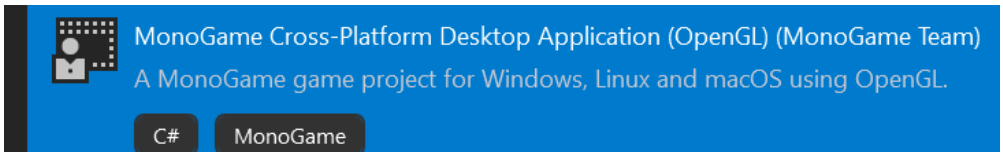
TD12 - INTRO MONOGAME – 6H

OBJECTIFS

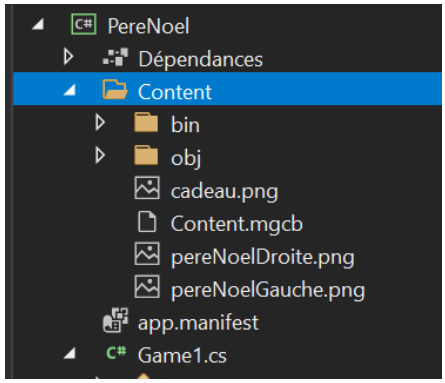
- Découvrir monogame

EXO 1 :

1. Créez un projet « TD12_IntroMonogame » dans une solution de même nom :



2. Lancez l'exécution.



3. Placez les images fournies sur le réseau dans le répertoire Content de votre projet

4. Puis ajoutez-la au fichier de configuration des contenus (images, sons…) :
   a. Clic droit sur le fichier « Content.mgcb » puis ouvrir avec « mgcb-editor-wpf » à chercher sur le c : de la machine.
   b. Menu Add Existing Item : sélectionner vos images :
   c. Actionner « Build »
   d. Sauvegarder et fermer l'éditeur

5. Créez des textures à partir de ces images :
   a. Au début de votre classe Game1.cs ajoutez 2 champs privés :
      - _texturePereNoel de classe Texture2D
      - _positionPereNoel de classe Vector2

   b. Ajoutez 2 constantes :
      - `public const int LARGEUR_PERE_NOEL = 200 ;`
      - `public const int HAUTEUR_PERE_NOEL = 154 ;`

   c. Au sein de la méthode LoadContent : ajoutez le code pour charger les images :

```
protected override void LoadContent()
    {
      _spriteBatch = new SpriteBatch(GraphicsDevice);
      // TODO: use this.Content to load your game content here
      _texturePereNoel = Content.Load<Texture2D>("pereNoelDroite");
    }
```

d. Au sein de la méthode Initialize : initialisez les positions :

```
protected override void Initialize()
 {
  // TODO: Add your initialization logic here
  // GraphicsDevice.Viewport.Height permet d'avoir la hauteur de la fenêtre
  _positionPereNoel = new Vector2(0, GraphicsDevice.Viewport.Height - HAUTEUR_PERE_NOEL);
   base.Initialize();
 }
```

e. Au sein de la méthode Draw : ajoutez le code pour dessiner les textures à l'aide des positions précédemment définies :

```
protected override void Draw(GameTime gameTime)
{
  GraphicsDevice.Clear(Color.CornflowerBlue);
  // TODO: Add your drawing code here
_spriteBatch.Begin();
_spriteBatch.Draw(_texturePereNoel, _positionPereNoel  , Color.White);
 _spriteBatch.End();
  base.Draw(gameTime);
 }
```

6. Reproduisez ces actions pour positionner le cadeau en haut de fenêtre et en x aléatoire.

7. Au sein de la méthode Initialize, changez la taille de la fenêtre de jeu : 800 par 800 : utilisez une constante : TAILLE_FENETRE

```
_graphics.PreferredBackBufferWidth = TAILLE_FENETRE;
_graphics.PreferredBackBufferHeight = TAILLE_FENETRE;
_graphics.ApplyChanges();
```

8. Au sein de la méthode Draw, changez la couleur de fond : `MidnightBlue`

9. Modifiez le positionnement du père noël pour qu'il soit au centre en bas de fenêtre.

10. Ajoutez un champ pour gérer le clavier :

```
private KeyboardState _keyboardState;
```

11. Ajoutez 2 champs pour gérer le sens et la grandeur du pas de déplacement :

```
private int _sensPereNoel;
private int _pasPereNoel; // à initialiser à 100 dans la méthode Initialize
```

12. Dans la méthode Update, ajoutez le code pour faire un déplacement en fonction des touches directionnelles. Testez. Puis complétez pour que le déplacement ne se fasse qu'une fois par appui de touche, et qu'il fonctionne dans les 2 sens.

```
// deltaTime stocke le temps écoulé entre 2 tous de boucles
// permet de rationaliser le deplacement en pixel en fonction du temps écoulé
float deltaTime = (float)gameTime.ElapsedGameTime.TotalSeconds;
 _keyboardState = Keyboard.GetState();

// si fleche droite
 if (_keyboardState.IsKeyDown(Keys.Right))
       _sens = 1;
_positionPereNoel.X +=    _sensPereNoel * _pasPereNoel * deltaTime;
```

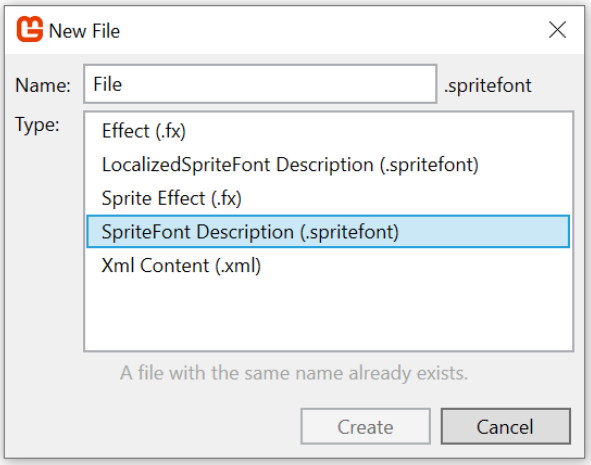
13. Dans la méthode Update, ajoutez le code pour faire tomber le cadeau jusqu'au sol, utilisez une variable `_pasCadeau` à initialiser `dans Initialize`.

14. Une fois tombé et entièrement disparu, un autre cadeau doit tomber à une autre position aléatoire en X.

15. Le père noël doit essayer de récupérer un maximum de cadeau. A chaque fois, qu'il en récupère un, cela doit augmenter le score : Faite en sorte que la descente du cadeau s'arrête sur le père noël (et non sur le bas de la fenêtre)

    a. Définissez 3 champs :
        - un champ _score de type int : pensez à l'initialiser à 0 dans Initialize et à l'incrémenter dans Update à chaque collision (cadeau/père noel) : soit vous déterminez vous-même si collision il y a , soit vous créez 2 rectangles avec les coordonnées et les tailles des textures pour utiliser Intersects
        - un champ _textScore de type SpriteFont pour afficher le score.
        - un champ _positionScore de type Vector2 : pensez à l'initialiser à 0,0 dans Initialize

    b. Ajoutez un fichier .spritefont pour définir la font à utiliser. Pour cela, ouvrez content.mgcb, ajoutez un nouvel élément et choisissez « SpriteFont Description) renommez tout de suite dans la fenêtre ci-dessous File en Font . Vous aurez un nouveau fichier dans votre projet, ouvrez-le : changez la taille « 24 » et le style en « bold »



    c. Dans la méthode LoadContent, chargez la police :
```
_textScore = Content.Load<SpriteFont>("Font");
```

    d. Dans la méthode Draw, dessinez le composant SpriteFont:
```
_spriteBatch.DrawString(_textScore, $"Score : {_score}", _positionScore, Color.White);
```

16. Ajoutez un chrono et affichez le en haut à droite : définissez un champ _chrono de type float à initialiser à 60 dans Initialize. Dans Update, décrémentez le temps écoulé, utilisez le deltaTime défini précédemment. Attention : faites l'arrondi uniquement pour l'affichage . La partie s'arrête, se fige quand le chrono arrive à 0.

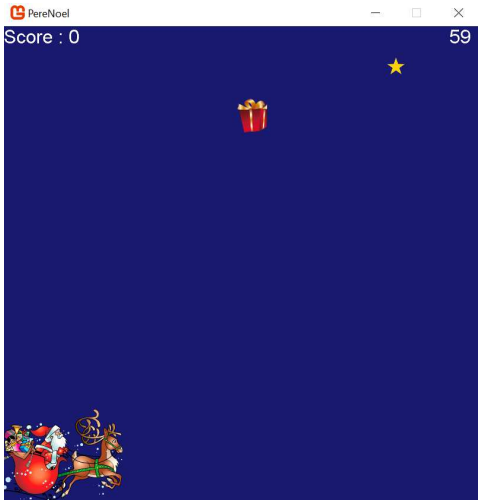
17. Affichez pendant 2 secondes une étoile positionnée aléatoirement et cela toutes les 10 secondes. Si le joueur arrive à la toucher avec un clic de souris, cela doit accélérer le jeu. Pour cela :

    a. Ajoutez un compteur de temps, dès que vous avez atteint ou dépassé 10 secondes, affichez une étoile, et remettez le compteur à 0. Dès que vous avez atteint ou dépassé 2 secondes, rendez l'étoile non visible. (position en dehors de l'écran)

    b. En cas de clic sur l'étoile , accélérez le pas de 50 pixels (créez un champ statique accélération à 50) pour le cadeau et le père noël. Pour gérer le clic : définissez un champ _mouseState de classe MouseState puis initialisez dans la méthode Update et testez l'état de la souris. Ex à adapter :

```
_mouseState = Mouse.GetState();
if ( _mouseState.LeftButton == ButtonState.Pressed)
      Console.WriteLine ( _mouseState.X + " " +_mouseState.Y) ;
```

    Vous pouvez utiliser la méthode Contains sur un rectangle défini avec la position et taille de l'étoile.



18. Ajoutez la possibilité de rejouer ! Posez la question et déclenchez Initialize si l'utilisateur veut rejouer, quitter l'appli sinon.

19. Ajoutez une touche pause à l'appui de la touche espace !

20. Améliorez en faisant descendre plusieurs cadeaux à la fois : utiliser un tableau de cadeaux.

21. Ajoutez les sons à votre jeu.

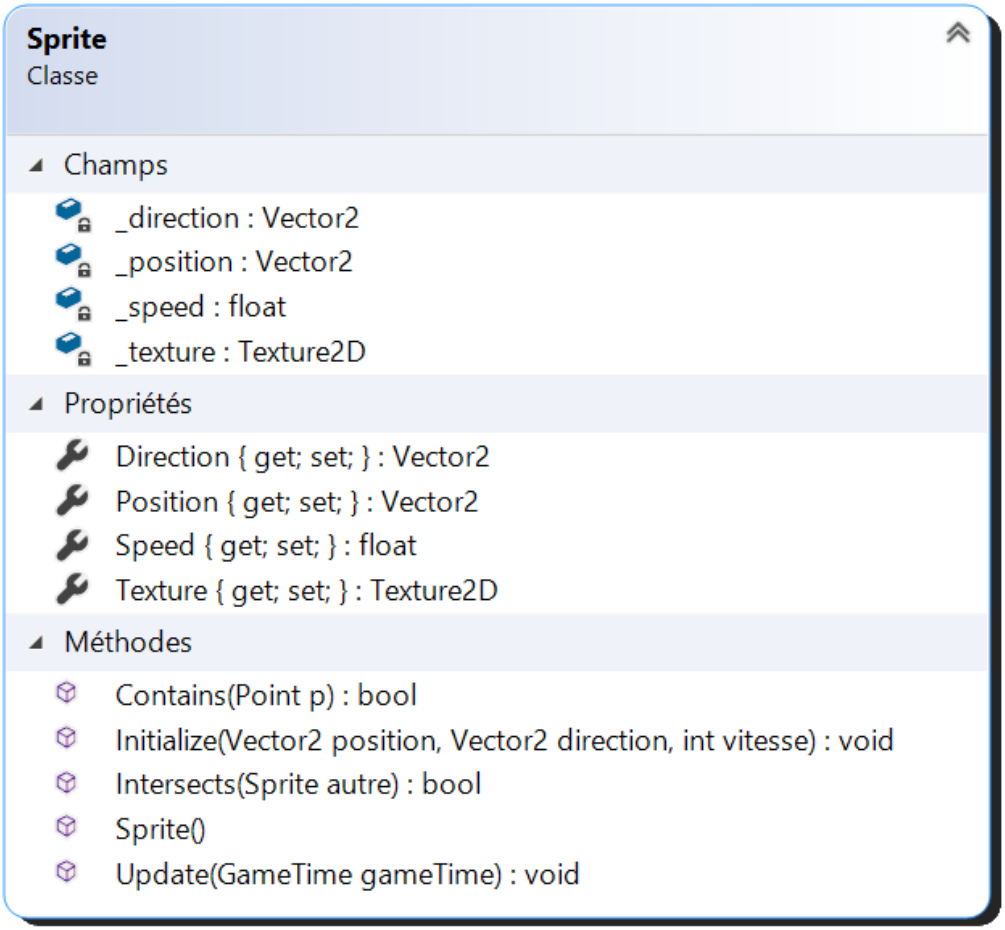
PARTIE 2 :

Ajoutez un projet à votre solution. On va reprendre le même sujet mais cette fois, on va re-travailler le code pour factoriser et organiser (Vous pouvez éventuellement changer de thème et d'images) vous définirez la classe Sprite ci-dessous :
Elle vous sera utile pour instancier les 2 sprites dans Game1.cs : _pereNoel et _cadeau

Elle regroupe la position et la texture, mais aussi la vitesse (ainsi on pourra éventuellement la différencier en fonction du sprite) mais aussi la direction qui sera un vecteur afin de gérer un déplacement en x et en y à la fois (mis à (0,0) si aucun mouvement n'est à faire)

Les méthodes Intersects et Contains de la classe Sprite s'appuieront sur les méthodes Intersects et Contains de Rectangle (Rappel : il faut instancier des rectangles à partir des positions et tailles des textures)

Remarque : le constructeur ne fait rien et c'est Initialize qui initialise réellement les champs pour coller au modèle de monogame : seule la texture ne sera pas initialisée car elle doit l'être dans la méthode LoadContent




```
public  void Update(GameTime gameTime)
{
 _position += _direction * _speed * (float)gameTime.ElapsedGameTime.TotalSeconds;

}
```

Il est possible d'ajouter encore la méthode Draw dans la classe Sprite :

```
public void Draw(SpriteBatch spriteBatch)
        {
            spriteBatch.Draw(_texture, _position, Color.White);
        }
```

Ainsi au lieu d'écrire dans Game1.cs

```
_spriteBatch.Draw(_cadeau.Texture, _cadeau.Position, Color.White);
```

Il suffira d'écrire :

```
_cadeau.Draw(_spriteBatch) ;
```

Vous pourrez aussi regrouper les variables utiles pour afficher le score et le chrono grâce à la classe LabelAvecValeur : c'est une classe conçue pour afficher un texte suivi d'une variable de type entier. Pensez à définir alors 2 LabelAvecValeur dans Game1.cs (attention, ne chargez qu'une seule fois le Spritefont)

Le ToString ici permet de concaténer la valeur au texte et de l'appeler au moment de l'utilisation de la méthode DrawString