Université de la Rochelle
INFO-12605C - IHM - TD8 : Dessin et sérialisation

© B. Besserer, R. Péteri

Année universitaire 2016-2017

1 Quelques questions C#

1. Le pixel de coordonnées (0, 0) est situé au centre exact de l'écran ? Vrai ☐ Faux ☐
2. Dans le système de coordonnées du C#, les valeurs en x croissent de gauche à droite ? Vrai ☐ Faux ☐
3. Tout `Control` a un objet graphique (Graphics) associé ? Vrai ☐ Faux ☐
4. Toutes les `Forms` héritent de la méthode `OnPaint` ? Vrai ☐ Faux ☐
5. Quel est la signification de l'interface IEnumerable<T> qu'un objet peut implémenter ?

   ☐
6. En C#, un fichier n'est qu'une suite de données binaires ; la structuration des informations contenues dans une fichier est du ressort de l'application utilisant ce fichier ? Vrai ☐ Faux ☐
7. La classe `StreamReader` hérite de la classe `Stream` ? Vrai ☐ Faux ☐
8. Sans modification, toute classe déclarée en C# peut être sérialisée dans un fichier. Vrai ☐ Faux ☐
9. La méthode `Seek` de la classe `FileStream` cherche toujours ses informations par rapport au début d'un fichier (comme `fseek`). Vrai ☐ Faux ☐

1) False. The coordinate (0,0) corresponds to the upper-left corner of a GUI component on which drawing occurs
2) True.
3) True.
4) True.
5) L'interface IEnumerable<T> décrit le comportement de tout objet qui peut disposer d'un itérateur (array, list, queue) et, en conséquence, propose des propriétés (count) et des méthodes (first(), last(), opérateur []) permettant l'accès au éléments.
6) True.
Generally, a file can contain arbitrary data in arbitrary formats. In some operating systems, a file is viewed as nothing more than a collection of bytes, and any organization of the bytes in a file (such as organizing the data into records) is a view created by the application programmer.

7) False. StreamReader inherits from TextReader

Textreader : *can read a sequential series of characters. so it read only characters.*

Streamreader : *reads characters from a byte stream in a particular encoding. It also read character from a stream.*

8) False. Only classes with the Serializable attribute can be serialized.
9) False. It seeks relative to the SeekOrigin enumeration member that is passed as one of the arguments (begin, current, end)

2 Dessin dans une fenêtre

On souhaite réaliser une application permettant de tracer des cercles. Il faut que, après avoir cliqué dans la zone de dessin pour marquer le centre du cercle, un déplacement de la souris trace le cercle CONTINUELLEMENT

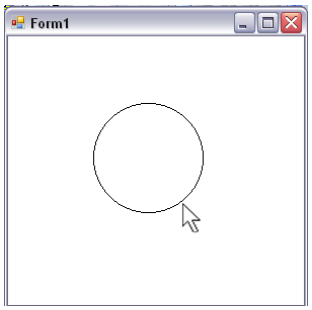

FIGURE 1 – Saisie d'un cercle

PENDANT LE DEPLACEMENT jusqu'au moment où l'on relâche le bouton. Ensuite, le cercle ainsi défini (centre, rayon) est ajouté à une liste ; sachant que l'instant seul le dernier cercle reste à l'écran (figure 1) en attendant une nouvelle saisie.

1. Définir une classe cercle nommée TheCircle, avec comme données membres son rayon et son centre. Utilisez les *properties*

```
namespace TD3_dotNet
{
    public class TheCircle
    {
        public Point Center {get;set;}
        public int Radius {get; set;}
        public TheCircle(Point c) // constructeur
        {
            Center = c;
            Radius = 0;
        }
    }
}
```

2. Où doit on définir la liste `List<TheCircle> maListe` qui stockera les différentes cercles tracés par l'utilisateur ?

   Comme donnée membre de la classe Form.

   `List<TheCircle> maListe = new List<TheCircle>();`

3. Lors du clic gauche, il faudra récupérer les coordonnées du clic, créer un cercle avec ces coordonnées comme centre, et rafraîchir le tracé tant que le bouton gauche de la souris sera enfoncé. A quels evenements faudra t'il attacher des méthodes ?

   Il faut réagir aux evenements MouseDown, MouseUp et MouseMouve.

4. Pour ce fonctionnement particulier (rectangle de sélection, rectangle élastique, etc...) à quel endroit du code doit on écrire le code permettant **le tracé graphique du cercle pendant sa création** ?

   Réponse : dans la methode en réaction à l'événement de type mousemove. De façon exceptionnelle, le tracé graphique ne se fait pas en réponse à l'événement PAINT. Probleme : Il faut donc créer un " contexte graphique " (classe Graphics) pour pouvoir y faire du dessin.

5. Écrire le code permettant le tracé graphique d'un cercle pendant sa création.
   Ici, exeptionnellement, le tracé ne peut se faire dans la méthode Paint :

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    maListe.Add(new TheCircle(((MouseEventArgs)e).Location));
    m_DrawCircle = true;
}

private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    if (m_DrawCircle)
    {
        TheCircle c = maListe.Last();
        int dX = e.X - c.Center.X;
        int dY = e.Y - c.Center.Y;
        int r = (int)Math.Sqrt((dX * dX) + (dY * dY));
        maListe.Last().Radius = r;
        // pour eviter une erreur dans le tracé, r > 0
        if(r > 0)
            using (Graphics gfx = this.CreateGraphics())
            {
                gfx.Clear(SystemColors.Window);
                Rectangle boundingRect = new Rectangle(c.Center.X - r, c.Center.Y - r, 2 * r, 2 * r);
```

```
                gfx.DrawArc(Pens.Black, boundingRect, 0, 360);
            }
        }
    }

    private void Form1_MouseUp(object sender, MouseEventArgs e)
    {
        m_DrawCircle = false;
    }
```

6. Est-il possible d'arrêter le tracé du cercle en cours autrement qu'en relâchant la touche de la souris ? Expliquez.
   Est-ce que les modalités d'interaction pour tracer un cercle peuvent convenir pour un écran tactile ? Expliquez *Le tracé du cercle est réactualisé à chaque déplacement (même infime) de la souris. Entre deux déplacement on revient à la boucle de traitement des message. On peut imaginer récupérer un appui sur une touche du clavier (ESCAPE) pour mettre à FAUX la variable m_DrawCircle*
   Pour tracer, l'usage d'un écran tactile est OK (touchdown, move, touchup). On pourrait imaginer utiliser deux doigts (marquer le centre, et dimensionner le cercle avec le second doigt, mais il faut que l'écran tactile supporte les appuis multiple (actuellement c'est vrai à 99.99% pour les écrans tactiles qui ont au moins 2 points de contacts, Les écrans qui ont 10 points de contacts sont moins fréquents). Il y a une suite à cette question pour le mode "investigation" ou il faut utiliser la durée de l'appui par exemple (appui court pour commencer le tracé d'un cercle, appui long pour "consulter")

Une fois les cercles dessinés et stockés dans la liste, on souhaite les afficher TOUS lors de l'appui sur la touche 'D' du clavier, selon un fonctionnement "toogle" (un appui sur D affiche tous les cercles, un autre appui sur D masque tous les cercles).

1. Dans quelle partie du code doit s'effectuer le tracé de tous les cercles ? *Cette fois ci, dans la surcharge de la méthode paint*
2. Écrire le code permettant le tracé de tous les cercles ; ainsi que la méthode gérant les actions sur la touche.

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    if ((e.KeyChar == 'd') || (e.KeyChar == 'D'))
    {
        bDoPaintAll = !bDoPaintAll;
        this.Invalidate(); ;
    }
}

private void Form1_Paint(object sender, PaintEventArgs e)
{
    if (bDoPaintAll)
    {
        if (maListe.Count > 0)
        {
            for (int i = 0; i < maListe.Count; i++)   // utilisation possible de foreach
            {
                Rectangle boundingRect = new Rectangle(maListe[i].Center.X - maListe[i].Radius,
                maListe[i].Center.Y - maListe[i].Radius, 2 * maListe[i].Radius, 2 * maListe[i].Radius);
                e.Graphics.DrawArc(Pens.Black, boundingRect, 0, 360);
            }
        }
    }
}
```

Enfin, il s'agit maintenant de détecter si les coordonnées d'un clic n'importe ou dans la zone de dessin sont contenues dans un des cercles de la liste, et si c'est le cas, on colorie ce cercle en rouge tant que le bouton de la souris reste appuyé.

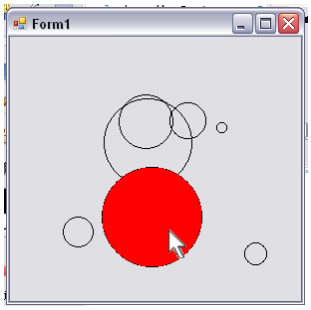

FIGURE 2 – Tracé d'un disque rouge lors d'un clic dans un cercle de la liste

1. Cliquer dans la zone de dessin peut donc signifier commencer le tracé d'un nouveau cercle ou bien déterminer si l'on est dans un cercle déjà existant. Dans un premier temps, un booléen permettra de discriminer entre ces deux mode. Au niveau IHM, proposer une manière de passer de l'un à l'autre de ces états ?

   Il faut mettre une variable boolean bDoDrawing qui sera testée pour savoir si on est dans une mode "dessin" ou "investigation". On peut par exemple utiliser un double-clic pour changer de mode, avec idéalement un rappel du mode effectif dans la barre d'état, par exemple. On peut aussi choisir judicieusement une action sur une touche du clavier. Enfin, et pour les interaction "post wimp", on peut aussi agir sur la durée de l'enfoncement de la touche de souris (clic long = investigation ; clic avec aussitôt mouvement = tracé d'un nouveau cercle)

2. Écrire le code correspondant au test (savoir si les coordonnées du clic sont à l'intérieur d'une cercle existant). Comment auriez-vous procédé pour savoir si les coordonnées sont dans un rectangle ?

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
  //    m_down = true;
  if (!m_endPaint)
  {
     m_doPaint = true;
     Circle newCircle = new Circle(new Point(e.X, e.Y));
     maListe.Add(newCircle);
  }
  else
  {
    for (int i = 0; i < maListe.Count; i++)
    {
      double distance = (int)System.Math.Sqrt(System.Math.Pow(maListe[i].m_center.X - e.X, 2) +
           System.Math.Pow(maListe[i].m_center.Y - e.Y, 2));
      if (distance <= maListe[i].m_radius)
      {
         Rectangle boundingRect = new Rectangle(maListe[i].m_center.X - 2*maListe[i].m_radius,
                maListe[i].m_center.Y - 2* maListe[i].m_radius, 2*2 * maListe[i].m_radius, 2*2 * maListe
         Graphics grx = this.CreateGraphics();
         grx.DrawArc(Pens.Black, boundingRect, 0, 360);
         grx.FillEllipse(new SolidBrush(Color.Red), boundingRect);
       }
     }
   }
}
```

   ***Pour savoir si un point se trouve dans un rectangle, pas besoin d'écrire du code**. La classe rectangle comporte une méthode Contains(Point p) indiquant si le Point p est dans le rectangle ou non.*

3 Sérialisation (préparation au TP)

Nous allons développer une application reprenant en partie le cahier des charges d'un TP effectué sous Qt, en développant une classe `Histogram` et en implémentant de surcroit la sérialisation

3.1 Mise en œuvre de la classe `Histogram`

Un objet de la classe `Histogram` doit représenter un histogramme contenant le nombre `histo_size` d'intervalles. Le nombre d'occurrence (hauteur) de chaque intervalle est sera déterminé par tirages aléatoires.

1. La classe `Histogram` doit permettre la création d'objets `Histogram` .
   Une définition de la classe `Histogram` pourrait être :

```
class Histogram
  {
  public int m_size;
  public List<Intervalle> m_list;

  public Histogram(int histo_size)
  {
   ...
  }
}
```

   Donnez la définition du constructeur de la classe `Histogram`.

```
public Histogram(int histo_size)
  {
    m_size = histo_size;
    m_list = new List<Intervalle>(histo_size);
  }
```

2. De la même manière, on définit la classe `Intervalle`, avec `m_x` la valeur du *bin* (la classe) de l'histogramme et `m_amount` le nombre d'occurences de la classe.

```
class Intervalle
{
  public int m_x;
  public int m_amount;
  public Intervalle(int a, int b)
  {
     m_x = a;
     m_amount = b;
  }
}
```

3.2 Implementation de l'application

3.2.1 Initialisation des valeurs de l'histogramme

1. On développe une application Windows Forms : où initialiseriez-vous les valeurs de l'histogramme ?

   Dans la fonction Form1_Load

2. Donnez le code d'initialisation des valeurs de l'histogramme, pour qu'il comprenne 10 bins, et que le nombre d'occurrences par bins soit initialisé avec une valeur aléatoire entre 0 et 99 (pour générer un nombre aléatoire, on utilisera la classe `Random` et sa méthode `Next(MIN_VALUE, MAX_VALUE))`.

```
private void Form1_Load(object sender, EventArgs e)
{
  Random rnd = new Random();

  myHisto = new Histogram(10);
  for (int i = 0; i < myHisto.m_size; i++)
    myHisto.m_list.Add( new Intervalle(i, rnd.Next(0, 99)));
}
```

3.2.2 Sérialisation des données

1. Que faut-il faire pour rendre une classe 'sérialisable'

   En indiquant au début des classes Histo et Intervalle l'attribut `[Serializable]`

2. Compléter le code ci-dessous permettant l'écriture de la classe `Histogramme` dans un fichier "test.dat".

```
System.IO.FileStream output = new System.IO.FileStream("test.dat",
             System.IO.FileMode.Create, System.IO.FileAccess.Write);

BinaryFormatter writer = new BinaryFormatter();

writer.Serialize(....);
output.Close();
```

   Element de correction : *C'est très simple, il suffit d'indiquer ces deux paramètres*

```
writer.Serialize(output, myHisto);
```

3. Ecrire le code permettant le chargement depuis le fichier "test.dat".

```
System.IO.FileStream input = new System.IO.FileStream("test.dat", System.IO.FileMode.Open,
  System.IO.FileAccess.Read);

BinaryFormatter loader = new BinaryFormatter();
myHisto = loader.Deserialize(input) as Histogram;
input.Close();
```

3.2.3 Interaction avec le clavier

Ecrire le code de l'application finale, permettant

1. De sauver les données de l'histogramme dans un fichier "test.dat" lors de l'appui sur la touche 'S' (save) du clavier.
2. De réinitialiser à 0 les données de l'histogramme lors de l'appui sur la touche 'C' (clear) du clavier.
3. De recharger les données de l'histogramme depuis fichier "test.dat" lors de l'appui sur la touche 'L' (load) du clavier.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
  if (e.KeyData == Keys.S)
  {
    m_saved = true;
    System.IO.FileStream output = new System.IO.FileStream("test.dat", System.IO.FileMode.Create,
                System.IO.FileAccess.Write);

    BinaryFormatter writer = new BinaryFormatter();

    writer.Serialize(output, myHisto);
    output.Close();

  }
  if (e.KeyData == Keys.L)
  {
    if (m_saved)
    {
      System.IO.FileStream input = new System.IO.FileStream("test.dat", System.IO.FileMode.Open,
                    System.IO.FileAccess.Read);
      BinaryFormatter loader = new BinaryFormatter();
      myHisto = loader.Deserialize(input) as Histogram;
      input.Close();
    }
  }
  if (e.KeyData == Keys.C)
  {
    for (int i = 0; i < myHisto.m_size; i++)
                    myHisto.m_list[i].m_amount = 0;
  }
  Invalidate();  // pour forcer le rafraichissment graphique
}
```