TP N°7
Animation

Gestion des évènements

GLFW permet de gérer les évènements d'entrées (liés à un Gamepad, Joystick, clavier, …).

Le clavier :

Afin de récupérer les évènements clavier, on appelle l'instruction callback suivante :
glfwSetKeyCallback(window, key_callback) ;

Lorsqu'un évènement clavier est détecté, la fonction key_callback est appelée:
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)

L'attribut « key » représente la touche clavier en question, tandis que l'attribut « action » peut prendre les valeurs suivantes : GLFW_PRESS ou GLFW_RELEASE.

Exemple1 :

```
static void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
   if (action == GLFW_PRESS)
    switch (key)
   {
      case GLFW_KEY_R :
         glClearColor(1.0f, 0.0f, 0.0f, 0.5f);
         break;
      case GLFW_KEY_V :
         glClearColor(0.0f, 1.0f, 0.0f, 0.5f);
         break;
      case GLFW_KEY_B :
         glClearColor(0.0f, 0.0f, 1.0f, 0.5f);
         break;
     case GLFW_KEY_ESCAPE:
         glfwSetWindowShouldClose(window, GLFW_TRUE);
         break;
     default:
         glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
   }
}
```

Exercice 1 :

- Que fait cette fonction ?
- Ajouter les instructions qui remettent l'arrière plan à sa couleur d'origine une fois les touches R,V et B relâchées.

On peut aussi récupérer l'état d'une touche clavier avec l'instruction :
int glfwGetKey (GLFWwindow* window, int key);

Exemple2 :

```
while (!glfwWindowShouldClose(window))
{
        int state = glfwGetKey(window, GLFW_KEY_E);
        if (state == GLFW_PRESS)    cout <<  "  le E  " ;

        …
}
```

La souris :

- La position du curseur de la souris peut êtres retournés avec l'instruction suivante :

```
void glfwGetCursorPos (  GLFWwindow* window, double xpos, double ypos).
```

Exemple 3 :

```
while (!glfwWindowShouldClose(window))
{
        double Xpos, Ypos ;
        glfwGetCursorPos (window, &Xpos, &Ypos );
        cout << "  La position du curseur :  " <<    Xpos  <<    "  ,   "  <<    Ypos  << endl ;
        …
}
```

- L'état d'un bouton donné, appuyé ou pas, peut aussi être récupérés avec l'instruction :

```
int glfwGetMouseButton(GLFWwindow* window, int button).
```

Exemple 4 :

```
while (!glfwWindowShouldClose(window))
{
        if ( glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT) == GLFW_PRESS)
                cout << "  Bouton gauche de la souris appuyé " ;

        ...
}
```

- Cependant, il existe aussi plusieurs instructions callback qui se déclenchent suite à un des évènement suivants :

1- Changement de la position du curseur :

```
glfwSetCursorPosCallback(window, cursor_position);
```

avec la fonction :

```
static void cursor_position (GLFWwindow* window, double xpos, double ypos)
```

Exemple 5 :

```
static void cursor_position(GLFWwindow* window, double x, double y)
{
        diffX = Xpos – x;
        diffY = y - Ypos;
        Model = rotate(Model,radians(float(diffX)),vec3(0.0f,0.0f,1.0f));
        Model = rotate(Model,radians(float(diffY)),vec3(1.0f,0.0f,0.0f));
}
```

Il faut initialiser les variables Xpos et Ypos dans la boucle d'affichage. Il faut aussi déplacer la définition de la matrice MVP et renvoyer la variable uniforme qui indique sa position au vertex shader :

```
while (!glfwWindowShouldClose(window))
{
    glfwGetCursorPos (window, &Xpos, &Ypos );

    mat4 MVP = Projection * View * Model;
    GLuint MatrixID = glGetUniformLocation(ShaderProgram, "MVP");

        …
}
```

2- Un bouton appuyé :

```
glfwSetMouseButtonCallback(window, mouse_button);
```

avec la fonction :

```
void mouse_button (GLFWwindow* window, int button, int action, int mods)
```

« button » peut prendre les valeurs : GLFW_MOUSE_BUTTON_RIGHT, GLFW_MOUSE_BUTTON_LEFT et GLFW_MOUSE_BUTTON_MIDDLE.
« action » prend les valeurs GLFW_PRESS ou GLFW_RELEASE.

Exercice 2 :
Utiliser les bouton souris pour faire agrandir et diminuer la taille de l'objet dessiné.

3- La roulette de la souris est utilisée (scroll) :

```
glfwSetScrollCallback(window, scroll);
```

avec la fonction :

```
void scroll (GLFWwindow * window, double xoffset, double yoffset)
```

« yoffset » représente la variation de la roulette.

Exemple 6 :
La variable float fov représente l'ouverture horizontale de la caméra. Elle est initialisée à 45° et utilisée dans la matrice Projection :

```
Projection = perspective(radians(fov), 4.0f / 3.0f, 0.1f, 100.0f) ;
```

La fonction qui gère l'évènement du scroll :

```
void scroll (GLFWwindow* window, double xoffset, double yoffset)
{
        if (fov >= 1.0f && fov <= 45.0f)
                fov -= yoffset;
        if (fov <= 1.0f)
                fov = 1.0f;
        if (fov >= 45.0f)
                fov = 45.0f;
}
```

Il faut penser à déplacer la création de la matrice Projection dans la boucle d'affichage, sinon le champs de vision ne change pas.

Faire bouger la caméra :

La gestion de la caméra revient à calculer la matrice de visualisation en fonction des paramètres utilisateur.

1- Trackball camera :

Ce type de caméra est assez simple, elle ne peut que tourner autour de la scène ou de l'objet ainsi qu'avancer et reculer.

Exemple 7 :

Nous allons faire tourner la caméra autour de la scène. Pour cela, nous allons positionner la caméra sur un cercle de rayon donné, dont le centre est le point de référence de visualisation.
Pour cela, insérer le code suivant au début de la fonction d'affichage :

```
float rayon = 5.0;
float Xcam = rayon * sin(glfwGetTime());
float Zcam = rayon * cos(glfwGetTime());
View = lookAt(vec3(Xcam, 0.0, Zcam),vec3(0,0,0),vec3(0,1,0) );
```

2- FreeFly camera :

Ce type de caméra permet de se déplacer selon différents axes dans toutes les directions. Elle permet aussi de tourner autour de soi.
La caméra est définie avec sa position, la direction de visualisation et le vecteur *ViewUp* :

```
vec3 camPos   = vec3(0.0f, 0.0f,  3.0f);
vec3 camDirection = vec3(0.0f, 0.0f, -1.0f);
vec3 camUp    = vec3(0.0f, 1.0f,  0.0f);
```

Le point de référence de visualisation est déduit à partir de *camPos* et *camDirection* :

```
View = lookAt(camPos, camPos + camDirection , camUp );
```

Exemple 8 :

Nous allons utiliser une constante, qui représentera la vitesse du mouvement, pour pouvoir incrémenter ou décrémenter *camPos* à chaque appuie sur les touches W et S respectivement, au début de la boucle d'affichage :

```
float camSpeed = 0.005f;
if (glfwGetKey (window, GLFW_KEY_W) == GLFW_PRESS)
    camPos += camSpeed * camDirection;
if (glfwGetKey (window, GLFW_KEY_S) == GLFW_PRESS)
    camPos -= camSpeed * camDirection;
```

Exercice 3 :

- Faire la même chose pour utiliser les touches A et D pour faire déplacer la caméra vers la droite ou la gauche respectivement, en suivant la direction du vecteur camRight = vec3(1.0f, 0.0f, 0.0f).