Projet SDL 60%

Remise : **jeudi 9 mai.** Vous devrez présenter votre travail en classe.

Vous pouvez vous placer en équipe de 2 (vous devez effectuer une recherche personnelle pour faire ce travail)

Je validerai cependant le travail avec chaque personne individuellement à l'aide de questions

Vous devez faire une animation qui contient :

1. Un menu (faire une image de background pour le menu si vous désirez) 4%

2. Vous devez utiliser les tuiles et une caméra afin que le fond bouge. 5%

   (Votre monde de tuiles n'a pas à être complexe. C'est-à-dire à utiliser plusieurs tuiles différentes.) Il doit quand même être plus large et plus haut que l'écran afin qu'il puisse bouger de gauche à droite et de bas en haut et doit présenter une collision et peut-être un endroit où une force sera appliquée(à voir plus loin).
   Note : votre monde de tuiles ne doit pas être celui de la démo.

3. Utiliser au moins 2 sprites par équipier de masse différente (voir note plus bas sur la masse) 2%

4. Chaque coéquipier doit avoir son sprite qui est suivi par la caméra à un moment donné Pour ce faire, vous pouvez appuyer sur une touche et c'est le sprite d'un autre coéquipier qui est géré par la caméra ou toute autre idée de votre choix. 5%

5. Votre sprite doit avoir au moins une collision avec les tuiles. 5%

   Soit vous créez un tableau pour indiquer si ce no de tuile est une collision ou non . (Toutes les tuiles pareilles seront une collision)

   Soit vous créez un tableau de rectangles de collision à votre sprite

   Ou encore, faire une autre matrice similaire au monde de tuiles, mais qui ne contient pas un no de tuile, mais s'il y a collision ou non à cet endroit

6. Prendre les données de votre matrice de tuiles dans un fichier (référence ma grosse matrice 32 par 32 avec les no de tuiles) 5%

7. Vous devez simuler une 3D. C'est-à-dire que vous devez ajouter quelques éléments de décor par-dessus. (Soit une autre palette de tuiles avec des tuiles transparentes et d'autres non transparentes ou encore de simples images). Il s'agit de dessiner le fond, de dessiner vos personnages sur le fond et dessiner les autres décors par-dessus afin de passer en dessous. 2%

8. Utiliser la libraire sdl_gfx pour rotation et flip pour votre autre sprite 2%

9. Trouver comment utiliser la librairie ttf pour écrire sur la fenêtre (cette librairie est déjà installée dans mes exemples de projets, mais je ne l'utilise pas. Je fais juste l'initialiser dans certaines démo.) Écrire le nom de la personne dont le sprite est suivi par la caméra à l'écran. 5%

10. Tous les sprites doivent utiliser la classe sprite et la classe vecteur2D (vous pouvez améliorer ces classes) 5%

11. Au moins un sprite par coéquipier doit bouger avec les touches (différentes pour chaque coéquipier) et faire une autre action sur une autre touche. L'autre sprite doit se promener comme vous voulez. 5%

12. Votre sprite qui bouge avec les touches doit utiliser la classe animation et avoir au moins une feuille de sprites (plusieurs états). Le ou les autres comme vous voulez 5%
13. Vos deux sprites personnels doivent subir deux mêmes forces de nature différente (cela peut-être aussi le fait de traverser une zone fluide) qui le ralentit ou l'accélère ou … (exemple : eau, une montée, etc..) 5%

    ICI pas besoin d'utiliser les autres formules plus réelles d'angle de pente, de friction, etc. (lien sur les forces en fin de tp) Mais pour ceux qui en ont envie… il y a un 3% au point 14 pour cela.

    Chaque coéquipier doit se voir subir des forces de nature différentes de ou des autres coéquipiers (pas juste une différence dans la valeur de la force!) Usez d'imagination!

    La gestion pourrait de faire par :

    a) dans une matrice de collision, au lieu de 0 libre et 1 collision, on peut placer 0 1 2 pour appliquer une force (pour ralentir, accélérer, voler, etc.).
    b) Ou encore, avec une collision avec un certain rectangle, on applique une force
    c) Ou encore un tableau de no de tuiles et sa force à appliquer.
    d) Ou l'appui d'une touche (mais pas pour vos 2 forces)

e) Ou une idée de votre choix

Vous devez cependant utiliser la méthode applyforce du sprite

Vous avez un exemple simplet de Applyforce dans la démo 11 (vent) le chat qui suit la souris (de l'ordinateur) reçoit du vent s'il s'en approche trop. J'aurai dû appliquer une résistance au fluide ou force de trainée, car je suis contraire au vent, mais à vous de chercher au point 14 si si cela vous tente d'avoir le 60%

void ApplyForce(Vecteur2D force) {

force.Div(masse); // masse est de 1 par défaut dans mes sprites
acceleration.add(force);}

L'accélération cumule toutes les forces et les pas à avancer s'il y a lieu (pas=5 par défaut dans mes sprite).
Cela dépend comment vous gérer la vitesse et l'accélération dans votre update

NOTE masse et vitesse d'accélération de base (masse et pas)

```
Sprite(Game * _game, SDL_Texture *_texture, int _x, int _y, int _w, int _h, int _pas = 5, int _masse = 1);
```

La masse est par défaut à 1 pour la classe sprite et le pas de 5

```
souris2 = new Sprite(this, texture, 0, 100, 58, 80);// pas besoin de passer le pas et la masse
```

pour passer le pas

```
souris2 = new Sprite(this, texture, 0, 100, 58, 80,6);
```

pour passer la masse il faudra obligatoirement passer le pas également

```
souris2 = new Sprite(this, texture, 0, 100, 58, 80,5,1.2f);
```

void update() {

// note : si la vitesse n'est pas remise à zéro, il va de plus en plus vite

On peut limiter la vitesse *1

vitesse.add(acceleration);

location.add(vitesse);

acceleration.mult(0);} // on remet à zéro l'accélération chaque fois

}

*****Attention : le Update de mes sprites animés ne gère pas l'accélération donc la fonction devra être modifiée pour en tenir compte!**
Vous avez le int pas si je veux qu'il avance de 4 avec une touche et je le place directement dans vitesse.x et. Y

*1 Pour limiter la vitesse à 15 pas si max=15:

```
void limit(float max) {  // le max pourrait être ajouté dans une donnée du sprite
   if (vitesse.lenght() > max) {
    vitesse.Normalize();
     vitesse.Mult(max);
   }
 }
```

14. Utilisation d'une force autre que celle de la démo dans la deuxième référence sur les forces (friction, fluidee frottement) une plus compliquée 5%

Références :

http://natureofcode.com/book/chapter-1-vectors/

http://natureofcode.com/book/chapter-2-forces/

attention : Dans leur langage, il n'est pas nécessaire de l'écrire, mais toutes les définitions des méthodes qui reçoivent un objet., par exemple :

```
void applyForce(PVector force)
```

veut dire
void applyforce(PVector &force)
Le paramètre est toujours modifiable. C'est la raison où quelquefois ils sont obligés de faire une copie avant de modifier pour ne pas modifier la force envoyée

```
Eux: void applyForce(PVector force) {
Making a copy of the PVector before using it!
  PVector f = force.get();
  f.div(mass);
  acceleration.add(f);
}
```

Sinon elle serait vraiment divisée par la masse du sprite

En c++ ce n'est pas le cas, car c'est déjà une copie si pas de & ou pas un pointeur!