Compte rendu TP2
Dessiner un beau dessin

Développement en C++
Semestre 2

De ce TP, nous pouvons retenir qu'un programmeur, ce doit d'être vigilant dans la rédaction de son travail. En effet il ne faut pas se limiter au fait que le programme marche, mais soigner son code, pour qu'il soit compréhensible par les autres, mais également pour nous même ulterieurement.
En effet il peut arriver qu'après coup on ne comprenne plus ce qu'on a voulut faire.

Ainsi, voici différentes conventions que nous avons essayé de corriger, cela rend le code plus propre, et plus lisible.

1 Variable

Les variables sont très importantes dans un programme, ainsi, il faut faire attention à leurs utilitées, leurs appellations, et leurs types.

1.1 Noms

Les noms de variables sont extrement important.
Dans le code fournit par l'IUT, tous les noms de variables s'appellaient vlx, c'était catastrophique pour la compréhension du code :
il était impossible de comprendre à quoi servait une variable.

1.2 Unicité d'utilité

On ne doit pas avoir une variable qui fait plusieurs choses différentes, cela permet d'améliorer la compréhension.

1.3 Variables globales

Les variables globales sont à proscrire, en effet, elles rendent le code sale, et utilisent beaucoup de mémoire, étant donné qu'elles sont disponibles dans tous les sous-programmes, même si on ne s'en sert pas.

2 Boucles

Il est préférable d'utiliser une boucle for lorsque que l'ont doit répéter un nombre d'actions connu à l'avance, plutôt que d'utiliser une boucle while. La boucle for est plus lisible, elle permet de déclarer une variable uniquement dans la boucle, et les informations sont condensés, donc plus claires, on saura tout de suite qu'une boucle for veut dire que l'on va répéter des actions un certain nombre de fois.

3 Sous-programmes

« Diviser pour mieux régner »

Il faut diviser le code avec le plus de sous-programmes possible : un sous-programme ne doit pas dépasser 40 lignes.
En effet il est plus simple de comprendre ce que fait le sous-programme, que de devoir comprendre tout un code, situé dans le programme principal.

Il faut également penser à donner un nom clair à son sous-programme, quitte à ce qu'il soit un peu long à écrire.

4 Indentation

L'indentation d'un code est utile : en effet un code bien indenté est beaucoup plus facilement compréhensible, on voit tout de suite quand commence une boucle ou une condition et quand elle finit.
Sans indentation, un code est tout simplement illisible.

5 Commentaires

« Si après avoir lu uniquement les commentaires d'un programme vous n'en comprenez pas l'utilité, jetez-le tout ! » IBM

Les commentaires d'un programmes sont eux aussi utiles, ils servent à aider les programmeurs dans le cas ou il relit sont code quelques temps plus tard, ou lorsqu'un autre programmeur lit le code, grâce aux commentaires, il compredra le code.
Cependant, il n'est pas utile de commenter toutes les lignes, en effet certaines lignes sont évidente, de plus si les fonctions ont des noms explicites, elles parleront toutes seules !

```
/*
|
   ------------------------------------------------------------------

|       TP2 Comment realiser un beau dessin !!
|
   ------------------------------------------------------------------

| dev2 - Developpement en C++
| Realise par Fabrice Valleix et Antoine de Roquemaurel (
   groupe 1F)
|
| Dessine pixel par pixel une maison et des sapins grace a la
   librairie SDL
|
   ------------------------------------------------------------------
   */

#include "BaseSDL.h"

enum Figure {triangle,rectangle,trapeze};


/*
|
   ------------------------------------------------------------------

|               Tracer une figure a l'ecran
|
   ------------------------------------------------------------------

|
| typeFigure(Figure): peut prendre triangle, trapeze ou
   rectangle,
                si il y a un autre parametre, la procedure ne
                   fera rien
| ec(ecran):       la variable contenant l'endroit ou l'on va
   dessiner
| positionX (int):    la position en abscisse de la figure
| positionY(int):   la position en ordonnee de la figure,
                est en mode mis-a-jour permet de savoir ou sera
                   dessine le tronc si
                c'est pour un sapin, dans les autres cas, maj
                   inutile
| hauteur(int):     hauteur de la figure
| largeur(int):     largeur de la figure
| couleur(char):    couleur de la figure, peut prendre
            J(Jaune), N(Noir), R(Rouge), V(Vert)
            la couleur sera Blanc dans tous les autres cas
|
| Necessite positionX, positionY, hauteur, largeur >= 0
```

```
35 |
     ----------------------------------------------------------------
   */
36 void tracerFigure(Figure typeFigure,//Type de la figure, (
   triangle, rectangle, trapeze)
37         ecran ec, //variable contenant le lieu ou il faut
             dessiner
38         int positionX, //position en abscisse de la figure
39         int &positionY, //position en ordonnee de la figure
40         int hauteur, //hauteur de la figure
41         int largeur, //largeur de la figure
42         char couleur); //couleur de la figure (Blanc, Vert,
             Rouge, Jaune, Noir)
43
44 /*
45 |
     ----------------------------------------------------------------

46 |                 Tracer un sapin a l'ecran
47 | procedure se servant de la procedure tracerFigure tracer le
     tronc et les branches
48 |
     ----------------------------------------------------------------

49 | ec(ecran):      La variable contenant l'endroit ou l'on va
     dessiner
50 | positionX (int):    La position en abscisse du sapin
51 | &positionY(int):    La position en ordonnee du sapin
52 | hauteurSapin(int):  Hauteur du sapin
53 | largeurTronc(int):  Largeur du tronc du sapin
54 | hauteurTronc(int):  Hauteurdu tronc du sapin
55 | nbBranche(int):   Nombre de branche qu'aura le sapin
56 |
57 | Necessite positionY, positionX, largeurTronc, hauteurTronc,
     hauteurSapin, nbBranche >= 0
58 |
     ----------------------------------------------------------------
   */
59 void tracerSapin(ecran ec,
60         int positionX, //position en abscisse du sapin
61         int positionY, //position en ordonnee du sapin
62         int hauteurSapin, //hauteur du sapin
63         int largeurTronc, //largeur du tronc du sapin
64         int hauteurTronc, //hauteur du tronc du sapin
65         int nbBranche); //nombre de branches du sapin
66
67 int main(void)
68 {
69   ecran ec;
70   int positionY;
71
```

```
72   //on creer la fen tre on l'on pourra dessiner, la variable
       est necessaire pour tracerFigure
73    ec = creerEcran();
74
75      /* on trace tous les sapins  */
76    tracerSapin(ec, 550, 10, 70, 15, 60, 2);
77    tracerSapin(ec, 450, 150, 50, 15, 50, 4);
78    tracerSapin(ec, 100, 350, 20, 5, 15, 2);
79    tracerSapin(ec, 500, 100, 50, 20, 50, 4);
80    tracerSapin(ec, 100, 200, 20, 5, 15, 3);
81    tracerSapin(ec, 130, 100, 70, 15, 60, 2);
82
83
84      /* on trace le corps de la maison */
85    positionY = 350;
86    tracerFigure(rectangle, ec, 150, positionY, 200, 300, 'R');
87    positionY = 250;
88    tracerFigure(trapeze, ec, 150, positionY, 300, 200, 'B');
89
90
91      /* on dessine la porte, puis les deux fenetres*/
92    positionY = 480;
93    tracerFigure(rectangle, ec, 275, positionY, 70, 50, 'J');
94    positionY = 400;
95    tracerFigure(rectangle, ec, 180, positionY, 50, 50, 'J');
96    positionY = 400;
97    tracerFigure(rectangle, ec, 370, positionY, 50, 50, 'J');
98
99
100   actualiser(ec); //actualise l'ecran avec les figures
101   attendre(); //attend un evenement
102   return EXIT_SUCCESS;
103 }
104
105
106 /* fonction qui se sert de tracerFigure pour tracer un sapin
      de nbBranche*/
107 void tracerSapin(ecran ec, int positionX, int positionY,
108           int hauteurSapin, int largeurTronc, int hauteurTronc
                , int nbBranche)
109 {
110   int placeTroncX;
111   int largeurTriangle = 1;
112
113   for (int i = 0; i < nbBranche; i++) //on parcourt la boucle
        autant de fois qu'il y a de branches
114   {
115     /* On trace un triangle, puis on augmente la largeur du
          triangle pour le suivant*/
116     tracerFigure(triangle, ec, positionX, positionY,
          hauteurSapin, largeurTriangle);
```

```
117       largeurTriangle += 10;
118
119     }
120
121     /* on positionne le tronc et on le trace*/
122     placeTroncX = positionX + (hauteurSapin - largeurTronc) / 2;
123     tracerFigure(rectangle, ec, placeTroncX, positionY,
          hauteurTronc, largeurTronc);
124   }
125
126
127   /* fonction qui trace une figure, triangle, rectangle ou
        trapeze*/
128   void tracerFigure(Figure typeFigure, ecran ec, int positionX,
129             int &positionY, int hauteur, int largeur, char
                couleur)
130   {
131     int longueurTraitCourant = 0;
132     int curseurTraitY = 0;
133     int posXTraitCourant;
134
135       /* les triangles et les trapezes se dessinent de la m me
            maniere*/
136     if(typeFigure == triangle || typeFigure == trapeze)
137     {
138        //si c'est un triangle, cette valeure sera egale a un,
             pour une pointe,
139       //sinon on commence avec la largeur du trapeze: un trapeze
             n'est qu'un triangle sans la pointe
140       longueurTraitCourant = largeur;
141       curseurTraitY = positionY; //on place le curseur a l'
            emplacement du haut du triangle/trapeze
142
143       for (int i= largeur; i<= hauteur; i++)
144       {
145         posXTraitCourant = (hauteur - longueurTraitCourant) / 2;
146         for(int j = 0; j < longueurTraitCourant; j++) //on trace
              le trait courant
147         {
148           afficherPixel(ec, positionX+posXTraitCourant+j,
                curseurTraitY, couleur);
149         }
150
151       /* Trait suivant + long et + bas */
152         longueurTraitCourant++;
153         curseurTraitY++;
154       }
155       positionY= positionY+ (hauteur-largeur);
156     }
157     else if(typeFigure == rectangle)
158     {
```

```
159       for(int i= 0; i <= hauteur; i++)
160       {
161         for(int j=0; j <= largeur; j++)
162         {
163           afficherPixel(ec, positionX+j, positionY+i, couleur);
164         }
165       }
166     }
167   }
```

Listing 1 – Code en C++n language