TD/TP de Programmation – Correction de la Planche 4
Affichage de fractales par fonctions récursives

1 Rotation et interpolation linéaire

```
double
Double_Lerp (double x0, double alpha, double x1)
{
  return x0 + alpha * (x1-x0);
}

Coord
Coord_Lerp (Coord p, double alpha, Coord q)
{
  double x= Double_Lerp (p.x, alpha, q.x);
  double y= Double_Lerp (p.y, alpha, q.y);
  return Coord_FromXY (x, y);
}
```

```
Coord
Coord_FromAngle (double theta)
{
  return Coord_FromXY (cos (theta),
                       sin (theta));
}
```

```
Coord
Coord_FromPolar (double rho, double theta)
{
  return Coord_ScaledBy (Coord_FromAngle (theta), rho);
}
```

```
Coord
Coord_ComplexProduct (Coord p, Coord q)
{
  double x= p.x * q.x - p.y * q.y;
  double y= p.x * q.y + p.y * q.x;
  return Coord_FromXY (x, y);
}
```

```
Coord
Coord_RotatedBy (Coord p, double angle)
{
  Coord complex= Coord_FromAngle (angle);
  return Coord_ComplexProduct (p, complex);
}
```

```
Coord
Coord_RotatedByAround (Coord p, double angle, Coord center)
{
  Coord relative= Coord_Difference (p, center);
  Coord rotated= Coord_RotatedBy (relative, angle);
  return Coord_Sum (center, rotated);
}
```

```
int
MainTest_LerpAndRotationDemo (void)
{
  bx_init ();
  char win_title []= "demo";
  Coord dim= Coord_FromXY (200, 100);
  bx_window win= bx_create_window (win_title, 0, 0, dim.x, dim.y);
  for (;;)  {
    Coord center= Coord_ScaledBy (dim, 0.5);
    bx_mouse mouse= bx_read_mouse (win);
    Coord cursor= Util_MouseCoord (mouse);
    Coord lerp= Coord_Lerp (center, 2.0/3.0, cursor);
    Coord rot = Coord_RotatedByAround (cursor, MATH_PI/3.0, center);
    bx_clear_canvas (win, bx_white());
    bx_set_color (bx_black());
    Util_DrawLine   (win, center, cursor);
    Util_DrawLine   (win, center, rot   );
    int radius= 5;
    bool filled= true;
    Util_DrawCircle (win, center, radius, filled);
    Util_DrawCircle (win, cursor, radius, filled);
    Util_DrawCircle (win, lerp  , radius, filled);
    Util_DrawCircle (win, rot   , radius, filled);
    int milliseconds_delay= 10;
    bx_show_canvas (win, milliseconds_delay);
  }
  bx_loop ();
  return 0;
}
```

2 Flocon de Koch

```
void
Util_DrawKochStep (bx_window window, Coord p, Coord q)
{
  Coord a= Coord_Lerp (p, 1/3.0, q);
  Coord b= Coord_Lerp (p, 2/3.0, q);
  Coord c= Coord_RotatedByAround (p, MATH_PI * 2/3.0, a);
  Util_DrawLine (window, p, a);
  Util_DrawLine (window, a, c);
  Util_DrawLine (window, c, b);
  Util_DrawLine (window, b, q);
}
```

```
void
Util_DrawKochLine (bx_window window, Coord p, Coord q, int level)
{
  if (level == 0) {
    Util_DrawLine (window, p, q);
    return;
  }
  Coord a= Coord_Lerp (p, 1/3.0, q);
  Coord b= Coord_Lerp (p, 2/3.0, q);
  Coord c= Coord_RotatedByAround (p, MATH_PI * 2/3.0, a);
  Util_DrawKochLine (window, p, a, level-1);
  Util_DrawKochLine (window, a, c, level-1);
  Util_DrawKochLine (window, c, b, level-1);
  Util_DrawKochLine (window, b, q, level-1);
}
}
```

```
void
Util_DrawKochFlake (bx_window window, Coord center, Coord corner1, int level)
{
  Coord corner2=  Coord_RotatedByAround (corner1, MATH_PI * +2/ 3.0, center);
  Coord corner3=  Coord_RotatedByAround (corner1, MATH_PI * -2/ 3.0, center);
  Util_DrawKochLine (window, corner1, corner2, level);
  Util_DrawKochLine (window, corner2, corner3, level);
  Util_DrawKochLine (window, corner3, corner1, level);
}
```

```
int
MainTest_KochFlakeDemo (void)
{
  bx_init ();
  bx_window window= bx_create_window ("Koch", 50, 50, 500, 500);
  for (int k= 0; /*forever*/; k= (k+1)%6) {
    Coord center= Coord_FromXY (250, 250);
    Coord cursor= Util_MouseCoord (bx_read_mouse (window));
    bx_clear_canvas (window, bx_white ());
    Util_DrawKochFlake (window, center, cursor, k);
    bx_show_canvas (window, 100);
  }
  bx_loop ();
  return 0;
}
```

3 Dragon de Heighway

```
void
Util_DrawDragonStep (bx_window window, Coord p, Coord q)
{
  Coord m= Coord_Lerp (p, 1/2.0, q);
  Coord t= Coord_RotatedByAround (p, MATH_PI * 1/2.0, m);
  Util_DrawLine (window, p, t);
  Util_DrawLine (window, q, t);
}
```

```
void
Util_DrawDragon (bx_window window, Coord p, Coord q, int level)
{
  if (level == 0) {
    Util_DrawLine (window, p, q);
    return;
  }
  Coord m= Coord_Lerp (p, 1/2.0, q);
  Coord t= Coord_RotatedByAround (p, MATH_PI * 1/2.0, m);
  Util_DrawDragon (window, p, t, level-1);
  Util_DrawDragon (window, q, t, level-1);
}
```

4 Dragon de Heighway à angle paramétrable

```
double
Coord_Distance2 (Coord p, Coord q)
{
  Coord pq= Coord_Difference (q, p);
  return Coord_Length2 (pq);
}

double
Coord_Distance (Coord p, Coord q)
{
  Coord pq= Coord_Difference (q, p);
  return Coord_Length (pq);
}

double
Coord_Angle (Coord p) {
  return atan2 (p.y, p.x);
}
```

```
void
Util_DrawDragon2_Rec (bx_window window, Coord p, Coord q, double angle, int level)
{
  bx_color col_t= bx_blend_color (col_p, ratio, col_q);
  if (Coord_Distance2 (p, q) < 25 || level > 64 ) {
    Util_DrawLine (window, p, q);
    return;
  }
  Coord m= Coord_Lerp (p, 1/2.0, q);
  Coord t= Coord_RotatedByAround (p, angle, m);
  Util_DrawDragon2_Rec (window, p, t, angle, level+1);
  Util_DrawDragon2_Rec (window, q, t, angle, level+1);
  }
}

void
Util_DrawDragon2 (bx_window window, Coord p, Coord q, double angle)
{
  Util_DrawDragon2_Rec (window, p, q, angle, 0);
}
```

```
int
MainTest_Dragon2Demo (void)
{
  bx_init ();
  bx_window window= bx_create_window ("dragon", 50, 50, 500, 500);
  for (;;) {
    Coord center= Coord_FromXY (250, 250);
    Coord cursor= Util_MouseCoord (bx_read_mouse (window));
    double angle= Coord_Angle (Coord_Difference (cursor, center));

    bx_clear_canvas (window, bx_white ());
    Util_DrawDragon2 (window, center, cursor, angle);
    bx_show_canvas (window, 100);
  }
  bx_loop ();
  return 0;
}
```