TPI 2020 - FlappySharp

AndrÃľ Gouveia de Oliveira
I.DA-P4A
ÃLcole d'informatique (CFPT-I)

08 Juin 2020

Table des matières

0.0.1 frmMain.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappySharp
{
    public partial class frmMain : Form
    {
        Jeu jeu;
        Dictionary<string, Bitmap> images;
        Sprite _spriteSelected;
        int zOrder;

        public frmMain()
        {
            InitializeComponent();
            DoubleBuffered = true;

            jeu = new Jeu();
        }

        private void pbxAjoutFlappy_Click(object sender, EventArgs e)
        {
            Dictionary<string, Bitmap> images = new Dictionary<string, Bitmap>();
            images.Add("Flappy_1.png", new Bitmap(Properties.Resources.Flappy_1));
            images.Add("Flappy_2.png", new Bitmap(Properties.Resources.Flappy_2));
            images.Add("Flappy_3.png", new Bitmap(Properties.Resources.Flappy_3));
            jeu.AddSprite("Flappy", new Size(50, 50), new Point(0, 0), images, 1, spSceneParam.Panel1);
        }

        private void pbxAjoutFond_Click(object sender, EventArgs e)
        {
            Dictionary<string, Bitmap> images = new Dictionary<string, Bitmap>();
            images.Add("Fond.jpg", new Bitmap(Properties.Resources.Fond));
            jeu.AddSprite("Fond", new Size(100, 100), new Point(0, 0), images, 1, spSceneParam.Panel1);
        }

        private void pbxAjoutSol_Click(object sender, EventArgs e)
        {
            Dictionary<string, Bitmap> images = new Dictionary<string, Bitmap>();
            images.Add("Sol.jpg", new Bitmap(Properties.Resources.Sol));
            jeu.AddSprite("Sol", new Size(100, 100), new Point(0, 0), images, 1, spSceneParam.Panel1);
        }

        private void pbxAjoutTuyau_Click(object sender, EventArgs e)
        {
            Dictionary<string, Bitmap> images = new Dictionary<string, Bitmap>();
            images.Add("Tuyau.jpg", new Bitmap(Properties.Resources.Tuyau));
            jeu.AddSprite("Tuyau", new Size(100, 100), new Point(0, 0), images, 1, spSceneParam.Panel1);
        }
```

```
58
59          private void pbxAjoutSprite_Click(object sender, EventArgs e)
60          {
61              frmAjoutSprite frmAjout = new frmAjoutSprite();
62
63              if (frmAjout.ShowDialog() == DialogResult.OK)
64              {
65                  jeu.AddSprite(frmAjout.GetNom(), frmAjout.GetTaille(), ↩
                        frmAjout.GetPosition(), frmAjout.GetImages(), ↩
                        frmAjout.GetCalque(), spSceneParam.Panel1);
66              }
67          }
68
69          private void btnUpZOrder_Click(object sender, EventArgs e)
70          {
71              zOrder = -1;
72          }
73
74          private void btnDownZOrder_Click(object sender, EventArgs e)
75          {
76              zOrder = 1;
77          }
78
79          private void btnSauveModif_Click(object sender, EventArgs e)
80          {
81              jeu.UpdateValueSpriteSelected(tbxNom.Text, new ↩
                    Size(Convert.ToInt32(tbxLargeur.Text), ↩
                    Convert.ToInt32(tbxHauteur.Text)), new ↩
                    Point(Convert.ToInt32(tbxPosX.Text), ↩
                    Convert.ToInt32(tbxPosY.Text)), images, ↩
                    Convert.ToInt32(tbxIntervalImage.Text), ↩
                    (int)nudCalque.Value, zOrder, cbxTag.Text, ↩
                    (int)nudRotation.Value);
82              zOrder = 0;
83          }
84
85          private void btnSupr_Click(object sender, EventArgs e)
86          {
87              _spriteSelected.SuprControlPanel(true);
88              jeu.Sprites.Remove(_spriteSelected);
89              tbxNom.Text = "";
90              nudCalque.Value = 0;
91              tbxHauteur.Text = "";
92              tbxLargeur.Text = "";
93              tbxPosX.Text = "";
94              tbxPosY.Text = "";
95              lbxImages.Items.Clear();
96              tbxIntervalImage.Text = "";
97              cbxTag.Text = "";
98              nudRotation.Value = 0;
99              btnSupr.Enabled = false;
100         }
101
102         private void tmp_Tick(object sender, EventArgs e)
103         {
104             if (_spriteSelected != jeu.GetValueSpriteSelected())
105             {
106                 _spriteSelected = jeu.GetValueSpriteSelected();
107                 tbxNom.Text = _spriteSelected.Name;
108                 nudCalque.Value = _spriteSelected.Calque;
109                 tbxHauteur.Text = _spriteSelected.Height.ToString();
110                 tbxLargeur.Text = _spriteSelected.Width.ToString();
111                 tbxPosX.Text = _spriteSelected.Location.X.ToString();
112                 tbxPosY.Text = _spriteSelected.Location.Y.ToString();
113                 lbxImages.Items.Clear();
114                 images = _spriteSelected.Images;
115                 foreach (var image in _spriteSelected.Images.Keys)
116                 {
117                     lbxImages.Items.Add(image);
118                 }
119                 tbxIntervalImage.Text = ↩
                        _spriteSelected.IntervalEntreImage.ToString();
```

```
120                 cbxTag.Text = _spriteSelected.TagSprite;
121                 nudRotation.Value = _spriteSelected.AngleRotation;
122                 btnSupr.Enabled = true;
123             }
124 
125             if (tbxNom.Text == null || tbxHauteur.Text == null || ↩
                    tbxLargeur.Text == null || tbxPosX.Text == null || ↩
                    tbxPosY.Text == null || lbxImages.Items.Count == 0 || ↩
                    tbxIntervalImage.Text == null || cbxTag.Text == null)
126             {
127                 btnSauveModif.Enabled = false;
128             }
129             else
130             {
131                 btnSauveModif.Enabled = true;
132             }
133         }
134 
135         private void runToolStripMenuItem_Click(object sender, ↩
                EventArgs e)
136         {
137             frmPlateauJeu plateauJeu = new frmPlateauJeu();
138             plateauJeu.Sprites = jeu.Sprites;
139             foreach (var sprite in plateauJeu.Sprites)
140             {
141                 sprite.ActiverDesactiverEvenement(false);
142             }
143 
144             plateauJeu.ShowDialog();
145             foreach (var sprite in plateauJeu.Sprites)
146             {
147                 sprite.ActiverDesactiverEvenement(true);
148             }
149             jeu.RefreshControl();
150         }
151 
152         private void enregistrerToolStripMenuItem_Click(object sender, ↩
                EventArgs e)
153         {
154             frmCreationProjet creationProjet = new frmCreationProjet();
155             if (jeu.Sprites.Count != 0)
156             {
157                 if (jeu.NomProjet == null)
158                 {
159                     if (creationProjet.ShowDialog() == DialogResult.OK)
160                     {
161                         jeu.CreationDossierProjet(creationProjet.GetCheminDossier(), ↩
                                creationProjet.GetNomProjet());
162                     }
163                 }
164 
165                 if (jeu.CheminDossierProjet != null)
166                 {
167                     jeu.XMLSerialize();
168                 }
169             }
170             else
171             {
172                 MessageBox.Show("Il␣faut␣avoir␣des␣Sprites␣sur␣la␣scène␣↩
                        pour␣sauvegarde␣le␣projet");
173             }
174         }
175 
176         private void ouvrirToolStripMenuItem_Click(object sender, ↩
                EventArgs e)
177         {
178             if (ofdXml.ShowDialog() == DialogResult.OK)
179             {
180                 jeu.CreateSpriteAfterDeserialize(spSceneParam.Panel1, ↩
                        ofdXml.FileName);
181             }
182         }
```

```
183
184         private void quitterToolStripMenuItem_Click(object sender, ↩
                EventArgs e)
185         {
186             this.Close();
187         }
188     }
189 }
```

Listing 1 – ./FlappySharp/frmMain.cs

0.0.2 frmAjoutSprite.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappySharp
{
    public partial class frmAjoutSprite : Form
    {
        Dictionary<string, Bitmap> _images;
        Point _position;
        Size _taille;
        string _nom;
        int _calque;

        public frmAjoutSprite()
        {
            InitializeComponent();
            _images = new Dictionary<string, Bitmap>();
            _position = new Point(0, 0);
            _taille = new Size(0, 0);
        }

        private void tbxName_TextChanged(object sender, EventArgs e)
        {
            if (tbxName.Text != string.Empty)
                _nom = tbxName.Text;
            VerifieValeurVide();
        }

        private void tbxPosX_TextChanged(object sender, EventArgs e)
        {
            if (tbxPosX.Text != string.Empty)
                _position.X = Convert.ToInt32(tbxPosX.Text);
            VerifieValeurVide();
        }

        private void tbxPosY_TextChanged(object sender, EventArgs e)
        {
            if (tbxPosY.Text != string.Empty)
                _position.Y = Convert.ToInt32(tbxPosY.Text);
            VerifieValeurVide();
        }

        private void tbxHauteur_TextChanged(object sender, EventArgs e)
        {
            if (tbxHauteur.Text != string.Empty)
                _taille.Height = Convert.ToInt32(tbxHauteur.Text);
            VerifieValeurVide();
        }

        private void tbxLargeur_TextChanged(object sender, EventArgs e)
        {
            if (tbxLargeur.Text != string.Empty)
                _taille.Width = Convert.ToInt32(tbxLargeur.Text);
            VerifieValeurVide();
        }

        private void tbxCalque_TextChanged(object sender, EventArgs e)
        {
            if (tbxCalque.Text != string.Empty)
                _calque = Convert.ToInt32(tbxCalque.Text);
            VerifieValeurVide();
```

```
70          }
71
72          // trouver ce bout de code sur ↩
               https://www.aspsnippets.com/Articles/Windows-Forms-WinForms-OpenFileDialo
73          private void btnAjoutImage_Click(object sender, EventArgs e)
74          {
75              if (lfd.ShowDialog() == DialogResult.OK)
76              {
77                  foreach (var fichier in lfd.FileNames)
78                  {
79                      _images.Add(Path.GetFileName(fichier), new ↩
                           Bitmap(fichier));
80                      lbxImage.Items.Add(Path.GetFileName(fichier));
81                  }
82              }
83              VerifieValeurVide();
84          }
85
86          private void VerifieNombre(object sender, KeyPressEventArgs e)
87          {
88              if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
89              {
90                  e.Handled = true;
91              }
92          }
93
94          private void VerifieValeurVide()
95          {
96              if (tbxName.Text == string.Empty || tbxPosX.Text == ↩
                   string.Empty || tbxPosY.Text == string.Empty || ↩
                   tbxLargeur.Text == string.Empty || tbxHauteur.Text == ↩
                   string.Empty || tbxCalque.Text == string.Empty || ↩
                   lbxImage.Items.Count == 0)
97              {
98                  btnOk.Enabled = false;
99              }
100             else
101             {
102                 btnOk.Enabled = true;
103             }
104         }
105
106         public string GetNom()
107         {
108             return _nom;
109         }
110
111         public int GetCalque()
112         {
113             return _calque;
114         }
115
116         public Point GetPosition()
117         {
118             return _position;
119         }
120
121         public Size GetTaille()
122         {
123             return _taille;
124         }
125
126         public Dictionary<string,Bitmap> GetImages()
127         {
128             return _images;
129         }
130     }
131 }
```

Listing 2 – ./FlappySharp/frmAjoutSprite.cs

0.0.3 frmCreationProjet.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.NetworkInformation;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappySharp
{
    public partial class frmCreationProjet : Form
    {
        public frmCreationProjet()
        {
            InitializeComponent();
        }

        private void TextChanged(object sender, EventArgs e)
        {
            if (tbxCheminDossier.Text != string.Empty && ↩
                tbxNomProjet.Text != string.Empty)
            {
                btnOk.Enabled = true;
            }
            else
            {
                btnOk.Enabled = false;
            }
        }

        private void btnCheminDossier_Click(object sender, EventArgs e)
        {
            if (fbdDossierProjet.ShowDialog() == DialogResult.OK)
            {
                tbxCheminDossier.Text = fbdDossierProjet.SelectedPath;
            }
        }

        public string GetNomProjet()
        {
            return tbxNomProjet.Text;
        }

        public string GetCheminDossier()
        {
            return tbxCheminDossier.Text;
        }
    }
}
```

Listing 3 – ./FlappySharp/frmCreationProjet.cs

0.0.4 frmPlateauJeu.cs

```
ïż£using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappySharp
{
    public partial class frmPlateauJeu : Form
    {
        List<Sprite> _sprites;
        public List<Sprite> Sprites { get => _sprites; set => _sprites = value; }

        public frmPlateauJeu()
        {
            InitializeComponent();
        }

        private void frmPlateauJeu_Load(object sender, EventArgs e)
        {
            foreach (var sprite in Sprites)
            {
                sprite.AjoutControlPlateauJeu(this);
                sprite.DemareAnimation();
            }
            tmp.Enabled = true;
        }

        private void tmp_Tick(object sender, EventArgs e)
        {
            foreach (var sprite in Sprites)
            {
                sprite.Deplacement();
            }
            CheckCollision();
            lblFps.Text = "FPS : " + (1000 / tmp.Interval);
        }

        private void frmPlateauJeu_FormClosing(object sender, FormClosingEventArgs e)
        {
            tmp.Enabled = false;
            foreach (var sprite in Sprites)
            {
                sprite.DemareAnimation();
            }
        }

        private void CheckCollision()
        {
            foreach (var spritePlayer in Sprites)
            {
                if (spritePlayer.TagSprite == "Player")
                {
                    foreach (var spriteCollision in Sprites)
                    {
                        Rectangle rect = Rectangle.Intersect(spriteCollision.Collision, spritePlayer.Collision);
                        if (!rect.IsEmpty && spritePlayer.TagSprite != spriteCollision.TagSprite && spriteCollision.Calque == spritePlayer.Calque)
                        {
                            this.Close();
```

```
64                          }
65                      }
66                  }
67              }
68          }
69
70          private void frmPlateauJeu_KeyDown(object sender, KeyEventArgs e)
71          {
72              if (e.KeyCode == Keys.Up)
73              {
74                  foreach (var sprite in Sprites)
75                  {
76                      if (sprite.TagSprite == "Player")
77                      {
78                          sprite.ChangePositionFinal(80);
79                      }
80                  }
81              }
82          }
83      }
84  }
```

Listing 4 – ./FlappySharp/frmPlateauJeu.cs

0.0.5 Jeu.cs

```
ï»¿using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Serialization;

namespace FlappySharp
{
    [Serializable()]
    class Jeu
    {
        List<Sprite> _sprites;
        List<SpriteSerialisable> _spriteSerialisables;

        Sprite _spriteSelected;

        string _nomProjet;
        string _cheminDossierProjet;

        internal List<Sprite> Sprites { get => _sprites; set => _sprites = value; }
        public string NomProjet { get => _nomProjet; set => _nomProjet = value; }
        public string CheminDossierProjet { get => _cheminDossierProjet; set => _cheminDossierProjet = value; }

        public Jeu()
        {
            Sprites = new List<Sprite>();
            _spriteSerialisables = new List<SpriteSerialisable>();
        }

        public void AddSprite(string nom, Size taille, Point position, Dictionary<string, Bitmap> images, int calque, Panel panelScene)
        {
            int zOrder = 0;

            if (Sprites.Count != 0 && Sprites.Where((sprite) => sprite.Calque == calque).Count() != 0)
            {
                zOrder = Sprites.Where((sprite) => sprite.Calque == calque).OrderByDescending(sprite => sprite.ZOrder).First().ZOrder;
                zOrder++;
            }

            Sprites.Add(new Sprite(CheckNomExist(nom), taille, images, calque, zOrder, position, panelScene));

            RefreshControl();
        }

        public void RefreshControl()
        {
            Sprites.ForEach(sprite => sprite.SuprControlPanel(false));

            Sprites = Sprites.OrderBy(s => s.Calque).ThenBy(s => s.ZOrder).ToList<Sprite>();

            Sprites.ForEach(sprite => sprite.AjoutControlPanel());
        }

        private void ModifZOrder(Sprite spriteZOrderChanger)
        {
```

```
60              if (Sprites.Count != 0)
61              {
62                  foreach (var spiteChangementZOrder in Sprites)
63                  {
64                      if (spriteZOrderChanger.Name != ↩
                            spiteChangementZOrder.Name && ↩
                            spriteZOrderChanger.ZOrder == ↩
                            spiteChangementZOrder.ZOrder && ↩
                            spriteZOrderChanger.Calque == ↩
                            spiteChangementZOrder.Calque)
65                      {
66                          if (spriteZOrderChanger.ModificationZOrder == "+")
67                          {
68                              spiteChangementZOrder.ZOrder -= 1;
69                              spriteZOrderChanger.ModificationZOrder = null;
70                          }
71                          else if (spriteZOrderChanger.ModificationZOrder ↩
                                == "-")
72                          {
73                              spiteChangementZOrder.ZOrder += 1;
74                              spriteZOrderChanger.ModificationZOrder = null;
75                          }
76                      }
77                  }
78              }
79              RefreshControl();
80          }
81
82          public void UpdateValueSpriteSelected(string nom, Size size, ↩
                Point location, Dictionary<string, Bitmap> images, int ↩
                intervalImage, int calque, int zOrder, string tag, int ↩
                rotation)
83          {
84              if (_spriteSelected.Calque == calque)
85              {
86                  if (_spriteSelected.ZOrder < _spriteSelected.ZOrder + ↩
                        zOrder)
87                  {
88                      _spriteSelected.UpdateValue(nom, size, location, ↩
                            images, intervalImage, calque, ↩
                            _spriteSelected.ZOrder + zOrder, tag, rotation);
89                      _spriteSelected.ModificationZOrder = "+";
90                      ModifZOrder(_spriteSelected);
91                  }
92
93                  if (_spriteSelected.ZOrder > _spriteSelected.ZOrder + ↩
                        zOrder)
94                  {
95                      _spriteSelected.UpdateValue(nom, size, location, ↩
                            images, intervalImage, calque, ↩
                            _spriteSelected.ZOrder + zOrder, tag, rotation);
96                      _spriteSelected.ModificationZOrder = "-";
97                      ModifZOrder(_spriteSelected);
98                  }
99
100                 if (_spriteSelected.ZOrder == _spriteSelected.ZOrder + ↩
                        zOrder)
101                 {
102                     _spriteSelected.UpdateValue(nom, size, location, ↩
                            images, intervalImage, calque, zOrder, tag, ↩
                            rotation);
103
104                     RefreshControl();
105                 }
106             }
107             else
108             {
109                 zOrder = 0;
110
111                 if (Sprites.Count != 0 && Sprites.Where((sprite) => ↩
                        sprite.Calque == calque).Count() != 0)
112                 {
```

```
113                     zOrder = Sprites.Where((sprite) => sprite.Calque == ↩
                            calque).OrderByDescending(sprite => ↩
                            sprite.ZOrder).First().ZOrder;
114                     zOrder++;
115                 }
116                 _spriteSelected.UpdateValue(nom, size, location, ↩
                        images, intervalImage, calque, zOrder, tag, rotation);
117 
118                 RefreshControl();
119             }
120         }
121 
122         public Sprite GetValueSpriteSelected()
123         {
124             foreach (var sprite in Sprites)
125             {
126                 if (sprite.Selected && sprite != _spriteSelected)
127                 {
128                     _spriteSelected = sprite;
129                     _spriteSelected.Selected = false;
130                 }
131             }
132 
133             return _spriteSelected;
134         }
135 
136         /// <summary>
137         /// Permet de vérifier si le nom du Sprite qu'on ajoute existe
138         /// </summary>
139         /// <param name="checkNomSprite">Le nom du sprite à ↩
                verifier</param>
140         /// <returns></returns>
141         private string CheckNomExist(string checkNomSprite)
142         {
143             string nomModifier = checkNomSprite;
144             int compteur = 1;
145             if (Sprites.Count != 0)
146             {
147                 while (Sprites.Where((sprite) => sprite.Name == ↩
                        nomModifier).Count() != 0)
148                 {
149                     nomModifier = checkNomSprite + compteur;
150                     compteur++;
151                 }
152             }
153             return nomModifier;
154         }
155 
156         public void CreationDossierProjet(string cheminDossierProjet, ↩
                string nomProjet)
157         {
158             NomProjet = nomProjet;
159             CheminDossierProjet = cheminDossierProjet;
160             Directory.CreateDirectory(Path.Combine(cheminDossierProjet, ↩
                    nomProjet));
161             foreach (var sprite in Sprites)
162             {
163                 sprite.CreationDossier(Path.Combine(cheminDossierProjet, ↩
                        nomProjet));
164             }
165         }
166 
167         public void XMLSerialize()
168         {
169             foreach (var sprite in Sprites)
170             {
171                 SpriteSerialisable spriteSerialise = new ↩
                        SpriteSerialisable();
172 
173                 List<string> nomImages = new List<string>();
174 
175                 foreach (var nomImage in sprite.Images.Keys)
```

```
                {
                    nomImages.Add(nomImage);
                }

                spriteSerialise.SetValue(nomImages, ↩
                    sprite.CheminDossierImage, sprite.Name, ↩
                    sprite.Location, sprite.Size, sprite.Calque, ↩
                    sprite.ZOrder, sprite.IntervalEntreImage, ↩
                    sprite.AngleRotation, sprite.TagSprite);

                _spriteSerialisables.Add(spriteSerialise);
            }

            Stream stream = File.Open(Path.Combine(CheminDossierProjet, ↩
                NomProjet + @"\" + NomProjet + ".xml"), FileMode.Create);
            XmlSerializer formatter = new ↩
                XmlSerializer(typeof(List<SpriteSerialisable>));
            formatter.Serialize(stream, _spriteSerialisables);
            stream.Close();
        }

        public void CreateSpriteAfterDeserialize(Panel panel, string ↩
            cheminXml)
        {
            foreach (var sprite in Sprites)
            {
                sprite.SuprControlPanel(true);
            }
            Dictionary<string, Bitmap> images;
            _spriteSerialisables = XMLDeserialize(cheminXml);
            foreach (var spriteSerialize in _spriteSerialisables)
            {
                _cheminDossierProjet = spriteSerialize.CheminDossier;
                images = new Dictionary<string, Bitmap>();

                foreach (var nomImage in spriteSerialize.NomImages)
                {
                    images.Add(nomImage, new ↩
                        Bitmap(spriteSerialize.CheminDossier + @"\" + spriteSerialize.Nom + ↩
                        @"\" + nomImage));
                }

                Sprites.Add(new Sprite(spriteSerialize.Nom, ↩
                    spriteSerialize.Taille, images, spriteSerialize.Calque, ↩
                    spriteSerialize.ZOrder, spriteSerialize.Position, panel));
            }
            RefreshControl();
        }

        public List<SpriteSerialisable> XMLDeserialize(string cheminXml)
        {
            Stream stream = File.Open(cheminXml, FileMode.Open);
            XmlSerializer formatter = new ↩
                XmlSerializer(typeof(List<SpriteSerialisable>));
            List<SpriteSerialisable> obj = ↩
                (List<SpriteSerialisable>)formatter.Deserialize(stream);
            stream.Close();
            return obj;
        }
    }
}
```

Listing 5 – ./FlappySharp/Jeu.cs

0.0.6 Sprite.cs

```
ïż£using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FlappySharp
{
    public class Sprite : PictureBox
    {
        Panel _zoneScene;
        Dictionary<string, Bitmap> _images;
        Dictionary<string, Bitmap> _imagesRot;
        Rectangle _collision;
        Timer _tmp;
        int _xPos;
        int _yPos;
        int _calque;
        int _zOrder;
        int _intervalEntreImage;
        int _angleRotation;
        int _imageAnime;
        bool _dragging;
        bool _selected;
        string _modificationZOrder;
        string _tagSprite;
        string _cheminDossierImage;
        int _positionFinal;
        float _spriteVYMonte = 1f;
        float _spriteVYDescend = 1f;
        int _vitesseLateral = 1;

        public int Calque { get => _calque; private set => _calque = value; }
        public int ZOrder { get => _zOrder; set => _zOrder = value; }
        public string ModificationZOrder { get => _modificationZOrder; set => _modificationZOrder = value; }
        public bool Selected { get => _selected; set => _selected = value; }
        public int IntervalEntreImage { get => _intervalEntreImage; set => _intervalEntreImage = value; }
        public string TagSprite { get => _tagSprite; set => _tagSprite = value; }
        public int AngleRotation {
            get => _angleRotation;
            set
            {
                _angleRotation = value;
                _imagesRot = new Dictionary<string, Bitmap>();
                foreach (var item in Images)
                {
                    _imagesRot.Add(item.Key, RotateImage(item.Value, _angleRotation));
                }
            }
        }
        public Dictionary<string, Bitmap> Images { get => _images; private set => _images = value; }
        public string CheminDossierImage { get => _cheminDossierImage; set => _cheminDossierImage = value; }
        public Rectangle Collision { get => _collision; private set => _collision = value; }

```

```
60          public Sprite(string nom, Size taille, Dictionary<string, ↩
                Bitmap> images, int calque, int zOrder, Point position, ↩
                Panel zoneScene)
61          {
62              base.SizeMode = PictureBoxSizeMode.StretchImage;
63
64              base.Name = nom;
65              base.Size = taille;
66              base.Location = position;
67              base.Image = images.Values.First();
68
69              Collision = new Rectangle(position, new Size(taille.Width - ↩
                    1, taille.Height - 1));
70              _tmp = new Timer();
71
72              Images = images;
73              _imagesRot = new Dictionary<string, Bitmap>(Images);
74              Calque = calque;
75              ZOrder = zOrder;
76              IntervalEntreImage = 1;
77              _zoneScene = zoneScene;
78              _positionFinal = base.Location.Y + 200;
79
80              base.MouseDown += pbx_MouseDown;
81              base.MouseMove += pbx_MouseMove;
82              base.MouseUp += pbx_MouseUp;
83              base.Paint += pbx_Paint;
84              base.Click += pbx_Click;
85
86              _tmp.Interval = 130;
87              _tmp.Enabled = false;
88              _tmp.Tick += tmp_Tick;
89
90              AjoutControlPanel();
91          }
92
93
94          public void AjoutControlPanel()
95          {
96              _zoneScene.Controls.Add(this);
97          }
98
99          public void SuprControlPanel(bool removeSprite)
100         {
101             _zoneScene.Controls.Remove(this);
102             if (removeSprite)
103             {
104                 this.Dispose();
105                 // :TODO demander au prof comment faire ça
106                 //Directory.Delete(CheminDossierImage + @"\" + ↩
                        base.Name, true);
107             }
108         }
109
110         public void AjoutControlPlateauJeu(frmPlateauJeu plateauJeu)
111         {
112             plateauJeu.Controls.Add(this);
113         }
114
115         public void UpdateValue(string nom, Size taille, Point ↩
                position, Dictionary<string,Bitmap> image, int ↩
                intervalImage, int calque, int zOrder, string tag, int ↩
                rotation)
116         {
117             base.Name = nom;
118             base.Size = taille;
119             base.Location = position;
120             Images = image;
121             IntervalEntreImage = intervalImage;
122             _tmp.Interval = intervalImage;
123             Calque = calque;
124             ZOrder = zOrder;
```

```
            TagSprite = tag;
            AngleRotation = rotation;
        }

        public void DemareAnimation()
        {
            _tmp.Enabled = !_tmp.Enabled;
        }

        public void CreationDossier(string cheminDossier)
        {
            CheminDossierImage = cheminDossier;
            Directory.CreateDirectory(Path.Combine(cheminDossier, base.Name));
            foreach (var image in Images)
            {
                Bitmap monImage = image.Value;
                monImage.Save(Path.Combine(cheminDossier, base.Name) + @"\" + image.Key, monImage.RawFormat);
            }
        }

        public void Deplacement()
        {
            switch (TagSprite)
            {
                case "Player":
                    if (_positionFinal < 0)
                    {
                        _positionFinal = 0;
                    }

                    if (base.Location.Y > _positionFinal)
                    {
                        _spriteVYDescend = 1f;
                        base.Location = new Point(base.Location.X, base.Location.Y - (int)_spriteVYMonte);
                        _spriteVYMonte += 0.08f;
                    }
                    else
                    {
                        _spriteVYMonte = 1f;
                        _positionFinal = base.Location.Y + 200;
                        base.Location = new Point(base.Location.X, base.Location.Y + (int)_spriteVYDescend);
                        _spriteVYDescend += 0.08f;
                    }
                    Collision = new Rectangle(base.Location, base.Size);
                    break;

                case "Ennemie":
                    if (base.Location.X <= -base.Width)
                    {
                        base.Location = new Point(_zoneScene.Width, base.Location.Y);
                    }
                    else
                    {
                        base.Location = new Point(base.Location.X - _vitesseLateral, base.Location.Y);
                    }
                    Collision = new Rectangle(base.Location, base.Size);
                    break;
            }
        }

        public void ChangePositionFinal(int valeurFinal)
        {
            _positionFinal = this.Location.Y - valeurFinal;
        }

        public void ActiverDesactiverEvenement(bool activerEnement)
```

```
        {
            if (activerEnement)
            {
                base.MouseDown += pbx_MouseDown;
                base.MouseMove += pbx_MouseMove;
                base.MouseUp += pbx_MouseUp;
                base.Click += pbx_Click;
            }
            else
            {
                base.MouseDown -= pbx_MouseDown;
                base.MouseMove -= pbx_MouseMove;
                base.MouseUp -= pbx_MouseUp;
                base.Click -= pbx_Click;
            }
        }

        public Bitmap RotateImage(Image image, float angle)
        {
            return Rotation(image, new PointF((float)image.Width / 2, (float)image.Height / 2), angle);
        }

        private static Bitmap Rotation(Image image, PointF pointRotation, float angle)
        {
            if (image == null)
                throw new ArgumentException("image");

            Bitmap bmpTourner = new Bitmap(image.Width, image.Height);
            bmpTourner.SetResolution(image.HorizontalResolution, image.VerticalResolution);

            Graphics g = Graphics.FromImage(bmpTourner);

            g.TranslateTransform(pointRotation.X, pointRotation.Y);

            g.RotateTransform(angle);

            g.TranslateTransform(-pointRotation.X, -pointRotation.Y);

            g.DrawImage(image, new Point(0, 0));

            return bmpTourner;
        }

        #region Event
        private void pbx_MouseUp(object sender, MouseEventArgs e)
        {
            var c = sender as PictureBox;
            if (null == c) return;
            _dragging = false;
            Collision = new Rectangle(base.Location, base.Size);
        }

        private void pbx_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button != MouseButtons.Left) return;
            _dragging = true;
            _xPos = e.X;
            _yPos = e.Y;
        }

        private void pbx_MouseMove(object sender, MouseEventArgs e)
        {
            var c = sender as PictureBox;
            if (!_dragging || null == c) return;
            c.Top = e.Y + c.Top - _yPos;
            c.Left = e.X + c.Left - _xPos;
        }

        private void pbx_Paint(object sender, PaintEventArgs e)
```

```
260 ␣␣␣␣␣␣␣␣{
261 ␣␣␣␣␣␣␣␣␣␣␣␣if␣(sender␣is␣Sprite)
262 ␣␣␣␣␣␣␣␣␣␣␣␣{
263 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣e.Graphics.DrawRectangle(Pens.Transparent,␣Collision);
264 ␣␣␣␣␣␣␣␣␣␣␣␣}
265 ␣␣␣␣␣␣␣␣}
266
267 ␣␣␣␣␣␣␣␣private␣void␣pbx_Click(object␣sender,␣EventArgs␣e)
268 ␣␣␣␣␣␣␣␣{
269 ␣␣␣␣␣␣␣␣␣␣␣␣if␣(sender␣is␣Sprite)
270 ␣␣␣␣␣␣␣␣␣␣␣␣{
271 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Selected␣=␣true;
272 ␣␣␣␣␣␣␣␣␣␣␣␣}
273 ␣␣␣␣␣␣␣␣}
274
275 ␣␣␣␣␣␣␣␣private␣void␣tmp_Tick(object␣sender,␣EventArgs␣e)
276 ␣␣␣␣␣␣␣␣{
277 ␣␣␣␣␣␣␣␣␣␣␣␣base.Image␣=␣_imagesRot.Values.ElementAt(_imageAnime);
278 ␣␣␣␣␣␣␣␣␣␣␣␣if␣((Images.Count␣-␣1)␣␣==␣_imageAnime)
279 ␣␣␣␣␣␣␣␣␣␣␣␣{
280 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣_imageAnime␣=␣0;
281 ␣␣␣␣␣␣␣␣␣␣␣␣}
282 ␣␣␣␣␣␣␣␣␣␣␣␣else
283 ␣␣␣␣␣␣␣␣␣␣␣␣{
284 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣_imageAnime++;
285 ␣␣␣␣␣␣␣␣␣␣␣␣}
286 ␣␣␣␣␣␣␣␣}
287 ␣␣␣␣␣␣␣␣#endregion
288 ␣␣␣␣}
289 }
```

Listing 6 – ./FlappySharp/Sprite.cs

0.0.7 SpriteSerialisable.cs

```
1  ï»¿using System;
2  using System.Collections.Generic;
3  using System.Drawing;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace FlappySharp
9  {
10     public class SpriteSerialisable
11     {
12         Point _position;
13         Size _taille;
14         List<string> _nomImages;
15         string _cheminDossier;
16         string _tagSprite;
17         string _nom;
18         int _calque;
19         int _zOrder;
20         int _intervalEntreImage;
21         int _angleRotation;
22
23         public Point Position { get => _position; set => _position = ↩
               value; }
24         public Size Taille { get => _taille; set => _taille = value; }
25         public List<string> NomImages { get => _nomImages; set => ↩
               _nomImages = value; }
26         public string CheminDossier { get => _cheminDossier; set => ↩
               _cheminDossier = value; }
27         public string TagSprite { get => _tagSprite; set => _tagSprite ↩
               = value; }
28         public string Nom { get => _nom; set => _nom = value; }
29         public int Calque { get => _calque; set => _calque = value; }
30         public int ZOrder { get => _zOrder; set => _zOrder = value; }
31         public int IntervalEntreImage { get => _intervalEntreImage; set ↩
               => _intervalEntreImage = value; }
32         public int AngleRotation { get => _angleRotation; set => ↩
               _angleRotation = value; }
33
34         public SpriteSerialisable()
35         {
36
37         }
38
39         public void SetValue(List<string> nomImages, string ↩
               cheminDossier, string nom, Point position, Size taille, int ↩
               calque, int zOrder, int intervalEntreImage, int ↩
               angleRotation, string tagSprite)
40         {
41             Position = position;
42             Taille = taille;
43             NomImages = nomImages;
44             CheminDossier = cheminDossier;
45             TagSprite = tagSprite;
46             Nom = nom;
47             Calque = calque;
48             ZOrder = zOrder;
49             IntervalEntreImage = intervalEntreImage;
50             AngleRotation = angleRotation;
51         }
52     }
53 }
```

Listing 7 – ./FlappySharp/SpriteSerialisable.cs