TD

Une cellule mutante dans une fenêtre graphique

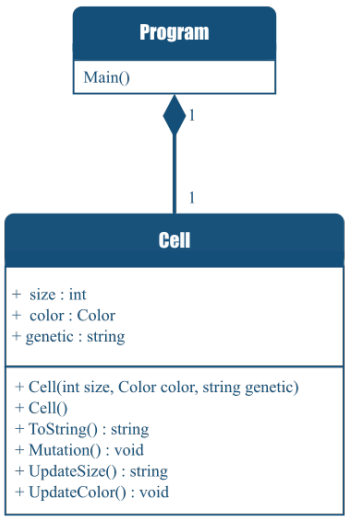
Octobre 2023

Présentation :

Cet exercice va nous amener à manipuler une nouvelle classe permettant de définir un objet Cellule. Cet objet va être amené à changer de taille et/ou encore de couleur. Nous allons donc dans un premier temps implémenter la classe attendue en nous appuyant une fois encore sur un diagramme de classes puis, dans un deuxième temps, nous allons implémenter une interface graphique pour observer les mutations dans une nouvelle fenêtre.

Implémentation de la classe Cellule :

Vous êtes libre de proposer l'implémentation de votre choix en accord avec le diagramme de classes suivant. Les détails de la méthode mutation sont donnés ci-dessous.



La méthode ***mutation*** correspond à la méthode qui va influencer, en fonction de différentes probabilités, l'information génétique de notre cellule. Cette nouvelle information génétique sera ensuite traitée par d'autres méthodes et pourra conduire à des modifications de l'aspect de notre cellule.
La méthode ***mutation*** parcourt un à un tous les caractères de l'attribut *genetic* de la cellule. En fonction des probabilités suivantes, les changements donnés sont susceptibles d'apparaître dans le code génétique de la cellule.

⋆ Il y a 15% de chance de voir apparaître une mutation du "A" en "T".
⋆ Il y a 7% de chance de voir apparaître une mutation du "T" en "AA".
⋆ Il y a 21% de chance de voir apparaître une mutation du "C" en "G".
⋆ Il y a 4% de chance de voir apparaître une mutation du "G" en "CG".

Entre chaque caractère lu dans le code génétique de la cellule, il y a de plus une probabilité de 5% de voir apparaître entre ces deux caractères un des quatre caractères "A", "C", "T", "G" choisit aléatoirement.
Une fois l'ensemble du code génétique parcourut, et l'ensemble des mutations appliquées, la cellule doit se mettre à jour et opérer des changements si besoin :

⋆ La **couleur** de la cellule est déterminée par le motif le plus fréquent parmi les motifs suivants :
- **TGT** - Noir (couleur par défaut de la cellule)
- **ATT** - Bleu
- **CTC** - Jaune
- **ACT** - Violet
- **GTC** - Orange
- **GAA** - Vert

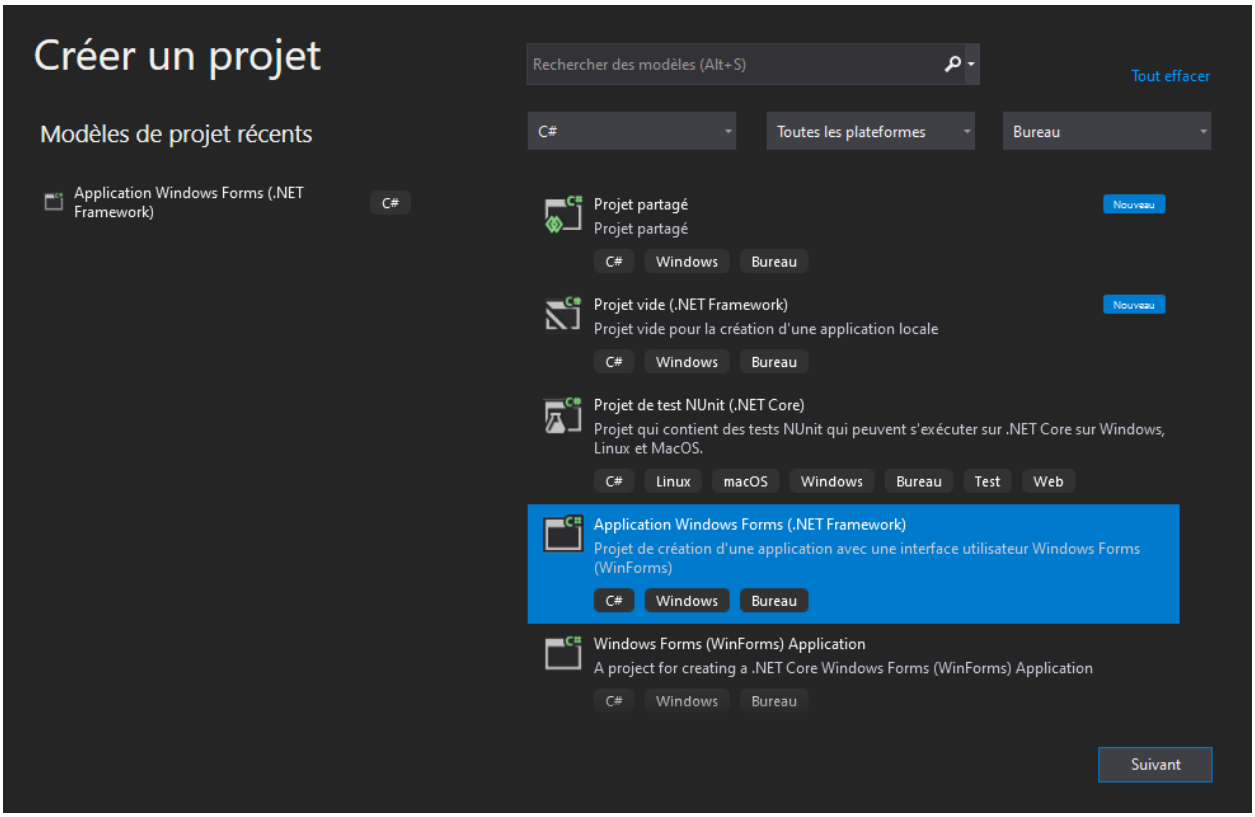
Si aucun de ces patterns n'est détecté dans le code génétique de la cellule, alors sa couleur devient le noir.
Si plusieurs patterns sont détectés avec la même occurrence, c'est à vous de proposer la méthode de votre choix pour le choix de la couleur (aléatoire, ordre de priorité, pourcentage de probabilité...).

Ressource vers des couleurs prédéfinies en C#

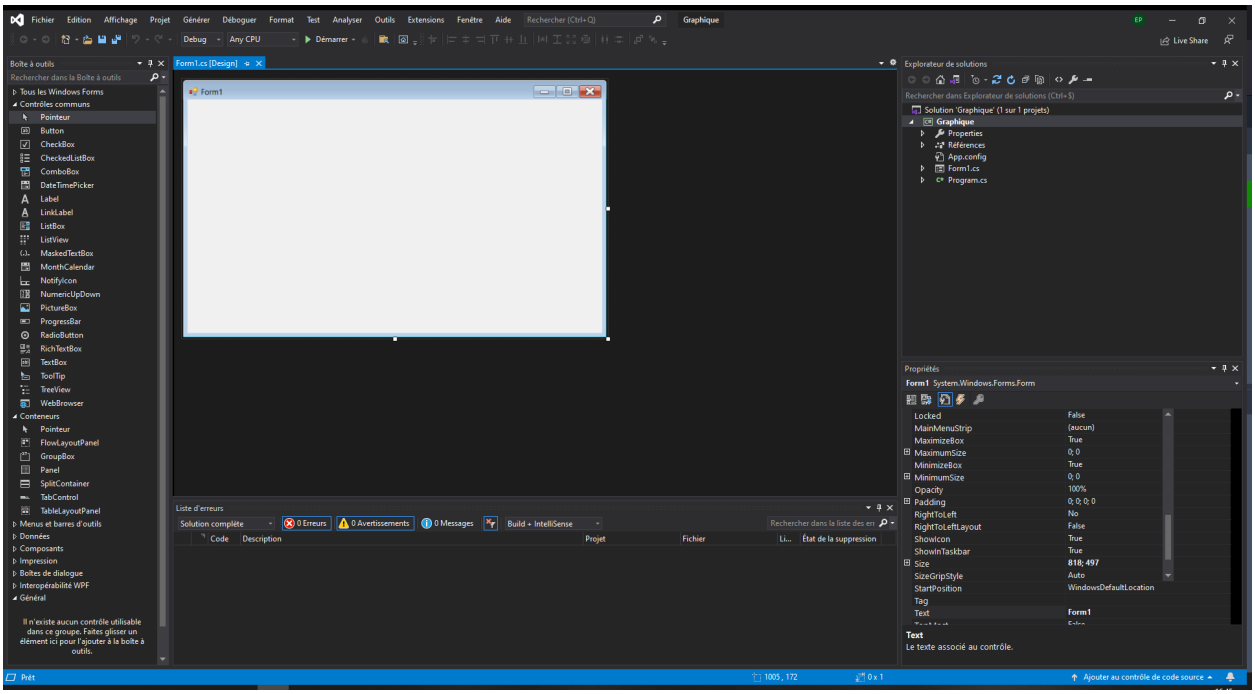
⋆ La cellule est représentée par un cercle donc la taille correspond au diamètre. Par défaut, au début de la simulation, la cellule a un diamètre de 10. La **taille** de la cellule est impactée par la longueur de son code génétique et par le nombre d'occurrence de la lettre "T" dans son code génétique. Elle se calcul comme suit :
- 10 +
- Taille du code génétique divisée par 5 +
- Le minimum entre le nombre d'occurrence de la lettre "T" et la taille précédente de la cellule

Implémentation graphique :

1. Ouvrir Visual Studio
2. Créer un nouveau projet en vous assurant de choisir comme modèle "Application Windows Forms (.NET Framework)".

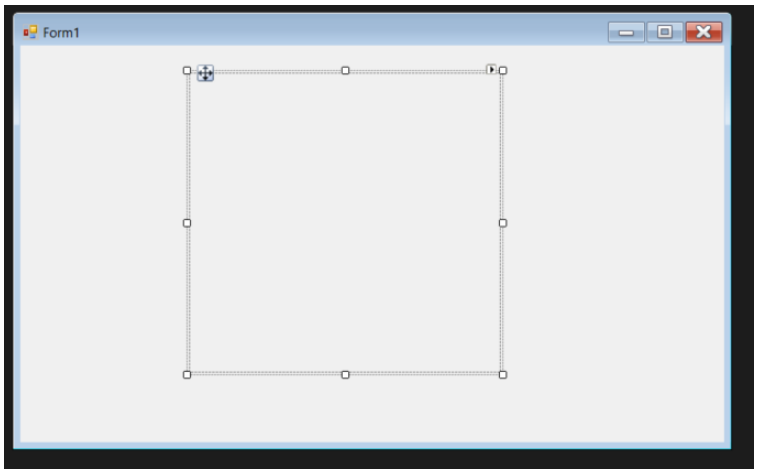


Vous arrivez sur un écran comme celui-ci. Si certains onglets ne s'affichent pas par défaut chez vous, vous pouvez aller les chercher dans le menu affichage (Boîte à outils, Explorateur de solutions).

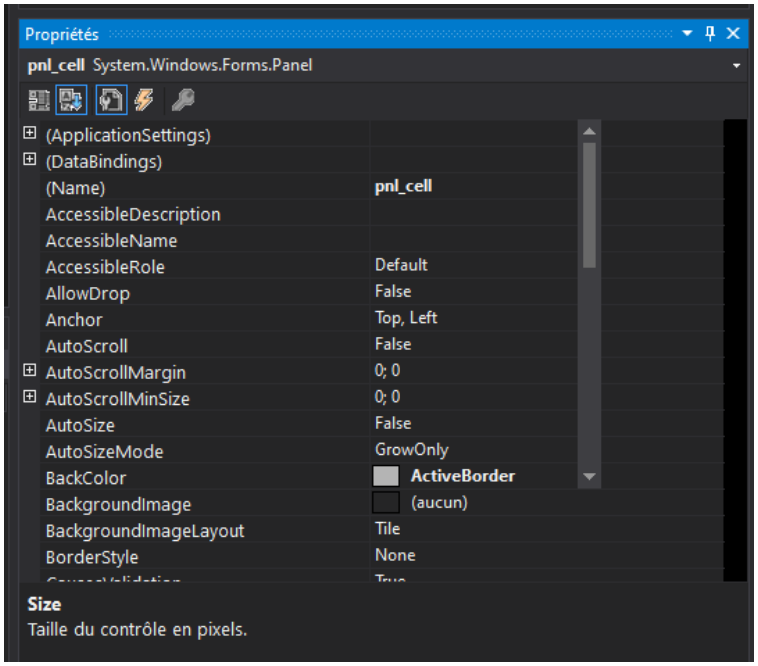


Nous allons travailler pour commencer dans la fenêtre du centre nommée Form1.cs [Design].

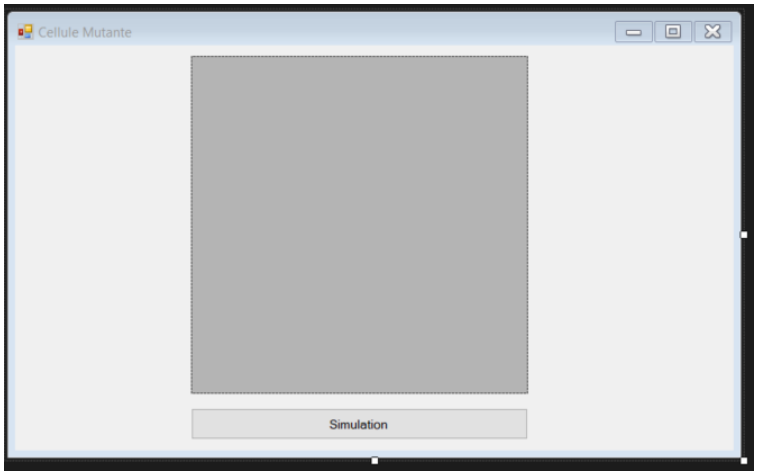
3. Cliquez dans la **boîte à outils**, dans la section **Conteneurs**, sur la ligne **Panel** pour dessinez un carré dans la fenêtre centrale.



4. Dans l'onglet **Propriétés** en bas à droite de l'écran, lorsque le **Panel** est sélectionné, vous accédez à un ensemble de propriétés caractérisant l'objet. Renommez le panel en **pnl_cell** puis attribuez-lui une couleur de fond.



5. Ajoutez de la même façon un bouton sur la fenêtre. Renommez-le **btn_simulation** et modifiez le texte par défaut en **Simulation**. Modifiez également le titre de la fenêtre en **Cellule Mutante**. Vous pouvez vous aider du menu format pour dimensionner et placer les éléments les uns par rapport aux autres.



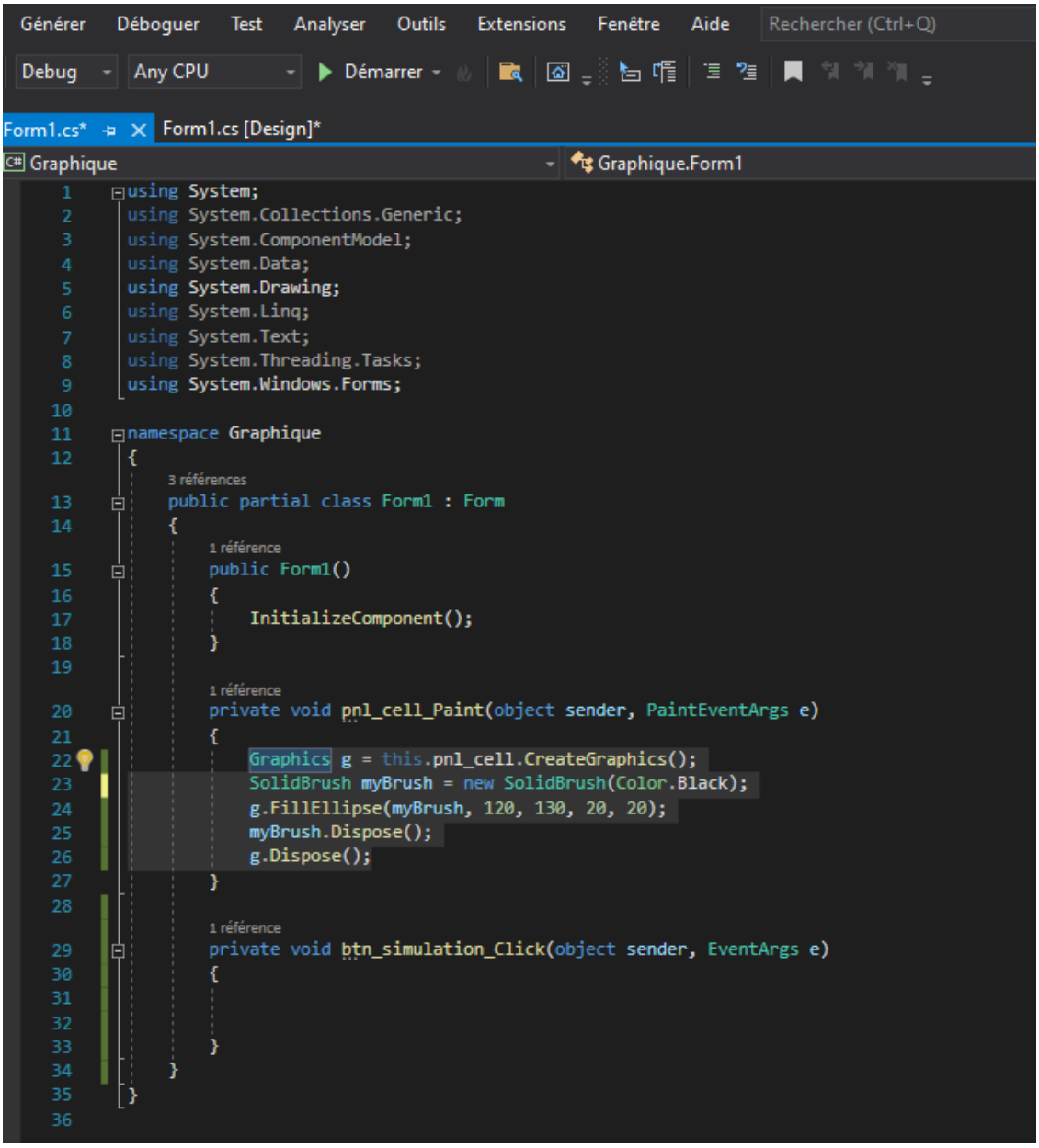
Nous avons maintenant mis en place dans les grandes lignes notre affichage graphique. Vous serez bien sûr vivement encouragés à l'améliorer et à le personnaliser. Pour le moment, concentrons-nous sur les bases et allons voir les lignes de codes qui se cachent derrière cet affichage.

6. Double-cliquez sur le panel dans la fenêtre graphique (ce qui vous conduit sur un nouveau document nommé Form1.cs).
7. Revernir sur Form1.cs [Design] et faire de même avec un double-clic sur le bouton.

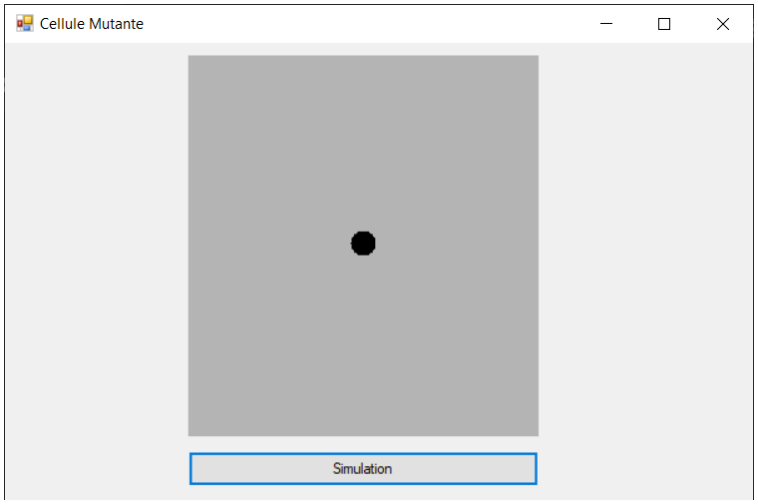
```
Form1.cs*  Form1.cs [Design]*
Graphique                          Graphique.Form1
10
11  namespace Graphique
12  {
        3 références
13      public partial class Form1 : Form
14      {
            1 référence
15          public Form1()
16          {
17              InitializeComponent();
18          }
19
            1 référence
20          private void pnl_cell_Paint(object sender, PaintEventArgs e)
21          {
22
23          }
24
            1 référence
25          private void btn_simulation_Click(object sender, EventArgs e)
26          {
27
28
29          }
30      }
31  }
32
```

8. Saisir les instructions suivantes dans la méthode **pnl_cell_Paint()** :

```
Graphics g = this.pnl_cell.CreateGraphics();
SolidBrush myBrush = new SolidBrush(Color.Black);
g.FillEllipse(myBrush, 120, 130, 20, 20);
myBrush.Dispose();
g.Dispose();
```



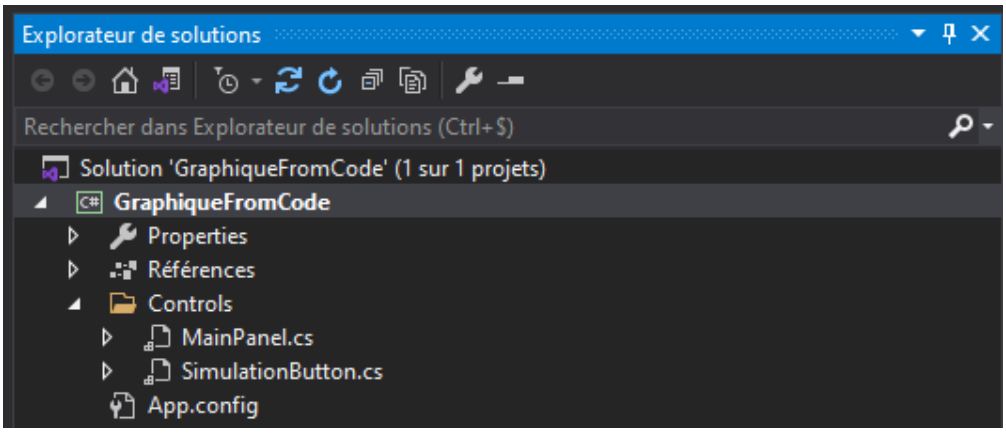
9. Cliquez sur le **bouton Démarrer précédé de la flèche verte** dans le menu de haut de l'écran.



Implémentation graphique depuis l'éditeur de code :

Nous venons de voir une façon de configurer notre fenêtre visuelle à partir d'éléments placés manuellement. Nous allons maintenant voir une nouvelle façon de construire notre fenêtre graphique avec plus de précision.

1. Créez un nouveau projet de type Application Windows Forms (.NET Framework)"
2. Double-Cliquez sur le fond de la fenêtre graphique pour accéder au fichier Form1.cs et créez automatiquement la méthode Form1_Load().
3. Dans l'onglet "Explorateur de solutions", faire un clic droit sur le nom de votre application pour choisir *Ajouter* > *Nouveau Dossier* et nommez ce dossier **Controls**.
4. Faire un clic droit sur ce dossier **Controls** pour ajouter une classe nommée **MainPanel.cs**. Faire de même pour ajouter la classe **SimulationButton.cs**.



5. Ouvrir le fichier **MainPanel.cs**.

Nous allons ici déclarer un nouvel objet personnalisé qui héritera de la classe Panel prédéfinie dans les applications Windows Form.

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace GraphiqueFromCode.Controls
{
    public class MainPanel : Panel
    {
        public MainPanel()
        {
            Name = "pnl_main";
            BackColor = Color.LightGray;
            Anchor = AnchorStyles.None;
            Size = new Size(300, 300);
            Dock = DockStyle.None;
        }
    }
}
```

Nous allons faire de même avec un bouton qui nous servira à lancer la simulation.

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace GraphiqueFromCode.Controls
{
    class SimulationButton : Button
    {
        public SimulationButton()
        {
            Name = "btn_simulation";
            Text = "Simulation";
            BackColor = Color.LightGray;
            ForeColor = Color.Black;
            Size = new Size(300, 40);
            Dock = DockStyle.None;
            Font = new Font("Arial", 14);
            Cursor = Cursors.Hand;
            SetStyle(ControlStyles.Selectable, false);
        }
    }
}
```

Vous êtes libre de personnaliser à loisir ces différents éléments dans votre application. Maintenant que nous avons nos deux éléments, nous allons pouvoir les placer précisément dans notre fenêtre en complétant le fichier **Form1.cs**.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using GraphiqueFromCode.Controls;

namespace GraphiqueFromCode
{
    public partial class Form1 : Form
    {
        // Déclaration des attributs Panel et Button que l'on va
        // afficher dans notre fenêtre
        Panel pnl_main;
        Button btn_simulation;

        public Form1()
        {
            InitializeComponent();

            // Initialisation d'un Panel en utilisant notre classe
            pnl_main = new MainPanel();
            pnl_main.Location = new Point((Size.Width - pnl_main.Width) / 2 - 10,
                (Size.Height - pnl_main.Height) / 2 - 40);
            pnl_main.Anchor = AnchorStyles.None;

            // Initialisation d'un Button en utilisant notre classe
            btn_simulation = new SimulationButton();
            btn_simulation.Location = new Point(90, 390);
            btn_simulation.Anchor = AnchorStyles.None;

            // Ajout des éléments à notre fenêtre
            Controls.Add(pnl_main);
            Controls.Add(btn_simulation);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Définir la taille et le titre de la fenêtre
            Size = new Size(500, 500);
            Text = "Cellule Mutante";
        }
```

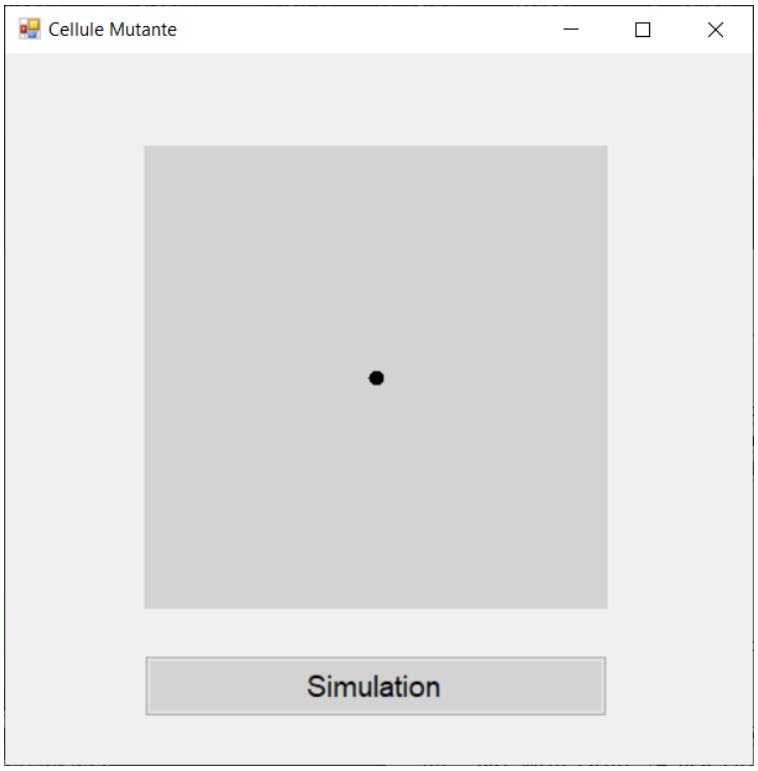
6. Démarrez la compilation pour observer le résultat dans la fenêtre graphique.
7. Ajoutez une méthode **Paint** sur le panel pour visualiser la cellule initiale et ajoutez une instruction pour la lier à notre objet **Panel** juste après les instructions sur son positionnement.

Méthode à ajouter :

```
private void pnl_main_Paint(object sender, PaintEventArgs e)
{
    var g = e.Graphics;
    g.Clear(pnl_main.BackColor);
    SolidBrush coloredBrush = new SolidBrush(Color.Black);
    g.FillEllipse(coloredBrush,  pnl_main.Width/2,  pnl_main.Width/2, 10, 10);
    g.Dispose();
}
```

Instruction à ajouter sous celles données :

```
pnl_main.Anchor = AnchorStyles.None;
pnl_main.Paint += new PaintEventHandler(pnl_main_Paint);
```



8. Ajoutez de même une méthode sur le bouton qui pour le moment affichera un message dans une fenêtre surgissante et ajoutez une instruction pour la lier à notre bouton.

Méthode à ajouter :

```
private void btn_simulation_Click(object sender, EventArgs e)
{
    MessageBox.Show("La simulation commence");
}
```

Instruction à ajouter sous celle donnée :

```
btn_simulation.Anchor = AnchorStyles.None;
btn_simulation.Click += new EventHandler(btn_simulation_Click);
```

9. Démarrez la compilation et observez l'action d'un clic sur le bouton Simulation.
10. Par un clic droit sur le nom de l'application dans l'onglet "Explorateur de solutions", ajoutez une nouvelle classe nommée **Cell.cs**.

Pour l'exemple, nous considérerons une classe **Cell** simplifiée. Il est bien sûr attendu de vous de la remplacer par la classe **Cell** sur laquelle vous avez travaillé.

```
using System;
using System.Drawing;

namespace GraphiqueFromCode
{
    public class Cell
    {
        public int size;
        public Color color;
        private Random rnd = new Random();

        public Cell()
        {
            size = 10;
            color = Color.Black;
        }

        public void Mutation()
        {
            size += 5;
            color = Color.FromArgb(rnd.Next(256), rnd.Next(256), rnd.Next(256));

        }
    }
}
```

11. Déclarez pour **Form1** un attribut de type **Cell**.
12. Initialisez dans le constructeur de **Form1** l'attribut de type **Cell** à partir de la classe **Cell**.
13. Modifiez la méthode **pnl_main_Paint()** pour qu'elle prenne en compte la taille et la couleur de la cellule et que l'affichage de la cellule reste centré.
14. Déclarez pour **Form1** un attribut de type **Timer**.
15. Initialisez dans le constructeur de **Form1** l'attribut de type **Timer** grâce aux instructions suivantes.

```
MyTimer = new Timer();
MyTimer.Interval = (600);
MyTimer.Tick += new EventHandler(UpdateCell);
```

16. Déclarez une nouvelle méthode **UpdateCell** comme suit et complétez la.

```
private void UpdateCell(object sender, EventArgs e)
{
    // instruction pour faire muter la cellule
    // AJOUTER LA BONNE INSTRUCTION ICI

    //Mise à jour de l'affichage
    this.Refresh();
}
```

17. Placez l'instruction "MyTimer.Start() ;" à l'endroit où la simulation démarre.
18. Démarrez la compilation et observer la simulation.