

# Introduction à la Visualisation

Michel GRAVE

*michel.grave@univ-lr.fr*

Michel GRAVE



## Objet de la Visualisation

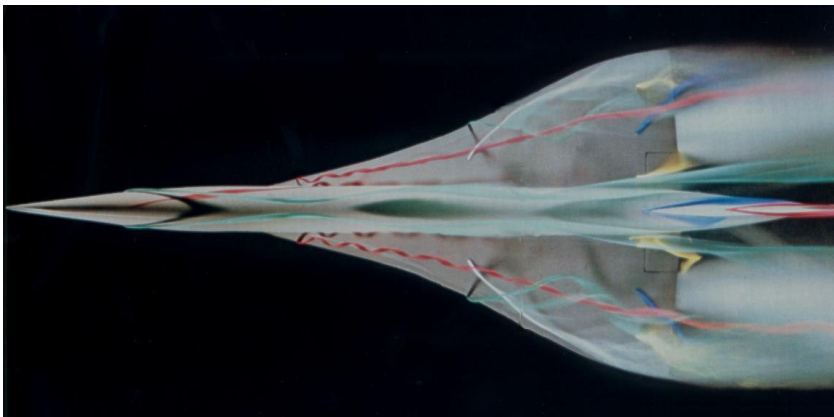
- Production d'images à partir de données.
  - calculées (simulations, ...)
  - d'expérimentation, de mesures, ...
- Interaction avec les données.
- Aide à la recherche d'information.
- Communication d'information

La visualisation sert à analyser et à comprendre les données, et non à produire des images !!

Michel GRAVE



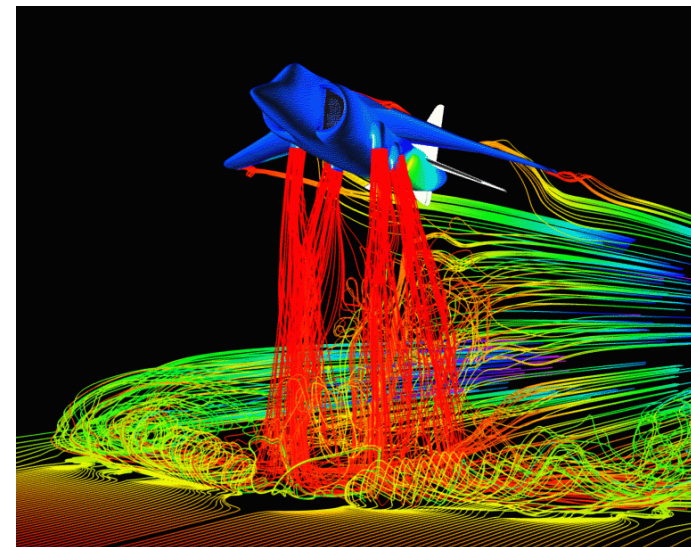
## Visualisation Expérimentale *(image ONERA)*



Michel GRAVE



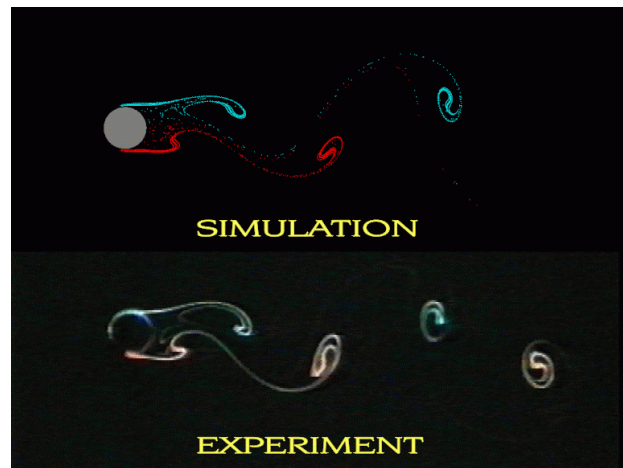
## Harrier *(Image NASA)*



Michel GRAVE



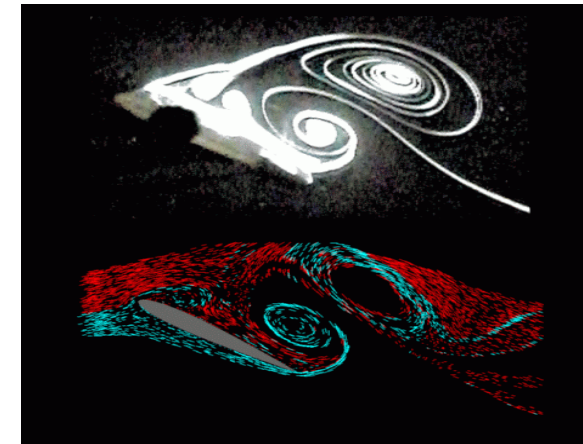
## Expérimental/Simulation (1) *(image ONERA)*



Michel GRAVE



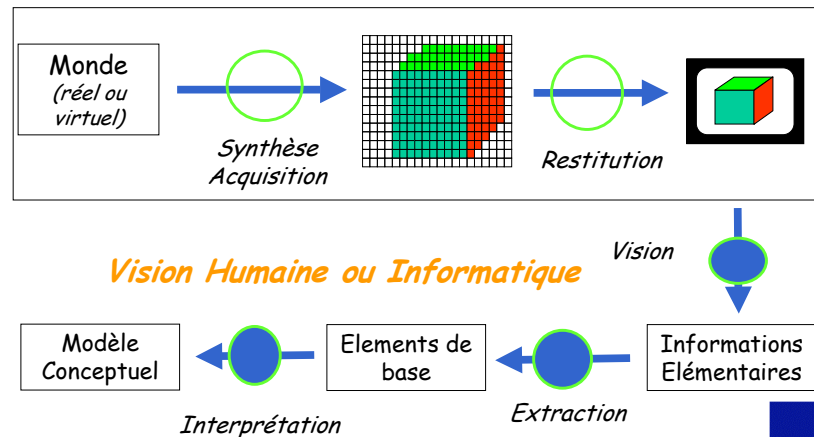
## Expérimental/Simulation (2) *(image ONERA)*



Michel GRAVE



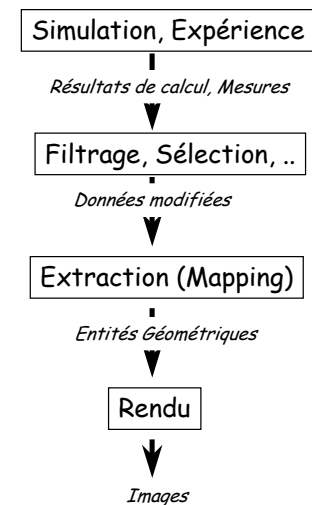
## Images & Modèles



Michel GRAVE



## La Chaîne de Visualisation Traditionnelle



### Extractions

- **Générales:** courbes, lignes ou surfaces isovaleurs, plans de coupe, ...
- **Spécifiques** de certains domaines d'application.
- **Pas neutres:** traitements numériques pouvant être "incompatibles" avec les simulations

Michel GRAVE



*(Ne s'applique pas au "rendu volumique" direct)*

## Hiérarchie des Logiciels

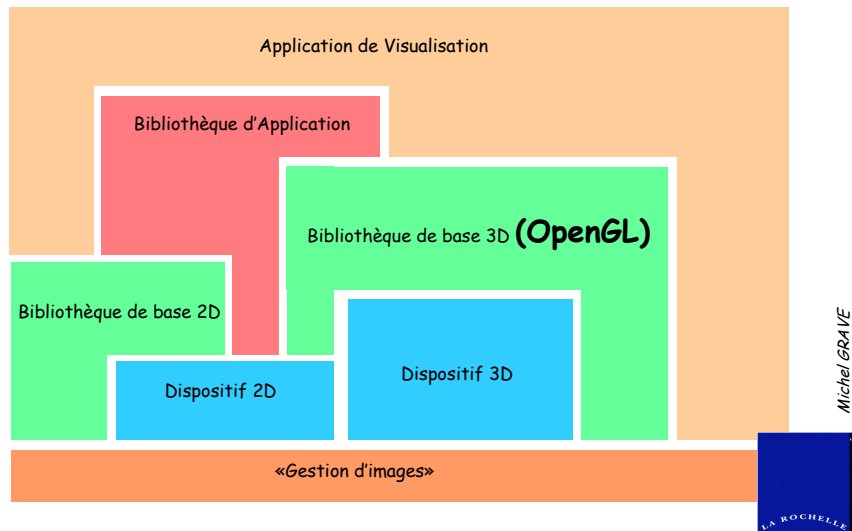
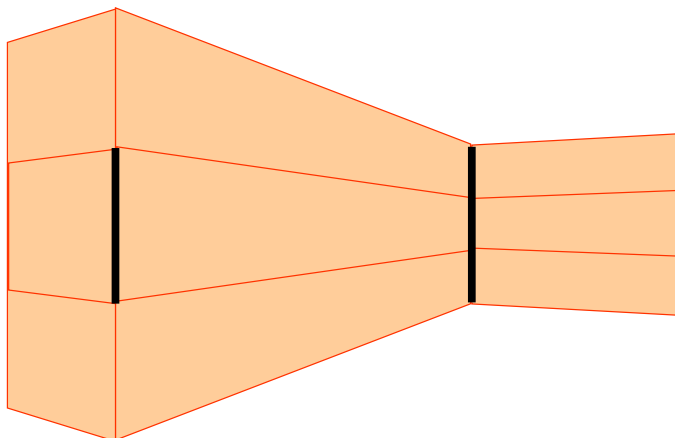


Image  $\neq$  « Réalité »  
(exemples)

*Michel GRAVE*



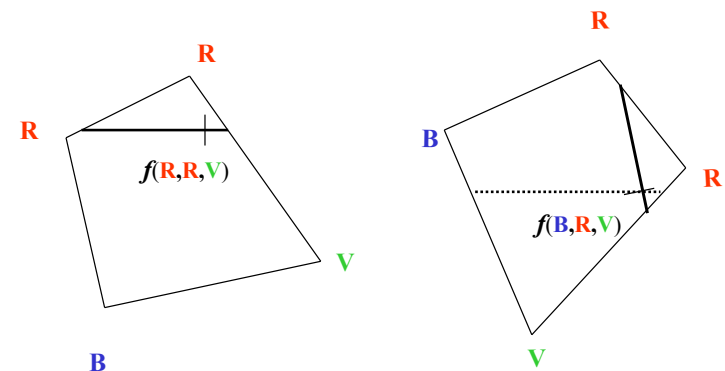
## Perception $\neq$ Réalité



*Michel GRAVE*



## Interpolation de Gouraud



*Michel GRAVE*



## Triangulation des polygones

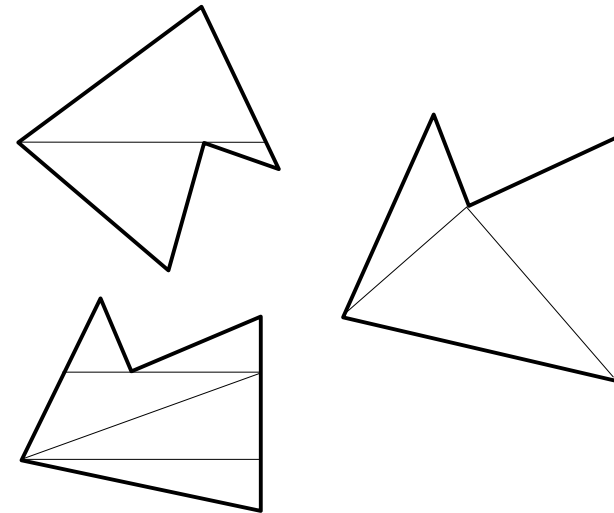
- Le tri et les calculs de «spans» sont des opérations assez complexes, surtout si l'on veut les mettre en œuvre en « hardware »
- Elles doivent être refaites chaque fois que le polygone est déplacé.

=> les processeurs graphiques ne traitent souvent que des triangles et laissent le soin aux logiciels de trianguler les polygones !

Michel GRAVE



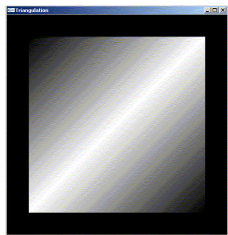
## Triangulations de Polygones



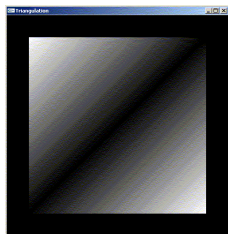
Michel GRAVE



## Triangulation & Interpolation



```
glBegin(GL_POLYGON);
glColor3f(1.0,1.0,1.0); glVertex2f(-0.8,-0.8);
glColor3f(0.0,0.0,0.0); glVertex2f(-0.8,0.8);
glColor3f(1.0,1.0,1.0); glVertex2f(0.8,0.8);
glColor3f(0.0,0.0,0.0); glVertex2f(0.8,-0.8);
glEnd();
```

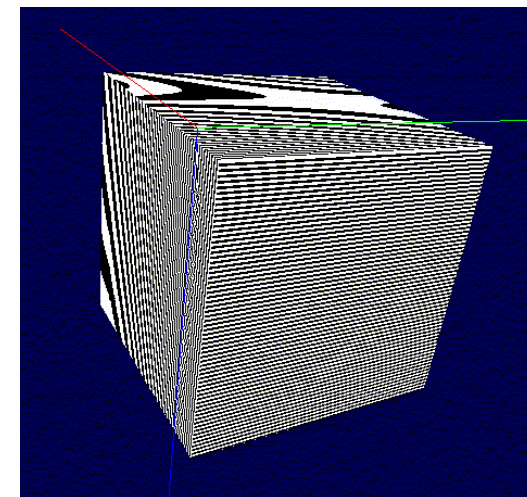


```
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,0.0); glVertex2f(-0.8,-0.8);
glColor3f(1.0,1.0,1.0); glVertex2f(-0.8,0.8);
glColor3f(0.0,0.0,0.0); glVertex2f(0.8,0.8);
glColor3f(1.0,1.0,1.0); glVertex2f(0.8,-0.8);
glEnd();
```

Michel GRAVE



## Cube texturé

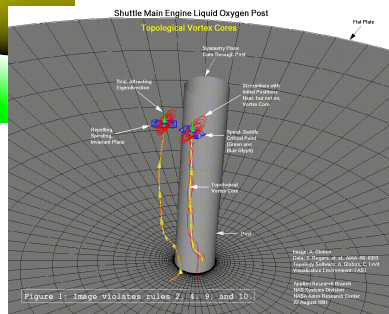


Michel GRAVE






LA ROCHELLE



LA ROCHELLE

(Al Globus & E. Raible)

- 



## 13 Erreurs (b)

(Al Globus & E. Raible)

- Ne pas connaître un minimum sur les données scientifiques manipulées.
- Ne jamais comparer les résultats obtenus avec diverses méthodes.
- Refuser les systèmes de visualisation évolutifs existants.
- Ne pas indiquer les sources de données dans les publications.
- Penser qu'on a tout testé avec un jeu de données.
- Cacher des erreurs avec des angles de vue bien choisis.
- Sous-estimer le coût de passage du 2D au 3D.

Michel GRAVE



## Etude de Synthèse 1 (Champs de vecteurs)

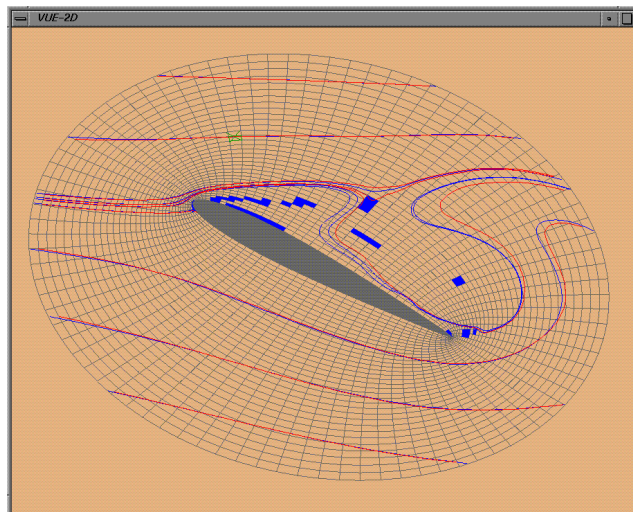
Michel GRAVE

*michel.grave@univ-lr.fr*

Michel GRAVE



### Visualisation de Champs de Vecteurs



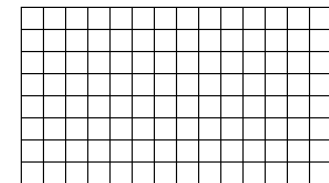
Michel GRAVE



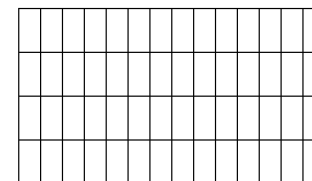
### Types de Maillages (1)



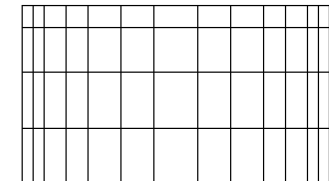
Données Dispersées



Maillage Structuré Cartésien



Maillage Structuré Régulier

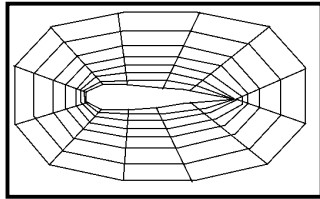


Maillage Structuré Rectilinéaire

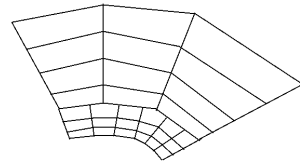
Michel GRAVE



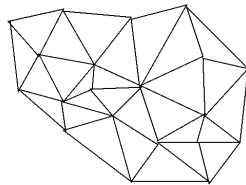
## Types de Maillages (2)



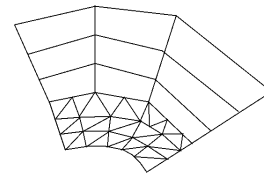
Maillage Structuré Curviligne



Maillage Structuré Multidomaine



Maillage Non Structuré



Maillage Hybride

Michel GRAVE



## Format de Données

```

3      3      90.0      nx ny Angle
-0.7 -0.7  0.0  0.0    x y u v (point 0 0)
-0.7  0.0  0.0  1.0    x y u v (point 0 1)
-0.7  0.7  0.0  0.0
0.0 -0.7 -1.0  0.0
0.0  0.0  0.0  0.0
0.0  0.7  1.0  0.0
0.7 -0.7  0.0  0.0
0.7  0.0  0.0 -1.0
0.7  0.7  0.0  0.0    x y u v (point 2 2)
    
```

Michel GRAVE



## Fichiers .h et .c

<b>voir.c</b>	Programme et sous-programmes principaux
<b>tp.c</b>	Sous-programmes à réaliser
<b>calcul.c</b>	Utilitaires de calcul
<b>maillage.h</b>	Maillage et variables associées
<b>calcul.h</b>	Variables communes aux calculs

Michel GRAVE



## maillage.h

```

#ifdef MAIN
#define EXTERN
#else
#define EXTERN extern
#endif

EXTERN int nx, ny;
EXTERN double Angle

EXTERN double x[100][100], y[100][100],
              u[100][100], v[100][100];
EXTERN double xmin,xmax,ymin,ymax;
EXTERN double umin,umax,vmin,vmax;
    
```

Michel GRAVE



## main (voir.c)

```
main(int argc, char *argv[ ]) {
    glutInit(&argc, argv);
    if (argc > 1)
        charger_objet(argv[1]);
    else
        charger_objet("aile");
    initialiser_graphique("VUE-2D");
    glutDisplayFunc(afficher_objet);
    glutReshapeFunc(v2d_reshape);
    glutKeyboardFunc(clavier);
    glutMouseFunc(pointage);
    glutMainLoop();
    return 0;
} /* main */
```

Michel GRAVE



## void afficher\_objet(void)

```
void afficher_objet(void) {
    /* initialisation OpenGL */
    afficher_profil(ggris);
    if (f_maillage)
        afficher_maillage(ggris);
    if (f_vitesses)
        afficher_vitesses(rouge);
    if (f_cellules)
        afficher_candidates(bleu, nc, ci, cj);
    afficher_pointage(vert, xpnt, ypnt, cci, ccj);
    afficher_ligne(bleu, cx1, cy1, n1);
    afficher_ligne(rouge, cx2, cy2, n2);
    afficher_ligne(bleu, cx3, cy3, n3);
    afficher_ligne(rouge, cx4, cy4, n4);
    glFlush();
} /* afficher_objet */
```

Michel GRAVE



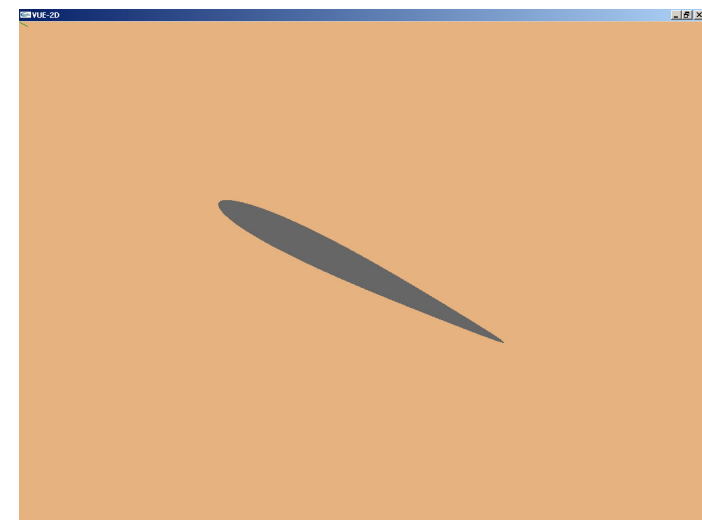
## Lecture/Affichage

```
void charger_objet(char *nom)
void afficher_profil(float couleur[3])
void afficher_maillage(float couleur[3])
void afficher_vitesses(float couleur[3])
void afficher_candidates(float couleur[3], int nc, int ci[], int cj[])
void afficher_pointage(float couleur[3], float xp, float yp, int ic, int jc)
void afficher_ligne(float couleur[3], double x[], double y[], int n)
```

Michel GRAVE



## Profil

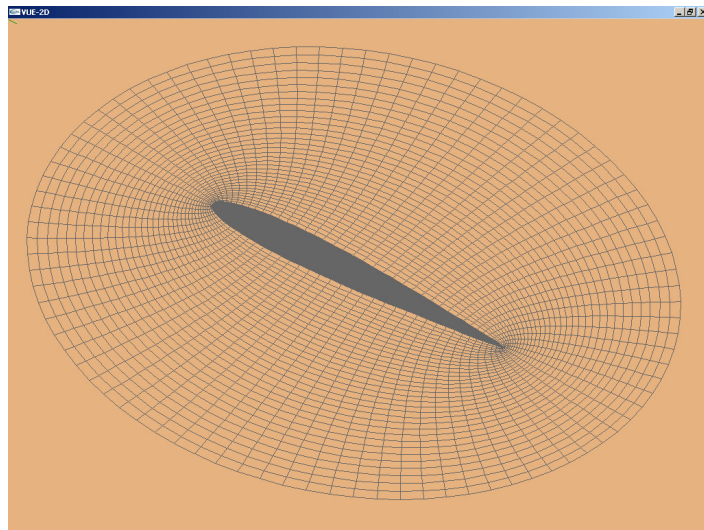


Michel GRAVE





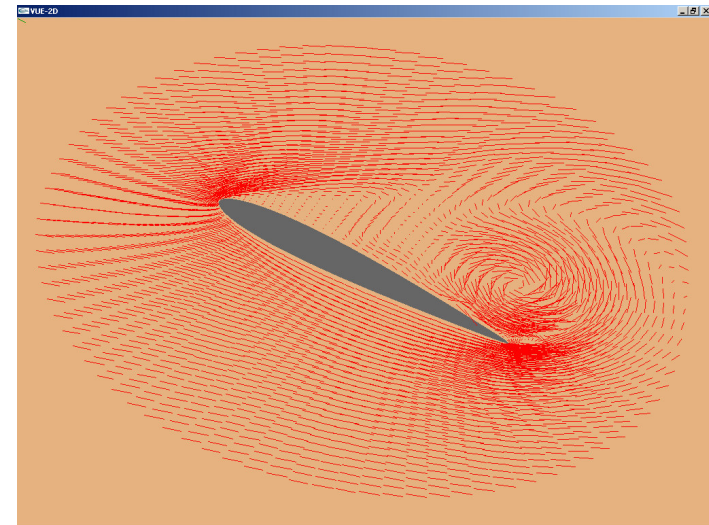
## Maillage



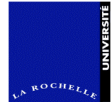
Michel GRAVE



## Vitesses



Michel GRAVE



## Calculs

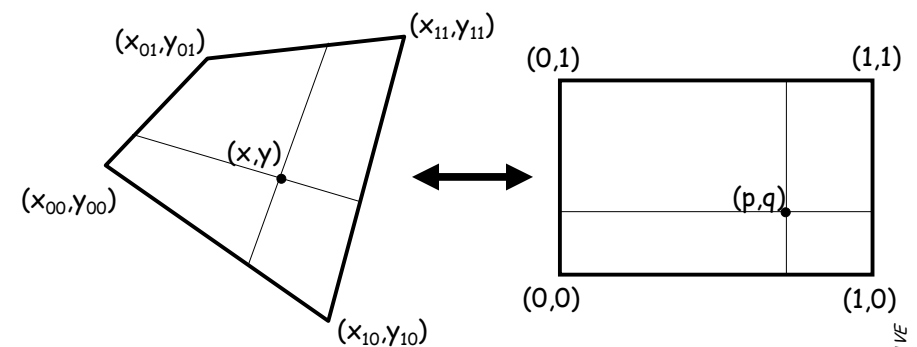
```
void cellules_candidates
(int *nc, int ci[], int cj[])
```

```
void calculer_ligne
(double step, double pe, double qe, int ci, int cj,
double sx[], double sy[], int *nns, int maxp)
```

Michel GRAVE



## Coordonnées réelles et paramétriques



$$x = (1-p)*((1-q)*x_{00} + q*x_{01}) + p*((1-q)*x_{10} + q*x_{11})$$

$$y = (1-p)*((1-q)*y_{00} + q*y_{01}) + p*((1-q)*y_{10} + q*y_{11})$$

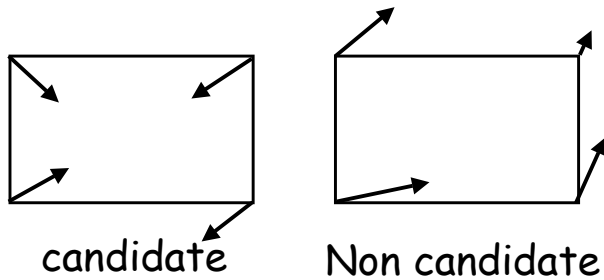
Michel GRAVE



## Cellules Candidates

Condition **nécessaire**, mais **non suffisante** :

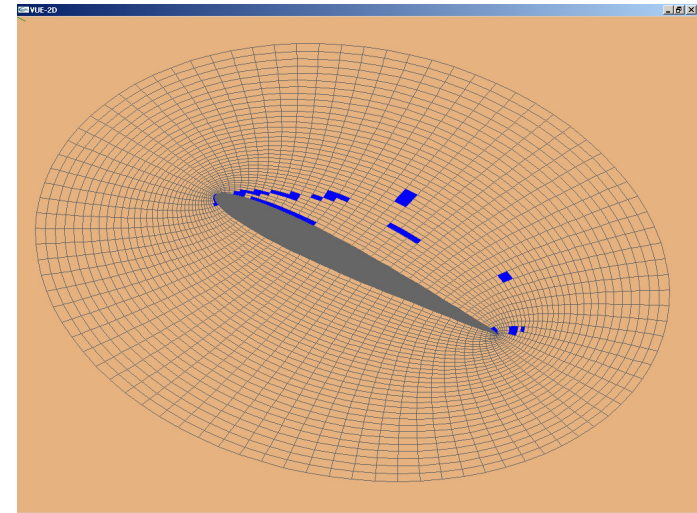
Il doit y avoir au moins un changement de signe, sur chaque composante du vecteur vitesse entre deux sommets pour qu'une cellule puisse contenir un point stationnaire. (*interpolation monotone supposée*)



Michel GRAVE



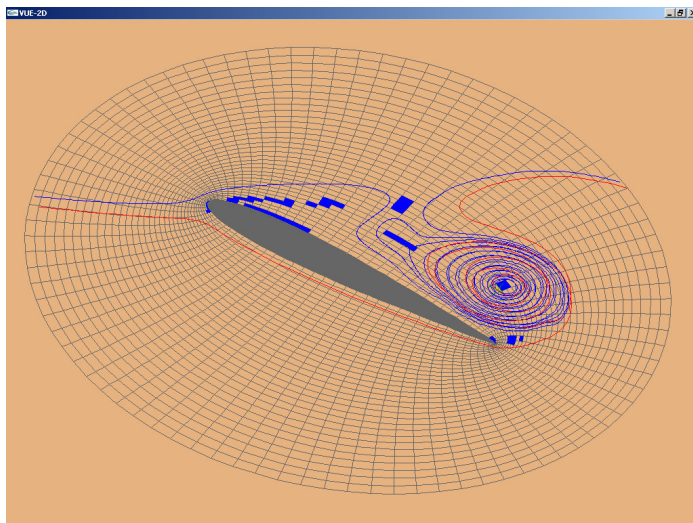
## Cellules Candidates



Michel GRAVE



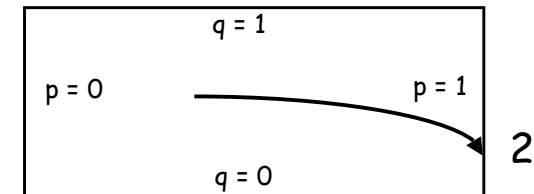
## Lignes de Courant



Michel GRAVE



## Tracer\_cellule



0 = Erreur

1 = sortie par le côté "p=0"

2 = sortie par le côté "p=1"

3 = sortie par le côté "q=0"

4 = sortie par le côté "q=1"

Michel GRAVE



## Calculer\_ligne

```
void calculer_ligne (float step, float pe, float qe, int ci, int cj,
    float sx[ ], float sy[ ], int *nns, int maxp)
{
    while ((*nns < maxp) && flag) {
        flag = tracer_cellule(icell,jcell,step,pe,qe,ps,qs,sx,sy,nns,maxp)
        switch(flag) {
            case 1 : se positionner sur la cellule suivante avec les pe et qe correspondants :
                break;
            case 2 : se positionner sur la cellule suivante avec les pe et qe correspondants :
                break;
            case 3 : se positionner sur la cellule suivante avec les pe et qe correspondants :
                break;
            case 4 : se positionner sur la cellule suivante avec les pe et qe correspondants :
                break;
            default : break;
        }; /* switch */
        Vérifier si les indices de cellule suivante sont valides.
        Si OUI alors flag = 1, si NON flag = 0;
    }; /* while */
} /* calculer ligne */
```

Michel GRAVE



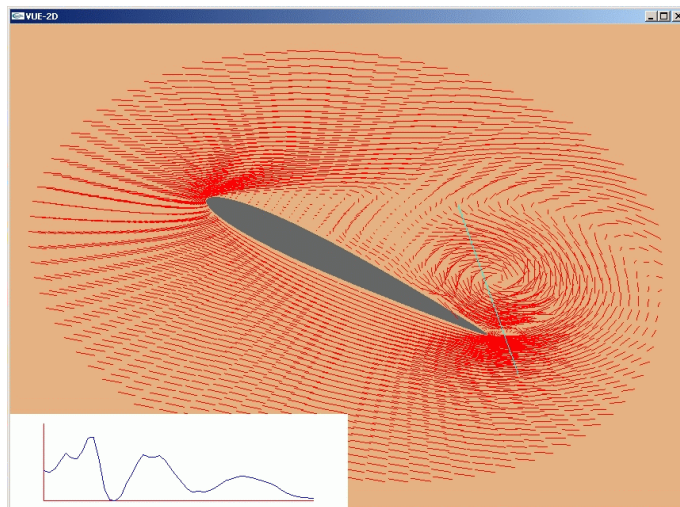
## pointage

```
void pointage(int button, int state, int mx, int my){
    GLdouble x,y,z;
    float c, s, p, q;
    if (state == GLUT_DOWN) {
        x = mx - xcentre; y = (vy2 - my) - ycentre;
        x = x / (vx2-vx1); y = y / (vy2-vy1); x = x * 2; y = y * 2;
        c = cosf((3.1415927 * Angle)/180.0); s = - sinf((3.1415927 * Angle)/180.0);
        xpnt = x*c - y*s; ypnt = x*s + y*c;
        localiser(xpnt,ypnt,&cci,&ccj);
        xy2pq(cci,ccj,xpnt,ypnt,&p,&q);
        n1 = n2 = n3 = n4 = 0;
        calculer_ligne( 0.005,p,q,cci,ccj,cx1,cy1,&n1,NMXPTS-1);
        calculer_ligne( 0.050,p,q,cci,ccj,cx2,cy2,&n2,NMXPTS-1);
        calculer_ligne(-0.005,p,q,cci,ccj,cx3,cy3,&n3,NMXPTS-1);
        calculer_ligne(-0.050,p,q,cci,ccj,cx4,cy4,&n4,NMXPTS-1);
        glutPostRedisplay();
    };
} /* pointage */
```

Michel GRAVE



## Coupe



Michel GRAVE

