3D Graphics Programming Tools

OpenGL Interactive Events

Dr. Xianhui Cherry Che

x.che@qmul.ac.uk

Learning Objectives

- Practice the basic use of input devices to trigger certain events within OpenGL
- Learn basic practice of implementing user interaction in OpenGL product

Topics

- Mouse Events
- Keyboard Events
- Menus

Objectives

- Learn to build interactive programs using GLUT callbacks
  - Mouse
  - Keyboard
- Introduce menus in GLUT

2

Recap: Event Types

- **Window:** resize, expose, iconify
- **Mouse:** click one or more buttons
- **Motion:** move mouse
- **Keyboard:** press or release a key
- **Idle:** nonevent
  - Define what should be done if no other event is in the queue

GLUT Callbacks

GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)

- **`glutDisplayFunc`** *(Already learned so far)*
- **`glutIdleFunc`** *(Already learned so far)*
- **`glutMouseFunc`**
- **`glutMotionFunc, glutPassiveMotionFunc`**
- **`glutKeyboardFunc, glutKeyboardUpFunc`**
  `glutSpecialFunc, glutSpecialUpFunc`
- **`glutReshapeFunc`** *(Already learned so far)*

Mouse Callback

```
glutMouseFunc(mymouse);
void mymouse(GLint button, GLint state, GLint x, GLint y)
 {}
```

- Returns
  - which button caused the event:
    - **GLUT_LEFT_BUTTON**
    - **GLUT_MIDDLE_BUTTON**
    - **GLUT_RIGHT_BUTTON**
  - state of that button
    - **GLUT_UP** (release the mouse)
    - **GLUT_DOWN** (press the mouse)
  - Position in window

Terminating a Program

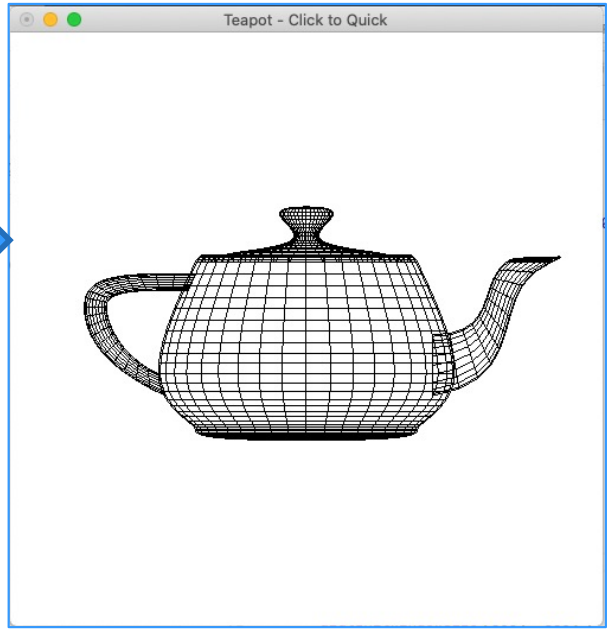
- An OpenGL program can be terminated with `exit(0);`
  - With the inclusion of **`#include <cstdlib>`**
- For example, to use a mouse event to close the program:

```
void mouse(int btn, int state, int x, int y)
{
  if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
    exit(0);
}
```

Example 1 – Zoom with Mouse Click

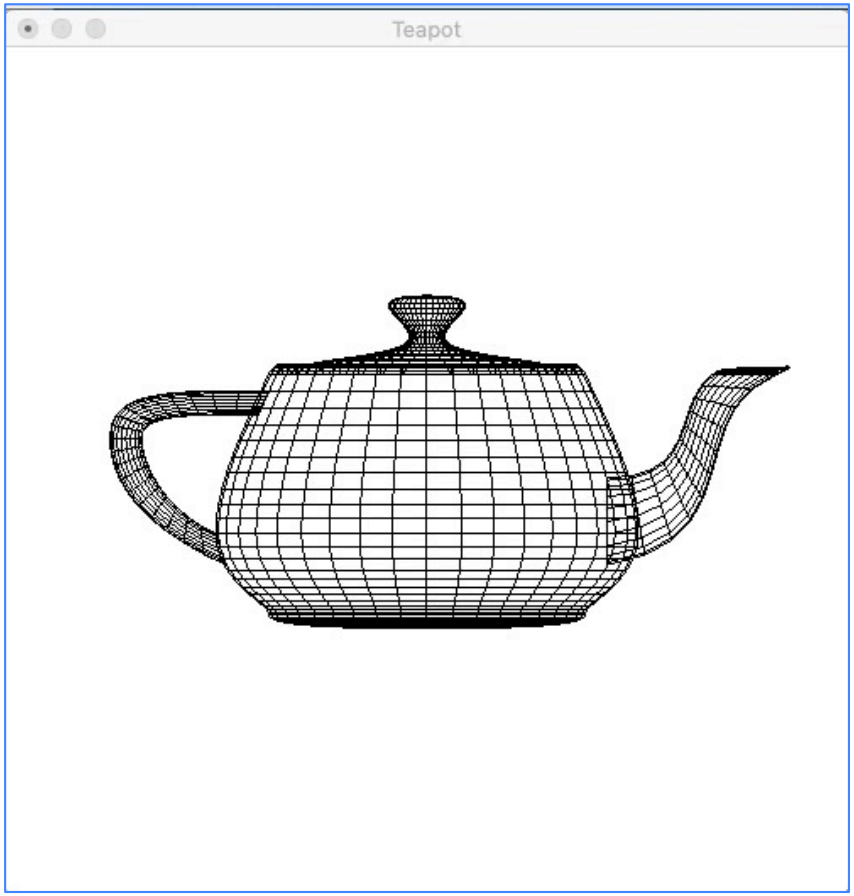
```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);
}
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();
}
void mymouse(int btn, int state, int x, int y){
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
exit(0);
}

int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot");
    glutDisplayFunc(mydisplay);
    glutMouseFunc(mymouse);
    myInit();
    glutMainLoop();
}
```

Left click on the mouse will close the window.



Example 2 – Zoom with Mouse Click

- Click a mouse to zoom in a teapot, click again to stop zooming, and repeat…



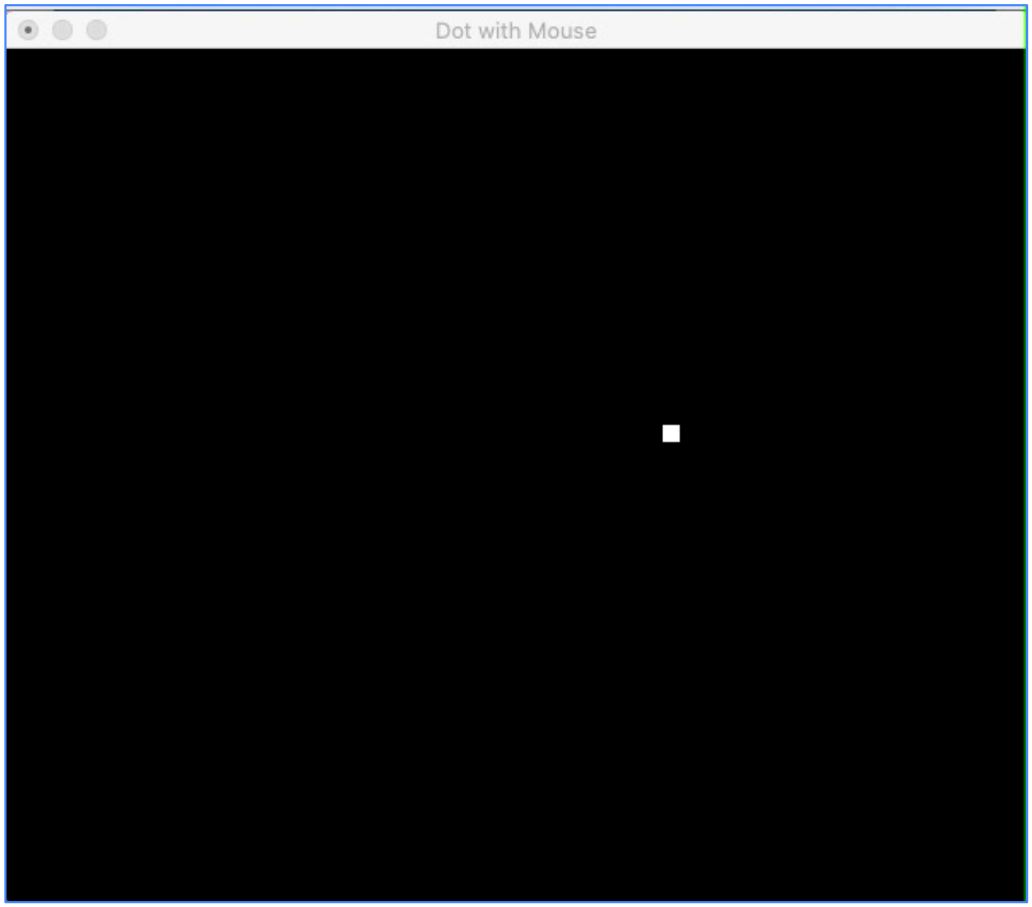
Example 2 – Zoom with Mouse Click

```
#include <GLUT/glut.h>
GLboolean animated = GL_FALSE;
GLdouble zoom = 1.0;

void myInit(void) {
   glClearColor(0.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);

}
void myidle() {
    if (!animated) return;
    if (zoom > 0) zoom -= 0.005;
    glutPostRedisplay();
}
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D (-zoom, zoom, -zoom, zoom);
    glutWireTeapot(0.5);
    glutSwapBuffers();
}
```

```
void mymouse
(GLint button, GLint state, GLint x, GLint y)
{
  if (button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        animated = !animated;
}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(W, H);
    glutCreateWindow("Teapot");
    glutDisplayFunc(mydisplay);
    glutIdleFunc(myidle);
    glutMouseFunc(mymouse);
    myInit();
    glutMainLoop();
}
```

Example 3 – Draw Dots with Mouse

- Draw a dot where the mouse clicks



Example 3 – Draw Dots with Mouse Click

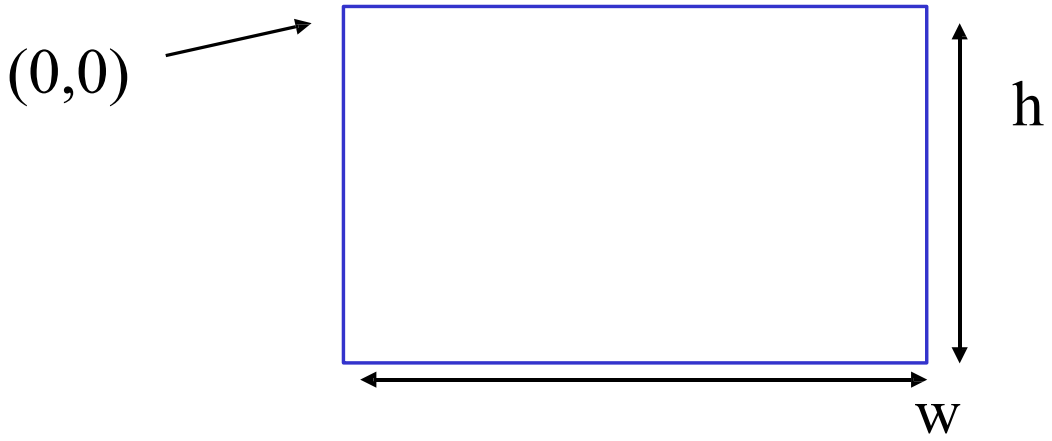
```
#include <GLUT/glut.h>
GLdouble W = 600.0;
GLdouble H = 500.0;
void myInit(void) {
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D (0, W, 0, H);}
void mydisplay(){}
void mymouse(GLint button, GLint state, GLint x, GLint y) {
    if (button==GLUT_LEFT_BUTTON && state==GLUT_DOWN) {
        glPointSize(10);
        glBegin(GL_POINTS);
        glVertex2i(x, H-y);
        glEnd();
        glFlush();}}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(W, H);
    glutCreateWindow("Dot with Mouse");
    glutDisplayFunc(mydisplay);
    glutMouseFunc(mymouse);
    myInit();
    glutMainLoop();}
```

View in traditional coordinates

Invert y position

Positioning in Screen

- The position in the screen window is usually measured in pixels with the origin at the top-left corner
  - Consequence of refresh done from top to bottom
- OpenGL uses a world coordinate system with origin at the bottom left
  - Must invert y coordinate returned by callback by height of window
  - $y = h - y;$



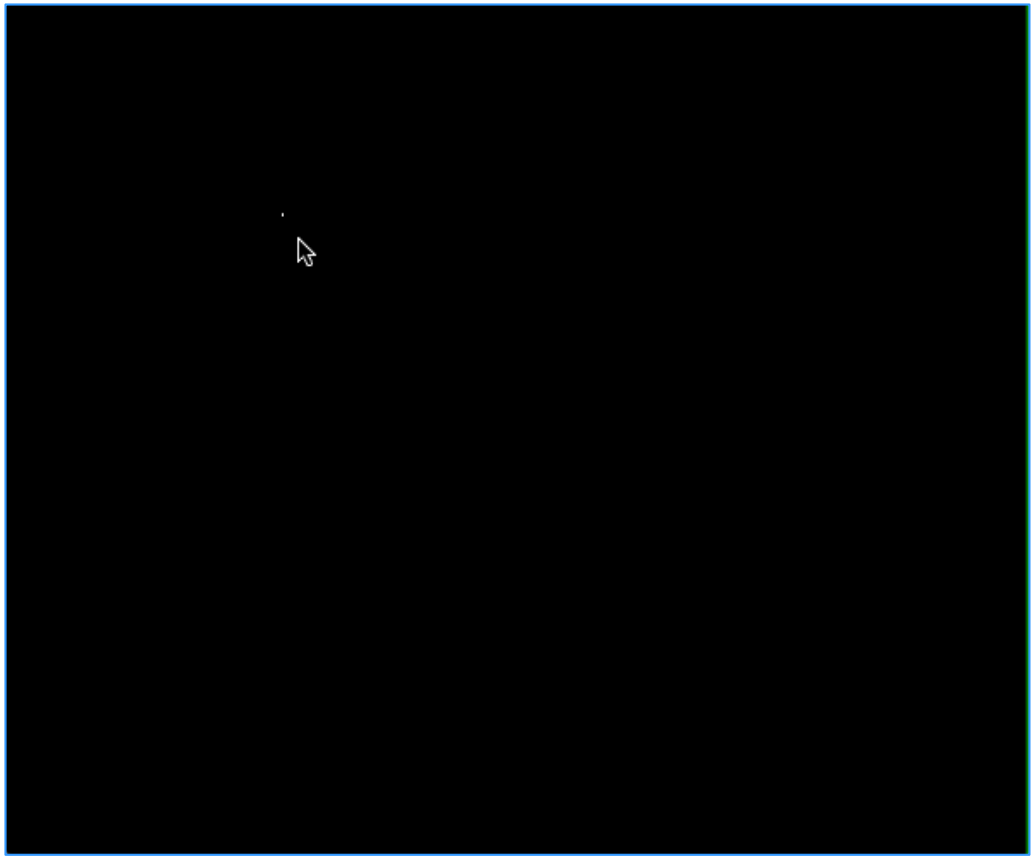
Motion Callback

- The motion callback for a window is called when the mouse moves within the window while one or more mouse buttons are pressed.
- It passes the position of the mouse.

```
glutMotionFunc(myMouseMotion);

void myMouseMotion(int x, int y){}
```

Example 1 – Draw Lines with Mouse

- Use the initial position of a mouse click at the starting point to draw a series of lines



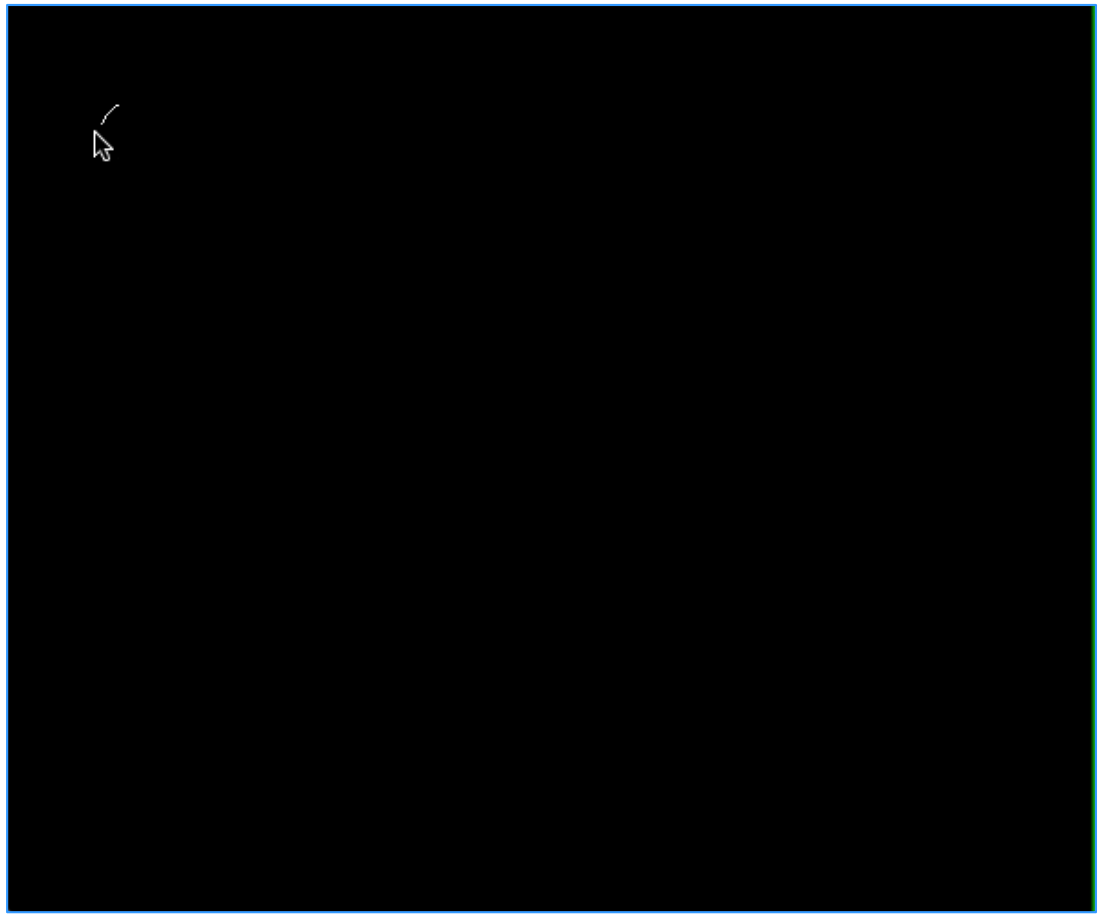
Example 1 – Draw Lines with Mouse

```
#include <GLUT/glut.h>
GLdouble W = 600.0;
GLdouble H = 500.0;
GLint xx = 0;
GLint yy = 0;
void mymouse(GLint button, GLint state, GLint x, GLint y) {
    glClear(GL_COLOR_BUFFER_BIT);
    if (button==GLUT_LEFT_BUTTON && state==GLUT_DOWN) {
        xx = x;
        yy = y;
    }
}
void myMouseMotion(GLint x, GLint y) {
    glBegin(GL_LINES);
    glVertex2i(xx, H-yy);
    glVertex2i(x, H-y);
    glEnd();
    glFlush();
}
void mydisplay(){
}
```

```
void myInit(void) {
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D (0, W, 0, H);
   glPointSize(10);
}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(W, H);
    glutCreateWindow("Dot with Mouse");
    glutDisplayFunc(mydisplay);
    glutMouseFunc(mymouse);
    glutMotionFunc(myMouseMotion);
    myInit();
    glutMainLoop();}
```

Example 2 – Drawing Along with Mouse

- Draw a continuous line along with mouse movement



Example 2 – Drawing Along with Mouse

```
#include <GLUT/glut.h>
GLdouble W = 600.0;
GLdouble H = 500.0;
GLint xx = 0;
GLint yy = 0;
void mymouse(GLint button, GLint state, GLint x, GLint y) {
    glClear(GL_COLOR_BUFFER_BIT);
    if (button==GLUT_LEFT_BUTTON && state==GLUT_DOWN) {
        xx = x;
        yy = y;
    }
}
void myMouseMotion(GLint x, GLint y) {
    glBegin(GL_LINES);
    glVertex2i(xx, H-yy);
    glVertex2i(x, H-y);
    xx=x;
    yy=y;
    glEnd();
    glFlush();
}
void mydisplay(){
}
```

```
void myInit(void) {
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D (0, W, 0, H);
   glPointSize(10);
}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(W, H);
    glutCreateWindow("Dot with Mouse");
    glutDisplayFunc(mydisplay);
    glutMouseFunc(mymouse);
    glutMotionFunc(myMouseMotion);
    myInit();
    glutMainLoop();}
```

Topics

- Mouse Events
- Keyboard Events
- Menus

Keyboard Callback

```
glutKeyboardFunc(mykey);
void mykey(unsigned char key, int x, int y)
{
 if(key == 'Q' | key == 'q')
    exit(0);
}
```

-Returns the ASCII code of key pressed

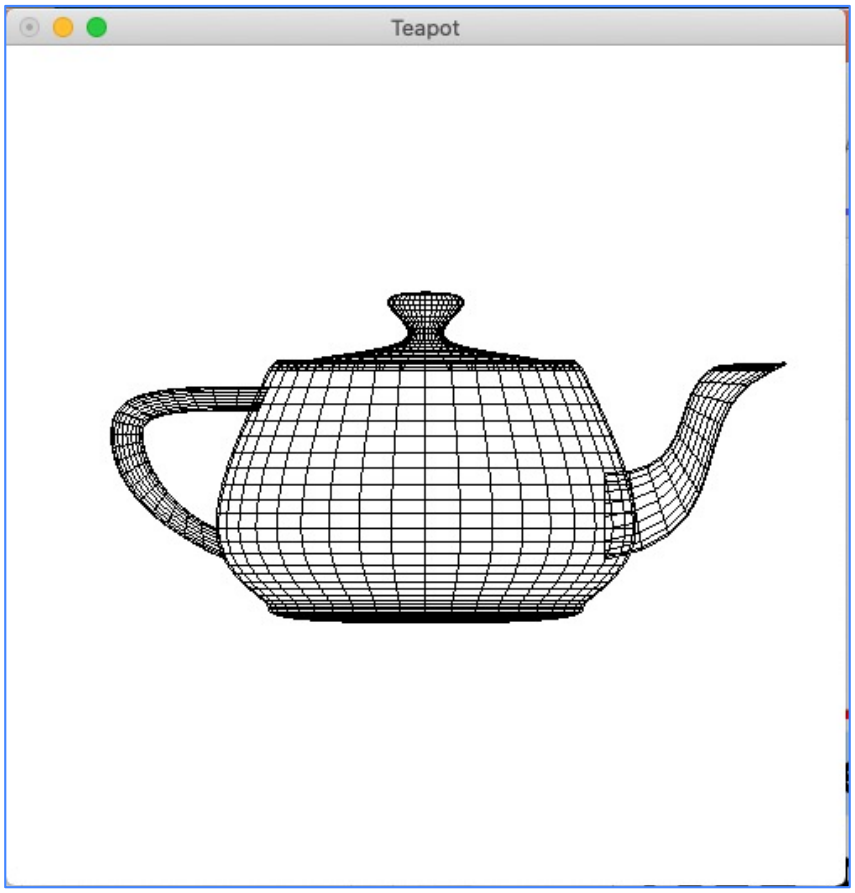
-The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. (Can be useful for gaming.)

ASCII Table

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

Example – Press “Q” to Quit

- Press the “Q” or “q” key to exit the program.



Example – Press “Q” to Quit

```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);}
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();}
void mykey(unsigned char key, int x, int y) {
    if(key == 'Q' | key == 'q') exit(0);}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot");
    glutDisplayFunc(mydisplay);
    glutKeyboardFunc(mykey);
    myInit();
    glutMainLoop();
}
```

Unsigned Char vs Signed Char

- Each char is stored in one byte.
- Signed char range: -128 ~ 127
  - Any char corresponds to an integer value, so a signed char can be used as an integer including a negative integer.
  - Signed char limits the range of chars that can be used.
  - Can also be used in special occasions such as encryption.
- Unsigned char range: 0 ~ 255
  - Ideal for computer graphics, such as presenting RGB colours.

Special Keyboard Callback

- `glutSpecialFunc` is used to detect non-ASCII key strokes.

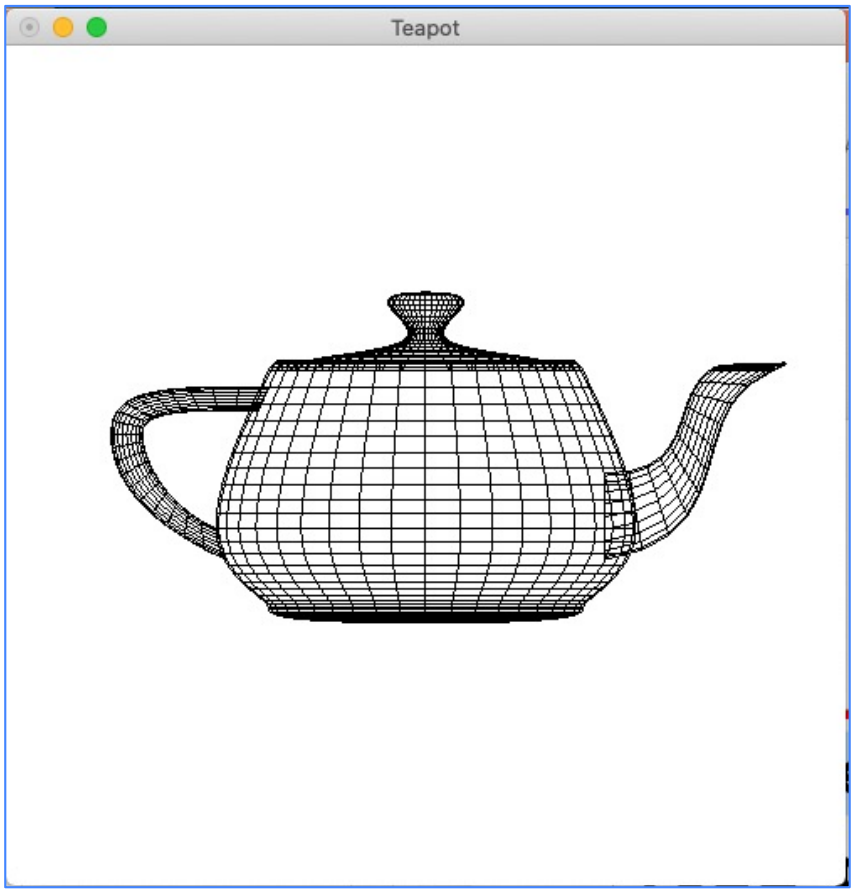
`GLUT_KEY_F1`	F1 function key.
`GLUT_KEY_F2`	F2 function key.
`GLUT_KEY_F3`	F3 function key.
`GLUT_KEY_F4`	F4 function key.
`GLUT_KEY_F5`	F5 function key.
`GLUT_KEY_F6`	F6 function key.
`GLUT_KEY_F7`	F7 function key.
`GLUT_KEY_F8`	F8 function key.
`GLUT_KEY_F9`	F9 function key.
`GLUT_KEY_F10`	F10 function key.
`GLUT_KEY_F11`	F11 function key.
`GLUT_KEY_F12`	F12 function key.

`GLUT_KEY_LEFT`	Left directional key.
`GLUT_KEY_UP`	Up directional key.
`GLUT_KEY_RIGHT`	Right directional key.
`GLUT_KEY_DOWN`	Down directional key.
`GLUT_KEY_PAGE_UP`	Page up directional key.
`GLUT_KEY_PAGE_DOWN`	Page down directional key.
`GLUT_KEY_HOME`	Home directional key.
`GLUT_KEY_END`	End directional key.
`GLUT_KEY_INSERT`	Inset directional key.

Note that the escape, backspace, and delete keys are generated as an ASCII character which can be dealt with by the **`glutKeyboardFunc`**.

Example – Press Left Arrow Key to Quit

- Press the left arrow key to exit the program.



Example – Press Left Arrow Key to Quit

```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);}
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();}
void mySpecialKey(int key, int x, int y) {
    if(key == GLUT_KEY_LEFT) exit(0);}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot");
    glutDisplayFunc(mydisplay);
    glutSpecialFunc(mySpecialKey);
    myInit();
    glutMainLoop();
}
```

Note the first parameter is an integer unlike the unsigned char in the normal keyboard callback.

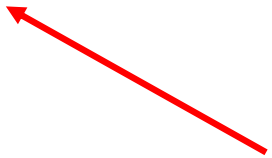
Modifier Keys

- Can also check if one of the modifiers
  - `GLUT_ACTIVE_SHIFT`
  - `GLUT_ACTIVE_CTRL`
  - `GLUT_ACTIVE_ALT`

  is pressed by `glutGetModifiers()`

Example – Detecting Shift Key Press

```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);}
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();}
void mySpecialKey(int key, int x, int y) {
    if ((key == GLUT_KEY_LEFT) && (glutGetModifiers() == GLUT_ACTIVE_SHIFT))
            exit(0);}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot");
    glutDisplayFunc(mydisplay);
    glutSpecialFunc(mySpecialKey);
    myInit();
    glutMainLoop();
}
```



The window will only close if the Left key and the Shift key are pressed at the same time.

Topics

- Mouse Events
- Keyboard Events
- Menus

Toolkits and Widgets

- Most window systems provide a toolkit or library of functions for building user interfaces that use special types of windows called widgets
- Widget sets include tools such as
  - Menus
  - Slidebars
  - Dials
  - Input boxes
- But toolkits tend to be platform dependent
- GLUT provides just a few widgets including menus

Menus

- GLUT supports pop-up menus
  - A menu can have submenus
- Three steps
  1. Define entries for the menu
  2. Define action for each menu item
     - Action is carried out if the entry is selected
  3. Attach menu to a mouse button

Defining a Simple Menu

In the OpenGL program initialisation stage:

```
GLint menu_id = glutCreateMenu(mymenu);
glutAddMenuEntry("clear screen", 1);
glutAddMenuEntry("exit", 2);
glutAttachMenu(GLUT_LEFT_BUTTON);
```

Menu Actions

- Menu callback

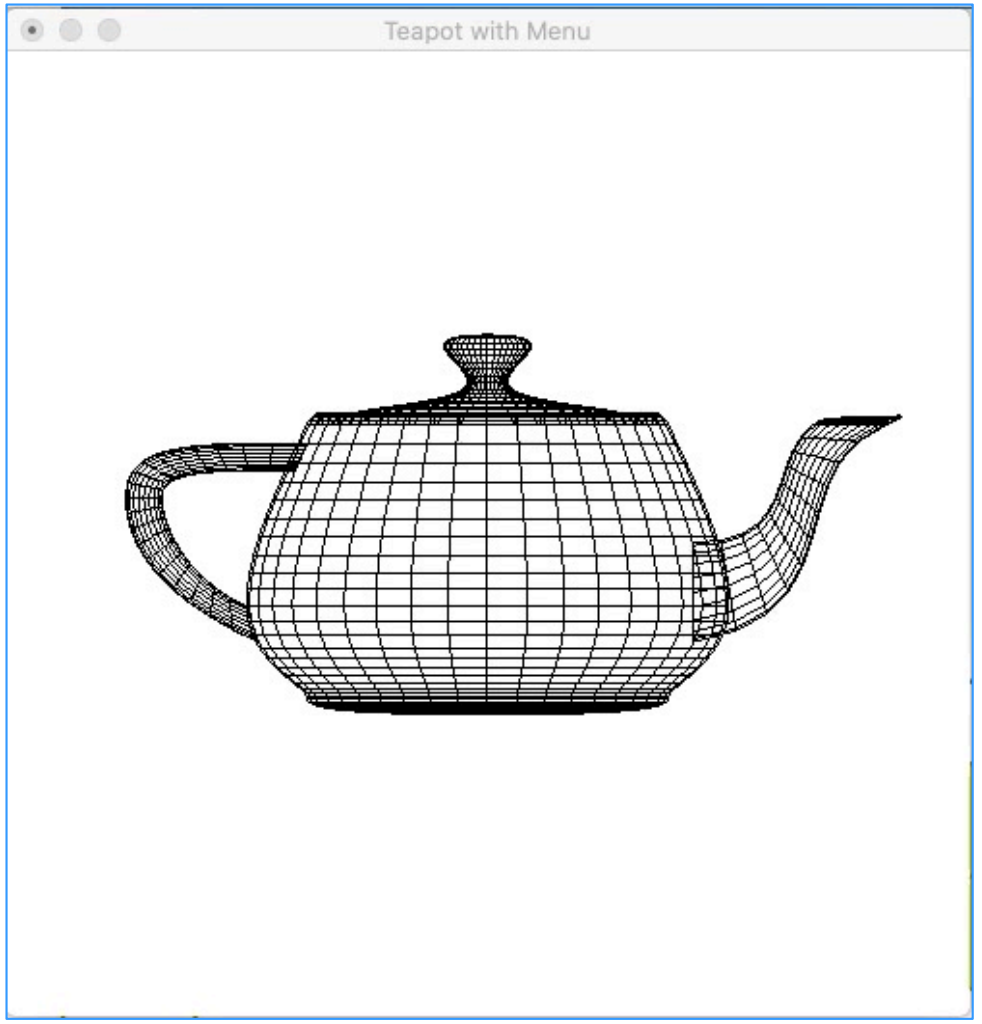
```
void mymenu(int id)
{
 if(id == 1) ...//Do something something;
 if(id == 2) ...//Do something something;
}
```

- The id of the menu is passed on to the function during callback.

Example – Menu

- A list of menu upon left click of mouse

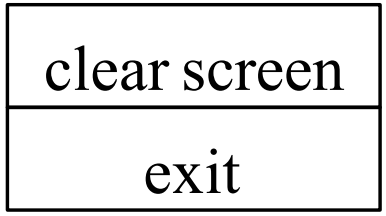
clear screen
exit



Example – Menu

```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);
}
void mydisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();
}
void mymenu(int id)
{
    if(id == 1) {
            glClear(GL_COLOR_BUFFER_BIT);
            glFlush();
            }

    if(id == 2) exit(0);
}
```

```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot with Menu");
    glutDisplayFunc(mydisplay);
    myInit();
    GLint menu_id = glutCreateMenu(mymenu);
    glutAddMenuEntry("clear screen", 1);
    glutAddMenuEntry("exit", submenu_id);
    glutAttachMenu(GLUT_LEFT_BUTTON);
    glutMainLoop();
}
```



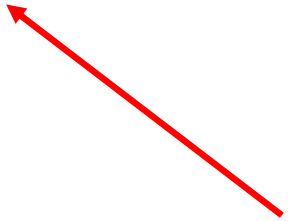
Menu with Submenu

- Declare a submenu in the same way as a menu, and declare a submenu id

```
GLint submenu_id = glutCreateMenu(mysubmenu);
glutAddMenuEntry(...);
glutAddMenuEntry(...);
glutAttachMenu(GLUT_LEFT_BUTTON);
```

- Add submenus to the main menu by

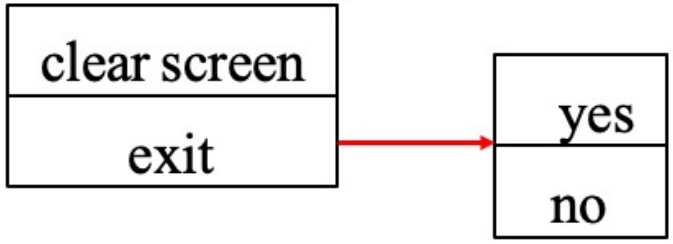
```
glutAddSubMenu(char *submenu_name, submenu id)
```

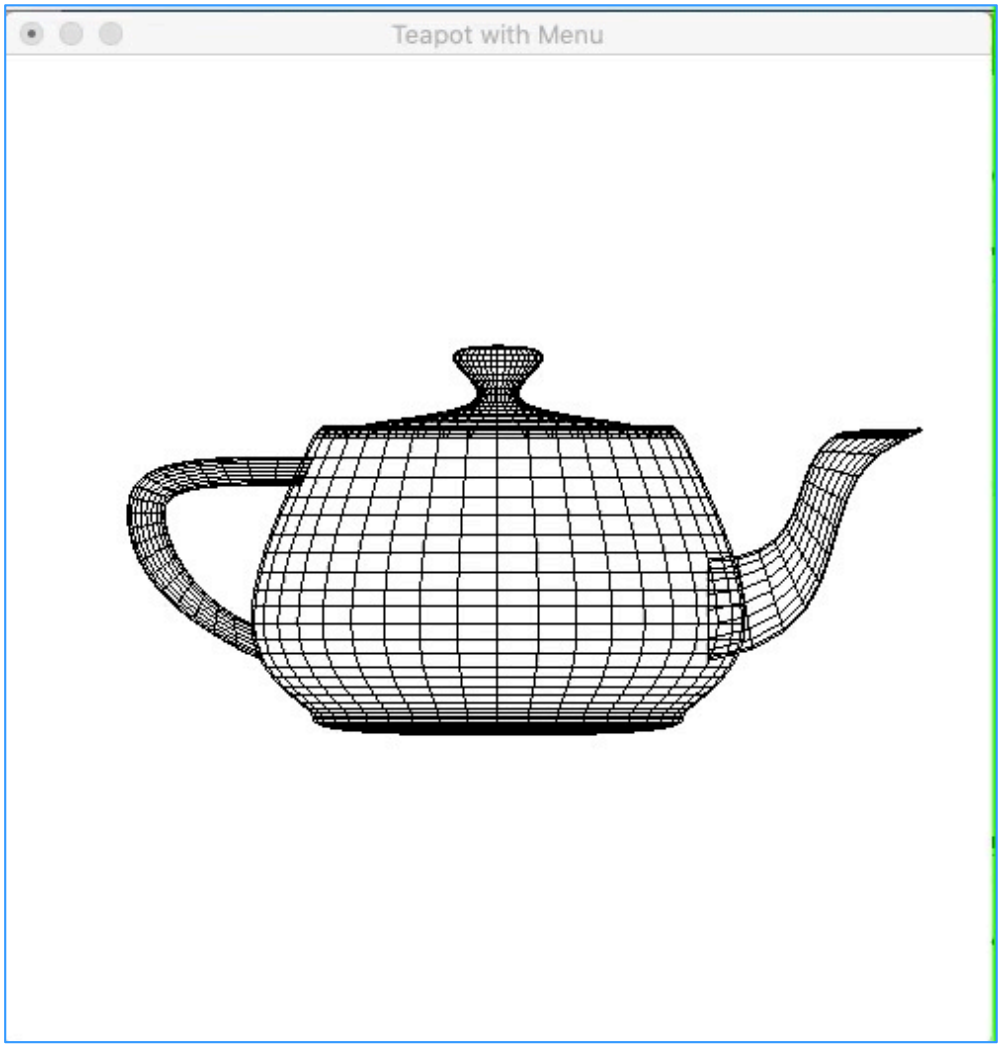


Entry in parent menu

Example – Menu and Submenu

- A list of menu with submenu upon left click of mouse

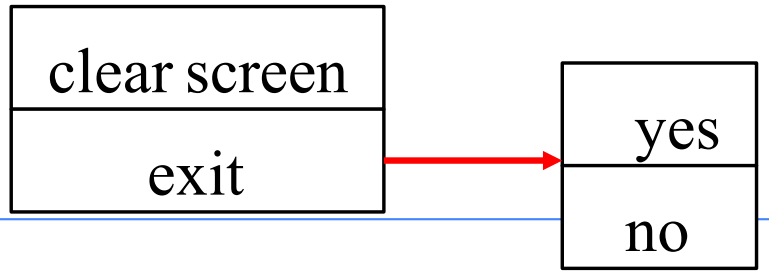




Example – Menu and Submenu

```
#include <GLUT/glut.h>
#include <cstdlib>
void myInit(void) {
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0, 0.0, 0.0);
}
void mydisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    glutWireTeapot(0.5);
    glFlush();
}
void mymenu(int id) {
    if(id == 1) {
        glClear(GL_COLOR_BUFFER_BIT);
        glFlush();
        }
}
void mysubmenu(int id) {
    if(id == 1) exit(0);
    if(id == 2) return;
}
```

```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Teapot with Menu");
    glutDisplayFunc(mydisplay);
    myInit();
    GLint submenu_id = glutCreateMenu(mysubmenu);
    glutAddMenuEntry("yes", 1);
    glutAddMenuEntry("no", 2);
    glutAttachMenu(GLUT_LEFT_BUTTON);
    GLint menu_id = glutCreateMenu(mymenu);
    glutAddMenuEntry("clear screen", 1);
    glutAddSubMenu ("exit", submenu_id);
    glutAttachMenu(GLUT_LEFT_BUTTON);
    glutMainLoop();
}
```



Questions?

x.che@qmul.ac.uk