

Atelier Programmation

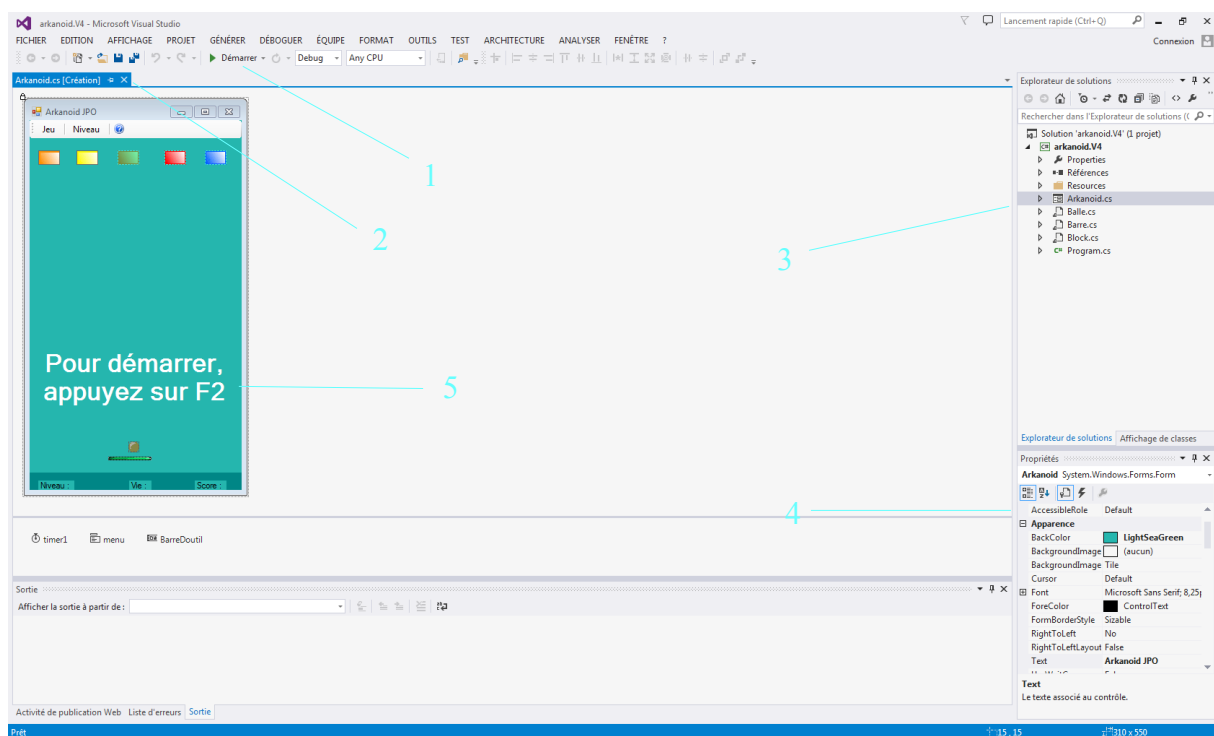
Jeu : Arkanoid

En cas de problème, n'hésitez pas à demander de l'aide !

Arkanoid est un jeu simple de casse-briques : une raquette, une balle et des briques à casser. En le découvrant, vous verrez qu'on peut y apporter quelques améliorations, mais aussi qu'il y a quelques problèmes à corriger. Votre mission aujourd'hui sera de le rendre meilleur, et ainsi comprendre le fonctionnement de la programmation. Pour cela, nous utiliserons Microsoft Visual Studio, un logiciel de développement pour Windows.

Voici l'environnement sous lequel vous allez passer les 20 prochaines minutes :

Vue globale de Visual Studio



1. Compiler et exécuter le programme
2. Onglets des différents fichiers
3. Explorateur des différents fichiers
4. Propriétés de l'objet sélectionné
5. Aperçu du programme

1. Pour débiter

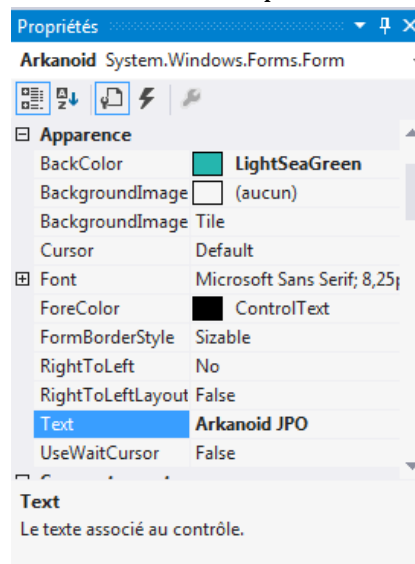
Tout d'abord, vous allez exécuter le programme : cliquez sur la flèche verte en haut ou appuyez sur F5, et jouez un peu avec le programme pour le découvrir. Pour lancer le jeu, allez dans le menu « jeu » ou appuyez sur F2. Pour relancer la balle après la perte d'un point de vie, pressez la barre d'espace ou F2.

Une fois que vous vous serez familiarisé avec le jeu, quittez-le et revenez à la fenêtre de Visual Studio. Si la vue centrale est vide, double-cliquez sur Arkanoid.cs dans l'explorateur de fichiers pour accéder au mode Design.

Modification des informations de la fenêtre

Modifiez le titre de la fenêtre du programme : sélectionnez la fenêtre principale par un simple clic dessus, puis dans la fenêtre de propriétés (voir ci-dessous), cherchez l'attribut `Text` et modifiez son champ d'origine « Arkanoid » (voir l'image ci-dessous). Vous pouvez aussi prendre connaissance des autres possibilités de modification.

La fenêtre Propriétés

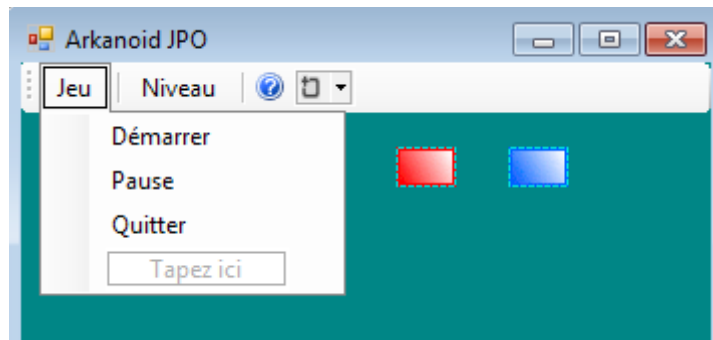


Changer la couleur de fond de jeu

Toujours en restant dans la fenêtre « Propriétés », cherchez l'attribut `BackColor` et choisissez une couleur parmi celles qui vous sont proposées dans le menu déroulant.

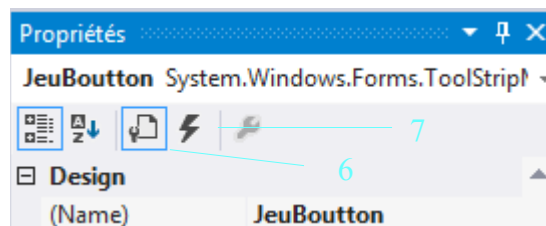
Nouvel élément de barre d'outils

Ajoutez un nouvel item « Quitter » dans le menu « Jeu » : Pour cela, cliquez sur le bouton « Jeu » directement dans la fenêtre du jeu à gauche, puis sur « Tapez ici » pour rajouter un item à ce menu déroulant. Ecrivez « Quitter », puis validez avec la touche Entrée (voir l'image ci-dessous).



2. Implémentations et modifications

Dans cette partie, nous allons regarder le code source lui-même, pour que les nouveaux objets aient des effets dans notre programme, et pour corriger certains points.



Implémentation du « Quitter »

Lorsque vous exécutez le programme, vous remarquerez que votre item Quitter n'a aucun effet. Pour remédier à cela, vous devrez écrire du code : fermez la fenêtre de jeu, puis cliquez sur « Jeu » et sur « Quitter » pour le sélectionner dans la fenêtre d'aperçu du jeu. Dans la fenêtre « Propriétés » qui désigne maintenant les propriétés de l'item « Quitter », allez dans l'onglet Evénements, symbolisé par un éclair (voir ci-dessus, bouton 7) pour voir les différents événements possibles avec l'objet sélectionné. Double-cliquez sur l'événement Click, le logiciel vous place alors immédiatement au milieu du code qu'il aura généré.

Ce code gère donc l'événement Click de « Quitter », c'est-à-dire le cas où l'utilisateur clique sur le bouton « Quitter ».

Ne déplacez pas le curseur, écrivez simplement `this.Dispose(true);`.

```
1 référence
private void quitterToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Dispose(true);
}
```

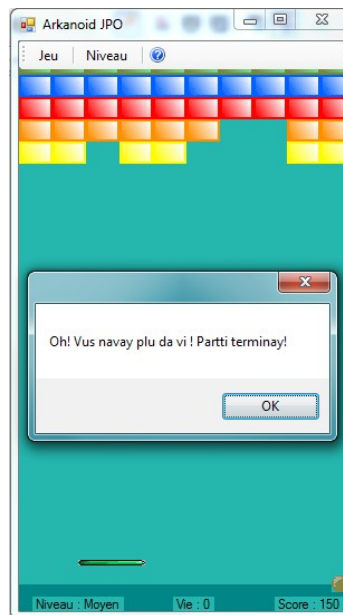
Exécutez le programme pour vérifier que votre programme fonctionne correctement. Après la vérification, n'oubliez pas de remettre l'affichage de la fenêtre Propriétés sur l'onglet Propriétés (voir ci-dessus, bouton 6).

Indice :

Pour revenir à l'affichage de l'aperçu du programme, sélectionnez l'onglet « Arkanoid.cs [Design] » , dans la barre des onglets de fichiers.

Corriger une faute

Fenêtre de jeu



Lorsque la balle touche la partie inférieure de la fenêtre, vous perdez une vie. Lorsque vous n'avez plus de vies, une nouvelle fenêtre apparaît. Le texte de cette fenêtre comporte des fautes d'orthographe que vous allez corriger : recherchez la fonction `perdu()` dans le fichier « Arkanoid.cs » (utilisez les onglets en haut de la fenêtre). Modifiez la fonction pour corriger les fautes.

```
// lorsque on a plus de vie
1 référence
private void perdu()
{
    MessageBox.Show("Oh! Vus navay plu da vi ! Partti terminay!");
    timer1.Enabled = false;    // le mouvement de la balle est arrêté
    musiquePerdu();
}
```

Ajouter une musique

On souhaite avoir une musique lorsqu'il y a une défaite du joueur. Pour cela, il faut que la fonction `perdu()` appelle la fonction `musiquePerdu()` , qui s'occupera de jouer le son.

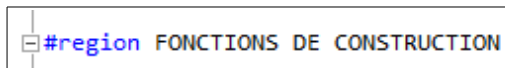
Indice :

Par exemple, pour appeler la fonction `toto()` dans la fonction `truc()`, on écrirait :

```
private void truc()  
{  
    toto() ;  
}
```

Changer le style des blocs

Ici, on veut changer un peu l'apparence des blocs. Si vous essayez de le modifier dans la fenêtre Propriétés de l'aperçu avec l'attribut `BorderStyle`, les changements ne prendront pas effet à l'exécution. Pour cela, il faut modifier le constructeur du jeu en revenant dans le code du « `Arkanoid.cs` », dans les fonctions de construction. Il ne vous reste plus qu'à changer le style « `None` » par « `FixedSingle` » ou « `Fixed3D` ».



Indice :

Aidez-vous des commentaires pour comprendre les lignes.

Gérer la pause

Vous avez remarqué qu'on prend le risque de perdre en mettant le jeu en pause en passant par la barre d'outils. Vous allez ajouter un raccourci clavier pour mettre en pause et reprendre le jeu, par exemple la touche « `P` ».

```
// si on presse la touche p on gère la pause/reprise du jeu ( pour decommenter retirez "/* et */" )  
/* if (e.KeyCode == Keys.P && etat_du_jeu == Etat.JOUE)  
    else if (e.KeyCode == Keys.P && etat_du_jeu == Etat.RELANCE)*/
```

Indice :

Retrouvez la fonction `appelPause` déjà écrite pour savoir l'appeler. Inspirez-vous des fonctions dans « `GESTION DES ACTIONS CLAVIER – SOURIS` ».

Diminuer la vitesse de la balle

Pour diminuer la difficulté du jeu, il est possible de modifier la vitesse de la balle.

Modifiez les valeurs de déplacement de la balle afin de la rendre plus lente.

Indice :

Les valeurs se trouvent dans le fichier propre à la balle.

3. Pour aller plus loin

Améliorer l'affichage

Vous aurez sûrement remarqué, en essayant le programme, que les blocs verts sont en partie cachés par la barre d'outils. Le problème vient quand on les construit : `Block(ordonnée i, ordonnée j, taille block, taille barre d'outil, ...)`. Ce constructeur est appelé dans une des fonctions de construction, à vous de mettre les bonnes valeurs.

Indice :

La taille de la barre d'outils vaut `BarreDoutil.Size.Height`.

Les blocs incassables

Pour corser le jeu, ainsi que l'atelier, nous allons rendre certains blocs indestructibles. Pour ce faire, nous allons modifier les fichiers *Block.cs*, *Arkanoid.cs* et *Balle.cs*.

Commencez par ajouter un booléen dans le fichier *Block.cs*, après la ligne `class Block : PictureBox {` qui permettra de savoir si un bloc est indestructible ou pas : ajoutez la ligne `bool estIndestructible = false;`. En effet, par défaut, tout bloc est destructible.

Ensuite, afin d'accéder à cette variable en lecture et en écriture, nous devons définir des accesseurs :

```
public void setVar(bool v) { var = v; }    // accesseur en écriture
public bool getVar() { return var; }       // accesseur en lecture
```

Maintenant que la variable est mise en place et accessible, nous allons pouvoir l'utiliser. Dans le fichier *Arkanoid.cs*, nous allons définir les blocs indestructibles à l'aide de l'accesseur en écriture. Faites appel à cette fonction à la fin du constructeur de block.

Indice :

Pour appeler une fonction sur un bloc de coordonnées (5, 4), il faut faire :

```
Blocks[5][4].fonction(paramètres);
```

Dans *Balle.cs*, la fonction `toucheBlock()` fait rebondir la balle si le bloc est visible. Par conséquent, si on veut qu'un bloc soit indestructible, il faut qu'il soit tout le temps visible. Vous allez insérer à la fin de cette fonction une condition, qui, si un bloc est indestructible et invisible, le remettra visible.

Indice :

Voici comment s'écrit une condition :

```
// si la condition1 et la condition2
// sont vérifiées toutes les deux
if( condition1 && condition2 )
{
    Action1; // alors effectuer l'action1
}
```

Inspirez-vous du reste de la fonction pour écrire cette portion de code.