l'expression de la transformation dans R4:

```
x' = x  soit    P =     |1  0   0   0   |
y' = y          |0  1   0   0   |
z' = H(1/F.z - 1)       |0  0   H/F -H  |
w' = z/F            |0  0   1/F 0   |
```

En concaténant à cette transformation perspective une translation qui amène l'observateur au centre de la fenêtre (z = z + F), nous exprimons finalement la transformation totale et son inverse sous la forme suivante:

```
P =     |1  0   0   0   | et    | 1 0 0 0 | si H= F= 1
        |0  1   0   0   |   | 0 1 0 0 |
        |0  0   H/F 0   |   | 0 0 1 0 |
        |0  0   1/F 1   |   | 0 0 1 1 |


P-1=    |1  0   0   0   | et    | 1 0 0 0 | si H= F= 1
        |0  1   0   0   |   | 0 1 0 0 |
        |0  0   F/H 0   |   | 0 0 1 0 |
        |0  0   -1/H    1   |   | 0 0-1 1 |
```

Les inverses des autres transformations (translation, rotation, homothétie) étant évidentes, nous avons tout ce qu'il nous faut pour conserver la transformation inverse de toute concaténation de transformations et donc de conserver un lien biunivoque entre un point de R4 et son transformé dans R4 , et donc en particulier entre un point de l'espace appartenant à un plan et sa projection sur l'écran. Le programme µ3d utilise ce lien pour assurer les interactions entre les formes et l'utilisateur: le déplacement du curseur sur l'écran induit par défaut le déplacement du point courant associé sur un plan parallèle au plan Oxy du repère courant de définition du point; une touche de modification, comme “option” ou “majuscule”, pourra astreindre le déplacement à une droite parallèle à l'axe Oz de ce même repère. On dispose donc ainsi d'un outil permettant de positionner ou de sélectionner un point dans l'espace 3D, à partir de la position du curseur sur l'écran.

Une petite réflexion libre sur la perspective maintenant. Etudions les variations dans une transformation perspective de la distance entre deux points P1 et P2 de R4 du plan w = 1 :

```
Δx' = x2' - x1' = x2 - x1               = Δx
Δy' = y2' - y1' = y2 - y1               = Δy
Δz' = z2' - z1' = ((H/F).z2 - H) - ((H/F).z1 - H)   = (H/F).Δz
Δw' = w2' - w1' = (1/F).z2 - (1/F).z1           = (1/F).Δz
```

Le carré de la distance des points initiaux est donné par:

```
Δs2     = Δx2 + Δy2 + Δz2 + Δw2
     = Δx2 + Δy2 + Δz2      // car Δw est toujours nul.
Δs'2    = Δx'2 + Δy'2 + Δz'2 + Δw'2
    = Δx2 + Δy2 + (H/F)2Δz2 + (1/F)2Δw2
```

$= \Delta x^2 + \Delta y^2 + (H/F)^2\Delta z^2$
$\neq \Delta s^2$

La métrique euclidienne classique n'est donc pas conservée, par contre, en définissant la métrique suivante:

$\Delta s^2 = \Delta x^2 + \Delta y^2 + \Delta z^2 - (H^2 - F^2)\, \Delta w^2$,

on obtient:

$$\begin{aligned}\Delta s'^2 &= \Delta x'^2 + \Delta y'^2 + \Delta z'^2 - (H^2 - F^2)\, \Delta w'^2\\ &= \Delta x^2 + \Delta y^2 + (H/F)^2\Delta z^2 - (H^2 - F^2)\, (1/F)^2\Delta z^2\\ &= \Delta x^2 + \Delta y^2 + \Delta z^2\\ &= \Delta s^2\end{aligned}$$

et donc conservation de la métrique. En relativité restreinte, la transformation de Lorentz est une rotation autour de l'axe du temps qui laisse invariante la métrique:

$\Delta s^2 = \Delta x^2 + \Delta y^2 + \Delta z^2 - c^2\Delta t^2$.

On peut s'interroger sur les similitudes qui apparaissent, et au delà se poser des questions sur la perception que nous avons de l'espace qui nous entoure.

Prenons un exemple: la perception bidimensionnelle (l'image) d'un cube qui se déplace à petite vitesse et loin d'un observateur est celle d'une forme invariante. Si ce cube se déplace à grande vitesse sa forme n'est plus invariante, l'observateur constate une contraction de la dimension dans le sens du déplacement; il imagine alors que ce cube est la projection dans R3 d'un hyper_cube de l'espace R4 muni de la métrique relativiste; il a ainsi construit un objet invariant qui ne subit qu'une simple rotation sur l'axe du temps. Si maintenant le cube se déplace lentement mais à faible distance de l'observateur, il est perçu en continuelle déformation; et l'on peut imaginer de même que ce cube est la projection dans R2 d'un hyper_cube de l'espace R4 muni de la métrique perspective , qui reste invariant et ne subit qu'un simple déplacement sur l'axe des profondeurs. Les effets (déformations) relativistes apparaissent aux grandes vitesses, les effets (déformations) perspectifs apparaissent aux faibles éloignements.

En tout cas, comme la construction relativiste, la perspective inverse apparait comme une opération qui consiste à passer de l'espace vu (virtuel ou réel?) dans lequel les formes ne sont pas invariantes dans les transformations, à un espace construit ou représenté (réel ou virtuel?) dans lequel elles deviennent invariantes. La perspective directe est un mode de représentation de l'espace 3D, et la perspective inverse peut être considérée comme un mode de conception de l'espace 3D. Les constructions de la géométrie descriptive (art de raisonner sur des objets 3D à l'aide exclusive de constructions 2D) sont des moyens d'imaginer et de travailler facilement sur des objets invariants, dont nous ne percevrons jamais que la projection bidimensionnelle et mouvante sur notre rétine. Peut-on envisager pour mieux visualiser les raisonnements relativistes des outils comme ceux que la géométrie descriptive apporte à la représentation et à la conception des formes géométriques 3D ? Faut-il mettre la relativité en perspective ?

Conclusion

Ces quelques chemins de traverse parcourus à l'occasion de l'écriture du projet “µ3d” font appel à quelques connaissances géométriques de base, simples et assez diverses. On peut se demander si elles sont exigibles pour le simple utilisateur ?

Prenons l'exemple du tableur qui a révolutionné la micro-informatique, cette partie de l'informatique qui est accessible au commun des mortels. L'idée forte du concept de tableur réside dans le fait que les entrées et les données sont gérées et affichées de façon unique par le logiciel, et que l'utilsateur dispose d'un langage simple et universel pour combiner à l'infini les relations entre ces données. Avec un tableur, on peut gérer sa comptabilité, les situations de travaux d'un chantier, mais également on peut calculer et afficher les déformées d'une poutre sur appuis continus, ou la forme d'une surface (presque) minimale satisfaisant à l'équation de Laplace et à certaines conditions aux limites. Tout celà suppose de l'utilisateur la connaissance du sujet à traiter!

Quel pourrait être le tableur de la 3D ? Le tableur de la 3D devra certainement avoir cette unicité d'entrée et d'affichage des données, et un langage minimum et universel pour la définition et la composition des formes, et comme le tableur, il supposera toujours de l'utilisateur la connaissance du sujet à traiter. C'est la raison pour laquelle je crois que l'enseignement de la géométrie dans les Ecoles d'Architecture, probablement sous de nouvelles formes restant à déterminer, restera toujours une nécessité.

... et pendant que nous nous posons ces graves questions, la jeune génération s'entraine sans complexes sur de superbes jeux vidéo qui préfigurent les outils de CAO de demain (interaction avec des espaces complexes et mouvants, déplacements en temps réel, contacts, textures riches, multimedia,...). Il est bien connu que demain, l'architecte ne tracera plus un plan, une coupe ou une élévation. Comme le proclament les vendeurs de logiciels 3D, qui savent ce qu'ils disent: “à bas l'abstraction castratrice du plan, seul le modèle 3D compte...”

Par analogie avec ce que l'on peut observer dans l'industrie, où les machines numériques des ateliers de production travaillent directement à partir des données 3D, on imagine les futurs chantiers du batiment avec des manoeuvres robots alimentés en données 3D par modem et satellite depuis l'ordinateur de l'architecte. Et pour construire ce modèle 3D, comme dans un jeu, on imagine l'architecte entrer dans un magasin virtuel, on l'imagine choisir des briques virtuelles sur des étagères virtuelles, les placer sur une brouette virtuelle, et aller courir le long de chemins 3D multiples les déposer soigneusement sur les murs en construction d'un batiment virtuel... On l'imagine recommencer ces gestes autant de fois que nécessaire pour peaufiner son oeuvre, et on le devine un peu fatigué par toutes ces allées et venues chaotiques... au point qu'on en vient à lui souhaiter de penser à mettre un peu d'ordre dans tout celà en prenant par exemple un crayon et un petit carnet quadrillé bien réels pour dessiner à la main - oui, à la main! -, quelques schémas abstraits, puis des plans, des coupes, des élévations, des axonos, et des croquis perspectifs... en bonne application de quelques règles de la géométrie vieilles de vingt siècles. Il pourra alors embaucher des manoeuvres sur console vidéo qui iront chercher à sa place les briques virtuelles, les monter sur des murs virtuels, etc...

Alain Marty, le 28/11/1995

En annexe: 5 pages A4 de planches graphiques

ala...