Informations sur la bibliothèque SFML

Mini-projets informatiques

22/03/2019

SFML est une bibliothèque multimédia qui permet de manipuler sons et images en C++. La bibliothèque doit être téléchargée et installée depuis le site internet. Elle contient plusieurs module à inclure sans le projet en fonction des besoins : graphique, sons et réseau. Cette bibliothèque est portable (Windows, Unix, MacOS,...), légère et facile à utiliser.

Nous allons utiliser cette bibliothèque dans les mini-projets pour pouvoir gérer une petite interface graphique permettant de dessiner des éléments ou d'afficher des images.

1 Installation de la SFML

Visual Studio : https://www.sfml-dev.org/tutorials/2.5/start-vc-fr.php

1. Télécharger la version `Visual C++ 15(2017) - 64 bits` sur le site officiel
2. Dézipper le dossier `SFML-2.5.1` à la racine de votre disque C:
3. Dans Visual Studio, créer un projet vide et inclure dans le projet un fichier `main.cpp`
4. En haut à gauche, à gauche du bouton `Débogueur Windows local`, vérifier que vous êtes bien dans la configuration `Debug` et `x64`
5. Ouvrir la page des propriétés dans `Projet -> Propriétés de < nom du projet >`
6. Tout en haut, vérifier que la configuration est `Debug` et la plateforme `x64`
7. Dans l'onglet `C/C++ -> Général`, dans le champ `Autre répertoires d'include`, indiquer `C:\SFML-2.5.1\include`
8. Dans l'onglet `Editeur de lien -> Général`, dans le champ `Répertoires de bibliothèques supplémentaires`, indiquer `C:\SFML-2.5.1\lib`
9. Dans l'onglet `Editeur de lien -> entrée`, cliquer sur `Dépendances supplémentaires` puis sur la flèche vers le bas puis `<Modifier...>`.
10. Dans la zone de texte, rentrer `sfml-system-d.lib`, `sfml-window-d.lib`, `sfml-graphics-d.lib`, `sfml-audio-d.lib` séparés de sauts de ligne.
11. Valider les changements, puis copier le code de test pour tester la configuration.
12. Avant de compiler, copier les .dll du dossier `C:\SFML-2.5.1\bin` dans le même dossier que le `main.cpp`

1.1 Code de test

Fichier `test-sfml.cpp`

```cpp
#include <SFML/Graphics.hpp>

int main()
{
    sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
    sf::CircleShape shape(100.f);
```

```
    shape.setFillColor(sf::Color::Green);

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear();
        window.draw(shape);
        window.display();
    }

    return 0;
}
```

Pour les autres systèmes d'exploitation, utiliser les tutoriels suivants :

- Linux : https://www.sfml-dev.org/tutorials/2.5/start-linux-fr.php
- Xcode : https://www.sfml-dev.org/tutorials/2.5/start-osx-fr.php

2 Utilisation de la bibliothèque

En guise d'introduction, vous pouvez commencer par étudier les exemples simples d'utilisation en lien avec les projets qui sont fournis dans l'archive. Ils sont repris dans les sous-paragraphes ci-dessous.

Pour une présentation plus complète, vous pouvez vous référer à la documentation proposée sur le site web de la SFML, en cliquant ici.

2.1 Chargement et affichage d'une image

Fichier `draw-image.cpp`

```
#include <SFML/Graphics.hpp>

int main() {
  sf::Texture texture;
  if (!texture.loadFromFile("lena.ppm"))
  {
      // erreur...
  }
  sf::Sprite sprite;
  sprite.setTexture(texture);

  // Création de la fenêtre
  sf::Vector2u sz=texture.getSize();
  sf::RenderWindow window(sf::VideoMode(sz.y, sz.x), "image");
```

```
  // Boucle principale
  while (window.isOpen())
  {
    sf::Event event;
    while (window.pollEvent(event)) {
      // Demande de fermeture de la fenêtre
      if (event.type == sf::Event::Closed)
        window.close();

      }

    // On efface la fenêtre (en blanc)
    window.clear(sf::Color::White);

    // Affichage du sprite
    window.draw(sprite);

    // Mise à jour de la fenêtre
    window.display();
  }

  return 0;
}
```

2.2 Utilisation du graphique pour gérer l'affichage d'un plateau de jeu

Fichier `morpion.cpp`

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <vector>


// Affichage de la grille
void drawGrid(sf::RenderWindow &window) {
  std::vector<sf::RectangleShape> lines;

  // Lignes verticales
  lines.push_back(sf::RectangleShape(sf::Vector2f(6, 300)));
  lines.push_back(sf::RectangleShape(sf::Vector2f(6, 300)));
  // Lignes horizontales
  lines.push_back(sf::RectangleShape(sf::Vector2f(300, 6)));
  lines.push_back(sf::RectangleShape(sf::Vector2f(300, 6)));

  // Positionnement des lignes
  lines[0].setPosition(97, 0);
  lines[1].setPosition(197, 0);
  lines[2].setPosition(0, 97);
  lines[3].setPosition(0, 197);
```

```
  // Lignes de couleurs noires
  for (size_t i = 0; i < lines.size(); i++)
    lines[i].setFillColor(sf::Color::Black);

  // Affichage
  for (size_t i = 0; i < lines.size(); i++)
    window.draw(lines[i]);
}

// Affichage d'une croix
void drawCross(sf::RenderWindow &window, size_t i, size_t j) {

  // Deux lignes tournées de 45°
  sf::RectangleShape r1(sf::Vector2f(4, 80));
  sf::RectangleShape r2(sf::Vector2f(4, 80));

  // Par défaut, l'origine est en haut à droite du rectangle
  // On met l'origine au milieu pour simplifier les calculs
  r1.setOrigin(2, 40);
  r2.setOrigin(2, 40);


  // Positionnement
  r1.setPosition(i * 100 + 50, j * 100 + 50);
  r2.setPosition(i * 100 + 50, j * 100 + 50);

  // Rotation
  r1.rotate(45);
  r2.rotate(-45);

  // Coueleur bleu
  r1.setFillColor(sf::Color::Blue);
  r2.setFillColor(sf::Color::Blue);

  // Affichage
  window.draw(r1);
  window.draw(r2);
}

// Affichage d'un cercle
void drawCircle(sf::RenderWindow &window, size_t i, size_t j) {
  // Rayon = 30px
  sf::CircleShape c(30);

  // Contour rouge d'une épaisseur de 4px
  c.setOutlineColor(sf::Color::Red);
  c.setOutlineThickness(4);

  c.setOrigin(30, 30);
  c.setPosition(i * 100 + 50, j * 100 + 50);
```

```
  window.draw(c);
}


int main()
{

  // Création de la fenêtre
  sf::RenderWindow window(sf::VideoMode(300, 300), "Jeu de morpion");

  // Tableau de jeu
  int tab[3][3];

  for (size_t i = 0; i < 3; i++)
    for (size_t j = 0; j < 3; j++)
      tab[i][j] = 0;

  // Indique le tour du joueur rouge
  bool redTurn = true;

  // Compte le nombre de cases vide
  int nbLeft = 9;


  // Boucle principale
  while (window.isOpen())
  {
    sf::Event event;
    while (window.pollEvent(event)) {
      // Demande de fermeture de la fenêtre
      if (event.type == sf::Event::Closed)
        window.close();

      // Appui sur le bouton gauche
      if ( (event.type == sf::Event::MouseButtonPressed)
        && (event.mouseButton.button == sf::Mouse::Left) ) {
        // Si on a plus de place, on vide le tableau
        if (nbLeft == 0) {
          for (size_t i = 0; i < 3; i++)
            for (size_t j = 0; j < 3; j++)
              tab[i][j] = 0;
          nbLeft = 9;
        }
        // Sinon, on place notre jeton
        else {
          // Position de la souris dans par rapport à la fenêtre
          sf::Vector2i localPosition = sf::Mouse::getPosition(window);

          // Position de la souris dans le tableau
          localPosition /= 100;
```

```
            // Affichage console des coordonées
            std::cout << localPosition.x << " " << localPosition.y << std::endl;

            // Si on tombe sur une case vide
            if (tab[localPosition.x][localPosition.y] == 0) {
              // On remplie en fonction du tour du joueur (1 pour rouge, 2 pour bleu)
              tab[localPosition.x][localPosition.y] = redTurn ? 1 : 2;
              // On change de joueur
              redTurn = !redTurn;
              // On met a jour le nombre de cases vides
              nbLeft--;
            }
          }
        }

      }

      // On efface la fenêtre (en blanc)
      window.clear(sf::Color::White);

      // On affiche la grille
      drawGrid(window);

      // Parcours du tableau
      for (size_t i = 0; i < 3; i++) {
        for (size_t j = 0; j < 3; j++) {

          // Affichage du jeton
          if (tab[i][j] == 1)
            drawCircle(window, i, j);
          else if (tab[i][j] == 2)
            drawCross(window, i, j);
        }
      }

      // Mise à jour de la fenêtre
      window.display();
    }

    return 0;
}
```