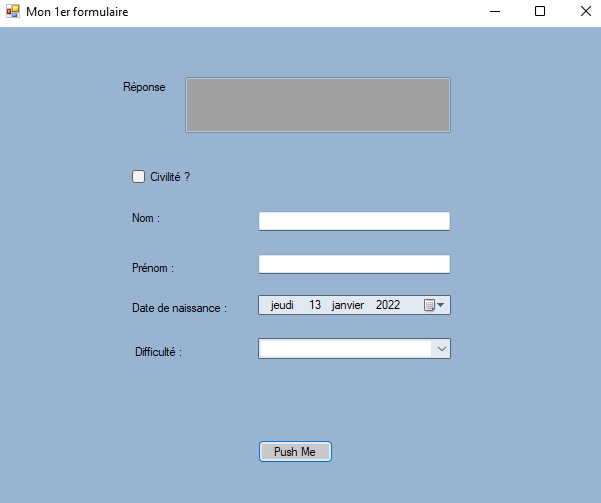
Prise en main de Visual Studio suite

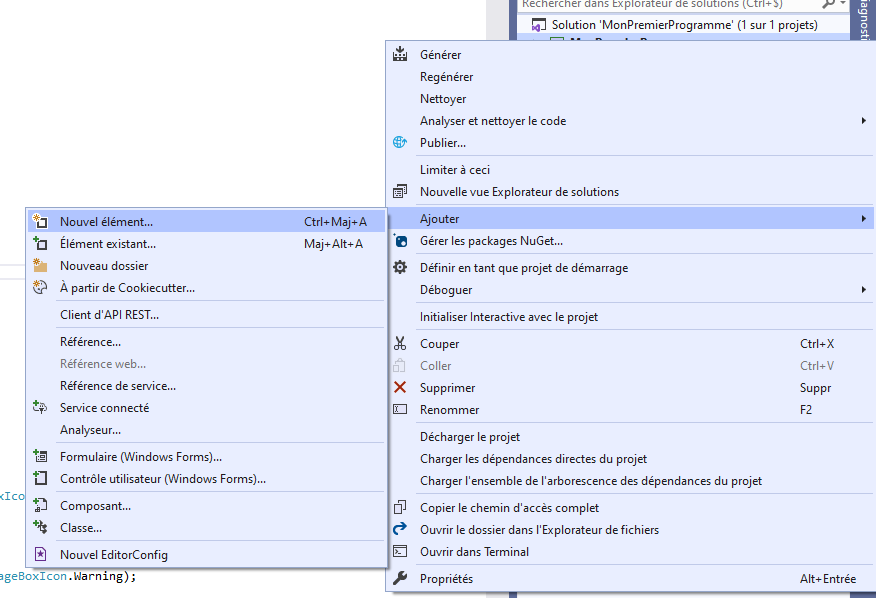
## Partie 1 : Découverte de nouveaux composants et créations de forms

### Phase 1 : Reprise du précédent projet et création d’un nouveau form

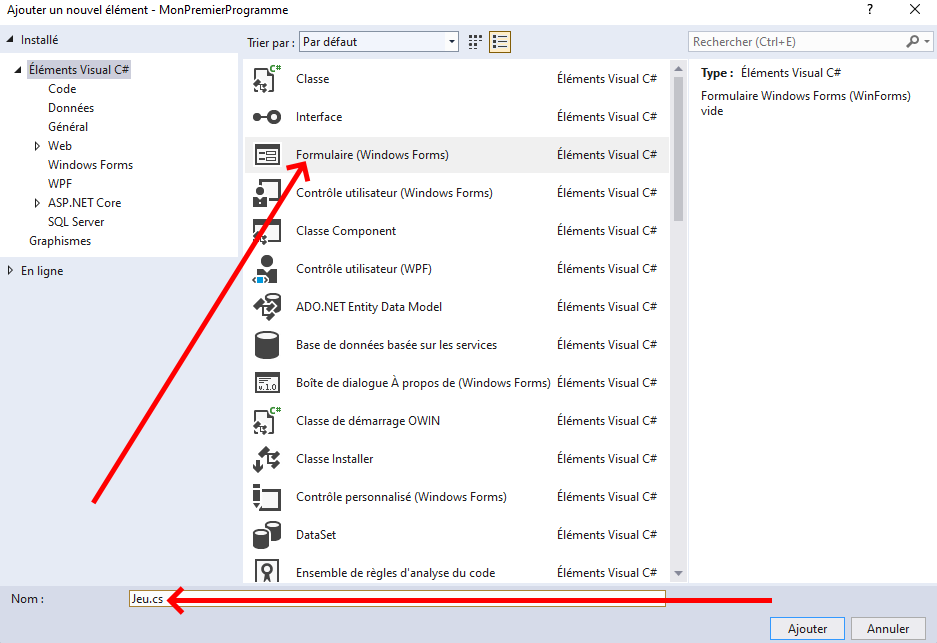
Il vous faut reprendre le projet précédent, il doit être terminé complétement.



Maintenant nous allons créer un nouveau Form, pour cela il faut cliquez droit sur le nom de votre projet et choisir : Ajouter > Nouvel élément

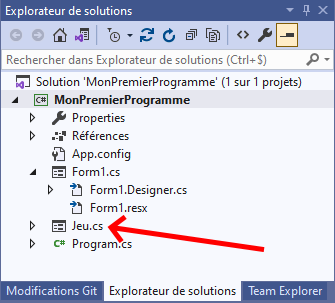


Il faut ensuite choisir WindowForm



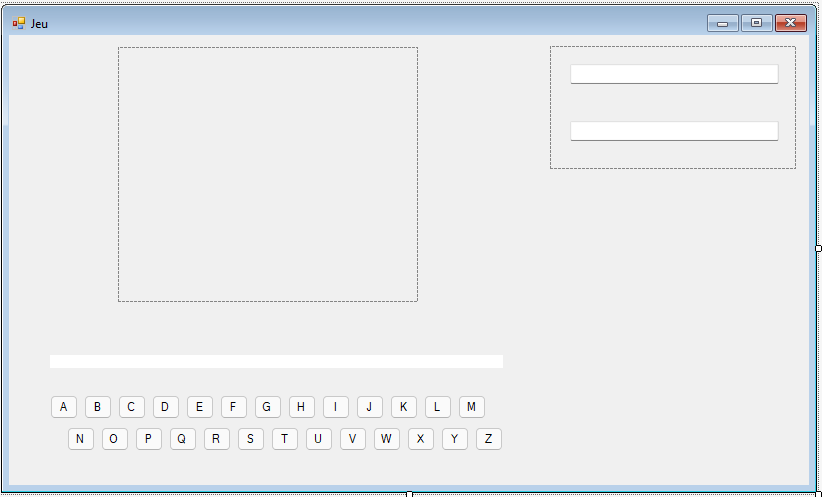
**N’oubliez pas de changer le nom de votre formulaire !! Puis cliquez sur Ajouter**

Normalement le nouveau formulaire devrait apparaitre dans votre explorateur de solution.

****

En double cliquant sur votre nouveau formulaire vous pourrez accéder au design

Reproduire l’affichage suivant :

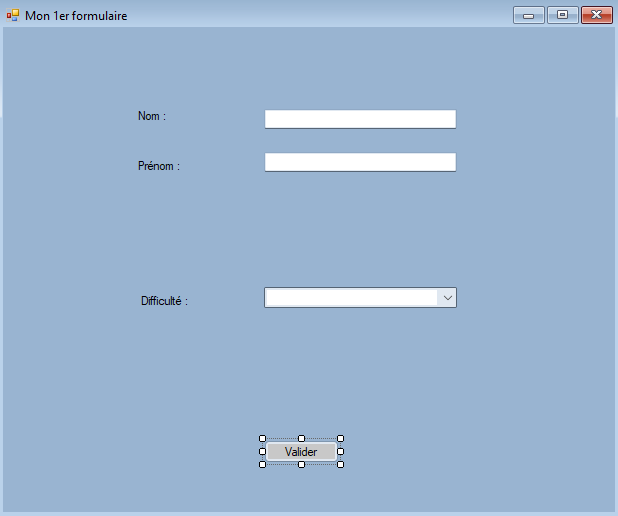


**N’oubliez pas de bien nommer vos boutons, les 3 textbox, ainsi que la picturebox et le panel.**

Les textbox ne doivent pas pouvoir être cliquable.

### Phase 2 : Modification de l’ancien form

Modifiez le Form1 pour qu’il est cet aspect



Des erreurs dans votre code devrais apparaitre, corrigez-les !!

Codez le contrôle de contenu des champs Nom, Prénom et Difficulté lors de l’appuis sur le bouton valider.

Pour contrôler qu’un combobox à une valeur sélectionnée, on utilise le code suivant <combobox>. SelectedIndex > -1

### Phase 3 : Appel d’un form à un autre

Une fois tous les contrôles réalisés, vous pouvez faire appel à votre 2éme Form en utilisant le code suivant :

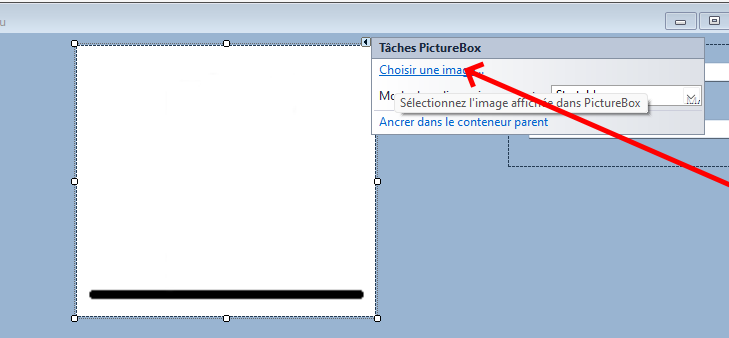
Jeu J = new Jeu();

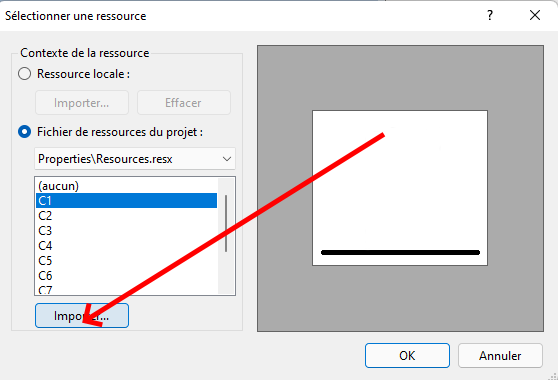
J.Show();

this.Hide();

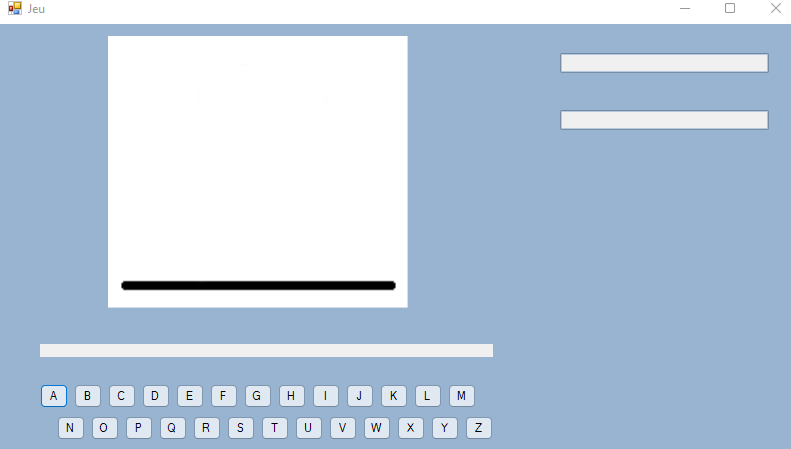
### Phase 4 : Affichage de l’image 1 dans le pictureBox

Pour ajouter les images il vous suffit de les importer





Et sélectionnez l’ensemble des images du projet. Puis une fois les images importées, reproduisez l’écran suivant.



Maintenant nous allons réaliser une fonction permettant de changer l’image en fonction d’un nombre.

Pour déclarer la fonction on utilise le code suivant

public static void ChangerIMG(Int32 nbcmpt,PictureBox pb\_pendu)

{ }

Et pour changer l’image on utilise le code suivant

pb\_pendu.Image = Properties.Resources.C1;

pb\_pendu.Refresh();

### Phase 5 : Coder le pendu

Nous allons déclarer 3 variables global, une pour stocker le mot à trouver, une pour le mot à afficher et une pour le compteur. Pour cela on utilise le code suivant :

public static string motatrouver = "";



Créer les 2 autres variables

Nous allons créer une nouvelle fonction qui permettra de vérifier si la lettre que l’on a choisie est présent dans le mot à trouver.

Pour cela créez une nouvelle fonction qui prend en paramètre une lettre. Appelez cette fonction vérification.

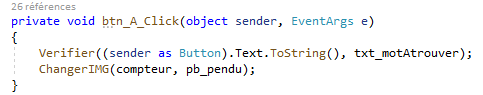
public static void Verifier(String lettretape,TextBox txt\_afficher)

Dans cette fonction, on vérifie que la lettre que l’on a tapée est présente dans le mot à trouver.

Si la lettre est présente, on met à jour le mot à afficher

Si la lettre n’est présente, on met à jour le compteur et l’image

Pour faire appel au 2 fonctions dans le bouton A on ajoute le code suivant :



Lier cet évènement sur le clic de tous les boutons. Vous pouvez désormais tester si votre code marche en affectant une valeur à motatrouver et en mettant le même nombre de \_ que de lettres à trouver dans motaafficher.

### Phase 6 : Génération du mot à afficher

Pour cela nous allons créer une fonction permettant de générer le mot à afficher en fonction du mot à trouver.

Pour cela il faut créer une fonction permettant de retourner une chaine de caractère ne contenant que le caractère « \_ » du même nombre que le nombre de lettre que le paramètre envoyé (une chaine de caractère). Pour cela on utilise le code suivant pour déclarer la fonction.

public string genererMotAfficher(String mottrouver)

{

Pour utiliser cette fonction, on utilise le code suivant :

motaafficher = genererMotAfficher(motatrouver);

### Phase 7 : Génération du mot à trouver

Pour le moment nous allons utiliser une liste de mot pour générer de manière aléatoire le mot à trouver.

Pour générer une liste, il faut utiliser le code suivant :

var listeMotaTrouver = new List<string> { "Francophile", "Chlorophylle", "Conspirateur", "Qualification", "Attraction", "Cornemuse", "Tourisme", "Diapason", "Brouhaha" };

Maintenant il faut réaliser une fonction permettant de sélectionner un mot de manière aléatoire à partir d’une liste et de renvoyer. Pour ajouter une liste obligatoire dans les paramètres, on utilise le code suivant :

List<String> listeATrouver

Pour générer un nombre aléatoire, on utilise le code suivant :

Random aleatoire = new Random();

int nbAleatoire = aleatoire.Next(listeATrouver.Count);

Pour sélectionner un mot dans la liste on utilise le code suivant :

string motchoisi = listeATrouver[nbAleatoire].ToUpper();

### Phase 8 : Rendre inutilisable les boutons qui ont déjà été appuyé

Pour cela vous devez ajouter une ligne de code à votre événement btn\_A\_Click

### Phase 9 : Gérer les conditions de victoire et de défaite

Pour cela nous allons vérifier à la fin du code dans l’événement btn\_A\_Click si le nombre maximum d’essai sans trouver le mot est atteint. Si le joueur gagne, un message apparait et propose de rejouer une partie. Si le joueur perd un message apparait montrant la solution et propose de rejouer une partie. Pour cela on utilise le code suivant :

DialogResult msg;

msg = MessageBox.Show("Bravo, vous avez gagné !! \r\nVous avez trouver le mot: " + motatrouver + "\r\nVoulez vous faire une autre partie ??",

"You win",

MessageBoxButtons.YesNo,

MessageBoxIcon.None,

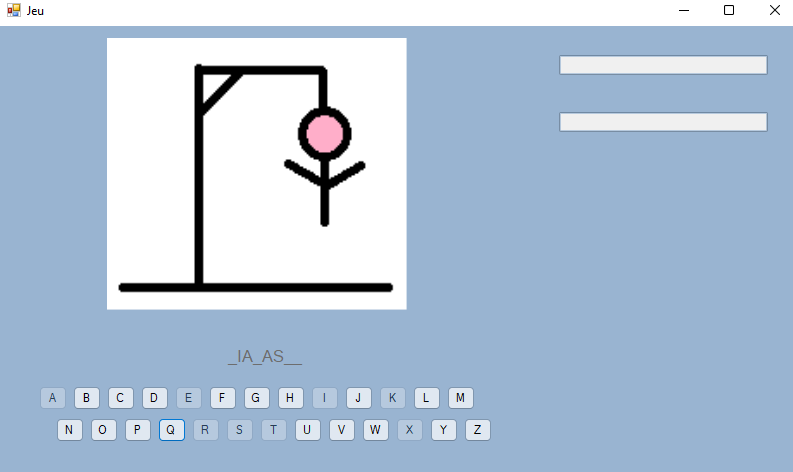
MessageBoxDefaultButton.Button1);

Pour contrôler le résultat du choix de l’utilisateur, on utilise le code suivant :

msg == DialogResult. Yes

Pour faire une nouvelle partie, il faut remettre à l’état initial l’écran de jeu ainsi que toutes les variables. Pour cela, il est conseillé de créer une fonction permettant de réaliser cela.

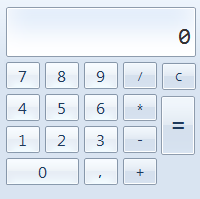
Une fois fini, le résultat devrait être le suivant :



N’oubliez pas de garder le projet une fois terminé nous le réutiliseront plus tard

# Partie 2 : Travail en autonomie

Vous devez réaliser une calculatrice ayant le visuel suivant :



Fonction :

La touche C sert à remettre la zone d'affichage à zéro.

La touche égale doit pouvoir donner le résultat.