COMPTE-RENDU

GMIN317 – Moteur de Jeux

TP2 – 25 Septembre 2015

Contenu

– Etudier et modifer la boucle (game loop) de rendu du TP1

– Afficher la scŁne dans 4 fenŒtres diffØrentes, avec 4 *timers*

– Animer la scŁne vitesse constante

– Bonus : Øtudier et modifier l'organisation logicielle des diffØrentes classes

SØbastien Beugnon
M2 IMAGINA 2015-2016

1. Questions

1 – Interactions

Ce TP met en place un rendu de terrain par *heightmap* avec la possibilitØde changer de carte en pleine exØcution ainsi que le rendu de cette derniŁre selon plusieurs modes : Affichage par points, par lignes (le maillage sans les faces), par faces (toutes blanches), par faces avec une coloration d'un triangle sur deux en vert sinon en blanc, et enfin un mode coloration en fonction de la position en Y :

- Y > 0.2 : blanc

- 0.1 < Y < 0.2 : cyan

- 0.05 < Y < 0.1 : rouge

- Y < 0.05 : bleu

De plus, des interactions supplØmentaires ont ØØrajoutØes pour modifier l'angle de vue en X ou Y de la camØra centrØe sur le terrain avec la possibilitØde zoomer et de dØzoomer. Ci-dessous se trouve la liste des touches avec les interactions reliØes :

Touche A : Rotation horaire de la camØra sur l'axe X

Touche E : Rotation anti-horaire de la camØra sur l'axe X

Touche Z : Zoom

Touche S : DØzoom

Touche Q : Rotation horaire de la camØra sur l'axe Y

Touche D : Rotation anti-horaire de la camØra sur l'axe Y

Touche W : Changement du mode de coloration et de rendu

Touche Z : Passage la *heightmap* suivante

Les paramŁtres *etat*, *ss*, *RotX*, *RotY* dØterminent respectivement l'Øtat actuel de l'affichage (le mode de coloration et de rendu), le *scaling* (ou le facteur de zoom/dØzoom appliquØ la camØra), la rotation en X de la camØra, la rotation en Y de la camØra. La crØation d'une classe *Camera* permet alors de dØplacer les variables liØs la position, la rotation, la direction et le facteur d'Øchelle de la camØra *OpenGL* dans un simple modŁle objet lisible. (Annexe 1) Ainsi la fenŒtre *GameWindow*, re oit une camØra qui permet de dØterminer le rendu l'Øcran et en offrant la posibilitØde changer cette camØra par une autre on peut facilement crØer un autre point de vue. (Annexe 2)

2 – Passage au QTimer

Afin d'utiliser correctement le *QTimer*, on retire les appels la mØthode *renderNow()* dans la classe *GameWindow* enfin que le rendu ne soit contr ler uniquement que par notre intance de *Qtimer*. (Annexe 3)

3 – Multiple fenŒtre avec diffØrents *framerate*

Gr ce l'ajout du nombre de frame souhaitØ par seconde dans le constructeur de fenŒtre, la fenŒtre possŁdera son propre taux de rafra chissement. De plus gr ce au passage d'un pointeur de *Camera* les interactions produites dans une fenŒtre se rØpercuteront sur les autres partageant la mŒme camØra.

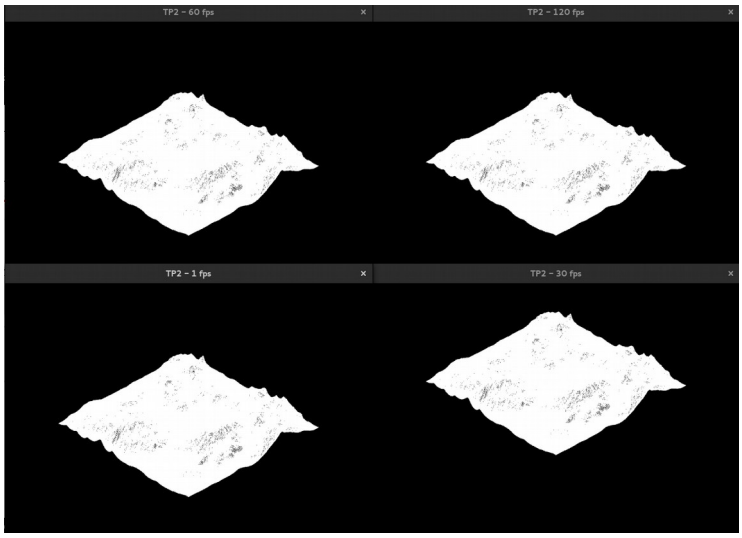

Illustration 1: 4 fenŒtres diffØrents taux de rafra chissement

4 et 5 – Rotation automatique et Contr le du taux de rafra chissement

On ajoute au *callback* des ØvØnements de clavier, une modification de la camØra pour qu'elle autorise (boolØen autoMove) l'incrØmentation chaque appel de la mØthod render de sa rotation en Y afin d'automatiser la rotation. Gr ce ceci, on peut observer que la fenŒtre un *fps* se rØactualise une seule fois par seconde rendant les autres fenŒtres beaucoup plus fluides. De mŒme en divisant par deux, le *framerate* des autres fenŒtres on constate le ralentissement de ces derniŁres affichant de plus en plus par -coups. Et lorsque l'on multiple par deux ce mŒme *framerate*, on contaste qu'il ne dØpasse pas une certaine limite en fonction de la puissance de la machine.

Bonus

Actuellement la classe *GameWindow* hØrite de la class *OpenGLWindow*, cela permet ainsi de surcharger les callbacks d'ØvØnements proposØs par *Qt* ainsi que la mØthode *render()* qui permet de dØfinir le rendu de la scŁne courante. Cette organisation permet de facilement mettre en place une scŁne en utilisant directement les objets *OpenGL* ; cependant on se trouve trop proche de la partie rendu et la crØation de scŁne se doit de s'abstraire tout comme les objets qui la compose (terrain, camØra) de la fa on dont ils sont affichØs. Ainsi, il semble plus juste de sØparer les mØcaniques de rendu de celles logiques. (camØra, terrain)

De ce fait, la question que l'on peut se poser et de savoir quelle entitØ contiendra l'autre. Pour cela il suffit simplement de savoir ce que chacune a besoin ; la classe *GameWindow* a besoin des ØvØnements de fenŒtre qu'intercepte la classe *OpenGLWindow* et cette derniŁre a besoin de la camØra principale et des objets dessiner.

Il faut alors ajouter comme composant *OpenGLWindow* une variable de classe *GameWindow* et rediriger les ØvØnements vers le *GameWindow*. Ainsi la gestion des ØvØnements se font hors de la fenŒtre de rendu et la fenŒtre n'a plus qu' rØcupØrer dans l'instance GameWindow les objets (gameobjects) qu'il faut dessiner et la camØra utiliser. Cette construction permet de sØparer la partie modØlisation de la partie rendu d'un programme.

Annexes

Annexe 1 - camera.h

```
//Caméra eulérienne pour TP2
class Camera
{
private:

    int etat;//état de rendu
    float rotX;//rotation en X
    float rotY;//rotation en Y
    float ss;//facteur de zoom/dézoom (scaling)
    bool autoMove;//Active la rotation automatique de la scène

public:

    Camera();
    virtual ~Camera();
    //Accessors
    void setScale(float _delta);
    void setEtat(int _etat);
    void setRotX(float _rotX);
    void setRotY(float _rotY);
    void incrEtat();
    void flipAutoMove();

    bool isAutoMove();
    float getScale();
    int getEtat();
    float getRotX();
    float getRotY();

};
```

Annexe 2 – gamewindow.h – Ajout du modŁle CamØra en pointeur

```
class GameWindow : public OpenGLWindow
{
public:
    GameWindow(Camera* camera);

    Camera* getCamera();

    void setCamera(Camera* camera);


private:

    int m_fps;
    QImage m_image;
    Camera* m_camera;
    point *p;

    int carte=1;

};
```

Annexe 3 – GameWindow.h – Ajout du QTimer

```
class GameWindow : public OpenGLWindow
{
public:
    GameWindow(Camera* camera, int fps);

// … methods

private:

    int m_fps=60;
    QImage m_image;
    QTimer* m_timer;
    Camera* m_camera;
    point *p;

    int carte=1;

};
// ...
// GameWindow.cpp
void GameWindow::initialize()
{
// ...
    m_timer = new QTimer(this);
    connect(m_timer,SIGNAL(timeout()),this,SLOT(renderNow()));
    m_timer->start(fps_to_timeout(m_fps));
// ...
}
```