Techniques de l’informatique – programmation de jeux vidéo

420-V32-SF Programmation de jeux vidéo III

Introduction à SFML

Partie 1 : Généralités sur la trigonométrie

Un *sprite* SFML contient plusieurs méthodes. À partir de l’aide en ligne, répondez aux questions suivantes

1. À quoi serve la méthode setColor?

La méthode SetColor sert à changer la couleur de la sprite.

1. À quoi serve la méthode setOrigin?

La méthode setOrigin sert à changer la position du point pivot de la sprite utilisé lors d’un déplacement et lors d’une rotation de la sprite.

1. À quoi serve la méthode setScale? Au fait, que signifie un facteur d’échelle de (1,1)?

setScale() permet de changer les dimensions de la sprite au niveau visuel. Pour répondre à votre autre question, un rapport (1,1) signifie que la dimension sera proportionnelle à l’échelle de référence.

Partie 2 : Deux questions de trigonométrie

1. La 15e diapositive vous propose une manière de déterminer l’angle entre un personnage et une cible. En fait, on y soulève un problème important. Lequel?

L’angle généré par la tangente peut être erroné, car il existe deux rapports possible par rapport à la cible au calcul.

1. Proposez une manière de procéder qui permet de contourner ce problème. En fait, je veux que vous inscriviez une autre manière que celle inscrite dans les diapositives. Orientez votre réponse en essayant de trouver une autre fonction à utiliser.

La fonction Math.ToDegrees(), selon mes recherches, gère ces genres d’exceptions. Il est aussi recommandé d’user de atan2(le calcul sin/cos)

Partie 3 : Vous aimiez les pokémons?

Pour cet exercice, on vous demande d’écrire un petit jeu dans lequel vous devez attraper un pokémon qui se déplace à l’écran et ce, afin de vous familiariser avec SFML et la trigonométrie.

**Version 1.**

Vous devez programmer deux classes : Pokeball et Pokemon. Une instance de Pokemon change de position à toutes les demi-secondes ou quand on clique dessus avec une Pokeball. La Pokeball doit suivre le curseur de souris. Chaque fois qu’on clique et que la Pokeball touche le Pokemon, Vous marquez un point.

La partie dure un temps limité (30 secondes, disons). Vous affichez les points et le temps restant.

Vous trouverez une version exécutable ainsi qu’une suggestion d’images à utiliser jointes à cet exercice.

**Version 2.**

Ralentissez les "téléportations" de votre Pokémon (aux 3 secondes environ). La Pokeball ne suit plus le curseur de la souris mais se déplace à une vitesse donnée vers l’endroit du curseur de souris quand on fait une action de click.

Quand la Pokeball et le Pokemon se touchent vous marquez un point, comme-ci-dessus.

**Note**

À la base utilisez des sphères de collision autour des deux sprites, mais vous remarquez que le pokémon fourni est plutôt rectangulaire et irrégulier. Donc…

**En Option**

Faites aussi un essai avec des rectangles de collision. Pour le moment, pas besoin de connaitre l’algorithme; explorez la classe IntRect (ou FloatRect) et servez-vous-en. Nous regarderons l’algorithme des rectangles plus tard.

**En Option, la suite**

Les plus habiles pourront essayer de coder un test de collision entre rectangle et sphère, mais ça ne servira pas dans les TP. C’est plutôt avancé; considérez cela comme un "défi algorithmique".