Introduction to Graphics Programming and its Applications

繪圖程式設計與應用

Practice

Instructor: Hung-Kuo Chu

Department of Computer Science

National Tsing Hua University

CS4505

Practice

1. Add new custom event
2. Control display loop

Practice 1

Add new custom event

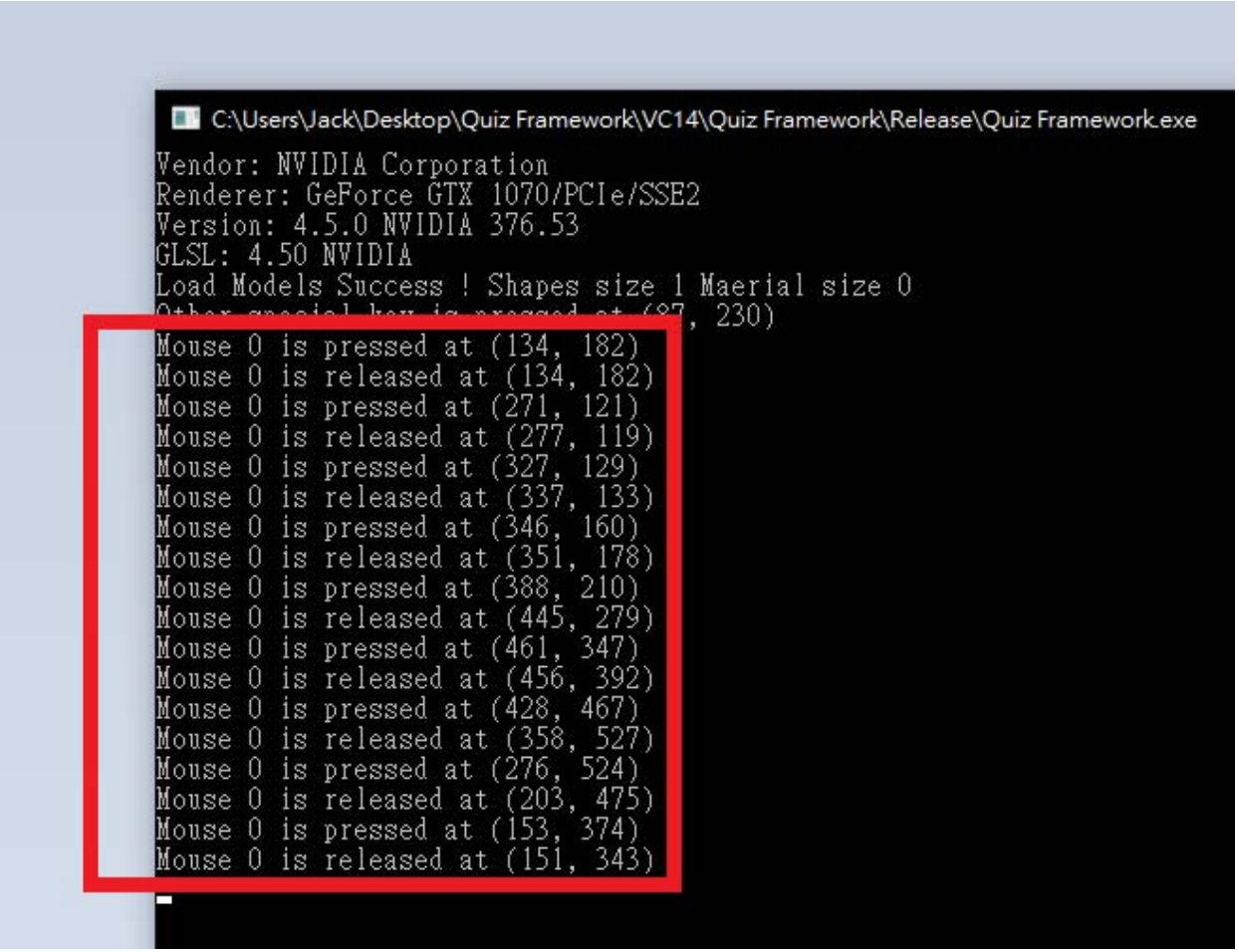
Objective

- Add new mouse event with GLUT
  - Output cursor coordinate when hit left button
  - Output cursor coordinate when release left button
- Add some new option in button list
  - At least 2 new options
  - You can do whatever you want in these new features

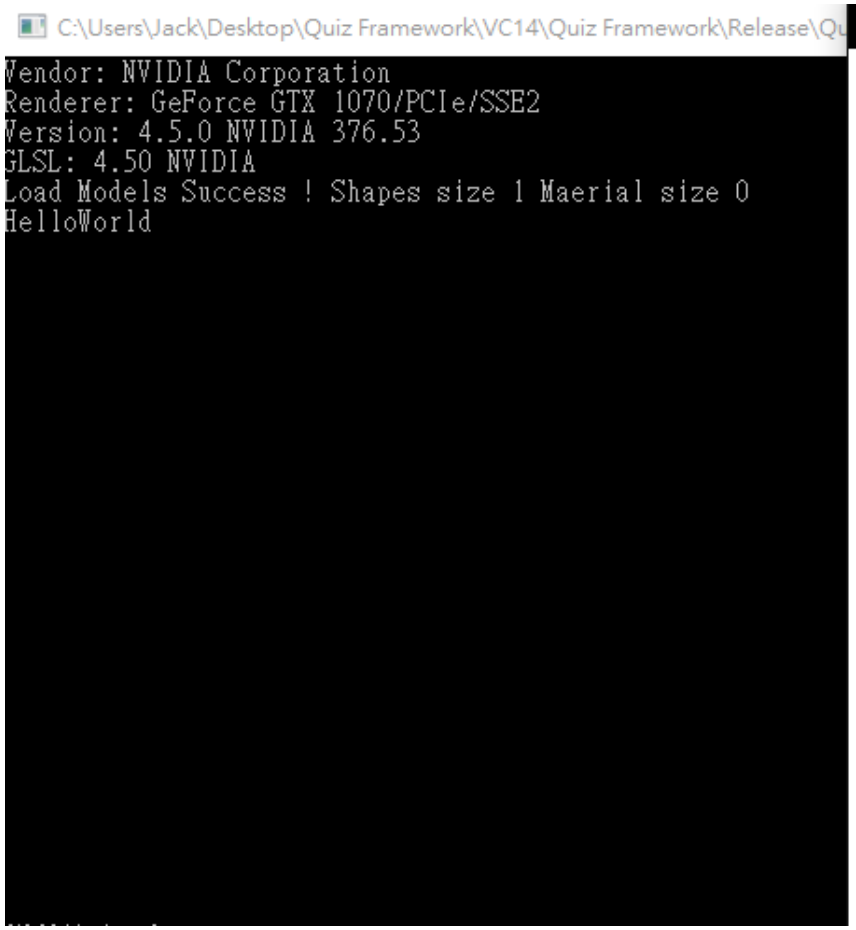
Hint

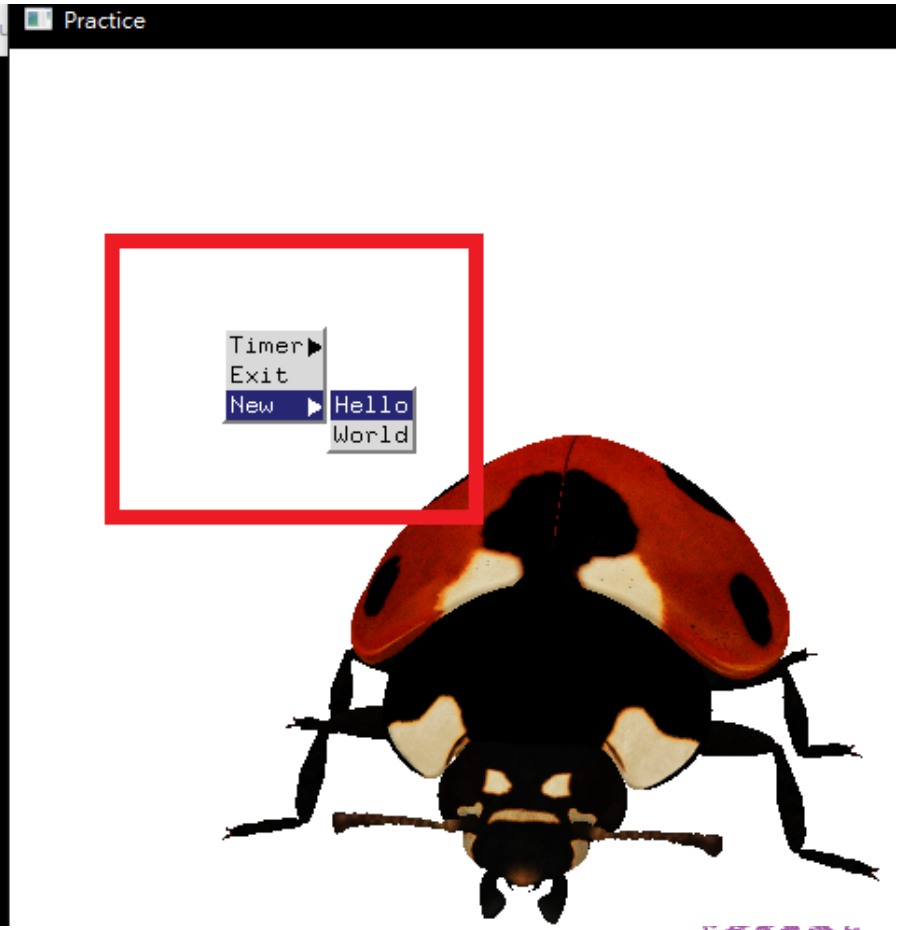
- Window-Specific Callback Registration Functions
- Commonly used call back function :
  - glutMouseFunc
  - glutMouseWheelFunc
  - glutKeyboardFunc
  - glutSpecialFunc
- Register events before entering main loop

Example


C:\Users\Jack\Desktop\Quiz Framework\VC14\Quiz Framework\Release\Quiz Framework.exe
Vendor: NVIDIA Corporation
Renderer: GeForce GTX 1070/PCIe/SSE2
Version: 4.5.0 NVIDIA 376.53
GLSL: 4.50 NVIDIA
Load Models Success ! Shapes size 1 Maerial size 0
, 230)
Mouse 0 is pressed at (134, 182)
Mouse 0 is released at (134, 182)
Mouse 0 is pressed at (271, 121)
Mouse 0 is released at (277, 119)
Mouse 0 is pressed at (327, 129)
Mouse 0 is released at (337, 133)
Mouse 0 is pressed at (346, 160)
Mouse 0 is released at (351, 178)
Mouse 0 is pressed at (388, 210)
Mouse 0 is released at (445, 279)
Mouse 0 is pressed at (461, 347)
Mouse 0 is released at (456, 392)
Mouse 0 is pressed at (428, 467)
Mouse 0 is released at (358, 527)
Mouse 0 is pressed at (276, 524)
Mouse 0 is released at (203, 475)
Mouse 0 is pressed at (153, 374)
Mouse 0 is released at (151, 343)

Example


C:\Users\Jack\Desktop\Quiz Framework\VC14\Quiz Framework\Release\Qu
Vendor: NVIDIA Corporation
Renderer: GeForce GTX 1070/PCIe/SSE2
Version: 4.5.0 NVIDIA 376.53
GLSL: 4.50 NVIDIA
Load Models Success ! Shapes size 1 Maerial size 0
HelloWorld


Practice
Timer
Exit
New
Hello
World

Practice 2

Control display loop

Objective

- Translate the model with keyboard
  - Either any direction or value is fine, but the model should stay visible for TAs to check
  - Any keyboard Input is allowed
- Rotate the model according to the elapsed time
  - Either any speed or direction is allowed, but the model should rotate upon itself but NOT rotate around original axis.

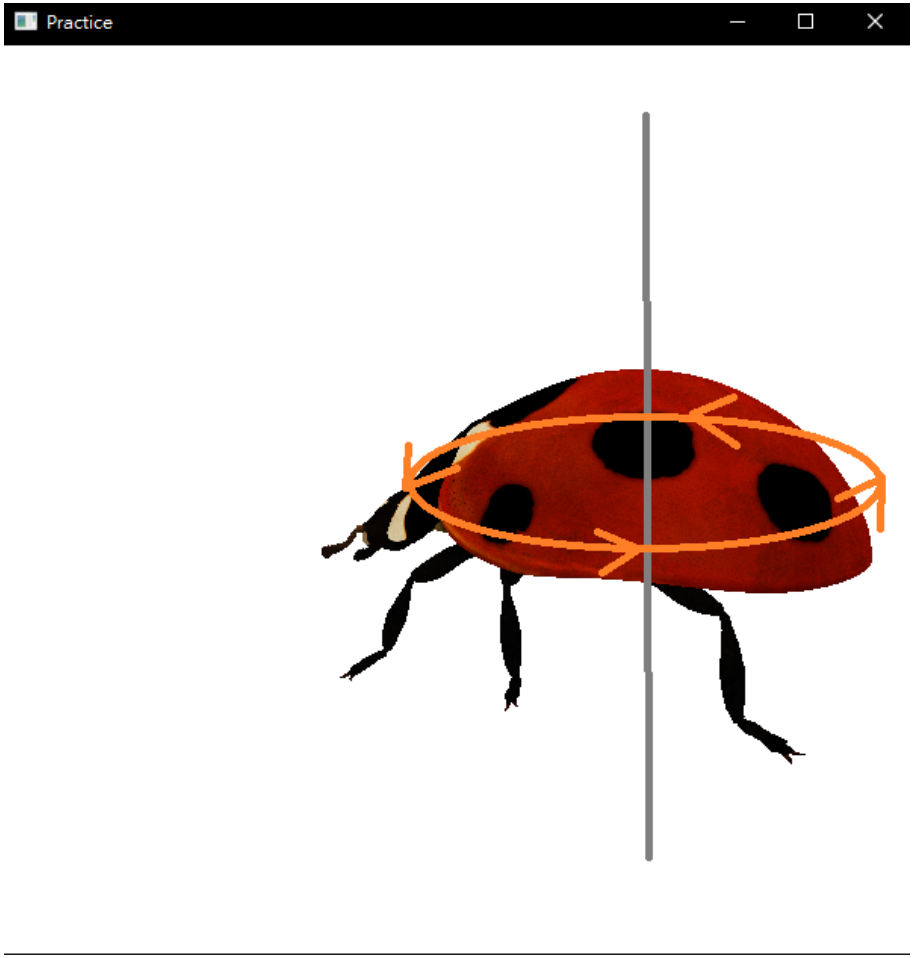
Hint

- Use glutGet(GLUT_ELAPSED_TIME) to get the value of time
- The predefined variable ‘model’ is a matrix which represents model’s transform matrix
- Apply some matrix operation to the variable ‘model’ to achieve the goal
- Check GLM transform API
  - glm::translate
  - glm::rotate
- Be careful of the operation order (first translate then rotate != first rotate then translate)

Hint

- The vertices information are already in the buffer, so all we need to do is try applying the transform matrix to these points.
- Transfer the time value as a parameter to glm::rotate to make the model spin according to time

Example



Answer – Practice 1

- To implement your own mouse event, you have to register a new event listener (glutMouseFunc) and a corresponding event (My_Mouse) in function main

```
// Register GLUT callback functions.
glutDisplayFunc(My_Display);
glutReshapeFunc(My_Reshape);
glutKeyboardFunc(My_Keyboard);
glutSpecialFunc(My_SpecialKeys);
glutTimerFunc(timer_speed, My_Timer, 0);
// Todo
// Practice 1 : Register new GLUT event listner here
// ex. glutXXXXX(my_Func);
// Remind : you have to implement my_Func
glutMouseFunc(My_Mouse);
```

Answer – Practice 1

- You can rename function My_Mouse to whatever name you like
- Implement the event you want to do in My_Mouse

```
void My_Mouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
    {
        if (state == GLUT_DOWN)
        {
            printf("Mouse %d is pressed at (%d, %d)\n", button, x, y);
        }
        else if (state == GLUT_UP)
        {
            printf("Mouse %d is released at (%d, %d)\n", button, x, y);
        }
    }
}
```

Answer – Practice 1

- Parameter ‘button’ means the button you interact with your mouse.
  - GLUT_LEFT_BUTTON : mouse left button
  - GLUT_RIGHT_BUTTON : mouse right button
- Parameter ‘state’ means the state you interact with your mouse.
  - GLUT_DOWN : hit mouse button
  - GLUT_UP : release mouse button
- Parameter ‘x’ and ‘y’, represents the cursor coordinate related to the window

Answer – Practice 1

- To add a new menu, you have to do 3 steps
  1. Define a menu index (optional)
  2. Create a menu in GLUT
  3. Implement the menu function

Define a menu index

- This step is optional, but you still have to give your menu event a index
- In other words, if you want to add 4 new menu events, you have to define 4 different constant for these events

```
//For GLUT to handle
#define MENU_TIMER_START 1
#define MENU_TIMER_STOP 2
#define MENU_EXIT 3
#define MENU_HELLO 4
#define MENU_WORLD 5
```

Create a menu in GLUT

- To put your menu in GLUT main menu, you should do like below

```
int menu_main = glutCreateMenu(My_Menu);
int menu_timer = glutCreateMenu(My_Menu);
int menu_new = glutCreateMenu(My_Menu);
```

- GLUT will return a integer as a index for your menu, these keys will tell GLUT which menu to handle now. (will see then)

Create a menu in GLUT

```
glutSetMenu(menu_main);
glutAddSubMenu("Timer", menu_timer);
glutAddSubMenu("New", menu_new);
glutAddMenuEntry("Exit", MENU_EXIT);

glutSetMenu(menu_timer);
glutAddMenuEntry("Start", MENU_TIMER_START);
glutAddMenuEntry("Stop", MENU_TIMER_STOP);

glutSetMenu(menu_new);                  // Tell GLUT to design the menu which id==menu_new now
glutAddMenuEntry("Hello", MENU_HELLO);  // Add submenu "Hello" in "New"(a menu which index is equal to menu_new)
glutAddMenuEntry("World", MENU_WORLD);  // Add submenu "Hello" in "New"(a menu which index is equal to menu_new)
```

Create a menu in GLUT

- glutSetMenu(int menu)
  - Tell GLUT which menu should it to register now by given menu index
- glutAddSubMenu(const char* name, int subMenu)
  - Add a sub menu to current hierarchy
  - name : submenu's name
  - subMenu : menu index of this submenu
- glutAddMenuEntry(const char* label, int value)
  - Add a menu entry for this menu
  - label : title of this entry
  - value : event index for this entry

Implement the menu function

```
void My_Menu(int id)
{
    switch(id)
    {
    case MENU_TIMER_START:
        if(!timer_enabled)
        {
            timer_enabled = true;
            glutTimerFunc(timer_speed, My_Timer, 0);
        }
        break;
    case MENU_TIMER_STOP:
        timer_enabled = false;
        break;
    case MENU_EXIT:
        exit(0);
        break;
    case MENU_HELLO:
        // do something
        break;
    case MENU_WORLD:
        // do something
        break;
    default:
        break;
    }
}
```

Answer – Practice 2

- Key point
  - How to control keyboard input?
  - How to make model rotate and translate?

Answer – Practice 2

- Keyboard Input
  - In fact, we already have our keyboard input controllable by register a keyboard callback function in main

```
// Register GLUT callback functions.
glutDisplayFunc(My_Display);
glutReshapeFunc(My_Reshape);
glutKeyboardFunc(My_Keyboard);
glutSpecialFunc(My_SpecialKeys);
glutTimerFunc(timer_speed, My_Timer, 0);
```

Answer – Practice 2

- My_Keyboard

```
void My_Keyboard(unsigned char key, int x, int y)
{
    printf("Key %c is pressed at (%d, %d)\n", key, x, y);
    if (key == 'd')
    {
        temp = temp + vec3(0, 0, 1);
    }
    else if (key == 'a')
    {
        temp = temp - vec3(0, 0, 1);
    }
    else if (key == 'w')
    {
        temp = temp + vec3(1, 0, 0);
    }
    else if (key == 's')
    {
        temp = temp - vec3(1, 0, 0);
    }
}
```

Answer – Practice 2

- Parameter
  - Key : which key is being pressed now
  - x : cursor x coordinate
  - y : cursor y coordinate
- Variable : temp
  - ‘temp’ is a global vector type predefined by TA’s, which value is (0.0, 0.0, 0.0) at initial
  - Use temp to record the position where ladybug should be

Answer – Practice 2

- We know how to move the translate vector(temp) by hitting wasd on keyboard now.
- To make ladybug rotate and translate, we need to compute the transformation matrix of ladybug and send it to GPU for computing final display result.
- Fortunately, the operation of sending matrix data to gpu's buffer has already done in this main.cpp, but we won't go into detail here, we just need to focus how to compute ladybug's transformation matrix.

Answer – Practice 2

- The vertices of ladybug are already in GPU's buffer, so we only need to compute our model (M of MVP) matrix and send it to GPU.

- In fact, we should let GPU to do this task in shader program to get better performance.but we havn't learned how to write GLSL for shaders now, so we need to compute these matrix not in shaders but in main.cpp.

- We know what to do now, but how? c++ does not provide any powerful library for us to do such these math works.

Answer – Practice 2

- We know what to do now, but how? c++ does not provide any powerful library for us to do such these math works.
- It’s a super heavy work to implement a series of mathematic operation in c++
- So, why not try using library GLM(OpenGL mathematic)?

Answer – Practice 2

- We know what to do now, but how? c++ does not provide any powerful library for us to do such these math works.
- It’s a super heavy work to implement a series of mathematic operation in c++
- So, why not try using library GLM(OpenGL mathematic)?

GLM

- Library GLM (OpenGL Mathematics)
- GLM provides classes and functions designed and implemented with the same naming conventions and functionalities than GLSL so that *anyone* who knows GLSL, can use GLM as well in C++.
- For more detail, you can check it's website or take a look in slides provided by TA's

GLM - Types

- Commonly-used Types
  - vec, vec2, vec3, vec4 : vec reprents vector, vecN represents N dimension vector
  - mat2, mat3, mat4 : mat reprenets matrix, matN represents NxN matrix
- GLM allow us to apply mathematic operation on types with same dimension.
  - mat3*mat3 (O)
  - vec3+vec3 (O)
  - mat4*mat3 (X)

GLM -Translate

- Build a translation matrix by calling glm::translate

```
//Build translation matrix
mat4 translation_matrix = translate(mat4(), temp);
```

- Translate(A,B) will return a 4x4 matrix, which pose of this object is an 4x4 matrix A and then move along with a 3-dimension vector B
- mat4() here represents an empty matrix

GLM - Rotate

- Build a rotation matrix by calling glm::rotate

```
//Build rotation matrix
GLfloat degree = glutGet(GLUT_ELAPSED_TIME) / 500.0;
vec3 rotate_axis = vec3(0.0, 1.0, 0.0);
mat4 rotation_matrix = rotate(mat4(), degree, rotate_axis);
```

- Use glutGet(GLUT_ELAPSED_TIME) to fetch time information, we will use it as a parameter ‘degree’ for glm::rotate

- Glm::rotate(A,B,C)
  - Return a 4x4 matrix which has original pose A, then rotate around axis C in B degrees.

Model Matrix

- After building translation matrix and rotation matrix, we can apply proper mathematic operation to combine these two matrixs into model matrix

```
//model = matrix_A*matrix_B;
model = translation_matrix*rotation_matrix;
```