**CS409/CS809 Tutorial 14: Mouse Input**

# This lab will show you how to control a game character by moving the mouse. Using the mouse as a pointer to click on things uses the same input functions with very different internal logic.

# Setting up an OpenGL Project with the Obj Library of Files

1. Start Microsoft Visual Studio 2015. Select File / New / Project…. Under the list on the left, choose Installed / Templates / Visual C++. Then choose a Win32 Console Application. Change the name to Tutorial14 or some suitable name and change the location to a hard drive located on the computer (typically C or D). You will need to remember this location for steps 3 and 5. Make sure the “Create folder for project” checkbox is not checked and that the “Add to source control” is not checked. Click OK.
	* If the Chose Location window keeps popping up again, you aren’t allowed to save files at the location you chose. Select a different location, such as the desktop.
	* Do not put your program on one of the external drives like H. The connection is slow, and your program may not compile correctly.
2. Click Next> or Application Settings (they go to the same place). Check the “Empty Project” box. Make sure the “Security development lifecycle check” is not checked. Click Finish.
3. Go to <http://www2.cs.uregina.ca/~anima/409/Terms/201810/Tutorials/index.html> and download the [Tutorial14.zip](http://www2.cs.uregina.ca/~anima/409/Terms/201810/Tutorials/Tutorial14.zip) and [ObjLibrary.zip](http://www2.cs.uregina.ca/~anima/409/Terms/201810/Tutorials/ObjLibrary.zip) files to your project directory.
4. Unzip the 2 zip files. You will have a folder named ObjLibrary and 14 other files.
5. In Visual C++, go to Project/Add Existing Item… and select main14.cpp (and holding control, continue to select), Sleep.h, Sleep.cpp, and GetGlut.h.
6. Ensure that Solution Explorer is visible (View / Solution Explorer). In the Solution Explorer, select Tutorial14 (the project name) and right click it. Choose Add / New Filter. A new folder icon should appear in the Solution Explorer. Name it ObjLibrary. (This creates an imaginary folder in the project, but it would not create a Windows folder on the computer).
7. In the Solution Explorer, right click on the ObjLibrary folder icon and select Add / Existing item. Go into the ObjLibrary folder on Windows and select all files. The files should appear in the Solution Explorer inside the ObjLibrary folder. You should get 26 files.
8. Compile and run the original program. You should see 2 windows: a text console and a graphical window labeled “Mouse Input”. The second window contains a set of XYZ axes, a bucket above a checkerboard grid, and a message in the corner that says “Sample Text”. Exit the windows. Closing the text window will close the graphical window, and usually vice versa.
	* You may see a pop-up window that says "This project is out of date:". If so, click "Yes". This tells Visual Studio to compile your program before trying to run it.
9. For the remainder of the instructions, it is assumed that you compile and run regularly so that you can see what each change does.

# Responding to Mouse Clicks

1. Add function mouseButton to handle the mouse button events after the keyboard function:
void mouseButton (int button, int state, int x, int y)
{
 switch(button)
 {
 case GLUT\_LEFT\_BUTTON:
 break;
 }
}
Add a prototype for this function with the other function prototypes:
void mouseButton (int button, int state, int x, int y);
In the main function, add a call to glutMouseFunc immediately after the call to glutKeyboardFunc:
glutMouseFunc(mouseButton);
2. Add a global variable to count the number of times that the mouse was clicked:
unsigned int click\_count;
In the mouseButton function, increment the variable whenever the left mouse button is used:
 case GLUT\_LEFT\_BUTTON:
 click\_count++;
 break;
3. Add #includes for string and sstream to after the #include for cmath:
#include <string>
#include <sstream>
4. In the drawText function, replace the text that says "Sample Text" with the number of clicks made so far:
 stringstream click\_count\_ss;
 click\_count\_ss << "Click count: " << click\_count;
 font.draw(click\_count\_ss.str(), 16, 16);
5. Compile and run the program, and try clicking the mouse in the window. The click count will go up by 2 for each left click. If you click and hold, you will see the count goes up by 1 when the mouse button is pressed and up by 1 again when it is released.
6. Changed the mouseButton function to only respond when the mouse is pressed:
 case GLUT\_LEFT\_BUTTON:
 if(state == GLUT\_DOWN)
 click\_count++;
 break;

# Responding to Mouse Position

1. Add an #include for Vector2.h before the #include for ObjModel.h:
#include "ObjLibrary/Vector2.h"
2. Declare a global variable of the Vector2 type to represent the mouse position:
Vector2 mouse\_position;
3. Add function mouseMotion to handle mouse movement after the mouseButton function. It should set the mouse position variable:
void mouseMotion (int x, int y)
{
 mouse\_position.set(x, y);
}
Add a prototype for this function with the other function prototypes:
void mouseMotion (int x, int y);
In the main function, add calls to glutMotionFunc to glutPassiveMotionFunc immediately after the call to glutMouseFunc:
 glutMotionFunc(mouseMotion);
 glutPassiveMotionFunc(mouseMotion);
The glutMotionFunc is used when at least one mouse button is pressed and the glutPassiveMotionFunc function is used no mouse button is pressed. In practice, you will normally set them to the same function.
4. In the drawText function, add another line of text to show the mouse position:
 stringstream mouse\_position\_ss;
 mouse\_position\_ss << "Mouse position: " << mouse\_position;
 font.draw(mouse\_position\_ss.str(), 16, 40);
5. Compile and run your program. The mouse coordinates should be in pixels, treating the top-left corner as (0, 0).
6. Change the mouseMotion function to transform the mouse coordinates into the range [-1, 1] along each axis:
 mouse\_position.set(x \* 2.0 / window\_width - 1.0,
 y \* 2.0 / window\_height - 1.0);
Compile and run your program. The mouse coordinates should now always be ±1 at the screen edge and 0 at the screen center. This should still be true even if you resize the window.
7. Change the camera height to depend on the mouse y coordinate:
 gluLookAt(2.0, 3.0 + mouse\_position.y \* 2.0, 4.0,
 0.0, 0.0, 0.0,
 0.0, 1.0, 0.0);
Compile and run your program. Moving the mouse up and down should change the angle that the bucket is shown from.

# Responding to Mouse Movement

1. Add a global variable to represent the rotation of the bucket:
double bucket\_degrees = 0.0;
Update the drawBucket function to rotate the bucket by that angle around the Y-axis:
 glRotated(bucket\_degrees, 0.0, 1.0, 0.0);
2. Rotate the bucket in the init function based on the X coordinate:
 bucket\_degrees -= mouse\_position.x \* 3.0;
Compile and run your program. The bucket should rotate to the left when the mouse is near the left side of the window and to the right when it is near the right side.
3. Change the display function to always place the camera behind the bucket:
 static const double PI = 3.14159265359;
 double bucket\_radians = bucket\_degrees \* PI / 180;
 Vector3 camera(2.0, 3.0 + mouse\_position.y \* 2.0, 4.0);
 camera.rotateY(bucket\_radians);
 glLoadIdentity();
 gluLookAt(camera.x, camera.y, camera.z,
 0.0, 0.0, 0.0,
 0.0, 1.0, 0.0);
Now the world should appear to turn while the bucket stays stationary.
4. Make sure that you can close the program with the [ESC] key before continuing.
5. Add a command to the update function to move the pointer back to the middle of the window. Also increase the bucket rotation speed:
 bucket\_degrees -= mouse\_position.x \* 30.0;
 glutWarpPointer(window\_width / 2, window\_height / 2);
Compile and run your program. You should now be able to turn the bucket right and left by moving the mouse. However, it will not be possible to move the mouse out of the program window, so you will have to close it with te [ESC] key.
	* Warning: Never call glutWarpPointer in the mouseMotion function. If you do, it will cause an infinite series of mouse position updates and your program will lock up.
6. It is annoying to not be able to click on things outside the window. Unfortunately, there is no good way to solve it, so we will hide it instead. Add a command in the initDisplay function to put the game in full screen mode:
 glutFullScreen();
Now there is nothing else to click on, so it doesn't matter that it you can't.
7. The cursor arrow stuck in the middle of the screen looks kind of silly. Add a command in the initDisplay function to replace it with a crosshair cursor:
 glutSetCursor(GLUT\_CURSOR\_CROSSHAIR);

# Exercises

1. Change the update function to only change the X coordinate for the mouse. The Y coordinate should keep its existing value.
2. Make the mouse cursor entirely invisible (GLUT\_CURSOR\_NONE). A complete list of cursor types can be found at <https://www.opengl.org/resources/libraries/glut/spec3/node28.html>.
3. Change the camera height to vary by ±5 instead of ±2. What happens if you move the camera below the checkerboard? Why?
	* Hint: Try commenting out glEnable(GL\_CULL\_FACE); in the initDisplay function and observe the effect.
4. Display a message on the screen while the right mouse button is held down and not otherwise.