Class CConstellation

(valeurs obtenues d’un dialogue)

m\_iEtoiles : Int, nombre d’étoiles (valeur obtenue d’un dialogue)

m\_iEcho : int, nombre d’écho(lignes répétées anciennes)

Fonctions :

**Dessine\_constellation();**

**SetLimites (maxX, minX, maxY, minY)** : : établit les limites de la constellation. Servira à calculer quand il faut rebondir sur une limite.

+++++++++++

Class CEtoile

ID Etoile : int (numéro de l’étoile)

N\_Echo : int array (nombre d’écho à maintenir) (valeur obtenue de CConstellation)

m\_APosition [N\_Echo] : <vector>? (int posX, int posY)  conserve les dernières positions en mémoire pour tracer les dernières « N » position de l’étoile.

m\_iEtoileActive : int. Qui point vers la position du m\_APosition [N] qui va être mise à jour.

Fonctions :

**InitEcho() :** initialiser les positions des « N-1 » Echo

**SetLimites (maxX, minX, maxY, minY)** : établit les limites de la constellation. Servira à calculer quand il faut rebondir sur une limite.

**SetPosition(int posX, int posY)** : position initiale d’une étoile au début

**SetVitesse(int Vx, int Vy)** : vitesse initiale d’une étoile au début

**CalculePosition**: calcule la prochaine position en utilisant les coordonnées et les vitesses. Si une des coordonnées (x ou y) dépasse les limites

**RebonditX(), RebonditY()** (2 fonctions séparées) : lorsque l’étoile atteint la limite (maxX, min X, max Y , min Y), l’étoile change de direction. La fonction calcule une nouvelle vitesse qui varie en fonction de la limite atteinte. Ex : si max X est atteint, la vitesse « x » est modifiée et la direction « x » s’inverse. Vy demeure inchangée. Utilise la fonction random pour changer la vitesse qui est affectée par le changement.

Initialiser les positions pour « N-1 » Echo.

Pour chaque étoile de 0 à N-1, x= pos X et y= pos Y (point de départ initial)

m\_iEtoileActive =0; (0 est l’étoile intiale qui se déplace)

Calcule position :

positionX = m\_APosition [m\_iEtoileActive].X;

positionY = m\_APosition [m\_iEtoileActive].Y;

newPositionX = positionX +Vx;

newPositionY = positionY + Vy;

if(newPositionX>maxX || newPositionX < minX)

RebonditX;

newPositionX = positionX +Vx

if(newPositionX>maxY || newPositionY < minY)

RebonditY;

newPositionY = positionY + Vy;

m\_iEtoileActive += (augmente le pointeur de 1)

if m\_iEtoileActive > N-1

m\_iEtoileActive = 0; (remet le pointeur au début)

m\_APosition [m\_iEtoileActive].X = newPositionX;

m\_APosition [m\_iEtoileActive].Y = newPositionY;

m\_APosition [m\_iEtoileActive] = ;

RebonditX ou RebonditY:

Génère nombre aléatoire (0-32768)

Ramener le nombre généré sur l’échelle de vitesse :

Random/32768 \* VXMax =newVx; (même opération pour Vy)

Inverser la direction newVx = -newVx;