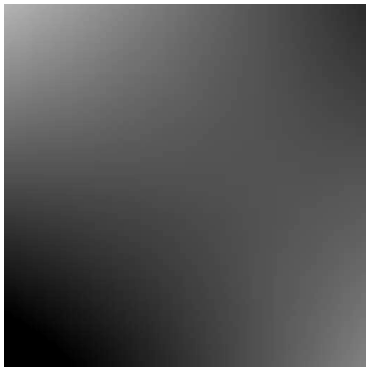
Textures

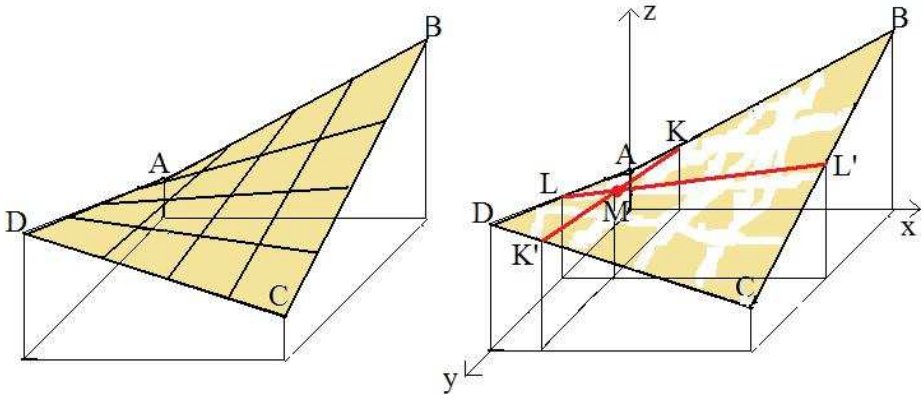
Ce chapitre donne quelques moyens de remplir des surfaces avec des motifs variés. On part d'une méthode de remplissage simple, par évolution régulière des couleurs (interpolation bilinéaire) puis à on procède à des transformations de cette méthode pour aboutir à des déformations plus complexes. Enfin on donne le moyen de plaquer ces surfaces ainsi texturées sur des formes en 3 dimensions, ici une sphère.

Dégradé de couleurs par interpolation bilinéaire



On commence par créer une palette de couleurs allant progressivement du noir au blanc, avec des nuances de gris intermédiaires.Chaque couleur possède son propre numéro, par exemple de 0 à 255. On peut préférer d'autres palettes, pourvu que l'évolution des couleurs soit régulière. Puis on prend un carré dont on se donne les couleurs aux quatre coins, prises au hasard dans la palette précédente. A partir de ces quatre points coloriés, on veut remplir l'intérieur du carré de façon à obtenir un dégradé harmonieux de couleurs, comme sur l'exempleci-contre

Pour cela, on procède à ce que l'on appelle une interpolation bilinéaire, c'est-à-dire à une évolution régulière dans les deux directions des côtés du carré.Pour comprendre, passons en trois dimensions, en plaçant le carré sur le sol, avec les numéros des couleurs devenant les altitudes des points. Au début on connaît seulement les altitudes des quatre coins. Il s'agit de construire une surface bordée par les segments [*AB*], [*BC*], [*CD*] et [*DA*]. Cette surface est gauche, n'ayant aucune raison d'être plane sauf cas exceptionnel. Comment avoir l'altitude *z* d'un point quelconque *M* de la surface ?



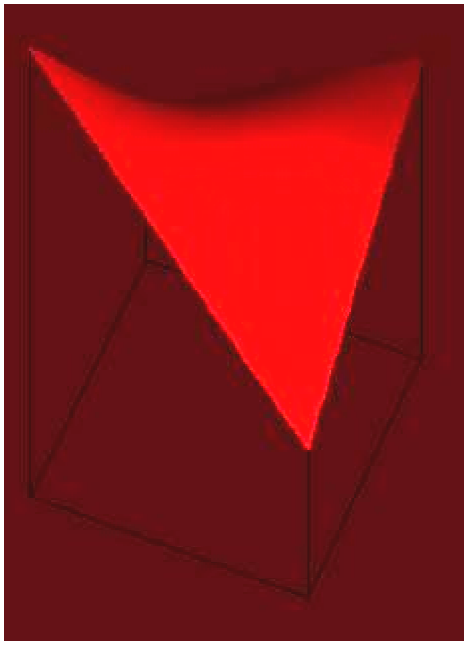
Dans le repère indiqué, les points *A*, *B*, *C*, *D* ont pour coordonnées (0, 0, *za*),(1, 0, *zb*),(1, 1, *zc*),(0, 1, *zd*) en supposant le carré initial de côté 1. Le plan vertical contenant *M* (*x*, *y*, *z*) et parallèle à *yOz* coupe *AB* et *CD* en *K* et *K*', avec $z_K = za + (zb - za)\, x$ et $z_{K'} = zd + (zc - zd)\, x$. On décide de prendre *M* sur [*KK*'], ce qui impose :

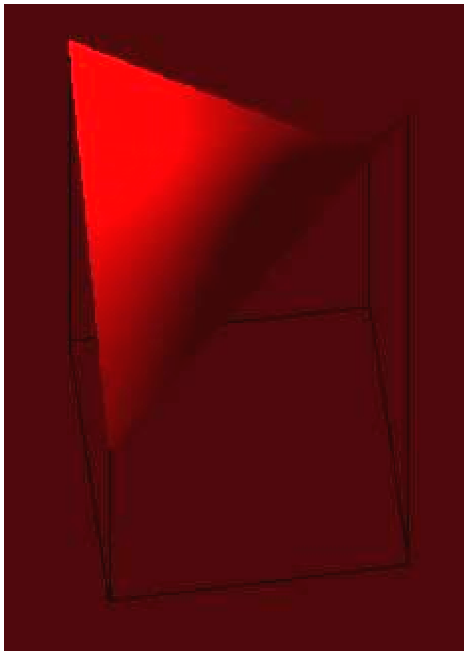
$z = z_K + (z_{K'} - z_K)\, y$. Tous calculs faits, cela donne :

$z = (1 - x)(1 - y)\, za + x(1 - y)\, zb + x\, y\, zc + y(1 - x)\, zd$

Cela s'appelle interpolation bilinéaaire, car on a pris une variation linéaire, en ligne droite, de *A* vers *B* (et de *C* vers *D*), ainsi qu'une variation linéaire entre *K* et *K*'. On aurait aussi bien pu tracer un plan vertical parallèle à *xOz*, ce qui donne les points *L* et *L*', et prendre *M* sur [*LL*'], en obtenant finalement le même résultat pour *z*.

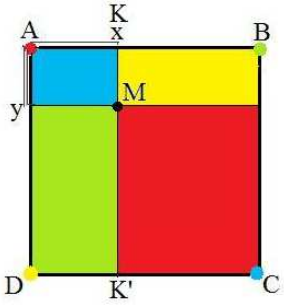
La surface gauche décrite par les points *M* est tapissée de droites dans deux directions, ce qui permet de tracer facilement un quadrillage sur la surface. En utilisant nos connaissances sur le tracé de surfaces et de leurs ombres en trois dimensions, on visualise aisément ce genre de surface.





En revenant en deux dimensions, on obtient le dégradé de couleurs recherché.

Lien avec les barycentres



Prenons le carré de côté *L*, où plus précisément $x_B = x_A + L$, avec un point *M* de coordonnées (*x*, *y*) à l'intérieur (*voir dessin*). Donnons aux sommets *A*, *B*, *C*, *D* du carré les poids respectifs $(L - x)(L - y)$, $x(L - y)$, xy, $y(L - x)$. Remarquons qu'il s'agit des aires des carrés construits autour du point *M* et qui leur sont opposés. Nous allons vérifier que le point *M* est le barycentre des quatre points pondérés *A*, *B*, *C*, *D*.

En effet, prenons le barycentre *K* des deux points pondérés $(A, (L - x)(L - y))$ et $(B, x(L - y))$. Il s'agit aussi du barycentre de $(A, L - x)$, (B, x), et le point *K* est tel que, vectoriellement :

AK = (x / L) **AB**. De même le barycentre *K*' des deux points pondérés $(D, y(L - x))$ et (C, xy) est tel que **DK'** = (x / L) **DC**. Le barycentre *G* des quatre points *A*, *B*, *C*, *D* est aussi le barycentre des deux points *K* et *K*' avec les poids $L - y$ et y. Il est tel que **KG** = (y / L) **KK'**. Le point *G* n'est autre que le point *M*.

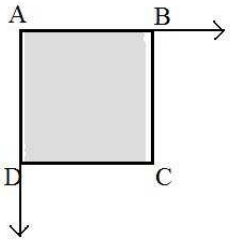
Finalement la couleur c_M donnée au point *M* (*x*, *y*) est la moyenne des couleurs des quatre coins du carrés pondérée par les poids de ces points lorsque *M* est leur barycentre, soit :

$$c_M = ((L - x)(L - y)\, c_A + x(L - y)\, c_B + xy\, c_C + y(L - x)\, c_D) / L^2$$

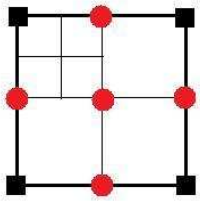
On en déduit le programme, où nous avons utilisé une palette de gris, les trois composantes RGB étant égales. Chaque nuance de gris est numérotée par un indice i allant de 0 (couleur noire) à 255 (couleur blanche). La formule d'interpolation précédente est appliquée sur les indices plutôt que sur les couleurs.

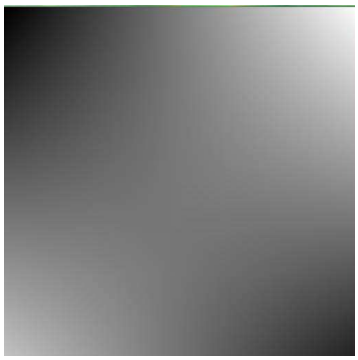
```
h0=rand()%256;    gris0=SDL_MapRGB(screen->format,h0,h0,h0);   putpixel(0,0,gris0); /* les 4 sommets */
h1=rand()%256;    gris1=SDL_MapRGB(screen->format,h1,h1,h1);   putpixel(L,0,gris1);
h2=rand()%256;    gris2=SDL_MapRGB(screen->format,h2,h2,h2);   putpixel(L,L,gris2);
h3=rand()%256;    gris3=SDL_MapRGB(screen->format,h3,h3,h3);   putpixel(0,L,gris3);
for(xe=0;xe<=L;xe++) for(ye=0;ye<=L;ye++)  /* parcours du carré et coloriage point par point */
  { icolor=((L-xe)*(L-ye)*h0+xe*(L-ye)*h1+xe*ye*h2+(L-xe)*ye*h3)/(L*L);
    color=SDL_MapRGB(screen->format,icolor,icolor,icolor);
    putpixel(xe,ye,color);
  }
```

Autre méthode par découpage du carré en carrés



Reprenons le carré, avec comme coordonnées de ses sommets (0, 0), (*L*, 0), (*L*, *L*), (0, *L*) dans le repère de l'écran. On choisit comme longueur *L* du côté une puissance de 2, par exemple 512 (sinon il faudra ajouter quelques aménagements). On définit aussi une palette de couleurs allant du noir au blanc en passant par les nuances de gris, ces couleurs étant indexées de 0 à 255. Comme auparavant, on commence par donner une couleur quelconque aux quatre sommets du carré. Mais maintenant on va remplir l'intérieur de ce carré en le découpant de façon récursive en quatre carrés de côté moitié (d'où l'intérêt de prendre une puissance de 2 comme longueur du côté). A chaque étape on connaît les couleurs des quatre coins du carré concerné et l'on calcule en fonction d'elles les couleurs du centre du carré, ainsi que des quatre milieux de chacun des côtés. Puis on recommence pour chacun des carrés obtenus. On prend comme couleur du centre du carré la moyenne des couleurs des quatre coins, en faisant la moyenne de leurs indices. Et on prend comme couleur du milieu d'un côté la moyenne des couleurs de ses deux extrémités. On retrouve ainsi l'interpolation bilinéaire, mais avec des calculs plus simples. Avec des carrés dont la longueur est divisée par deux à chaque étape, on s'arrête lorsque chaque point du carré est colorié.





Programme :

Le programme principal se contente essentiellement d'appeler la fonction d'interpolation sur les deux sommets opposés (0, 0) et (*L*, *L*) du carré initial.

```
h0=rand()%256;    gris0=SDL_MapRGB(screen->format,h0,h0,h0);putpixel(0,0,gris0);
h1=rand()%256;    gris1=SDL_MapRGB(screen->format,h1,h1,h1);putpixel(L,0,gris1);
h2=rand()%256;    gris2=SDL_MapRGB(screen->format,h2,h2,h2);putpixel(L,L,gris2);
h3=rand()%256;    gris3=SDL_MapRGB(screen->format,h3,h3,h3);putpixel(0,L,gris3);
interpo(0,0,L,L);
```

Reste à faire la fonction d'interpolation *interpo(x0,y0,x2,y2)* s'appliquant à un carré de sommets opposés (x_0, y_0) et (x_2, y_2), les deux autres sommets étant numérotés 1 et 3, avec 0123 dans l'ordre des aiguilles d'une montre. Comme chaque carré est divisé en quatre carrés, cette fonction se rappelle quatre fois sur les petits carrés. Connaissant les couleurs des quatre sommets du carré, il s'agit de récupérer les indices de ces couleurs pour pouvoir faire des moyennes sur ces indices. C'est l'occasion d'utiliser une nouvelle fonction de *SDL*, de la forme :

SDL_GetRGB(*couleur, ecran->format, &indicerouge, &indicevert,&indicebleu*).

A partir d'une *couleur* donnée, elle récupère les indices (entre 0 et 255) de ses trois couleurs *R*, *G*, *B*, que l'on peut ensuite utiliser pour connaître l'indice de la *couleur* (entre 0 et 255) dans la palette originelle. Dans le cas présent, avec les nuances de gris, les trois indices *R*, *G*, *B* sont égaux, ils sont récupérés dans *indicegris*, et déclarés comme entiers *Uint8*. Rappelons que dans cette palette de gris allant du noir au blanc, la couleur numéro *i* a comme valeurs *RGB* : *i*, *i*, *i*.

```
void interpo(int x0,int y0,int x2,int y2)
{
   if (x2-x0>1 )  /* l'arrêt se produit lorsque le carré a un côté unité */
    {
      cx=(x0+x2)/2;cy=(y0+y2)/2;  /* centre du carré  */
      couleur0=getpixel(x0,y0); couleur1=getpixel(x2,y0);
      couleur2=getpixel(x2,y2); couleur3=getpixel(x0,y2); /* couleurs des 4 sommets */
      SDL_GetRGB(couleur0,screen->format,&indicegris0,&indicegris0,&indicegris0);
      SDL_GetRGB(couleur1,screen->format,&indicegris1,&indicegris1,&indicegris1);
      SDL_GetRGB(couleur2,screen->format,&indicegris2,&indicegris2,&indicegris2);
      SDL_GetRGB(couleur3,screen->format,&indicegris3,&indicegris3,&indicegris3);
      ic=(indicegris0+indicegris1+indicegris2+indicegris3)/4; /* indice ic de la couleur du centre */
      couleurc=SDL_MapRGB(screen->format,ic,ic,ic);
      putpixel(cx,cy,couleurc); /* dessin du point central */

      i01= (indicegris0+indicegris1)/2; /* indice de la couleur du milieu du côté 01 */
      couleurc=SDL_MapRGB(screen->format,i01,i01,i01);   putpixel(cx,y0,couleurc);

      i12=(indicegris1+indicegris2)/2;
      couleurc=SDL_MapRGB(screen->format,i12,i12,i12);
      putpixel(x2,cy,couleurc);

      i23=(indicegris2+indicegris3)/2;
      couleurc=SDL_MapRGB(screen->format,i23,i23,i23);
      putpixel(cx,y2,couleurc);

      i30=(indicegris3+indicegris0)/2;
      couleurc=SDL_MapRGB(screen->format,i30,i30,i30);
      putpixel(x0,cy,couleurc);

      interpo(x0,y0,cx,cy); interpo(cx,y0,x2,cy);  /* rappel sur les quatre petits carrés */
      interpo(cx,cy,x2,y2); interpo(x0,cy,cx,y2);
    }
}
```

Nous allons maintenant appliquer et modifier ce schéma d'interpolation bilinéaire pout créer des textures plus complexes.

Plasma

Le plasma est le quatrième état de la matière, au-delà de l'état gazeux, et il est obtenu à de très fortes températures. Le programme qui suit va donner un dessin qui évoque artistiquement le plasma, avec des formes ondoyantes aux couleurs vives, et si l'on garde une palette de niveaux de gris, le résultat ressemble à des nuages.

Pour cela, on reprend la méthode récursive précédente, celle des milieux, avec des carrés découpés chaque fois en quatre carrés plus petits. On donne toujours au centre du carré la moyenne des quatre couleurs des coins. Mais pour les couleurs de milieux des côtés, on définit une variation *var* positive ou négative, dont la valeur absolue est prise aléatoirement entre 0 et la longueur du côté du carré concerné. Puis en multipliant cette valeur par un coefficient de proportionnalité *kk*, on ajoute le résultat obtenu à la valeur moyenne des couleurs des extrémités. Ainsi pour les points qui sont proches, dans de petits carrés, la variation des couleurs est faible, ce qui préserve un dégradé des couleurs à petite échelle, mais entre deux points éloignés la couleur de leur milieu subit éventuellement de grandes variations, ce qui donne un dessin final complexe, avec des couleurs variées, mais où le dégradé persiste.

Lors du phénomène récursif, un côté d'un carré est en général aussi le côté d'un autre carré, et son milieu va être colorié deux fois, ce qui provoque une rupture des couleurs entre les deux carrés, à cause des choix aléatoires différents. Pour éviter cela et préserver le dégradé, on décide que le milieu colorié une première fois garde ensuite cette couleur. Il convient aussi de prendre une décision lorsque la couleur dépasse les limites de la palette de couleurs, en la bloquant sur la limite. Le programme suivant utilise une fonction *carre*() prenant comme variables les coordonnées x_0, y_0 du sommet en haut à gauche, et celles x_2, y_2 du sommet en bas à droite.

Le programme s'ensuit.

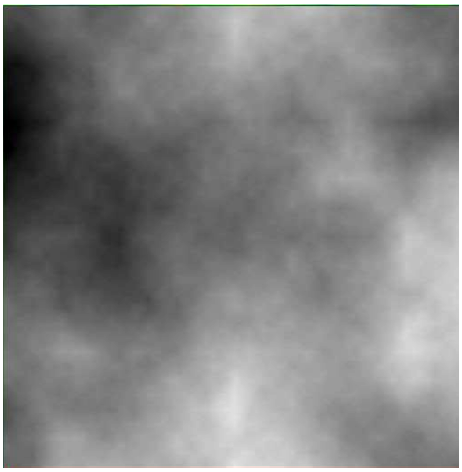
Programme principal :

```
couleurfondecran=SDL_MapRGB(ecran->format,255,255,254);
SDL_FillRect(ecran,0,rouge); srand(time(NULL));
 h=rand()%256;    gris=SDL_MapRGB(ecran->format,h,h,h);putpixel(0,0,gris);
 h=rand()%256;    gris=SDL_MapRGB(ecran->format,h,h,h);putpixel(L,0,gris);
 h=rand()%256;    gris=SDL_MapRGB(ecran->format,h,h,h);putpixel(L,L,gris);
 h=rand()%256;    gris=SDL_MapRGB(ecran->format,h,h,h);putpixel(0,L,gris);
 kk=0.2;
 carre(0,0,L,L);
```

Fonction carré():

```
void carre(int x0,int y0,int x2,int y2)
{  int cx,cy,moyenne,varmax;   float var;
   Uint32 couleur1,couleur2,couleur3,couleur4,couleurc;
   Uint8 indicegris1,indicegris2,indicegris3,indicegris4;

   if (x2-x0>1 )   /* test d'arrêt */
    {
      cx=(x0+x2)/2;cy=(y0+y2)/2;  /* centre du carré  */
      couleur1=getpixel(x0,y0); couleur2=getpixel(x2,y0);
      couleur3=getpixel(x2,y2); couleur4=getpixel(x0,y2);
      SDL_GetRGB(couleur1,ecran->format,&indicegris1,&indicegris1,&indicegris1);
      SDL_GetRGB(couleur2,ecran->format,&indicegris2,&indicegris2,&indicegris2);
      SDL_GetRGB(couleur3,ecran->format,&indicegris3,&indicegris3,&indicegris3);
      SDL_GetRGB(couleur4,ecran->format,&indicegris4,&indicegris4,&indicegris4);
      moyenne=((int)indicegris1+(int)indicegris2+(int)indicegris3+(int)indicegris4)/4;
      couleurc=SDL_MapRGB(ecran>format,moyenne,moyenne,moyenne);
```

```
        putpixel(cx,cy,couleurc);

        couleur1=getpixel(x0,y0);couleur2=getpixel(x2,y0);  /* milieu de AB  */
        SDL_GetRGB(couleur1,ecran->format,&indicegris1,&indicegris1,&indicegris1);
        SDL_GetRGB(couleur2,ecran->format,&indicegris2,&indicegris2,&indicegris2);
        moyenne=((int)indicegris1+(int)indicegris2)/2;
        varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
        moyenne+=var;
        if(moyenne<0) moyenne=0; if(moyenne>255) moyenne=255;
        couleurc=SDL_MapRGB(ecran>format,moyenne,moyenne,moyenne);
        if (getpixel(cx,y0)==couleurfondecran) putpixel(cx,y0,couleurc);

        couleur1=getpixel(x2,y0);couleur2=getpixel(x2,y2);  /* milieu de BC   */
        SDL_GetRGB(couleur1,ecran->format,&indicegris1,&indicegris1,&indicegris1);
        SDL_GetRGB(couleur2,ecran->format,&indicegris2,&indicegris2,&indicegris2);
        moyenne=((int)indicegris1+(int)indicegris2)/2;
        varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
        moyenne+=var;
        if(moyenne<0) moyenne=0; if(moyenne>255) moyenne=255;
        couleurc=SDL_MapRGB(ecran>format,moyenne,moyenne,moyenne);
        if (getpixel(x2,cy)==couleurfondecran) putpixel(x2,cy,couleurc);

        couleur1=getpixel(x0,y2);couleur2=getpixel(x2,y2);  /*  milieu de CD  */
        SDL_GetRGB(couleur1,ecran->format,&indicegris1,&indicegris1,&indicegris1);
        SDL_GetRGB(couleur2,ecran->format,&indicegris2,&indicegris2,&indicegris2);
        moyenne=((int)indicegris1+(int)indicegris2)/2;
        varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
        moyenne+=var;
        if(moyenne<0) moyenne=0; if(moyenne>255) moyenne=255;
        couleurc=SDL_MapRGB(ecran>format,moyenne,moyenne,moyenne);
        if (getpixel(cx,y2)==couleurfondecran) putpixel(cx,y2,couleurc);

        couleur1=getpixel(x0,y0);couleur2=getpixel(x0,y2);  /* milieu de DA  */
        SDL_GetRGB(couleur1,ecran->format,&indicegris1,&indicegris1,&indicegris1);
        SDL_GetRGB(couleur2,ecran->format,&indicegris2,&indicegris2,&indicegris2);
        moyenne=((int)indicegris1+(int)indicegris2)/2;
        varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
        moyenne+=var;
        if(moyenne<0) moyenne=0; if(moyenne>255) moyenne=255;
        couleurc=SDL_MapRGB(ecran>format,moyenne,moyenne,moyenne);
        if (getpixel(x0,cy)==couleurfondecran) putpixel(x0,cy,couleurc);

        carre(x0,y0,cx,cy); carre(cx,y0,x2,cy); carre(cx,cy,x2,y2); carre(x0,cy,cx,y2);
    }
}
```



Exercice 1 : D'autres applications : surface rouillée, flamme…

1) *Créer une palette de couleurs allant du noir au rouge vif, puis du rouge au jaune.*

Voici une façon de faire, avec les couleurs numérotées de 0 à 511 :

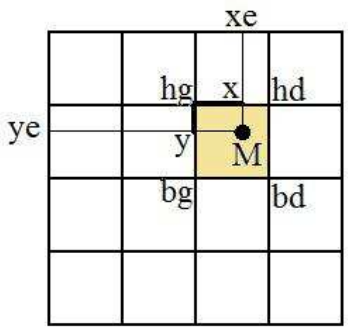
```
for(i=0;i<256;i++) c[i]=SDL_MapRGB(ecran->format,i,0,0);
for(i=256;i<512;i++)
c[i]=SDL_MapRGB(ecran->format,255,i-256,0);
```

2) *On considère un carré que l'on divise en N^2 petits carrés, chacun de côté L. La longueur du grand carré est donc N L. Puis on donne une couleur au hasard à tous les sommets de ces petits carrés. Ensuite on pratique une interpolation bilinéaire dans chacun des petits carrés, ce qui permet de calculer la couleur à donner à chaque point (xe, ye) situé dans le grand carré. Programmer.*

On commence par colorier chacun des sommets des petits carrés

On se donne N et L, par exemple N = 32 et L = 16

```
for(xe=0; xe<=N*L; xe+=L)   for(ye=0; ye<=N*L; ye+=L)
  { h=rand()%512;
    putpixel(xe,ye,c[h]);
  }
```

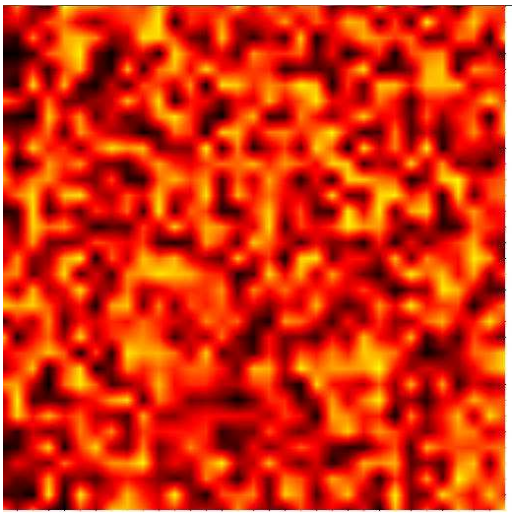


Ensuite pour chaque point *M* (*xe*, *ye*) du grand carré, on détermine ses coordonnées (*x*, *y*) dans le repère associé au petit carré dans lequel ce point *M* se trouve, ce qui nous sera utile pour pouvoir appliquer la formule d'interpolation bilinéaire. Puis on cherche les coordonnées de chaque sommet du carré où le point *M* se trouve, soit le point en bas à gauche (*bg*), le point en bas à droite (*bd*), le point en haut à gauche (*hg*) et le point en haut à droite (*hd*). Notamment le sommet en haut à gauche a pour abscisse (*xe* / *L*) *L*, avec (*xe* / *L*) qui est une division en entiers, et pour ordonnée (*ye* / *L*) *L*. On récupère par *getpixel*() les couleurs de ces quatre sommets, ce qui permet ensuite de connaître les indices de leurs couleurs *RGB*, grâce à *GetRGB*(). Il reste à retrouver le numéro de la couleur de ces quatre sommets, soit *icbg*, *icbd*, *ichg* et *ichd*. Cela permet enfin d'appliquer la formule d'interpolation bilinéaire, pour avoir le numéro *ic*[*xe*][*ye*] de la couleur du point (*xe*, *ye*), et de colorier ce point.

```
for(xe=0;xe<N*L;xe++)  for(ye=0;ye<N*L;ye++)
  { x=xe%L;y=ye%L;
    xhg=(xe/L)*L;yhg=(ye/L)*L; xhd=xhg+L;yhd=yhg;
    xbg=xhg; ybg=yhg+L;  xbd=xbg+L;ybd=ybg;
    chg=getpixel(xhg,yhg); chd=getpixel(xhd,yhd);
    cbg=getpixel(xbg,ybg); cbd=getpixel(xbd,ybd);
    SDL_GetRGB(chg,ecran->format,&r0,&g0,&b0);
    SDL_GetRGB(chd,ecran->format,&r1,&g1,&b1);
    SDL_GetRGB(cbg,ecran->format,&r2,&g2,&b2);
    SDL_GetRGB(cbd,ecran->format,&r3,&g3,&b3);
```

```
  if (g0==0) ichg=r0; else ichg=256+g0;  /* on retrouve le numéro de la couleur */
  if (g1==0) ichd=r1; else ichd=256+g1;
  if (g2==0) icbg=r2; else icbg=256+g2;
  if (g3==0) icbd=r3; else icbd=256+g3;
  ic[xe][ye]=((L-x)*(L-y)*ichg+x*(L-y)*ichd+x*y*icbd +(L-x)*y*icbg)/(L*L);
 }
 for(xe=0;xe<N*L;xe++)  for(ye=0;ye<N*L;ye++)  putpixel(xe,ye,c[ic[xe][ye]]);
```

On a un résultat de ce style :



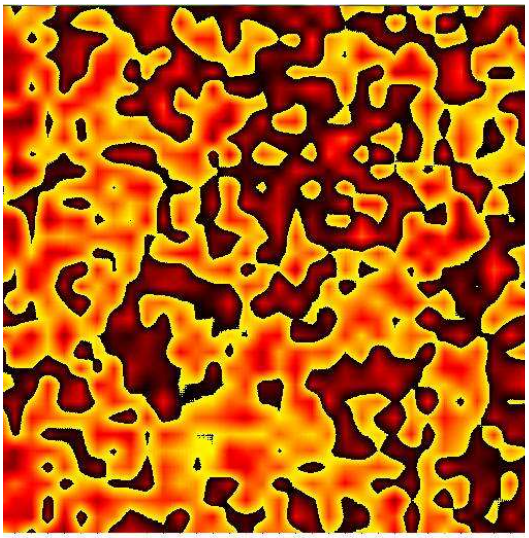
3) *On va brouiller le dessin précédent pour obtenir un aspect de surface écaillée et rouillée. D'abord on reprend les couleurs aléatoires aux sommets des petits carrés afin de pratiquer l'interpolation bilinéaire. On calcule en chaque point (xe, ye) du grand carré la couleur c[ic[xe][ye]]) comme on l'a fait précédemment, mais maintenant on lui ajoute la moitié de la couleur obtenue au point (xe/2, ye/2)puis le quart de la couleur du point (xe/4, ye/4), etc. Cela préserve la régularité du dégradé, pour des raisons de continuité, mais l'addition des couleurs provoque un dépassement de la limite à droite de la palette, ce qui fait que le jaune est suivi de noir puis de rouge. Cela provoque la formation de petites ou grandes taches qui évoquent de la rouille.Programmer.*

Il suffit d'ajouter au programme précédent le calcul de la nouvelle couleur, soit dans le programme principal :

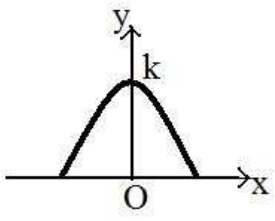
```
for(xe=0;xe<N*L;xe++)  for(ye=0;ye<N*L;ye++)  nic[xe][ye]=newicouleur(xe,ye);
for(xe=0;xe<N*L;xe++)  for(ye=0;ye<N*L;ye++) putpixel(xe,ye,c[nic[xe][ye]]);
```

avec la fonction *newicouleur*() qui ramène le numéro de cette couleur :

```
int newicouleur(int xe, int ye)
{ int iccolor; float pas,icolor=0;
  pas=1.;
  while(pas<33.)
    { icolor+=(float)ic[(int)((float)xe/pas)][(int)((float)ye/pas)]/pas;   pas=2*pas;}
  iccolor=(int)icolor; while (iccolor>=511) iccolor-=511;
  return iccolor;
}
```



4) *Flamme en mouvement : On commence par dessiner sur l'écran une forme parabolique remplie d'un dégradé de couleurs allant du jaune au rouge puis au noir. Puis on ajoute à cette couleur une petite proportion de la couleur correspondant à newicouleur() du phénomène de rouille précédent. La forme parabolique parfaite va se tordre et se brouiller en prenant l'allure d'une flamme. La succession de plusieurs images obtenues ainsi, avec l'aspect changeant de la rouille à cause des couleurs aléatoires données initialement, simule le mouvement d'une flamme. Programmer.*



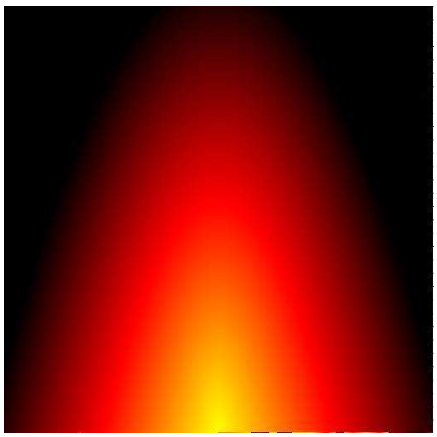
Une parabole dont le sommet est à la hauteur *k* sur l'axe des *y* (on est dans la zone des calculs en flottants avec son repère *Oxy*) a pour équation :

$y = k - (4 / k)\, x^2$.

Pour obtenir la forme parabolique qui remplit l'écran, il suffit de dessiner un grand nombre de paraboles avec *k* augmentant par petites quantités à partir de 0, d'où le programme :

```
for(kk=0.01;kk<=8.5;kk+=0.01)   for(xx=-kk/2.; xx<kk/2.; xx+=0.001)
{ yy= kk-4./kk*xx*xx;
  xe=xorig+zoom*xx; ye=yorig-zoom*yy;   /* passage en zone écran */
  if(ye<600 && ye>=0 && xe>=0 && xe<512)
  if (kk<5.5)   putpixel(xe,ye,c[511-(int)(kk*511./5.5)]);
  else putpixel(xe,ye,c[0]);
}
```

Il reste à calculer l'indice de la couleur obtenue en chaque point et de lui ajouter une petite partie de la couleur obtenue dans le programme de la rouille, cela étant répété dans une boucle, de façon à voir la flamme bouger.



forme parabolique

flamme

Exercice 2 : Placage d'un plasma sur une sphère qui tourne

1) *Refaire la palette de couleurs cyclique où chaque couleur fondamentale, le rouge, le vert, le bleu, prend la prédominance à tour de rôle. On utilise pour cela 768 numéros de couleurs.*

```
for (i=0;i<768;i++)
 if (i<256) co[i]=SDL_MapRGB(ecran->format,255-i,i,0);
 else if (i<512) co[i]=SDL_MapRGB(ecran->format,0,255-(i-256),i-256);
 else co[i]=SDL_MapRGB(ecran->format,i-512,0,255-(i-512));
```

2) *Utiliser cette palette pour faire un plasma en couleurs, à l'image de celui fait avec des nuances de gris. Ce plasma doit occuper une zone carrée de longueur L+1 en haut à gauche de l'écran, avec des abscisses et des ordonnées de points comprises entre 0 et L (on pourra prendre L = 256). Ce plasma devra avoir les contraintes suivantes sur ces bordures :*

- *tous les points de la bordure horizontale haute ont la même couleur*
- *tous ceux de la bordure horizontale basse ont la même couleur*
- *Connaissant les couleurs aux extrémités de la bordure verticale gauche, grâce à ce qui précède, utiliser la méthode des milieux avec son aspect aléatoire pour colorier tous les points de la bordure verticale gauche.*
- *Donner aux points de la bordure verticale droite les mêmes couleurs que ceux de la bordure verticale droite.*

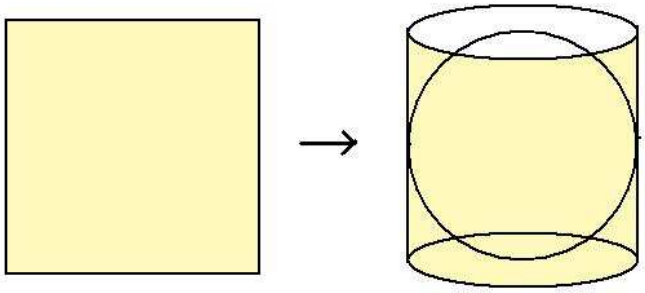
Une fois cela fait, reprendre le programme du plasma, par découpage du carré en carrés de longueur moitié, en ne coloriant que les points pas encore coloriés.

3) *Reprendre le programme permettant de construire une sphère de rayon R= 1, avec un zoom pour la dessiner sur l'écran, à côté du plasma déjà dessiné. Rappelons que l'on trace une série de points sur la sphère indexés par un nombre k, et qu'à partir de la valeur de k, on peut retrouver la longitude φ et la latitude λ du point. Les points ainsi construits sont les sommets des facettes qui recouvrent la sphère, ces facettes étant essentiellement en forme de quadrilatère, seules celles situées près des pôles étant des triangles.*

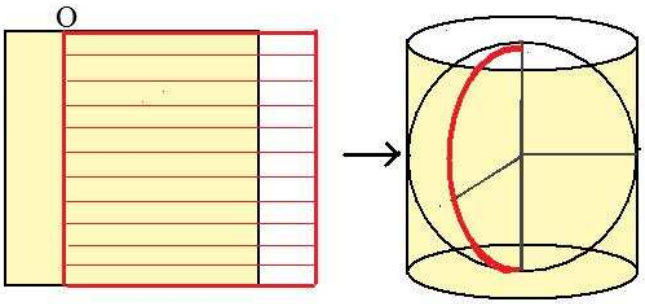
4) *Et maintenant, la nouveauté : on va plaquer le plasma sur la sphère. D'abord le carré du plasma est transformé en cylindre. Le fait d'avoir pris des bordures verticales identiques aux extrémités du carré permet d'éviter toute rupture de couleurs lorsque l'on passe au cylindre. Puis, en pratiquant un changement d'échelle, on fait en sorte que ce cylindre se colle autour de la sphère. Ainsi la longueur L+1 du carré du plasma correspondl à la circonférence de la sphère, soit 2π pour R*

= 1 (on est ici en zone calcul sur la sphère), et la hauteur L+1 du plasma correspond au diamètre de la sphère, soit 2. Le passage de la sphère au plasma s'obtient par deux zooms :

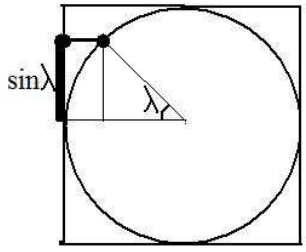
```
zoomp=(L+1)/deuxpi;   zoomz=(L+1)/2 ;
```



Faisons maintenant un décalage cyclique du plasma, la bordure verticale en O étant associée à la longitude φ = 0.



Comment passer d'un point (φ, λ) sur la sphère à un point (xeplasma, yeplasma) du carré cylindrique du plasma ? Il suffit d'effectuer une projection horizontale du point du plasma sur le point de la sphère, comme indiqué sur le dessin :



La formule de passage est donc :

```
xeplasma=xeO+zoomp*phi;
 if (xeplasma>L) xeplasma-=L+1
yeplasma=L/2.-zoomz*sin(lambda);
```

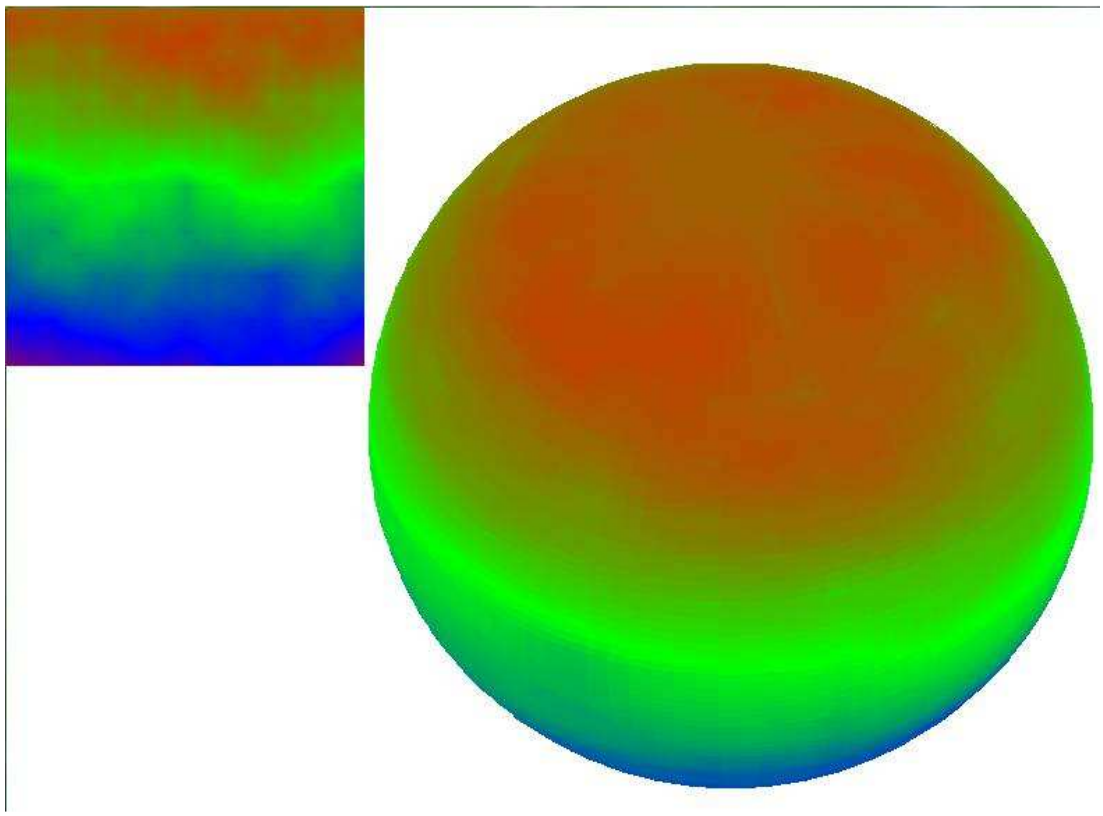
Il reste à colorier la facette de la sphère ayant comme sommet le point k avec la couleur correspondante du plasma. Enfin, en faisant avancer au coup par coup le point O, et en envoyant le plasma sur la sphère à chaque fois, on aura l'impression que la sphère tourne. Programmer.

Dans le programme principal, on ajoute ce mouvement du point *O* :

```
xeO=0;
for (j=0;j<5*L;j++)
  {  remplissagefacettes();  SDL_Flip(ecran);  xeO+=2;  }
```

Et la fonction de remplissage des facettes est la suivante :

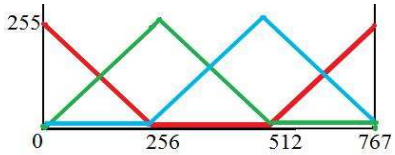
```
void remplissagefacettes(void)
{
   int k,xeplasma,yeplasma,xg,yg; float zoomp,zoomz;  Uint32 color;
   zoomp=(L+1)/deuxpi;   zoomz=(L+1)/2.;
   for (k=0;k<NNN;k++)
      if ( (k+1) % (NN+1) !=0 )    /* si l'on n'est pas à proximité des pôles */
      {   phi=k/(NN+1)*dp; lambda=(k%(NN+1))*dl-pi/2.+ll;
         xeplasma=xeO+zoomp*phi;
         if (xeplasma>L) xeplasma-=L+1;  if (xeO>L) xeO-=L+1;
         yeplasma=L/2.-zoomz*sin(lambda);
         if (dist[k]<0.)        /* facettes visibles */
         {   color=getpixel(xeplasma,yeplasma);
             xq[0]=xe[k];yq[0]=ye[k];
            xq[1]=xe[(k+NN+1)%NNN];yq[1]=ye[(k+NN+1)%NNN];
            xq[2]=xe[(k+NN+2)%NNN]; yq[2]=ye[(k+NN+2)%NNN];
            xq[3]=xe[k+1];yq[3]=ye[k+1];
            remplirquadri(color);    /* fonction déjà faite */
         }
      }
      else if (dist[k]<0)
      {
         phi=k/(NN+1)*dp;   lambda=(k%(NN+1))*dl-pi/2.+ll;
         xeplasma=xeO+zoomp*phi;
         if (xeplasma>L) xeplasma-=L+1;   if (xeO>L) xeO-=L+1;
         yeplasma=L/2.-zoomz*sin(lambda);   color=getpixel(xeplasma,yeplasma);
         xq[0]=xe[k];yq[0]=ye[k];
         xq[1]=xe[(k+NN+1)%NNN];yq[1]=ye[(k+NN+1)%NNN];
         xq[2]=xorig; yq[2]=yorig-C;
         remplirtriangle(color);
      }
}
```



Et elle tourne !

Exercice 3 : Plasma en couleurs

1) Reprendre la palette cyclique de 768 couleurs où le rouge, le vert, puis le bleu dominent à tour de rôle, selon le schéma suivant :




```
for(i=0;i<768;i++)
  { if (i<256) couleur[i]=SDL_MapRGB(screen->format,255-i,i,0);
    else if (i<512) couleur[i]=SDL_MapRGB(screen->format,0,255-(i-256),i-256);
    else if (i<768) couleur[i]=SDL_MapRGB(screen->format,i-512,0,255-(i-512));
  }
for(i=0;i<768;i++) line(i,0,i,50,couleur[i]); /* dessin de la palette */
```

2) Lorsque l'on connaît les composantes RGB d'une couleur de la palette précédente, soit iR, iG et iB (déclarés en Uint8), comment retrouver l'indice de cette couleur ?

Récupérons la couleur *color* d'un point (*x*, *y*), puis ses composantes *iR*, *IG*, *iB* grâce à la fonction *GetRGB*(). Pour retrouver l'indice *i* de cette couleur, on distingue trois cas, selon que c'est *iR* ou *iG* ou *iB* qui est à 0.

```
color=getpixel(x,y);
SDL_GetRGB(color,screen->format,&iR,&iG,&iB);
if (iB==0) i=iG;
else if (iR==0) i=iB+256;
else if (iG==0) i=iR+512;
```

3) Se donner les couleurs des quatre sommets 0123 du carré initial et pratiquer l'interpolation bilinéaire sans avoir besoin d'utiliser la fonction GetRGB(). Pour cela on programmera la fonction interpo(x0,y0,x2,y2,i0,i1,i2,i3) où les variables sont les coordonnées (x_0,y_0) et (x_2,y_2) des deux sommets opposés 0 et 2, ainsi que les indices i_0,i_1,i_2,i_3 des quatre couleurs des sommets 0123.

```
void interpo(int x0,int y0,int x2,int y2,int i0, int i1, int i2, int i3)
{  int cx,cy,ic,i01,i12,i23,i30;
   if (x2-x0>1 )
    {
      cx=(x0+x2)/2;cy=(y0+y2)/2;  /* centre du carré  */
      ic=(i0+i1+i2+i3)/4;  putpixel(cx,cy,couleur[ic]);
      i01=(i0+i1)/2;  putpixel(cx,y0,couleur[i01]); /* milieu du côté 01 */
      i12=(i1+i2)/2;   putpixel(x2,cy,couleur[i12]); /* milieu du côté 12 */
      i23=(i2+i3)/2;   putpixel(cx,y2,couleur[i23]); /* milieu du côté 23 */
      i30=(i3+i0)/2;   putpixel(x0,cy,couleur[i30]); /* milieu du côté 30 */
      interpo(x0,y0,cx,cy,i0,i01,ic,i30);
      interpo(cx,y0,x2,cy,i01,i1,i12,ic);
      interpo(cx,cy,x2,y2,ic,i12,i2,i23);
      interpo(x0,cy,cx,y2,i30,ic,i23,i3);
    }
}
```

Ce programme est nettement plus simple que celui de la fonction *interpo*() vue au début de ce chapitre (*pages 3-4*). Le programme principal reste inchangé et se réduit à :

```
h0=rand()%768;   putpixel(0,0,couleur[h0]);      h1=rand()%768;   putpixel(L,0,couleur[h1]);
h2=rand()%768;    putpixel(L,L,couleur[h2]);    h3=rand()%768;    putpixel(0,L,couleur[h3]);
interpo(0,0,L,L,h0,h1,h2,h3);
```

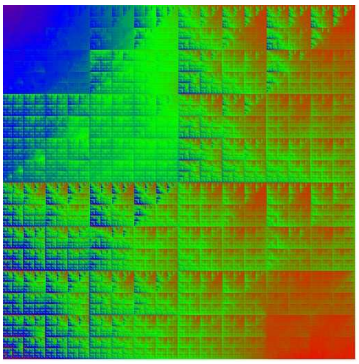
4) On veut aménager la fonction d'interpolation pour avoir le plasma en couleurs, en rajoutant du hasard comme on l'a fait précédemment avec la fonction carre() en niveaux de gris. Voici ce qui est proposé :

```
void carre(int x0,int y0,int x2,int y2,int i0, int i1, int i2, int i3)
{  int cx,cy, ic,i01,i12,i23,i30,varmax;   float var; Uint32 couleurc;  Uint8 iR,iG,iB;
   if (x2-x0>1 )
    {
      cx=(x0+x2)/2;cy=(y0+y2)/2;  /********** centre du carré ********/
      ic=(i0+i1+i2+i3)/4;  putpixel(cx,cy,couleur[ic]);
      i01=(i0+i1)/2;  /********** 01 **********/
      varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
      i01+=var;
      if(i01<0) i01=0; if(i01>767) i01=767;
      couleurc=couleur[i01];
      if (getpixel(cx,y0)==couleurfond) putpixel(cx,y0,couleurc);
      i12=(i1+i2)/2;    /********** i12 **********/
      varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
      i12+=var;
      if(i12<0) i12=0; if(i12>767) i12=767;
      couleurc=couleur[i12];
      if (getpixel(x2,cy)==couleurfond)putpixel(x2,cy,couleurc);
      i23=(i2+i3)/2;   /********** i23 **********/
      varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
      i23+=var;
      if(i23<0) i23=0; if(i23>767) i23=767;
      couleurc=couleur[i23];
      if (getpixel(cx,y2)==couleurfond) putpixel(cx,y2,couleurc);
     i30=(i3+i0)/2;  /********* i30 **********/
      varmax=x2-x0; var=2*(rand()%(varmax+1))-varmax; var=kk*var;
      i30+=var;
      if(i30<0) i30=0; if(i30>767) i30=767;
      couleurc=couleur[i30];
      if (getpixel(x0,cy)==couleurfond)putpixel(x0,cy,couleurc);

      carre(x0,y0,cx,cy,i0,i01,ic,i30);
      carre(cx,y0,x2,cy,i01,i1,i12,ic);
      carre(cx,cy,x2,y2,ic,i12,i2,i23);
      carre(x0,cy,cx,y2,i30,ic,i23,i3);
    }
}
```

Le programme principal se réduit à :

```
SDL_FillRect(screen,0,couleurfond); /* on peut prendre la couleur blanche comme couleur de fond */
putpixel(0,0,couleur[h0]);  putpixel(L,0,couleur[h1]);
putpixel(L,L,couleur[h2]);  putpixel(0,L,couleur[h3]);
kk=0.4;
carre(0,0,L,L,h0,h1,h2,h3);
```

Voici ce que l'on obtient :



Pourquoi ce programme ne marche pas ?

Prenons l'exemple du milieu des sommets 01, le même problème se posant pour les autres milieux. On sait que ce point peut être colorié plusieurs fois lors de la récursivité, mais on choisit de lui donner la première couleur obtenue d'indice *i01*, d'où le test :

```
couleurc=couleur[i01];
if (getpixel(cx,y0)==couleurfond) putpixel(cx,y0,couleurc);
```

Mais l'indice *i01* peut prendre plusieurs valeurs, et lorsqu'il est transmis aux rappels récursifs de la fonction *carre*() ce n'est pas forcément celui qui est le bon –c'est-à-dire le premier. On verra donc apparaître dans le dessin sur l'écran des discontinuités.

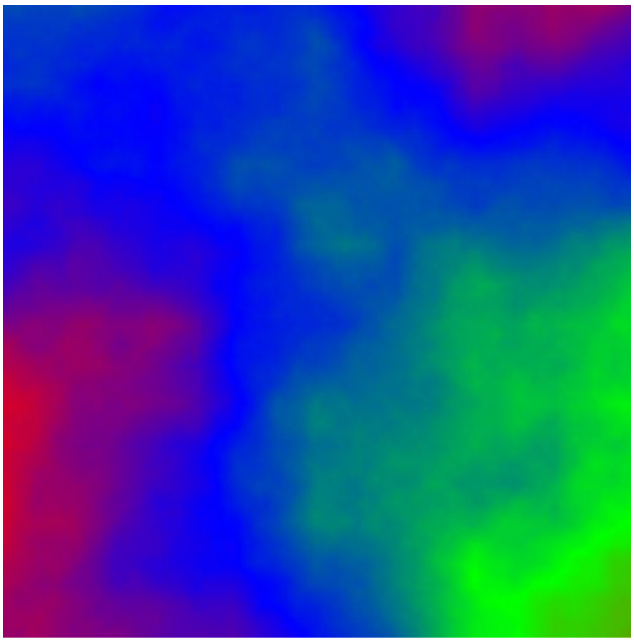
5) Rectifier le programme précédent pour obtenir le plasma.

Pour conserver la première valeur obtenue, voici ce qu'il convient de rajouter pour chaque milieu, comme ici pour le milieu de 01 :

```
  couleurc=couleur[i01];
  if (getpixel(cx,y0)==couleurfond) putpixel(cx,y0,couleurc);
else     /* sinon on récupère la valeur i01 que l'on avait déjà  (cf. question 2) */
  { couleurc=getpixel(cx,y0);
    SDL_GetRGB(couleurc,screen->format,&iR,&iG,&iB);
    if (iB==0) i01=(int)iG;
    else if (iR==0) i01=(int)iB+256;
    else if (iG==0) i01=(int)iR+512;
  }
```

6) A partir du dessin obtenu, faire une animation en augmentant de quelques unités, à chaque étape de temps, l'indice de couleur de chaque point du dessin. On profite ici du caractère cyclique de la palette.

```
for(etape=1;etape<200;etape++)
 { h=rand()%10;
   for(i=0;i<=L;i++) for(j=0;j<=L;j++) /* parcours de l'image */
    { couleurr=getpixel(i,j);
      SDL_GetRGB(couleurr,screen->format,&iR,&iG,&iB);
      if (iB==0) ii=iG;
      else if (iR==0) ii=(int)iB+256;
      else if (iG==0) ii=(int)iR+512;
      putpixel(i,j,couleur[(ii+2+h)%768]);  /* augmentation de l'indice */
    }
  SDL_Flip(screen);SDL_Delay(20);
 }
```



6) Recommencer tout cela en utilisant une palette plus large, la palette arc-en-ciel avec 1536 couleurs, ainsi définie :