OpenGl Course 2020

Keyboard

Use Keyboard

Keyboard

- In order to make graphics interactive we must improve a way to interact with user. There are many ways to interact with users. One of common ways is using the keyboard.
- OpenGL supplies a way to interact with keyboard using the following array:

```
bool keys[256];
```

- Which is predefined in the template we're using.
- Its indexes are the virtual keys in the keyboard. These virtual keys are accessed in OpenGL by writing VK abbreviation before them.

Keyboard

VK_UP

- Means the up arrow button on your keyboard. When it's pressed the boolean :

keys [VK_UP]

is true.

Keyboard

- So we can write now in the DrawGLScene function the following code:

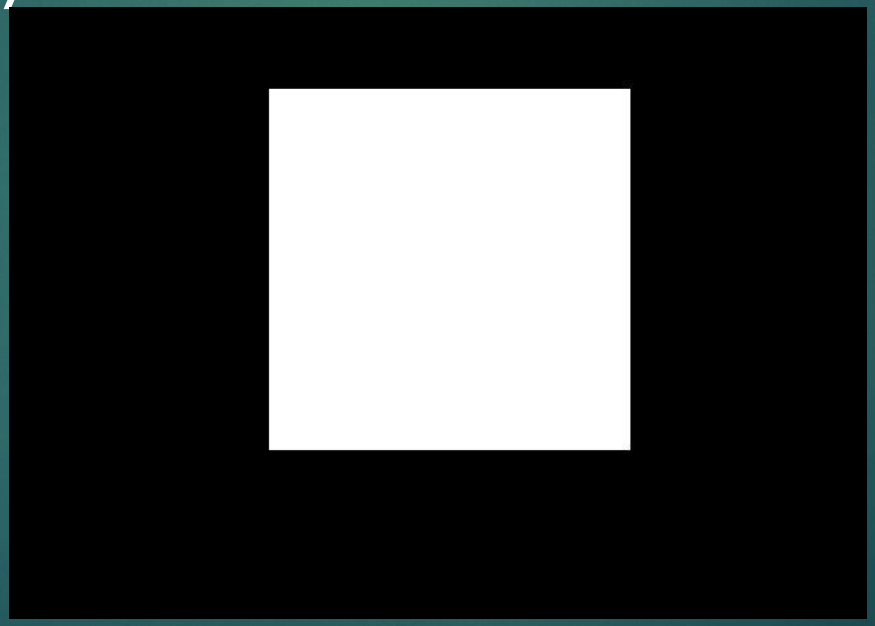
```
if (keys[VK_UP])
    //do something when it's pressed while running
```

Keyboard Task 1

Task 1 ☺

Keyboard Task 1

- Draw a square and move it by keyboard.



Mouse

Use Mouse

Mouse

- In order to use mouse we must add the following cases in the function LRESULT CALLBACK WndProc

```
case WM_MOUSEMOVE:
{
    mouseX = (int)LOWORD(lParam);   mouseY =
(int)HIWORD(lParam);
    isClicked   = (LOWORD(wParam) & MK_LBUTTON) ? true : false;
    isRClicked  = (LOWORD(wParam) & MK_RBUTTON) ? true : false;
    break;
}
case WM_LBUTTONUP:
    isClicked   = false;     break;
case WM_RBUTTONUP:
    isRClicked  = false;   break;
case WM_LBUTTONDOWN:
    isClicked   = true;      break;
case WM_RBUTTONDOWN:
    isRClicked  = true;     break;
```

Mouse

- We mustn't understand now what do these words mean; because they are used just from the system to receive events from the mouse.
- Then we must declare the following variables:

```
int mouseX=0,mouseY=0;
bool isClicked=0,isRClicked=0;
```

- After that we must declare the function:

```
void mouse (int mouseX, int mouseY, bool isClicked, bool isRClicked)
{
    //do something here
}
```

- Be careful with variables' names. Keep their syntax.

Mouse

- Finally call the function mouse in

  DrawGLScene function with passing the four variables we've declared before:

- Now you can enjoy interacting with your scene using mouse ☺.

mouse (mouseX, mouseY,I sClicked, isRClicked);

Mouse

```
GLfloat k=0;
int mouseX=0, mouseY=0;
bool isClicked=false, isRClicked=false;
void mouse (int mouseX, int mouseY,bool isClicked,bool isRClicked)
}
int DrawGLScene(GLvoid)// Here's Where We Do All The Drawing
{
	//....
	mouse(mouseX, mouseY, isClicked, isRClicked);
	{ if (isClicked)   k=float((mouseX-320)*10)/640; glTranslated(k,0,0);
	glBegin(GL_TRIANGLES);
		glVertex2d(1,0);
		glVertex2d(0.5,1);
		glVertex2d(0,0);
	glEnd();
	//...
}
```

Mouse

- Example:
  - In the last example we are moving the triangle right and left when the mouse is moved right and left while clicking on the left button of the mouse.
  - Nothing is changing when the cursor is moved up and down.
- Important Notice: use a mouse instead of touchpad if you're using a laptop.

Viewing Transformations

Viewing Transformations

- The viewing transformation allows you to place the point of observation anywhere you want and look in any direction. Determining the viewing transformation is like placing and pointing a camera at the scene.
- We can control the camera by using the following function:

void **gluLookAt**(GLdouble *eyeX*, GLdouble *eyeY*, GLdouble *eyeZ*,
GLdouble *centerX*, GLdouble *centerY*, GLdouble *centerZ*,
GLdouble *upX*, GLdouble *upY*, GLdouble *upZ*);

Viewing Transformations

- **The parameters:**
  - eyeX, eyeY, eyeZ: specifies the place of the observer, or the camera position in the scene. By default, its in position (0,0,0).
  - centerX, centerY, centerZ: specifies the position where the camera is looking. Which is by default looking in the direction (0,0,-1).
  - upX, upY, upZ: it's the direction that's perpendicular to the camera. It's called the UP direction. Which is by default it pointing to the direction (0,1,0).

Viewing Transformations

- It's very useful and important to understand the parameters of gluLookAt function in order to move in the scene like a moving human.
- The idea behind simulating the way we move is by choosing appropriate values of the parameters of gluLookAt function.
- For example:
  - Eye values usually chosen to be a sin or cos wave in the Y axis. Which will appear like a human walking.
  - Center values are chosen to be like a circle equation to simulate the human looking criteria.
  - Up values are left 0,1,0.
- It's not necessary to use the previous rules but we use them when we want to simulate the human movements in a scene.