|  |  |
| --- | --- |
| Résultat de recherche d'images pour "menu mdi c#"  Dossier de programmation N°2  Menu MDI | Résumé  Développement d’un menu MDI afin d’accueillir une création et gestion d’un joueur    Saint Adjutor, Vernon |

**Sommaire :**

1. Présentation du dossier
   1. Contexte
   2. Liste des compétences
2. Structures de données
   1. Structure globale
   2. Classes
3. Algorithmes utilisés
   1. Menu global
   2. Nouveau joueur
   3. Voir Joueur
   4. Modifier le mail
   5. Modifier l’avatar
   6. Modifier le niveau
   7. Modifier les points
4. Jeu d’essai
   1. Les différents essais effectués
5. Conclusion
   1. Conclusion globale
   2. Améliorations possibles
6. Présentation du dossier
   1. Présentation globale

Le dossier de programmation qui nous a été confié par la société BABYLONE, une société de développement spécialisé dans les logiciels de jeux et qui est désireuse de crée une classe Joueur. Celle-ci permettra d’intégrer à n’importe quel jeu, un ou plusieurs joueurs créent à partir de cette classe notamment pour des jeux de plateformes à plusieurs niveaux.

Ce projet sera à développer en langage C# en développant un projet en Windows Form comprenant un menu MDI (Multiple Document Interface) et toutes les formes nécessaires pour satisfaire le besoin du client. Elle devra également être composée de 2 classes : Globale et Joueur dont la structure nous a été donnée en pseudo-code.

Cet interface devra être capable de :

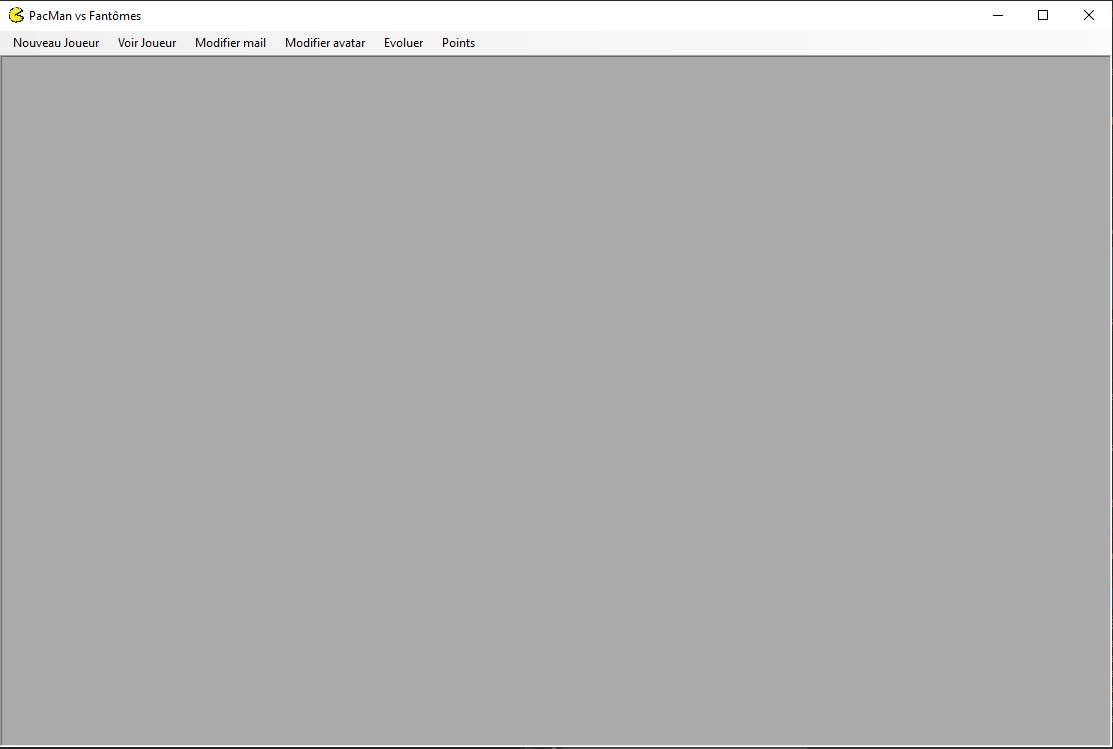
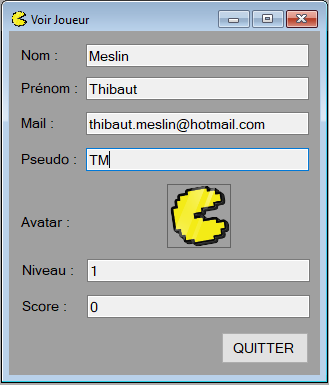
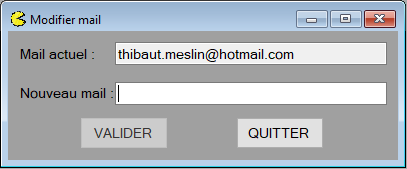
* Instancier un nouveau joueur de la classe Joueur en définissant les données de bases nécessaires à son instanciation (nom, prénom, mail, pseudonyme et avatar). Le type de donnée devra être vérifié et l’adresse mail du type : [identification@serveur.ext](mailto:identification@serveur.ext)
* Voir les informations connus sur le Joueur
* Changer le mail (en gardant les critères de validité ci-dessus)
* Changer l’avatar
* Evoluer de niveau via une interface sans dépasser le niveau maximum
* Changer son nombre de points et si celui-ci dépasse le plafond du niveau passer au suivant automatiquement
  1. Liste des compétences

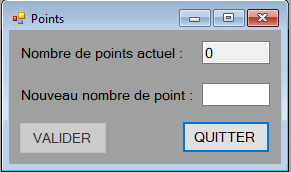
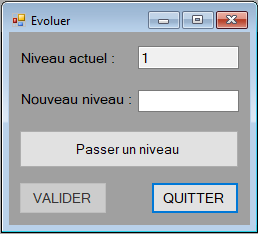
Les compétences associées sont les suivantes :

* A4.1.6 Gestion d’environnements de développement et de test
* A4.1.7 Développement, utilisation et adaptation de composants logiciels
* A4.1.8 Réalisation des tests nécessaires à la validation d’éléments adaptés ou développés
* A4.1.9 Rédaction d’une documentation technique

1. Structures de données
   1. Structure globale

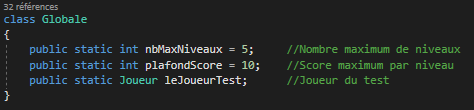
Mon projet se compose des deux classes traduites en langage C# et de 7 form, une servant de menu principale étant donc la form PARENT des autres. Chacune des autres form ont une fonction spécifique dans celles listées au-dessus. Le menu principal est un conteneur MDI possédant une barre de navigation supérieure permettant d’ouvrir les autres menus.



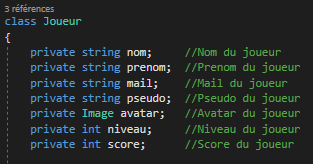


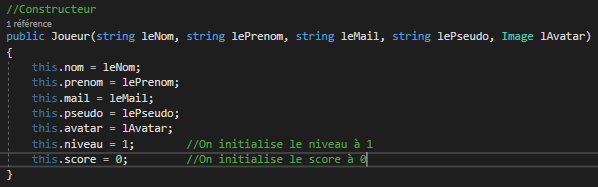
* 1. Structure globale

Mon projet dispose de 2 classes une classe Globale contenant le Joueur utilisé pour le test et les variables pour le nombre de niveau et le score maximum déclarer en static integer.



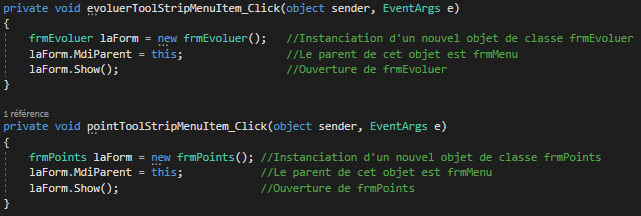
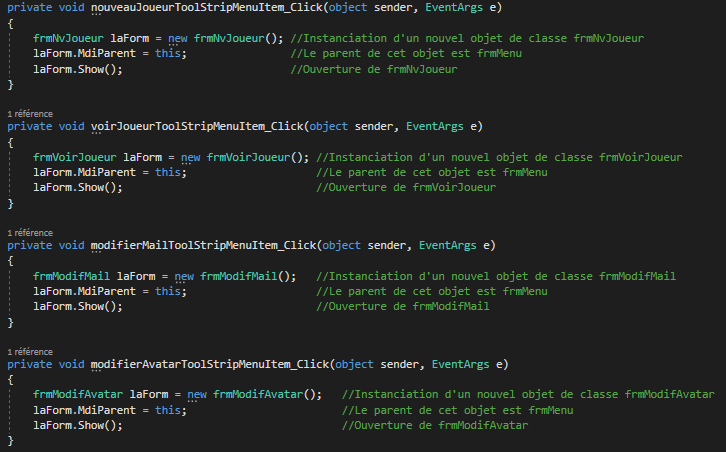
La seconde classe est la classe Joueur servant de base à l’instanciation des futurs joueurs du jeu et contient donc toutes les informations sur ceux-ci (nom, prénom, mail, …).



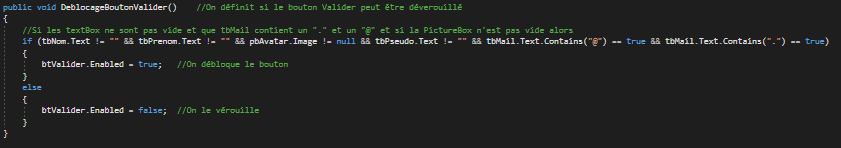
 Son constructeur avait des informations à respecter. Le joueur devait voir son niveau instancié à 1 et son score instancié lui à 0. L’avatar choisi par le joueur devait-être stocké dans l’objet créé par cette classe, il est donc présent ici reprenant une structure de classe Image.

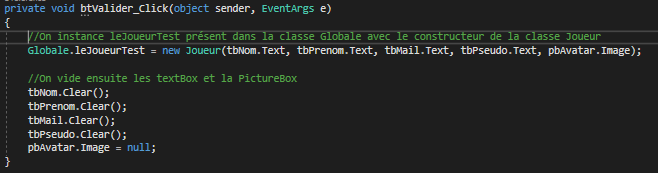
1. Algorithmes utilisés
   1. Menu global

Le menu global est assez simple et est seulement présent afin d’être le PARENT des autres form, c’est-à-dire de les contenir. Celui-ci avant de devenir un menu MDI était simplement une form. Afin de le faire devenir un menu MDI, la propriété « isMdiContainer » activé, c’est-à-dire isMdiContainer = true.

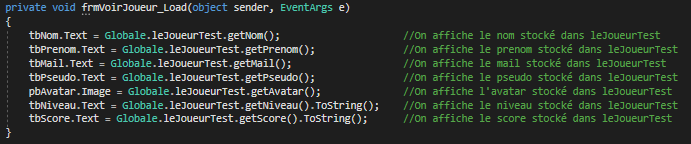
Le menu comme son nom l’indique permet d’ouvrir les autres form qui lui sont rattachées et lors de l’appui sur un bouton du menu, de les ouvrir. Pour cela il faut instancier un nouvel objet de la form désirée, définir le form du menu comme la form PARENT de celle-ci et ensuite on a simplement à ouvrir l’instance crée plus tôt.

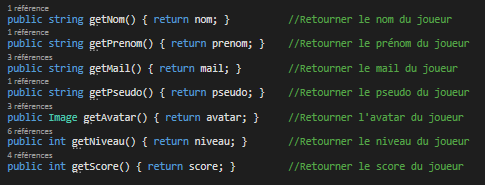
* 1. Nouveau joueur

Afin de créer un nouveau joueur, ici on instanciera seulement le joueur test présent dans la classe Globale, il faut instancier un nouvel objet de la classe Joueur. Mais avant cela le cahier des charges précises que les informations doivent toutes être présentes et que l’adresse mail doit suivre certaines contraintes définies plus haut. Pour cela, la méthode DeblocageBoutonValider a été créée. Elle sera recopiée et adaptée dans toutes les form pour vérifier qu’une information a été rentrée dans l’élément nécessaire. Ici on vérifie simplement que les textbox ne sont pas vident grâce à un « != « » » et si le mail contient un point et un arobase grâce à un .Contain(« . ») et un .Contain(« @ »). On appelle ensuite la méthode à chaque évènement Text\_Changed afin de vérifier chaque saisie de la part du joueur.

Ensuite on instancie leJoueurTest de la classe Globale nous servant de base à notre test afin de vérifier la validité de nos actions et de nos algorithmes. On y stocke les informations rentrées dans les textbox par le joueur et l’image entrée de son pc (cf. 3.5 Modifier l’avatar pour le code d’ouverture du fichier).

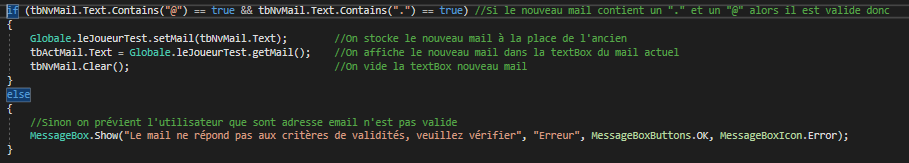
* 1. Voir Joueur

Le code de cette form est très simple. En effet le seul besoin de cette form est d’afficher les informations du joueur, or ici nous ne traitons que du joueur test de la classe Globale donc pas besoin de comboBox et de gestion d’index ici. Pour afficher les informations nous avons seulement à appeler nos getters / accesseurs présent dans la classe Joueur à l’objet leJoueurTest et à assigner les informations relevées à la textbox correspondante ou bien à la PictureBox pour l’image.

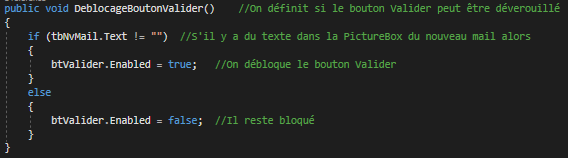


* 1. Modifier le mail

Modifier le mail requière de vérifier une nouvelle fois la validité de l’adresse mail en entré afin de voir si elle répond aux critères de validité fournis par la société BABYLONE. Pour cela on réutilise le .Contain() et on y réaffecte le point et l’arobase. Ensuite on fait appel à un setter / mutateur de la classe Joueur afin de remplacer l’ancienne adresse par celle entrée maintenant puis on met à jour l’affichage afin de montrer la nouvelle adresse dans le champ « mail actuel ». Si toute fois le mail entré ne répondait pas aux critères, on informera l’utilisateur via une messageBox et remettrai à vide la textBox d’entrée.

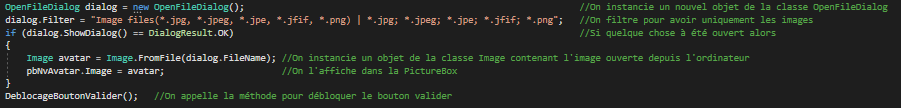


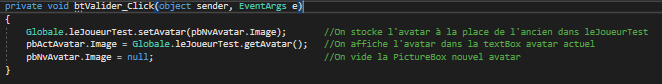
Pour le déblocage du bouton valider, on teste simplement s’il y a quelque chose dans la zone d’écriture.





* 1. Modifier l’avatar

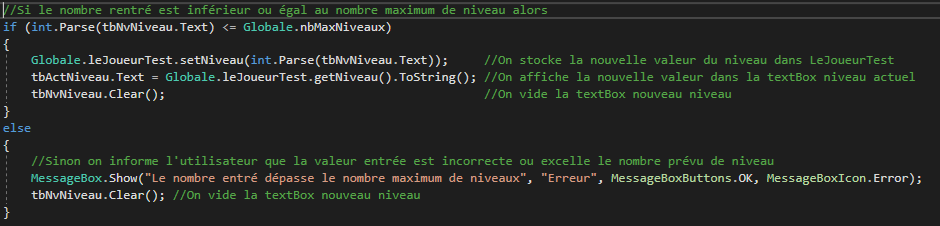
Pour modifier l’avatar on reprend l’algorithme de la création d’un nouveau joueur pour l’ouverture du fichier. On va ouvrir par le clic sur le bouton « avatar » une page de l’ordinateur du joueur afin de lui faire ouvrir une image. Une restriction est appliquée pour que seuls les fichiers images tels que jpeg ou png soit sélectionnable et donc ne pas créer une erreur non voulue. Ensuite si quelque chose a été ouvert par l’utilisateur alors on instancie cet élément via la classe image afin de pouvoir la sauvegarder dans l’instance d’objet du joueur test. Ensuite on utilise simplement un setter / mutateur de la classe Joueur afin de remplacer l’ancien avatar par le nouveau.

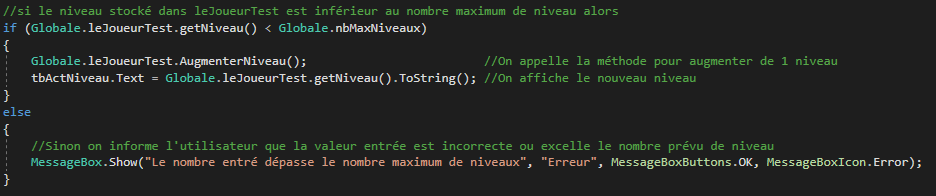




* 1. Modifier le niveau

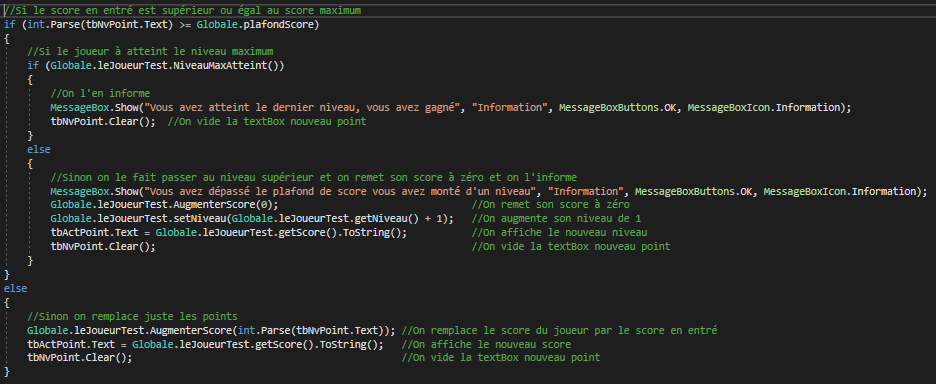
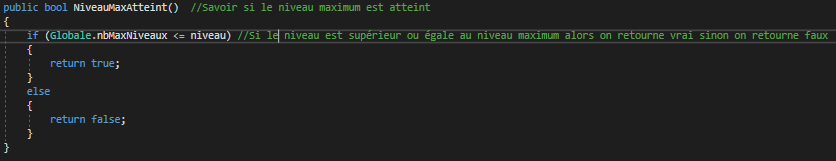
Avant de modifier le niveau il faut définir le nombre de niveau que le jeu possède. Pour cela on modifie la valeur de la variable « nbMaxNiveau » dans la classe Globale. Pour la phase de test je l’ai initialisé à 5.

Ensuite pour enfin modifier le niveau, il y a plusieurs choses à respecter. D’abord le niveau saisi par l’utilisateur ne doit pas excéder le nombre maximum de niveau contenus dans le jeu et ensuite un bouton a été créé pour passer directement au niveau supérieur en faisant appel à la méthode de classe Joueur AugmenterNiveau(), celle-ci incrémente simplement le niveau. Ensuite pour le choix du niveau, on fait appel à la méthode de classe Joueur setNiveau().



* 1. Modifier les points

Avant de modifier le nombre de points, il faut comme pour le niveau définir le plafond de score dans la classe Globale définie par la variable « plafondScore ». Ici pour le test le plafond est à 10 points.

Ensuite pour modifier les points quelques précisions s’imposent. Il est précisé que lorsqu’un nombre de points supérieur au plafond sera entré le jeu passera automatiquement au niveau suivant en remettant son score à 0. Puis vient également le cas lorsque le jeu a atteint son niveau final et donc ne peut plus augmenter en niveau. Pour le premier cas, on compare la valeur entrée au plafond si celle-ci est supérieure ou égale alors on incrémente le niveau et remet le score à 0. Pour le second, si le niveau dépasse le nombre de niveau on affiche un message pour dire que le jeu est terminé. Sinon on change juste via un setter / mutateur le nombre de point de l’instance.

1. Jeu d’essai

|  |  |  |  |
| --- | --- | --- | --- |
| **Numéro de test** | **Scénario** | **Description** | **Résultat** |
| 1 | Ouverture | Ouverture de frmMenu | Succès |
| 2 | Fermeture | Fermeture de frmMenu | Succès |
| 3 | Ouverture | Ouverture de frmNvJoueur | Succès |
| 4 | Fermeture | Fermeture de frmNvJoueur | Succès |
| 5 | Ouverture | Ouverture de frmVoirJoueur | Echec |
| 6 | Ouverture | Ouverture de frmVoirJoueur | Succès |
| 7 | Affichage | Affichage des données de frmNvJoueur dans frmVoirJoueur | Succès |
| 8 | Fermeture | Fermeture de frmVoirJoueur | Succès |
| 9 | Ouverture | Ouverture de frmModifMail | Succès |
| 10 | Affichage | Affichage du mail stocké dans leJoueurTest | Succès |
| 11 | Modification | Modification du mail stocké dans leJoueurTest | Succès |
| 12 | Fermeture | Fermeture de frmModifMail | Succès |
| 13 | Ouverture | Ouverture de frmModifAvatar | Succès |
| 14 | Affichage | Affichage de l’avatar stocké dans leJoueurTest | Succès |
| 15 | Modification | Modification de l’avatar stocké dans leJoueurTest | Succès |
| 16 | Fermeture | Fermeture de frmModifAvatar | Succès |

* 1. Les différents essais effectués

|  |  |  |  |
| --- | --- | --- | --- |
| **Numéro de test** | **Scénario** | **Description** | **Résultat** |
| 17 | Ouverture | Ouverture de frmEvoluer | Succès |
| 18 | Affichage | Affichage du niveau stocké dans leJoueurTest | Succès |
| 19 | Modification | Modification du niveau stocké dans leJoueurTest | Echec |
| 20 | Modification | Modification du niveau stocké dans leJoueurTest | Echec |
| 21 | Modification | Modification du niveau stocké dans leJoueurTest | Succès |
| 22 | Fermeture | Fermeture de frmEvoluer | Succès |
| 23 | Ouverture | Ouverture de frmPoints | Succès |
| 24 | Affichage | Affichage du score stocké dans leJoueurTest | Succès |
| 25 | Modification | Modification du score stocké dans leJoueurTest | Echec |
| 26 | Modification | Modification du score stocké dans leJoueurTest | Succès |
| 27 | Fermeture | Fermeture de frmPoints | Succès |
| 28 | Test Finale | Vérification de la totalité du programme | Succès |

1. Conclusion
   1. Conclusion globale

Pour conclure ce dossier de programmation, je dirai que le projet était très utile car il peut être utilisé dans une infinité de contexte différents et que chacun trouvera une application qui lui est propre et trouvera moyen de le modifier pour arriver à lui donner la forme et les fonctionnalités qu’il veut. Ce dossier étaient assez facile mais plutôt long à mettre en place mais dans les temps imposé cela restait tout de même très correct et largement suffisant (3 semaines). Je dirais que ce projet m’a apporté beaucoup de choses notamment le savoir de faire ce genre de menu qui comme je l’ai dit est très pratique et modifiable à notre gré.

* 1. Améliorations possibles

Ce projet peut être décliné de nombreuses façons et dans de nombreux domaines et pourrait avoir :

* Un tableau de joueur afin d’en gérer un plus grand nombre
* Des comboBox afin de gérer l’affichage des éléments du tableau ci-dessus
* Implémenter un jeu de plateforme pour y jouer
* Automatiser les points afin qu’ils augmentent et donne de la difficulté aux joueurs
* Ajouter de nouveaux ennemis en fonction du niveau
* Ajouter de nouvelles structures en fonction du niveau
* …