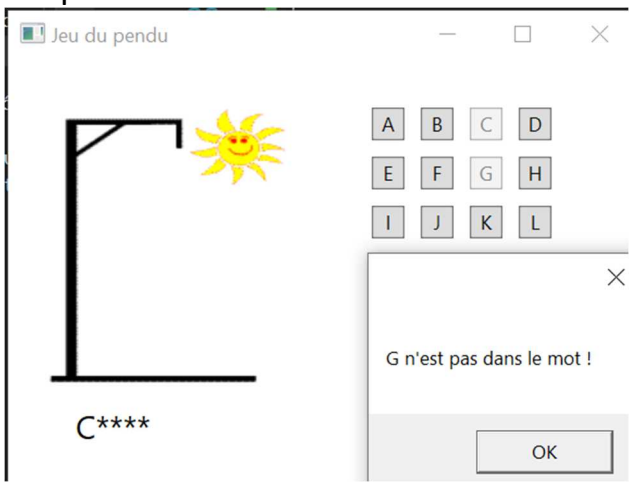
SEQUENCE 5– TABLEAUX
SEANCE 6 – JEU DE PENDU WPF

OBJECTIFS

Savoir-faire quelques algos classiques de tris sur un tableau

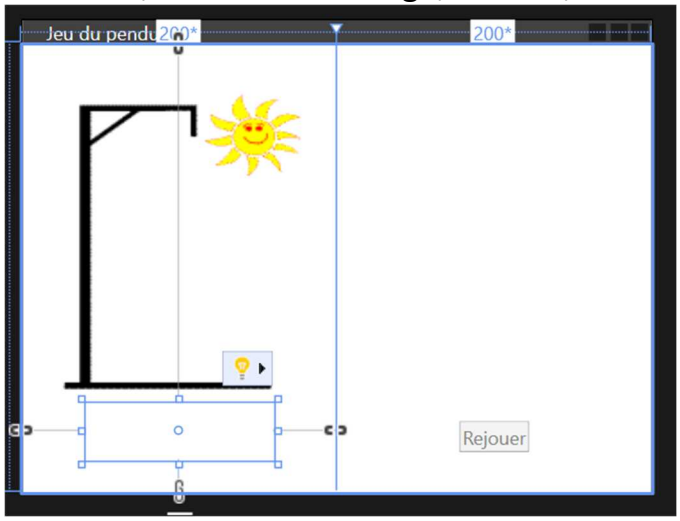
EXO : JEU DU PENDU

Objectif : faire le jeu du pendu en mode graphique.
Pour cela suivez les étapes à venir



ETAPE 1 : L IHM (INTERFACE GRAPHIQUE)

1. Vous définirez en XAML, seulement l'image, le label, le bouton rejouer.



2. Les 26 boutons seront fait à l'aide de codeC# (dans le code behind). Voici un exemple pour un bouton, il suffit de redéfinir toute les propriétés mais en C#. Copiez, testez. Remarque : la variable bouton est placée au-dessus des méthodes pour être globale et pouvoir être utilisée dans toutes les méthodes.

```
public Button unBouton;

public MainWindow()
        {
            InitializeComponent();
            InitialiseBouton();
        }

public void InitialiseBouton()
        {
            unBouton = new Button();
            unBouton.Content = "A";
            unBouton.Width = 20;
            unBouton.Height = 20;
            unBouton.VerticalAlignment = VerticalAlignment.Top;
            unBouton.HorizontalAlignment = HorizontalAlignment.Left;
            unBouton.Margin = new Thickness(10, 10, 0, 0);
            this.grille.Children.Add(unBouton);
            Grid.SetColumn(unBouton, 1);
            // ici il est placé dans la 2eme colonne da ma grille
            unBouton.Click += this.UnBouton_Click;
        }
private void UnBouton_Click(object sender, RoutedEventArgs e)
        {
           // on récupère le bouton qui a été cliqué
           Button bouton = ((Button)sender);
           //on le rend inutilisable par la suite
           bouton.IsEnabled = false;
           //on récupère la lettre du bouton sous forme d'un caractère
           char lettre = bouton.Content.ToString()[0];
        }
```

3. Généralisez le principe, modifiez le code ci-dessus pour ne plus initialiser un bouton mais les 26 boutons.

   Remplacez

   ```
   Button unBouton;
   ```

   Par

   ```
   Button [ ] lesBoutons = new Button[26];
   ```

   Remplacez `InitialiseBouton` par `InitialisesLesBoutons`

   Faites les modifications nécessaires.

ETAPE 2 : LOGIQUE DU JEU

1. Ajoutez en variables globales (au-dessus du main) toutes les variables utiles au jeu (les mêmes qu'à la séance 5)

```
public partial class MainWindow : Window
    {
        public Button[] lesBoutons;
        public char[] motAdeviner;
        public char[] motCrypte;
        public int nbEssais = 0, nbErreurs = 0;
        public bool gagne = false;

        public MainWindow()
        {
            InitializeComponent();
            motAdeviner = new char[] {'C','H','I','E','N'} ;
```

2. Puis utilisez les méthodes : **initialiseMotCrypte, ChercheEtRemplaceLettre et AGagne** (de la séance 5) afin de faire fonctionner votre jeu.

3. Ajoutez une nouvelle fenêtre « `SaisieMotAdeviner` » à votre projet pour faire une boite de dialogue pour récupérer le mot défini par le joueur 1. Instanciez cette fenêtre dans le constructeur du MainWindow et affichez-la :

```
SaisieMotAdeviner fenetre = new SaisieMotAdeviner();
fenetre.ShowDialog();
if ( fenetre.DialogResult == true)
   motAdeviner = fenetre.txtMotAdeviner.ToString().ToCharArray();
```

   Remarque : vous devrez dans cette fenêtre vérifier que le joueur fait bien une saisie et dans ce cas il faut mettre à jour la propriété **DialogResult à true. Ainsi la main window pourra décider de prendre la valeur ou de prendre un mot pas défaut.**

4. Prévoyez un mode de tirage aléatoire parmi plusieurs mots prédéfinis.