Techniques de l’informatique – programmation de jeux vidéo

420-V32-SF Programmation de jeux vidéo III

Introduction à SFML

Partie 1 : Généralités sur SFML et questions sur les exemples fournis.

Récupérez la solution ExemplesSFML puis répondez aux questions suivantes.

1. La boucle principale du jeu se trouve dans la méthode run de la classe Game. Elle contient les instructions suivantes :

while (mainWin.isOpen())

{

// Séquence pour un frame:

// a) Lecture des entrées du joueur

getInputs();

// b) Mise à jour du jeu (déplacements des personnages, physique, etc)

update();

// c) Mise à jour de l'affichage

draw();

}

Expliquez en vos mots pourquoi il est important de bien séparer la lecture des entrées, la mise à jour de la logique du jeu ainsi que l’affichage du jeu.

Parce que les instructions doivent s’exécuter dans l’ordre. Par exemple, le déplacement d’un objet doit être fait avant le rafraîchissement de l’écran. Il faut séparer les tâches et les exécuter dans un ordre bien précis.

1. L’affichage des éléments du jeu se fait dans la méthode draw de la classe Game. Cette méthode contient les instructions suivantes :

mainWin.clear();

// Affichage ici des éléments.

// …

mainWin.display();

Expliquez en vos mots le rôle des instructions mainWin.clear() et mainWin.display().

L’écran en 2D est séparé en deux affichages s’alternants à chaque frame. Lorsque le jeu doit changer la position d’un objet à l’écran ou changer une couleur, celui-ci noircis l’écran frontale et colore celle arrière, puis échange les deux.

mainWin.clear()

La fonction mainWin.clear() sert donc à noircir l’écran frontale.

mainWin.display()

La fonction mainWin.display() sert donc à colorer l’écran noircie envoyée.

1. Dans le projet ExempleSFML1, les instructions suivantes sont présentes dans la déclaration de la classe Game.

static const int LARGEUR;

static const int HAUTEUR;

Expliquez en vos mots pourquoi il est nécessaire de produire une définition (dans le fichier Game.cpp) « spéciale » pour ces deux propriétés.

Ces deux propriétés représentent des valeurs qui sont censées être fixes dans le programme. Les modificateurs static et const semblent donc convenir parfaitement.

1. Dans le fichier Game.cpp du projet ExempleSFML5, on retrouve dans la méthode init le code

retval = music.openFromFile("Assets/Music/Carl\_Sagan\_-\_Pale\_Blue\_Dot.ogg") && retval;

Est-il équivalent au code suivant? Si votre réponse est « non », expliquez-en la (les) différence(s).

retval = retval && music.openFromFile("Assets/Music/Carl\_Sagan\_-\_Pale\_Blue\_Dot.ogg");

Non, car l’instruction music.openFromFile a le potentiel de déclencher un erreur alors que si jamais retval venait à être faut, l’erreur ne serait pas déclenchée, car l’instruction s’arrêterait là.

1. Dans le fichier Game.cpp du projet ExempleSFML5, la méthode draw possède le code suivant :

mainWin.draw(backgroundSprite);

mainWin.draw(text);

mainWin.draw(ballSprite);

Inversez l’ordre des instructions (affichez l’arrière-plan en dernier). Que pouvez-vous déduire?

Je déduis que l’affichage se fait dans l’ordre des appels.

Partie 2 : Gestion du redimensionnement

1. Modifiez le code (à vous de décider ce qui doit être modifié!) du code du projet ExempleSFML2 de manière à ce que le cercle soit redimensionné ET toujours positionné au centre de l’écran lorsque l’on redimensionne ce dernier.

Vous n’avez pas beaucoup de code à écrire et/ou modifier. Vous devez quand même trouver lequel.

Indice :

1. la gestion du redimensionnement se fait dans la méthode getInputs de la classe Game.
2. Vous devrez utiliser adéquatement la méthode setScale sur la propriété shape de la classe SFMLCircle.
3. Vous devrez probablement ajouter une propriété pour stocker la position plutôt que de procéder comme cela est présentement fait dans la classe SFMLCircle.

Partie 3 : La balle qui rebondit

Pour cet exercice, on vous demande d’écrire un programme dans lequel une balle rebondit sur les côtés de l’écran. Vous avez déjà réalisé ce type d’application à l’hiver dernier. Vous devez cette fois-ci le faire en C++.

La balle doit partir du centre dans une direction au hasard et sa vitesse doit être la même peu importe son angle. Quand elle frappe un mur elle rebondit (l’angle d’entrée est le même que l’angle de sortie).

Peu importe la taille de votre vue, ce prototype devrait fonctionner correctement. Ce n’est pas une question de changement de taille de la fenêtre; pour que le fond d’écran soit affiché correctement, votre vue doit faire 1302\*666 (la même taille que terrain.png) mais le prototype doit fonctionner peu importe la taille de la vue définie.

Vous pouvez sortir votre trigonométrie si vous voulez mais c’est encore un peu tôt. En fait, ici la solution est plutôt simple. Observez une balle de billard qui entre contre un rebord de la table et décortiquez son mouvement en X et Y. Cette observation vous dira ce qu’il faut faire.

Vous pouvez utiliser le projet de démarrage fourni.