glutInit is used to initialize the GLUT library.

Usage

```
void glutInit(int *argcp, char **argv);
```

Description

glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options.

glutInit also processes command line options, but the specific options parse are window system dependent.

---

glutInitDisplayMode sets the initial display mode.

Usage

```
void glutInitDisplayMode(unsigned int mode);
```

mode
: Display mode, normally the bitwise OR-ing of GLUT display mode bit masks. See values below:

GLUT_RGBA
: Bit mask to select an RGBA mode window. This is the default if neither GLUT_RGBA nor GLUT_INDEX are specified.

GLUT_RGB
: An alias for GLUT_RGBA.

GLUT_INDEX
: Bit mask to select a color index mode window. This overrides GLUT_RGBA if it is also specified.

GLUT_SINGLE
: Bit mask to select a single buffered window. This is the default if neither GLUT_DOUBLE or GLUT_SINGLE are specified.

GLUT_DOUBLE
: Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.

GLUT_ACCUM
: Bit mask to select a window with an accumulation buffer.

GLUT_ALPHA
: Bit mask to select a window with an alpha component to the color buffer(s).

GLUT_DEPTH
: Bit mask to select a window with a depth buffer.

GLUT_STENCIL
: Bit mask to select a window with a stencil buffer.

GLUT_MULTISAMPLE
: Bit mask to select a window with multisampling support. If multisampling is not available, a non-multisampling window will automatically be chosen. Note: both the OpenGL client-side and server-side implementations must support the GLX_SAMPLE_SGIS extension for multisampling to be available.

GLUT_STEREO
: Bit mask to select a stereo window.

GLUT_LUMINANCE
: Bit mask to select a window with a ``luminance'' color model. This model provides the functionality of OpenGL's RGBA color model, but the green and blue components are not maintained in the frame buffer. Instead each pixel's red component is converted to an index between zero andglutGet(GLUT_WINDOW_COLORMAP_SIZE)-1 and looked up in a per-window color map to determine the color of pixels within the window. The initial colormap of GLUT_LUMINANCE windows is initialized to be a linear gray ramp, but can be modified with GLUT's colormap routines.

Description

The initial display mode is used when creating top-level windows, subwindows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

Note that GLUT_RGBA selects the RGBA color model, but it does not request any bits of alpha (sometimes called an alpha buffer or destination alpha) be allocated. To request alpha, specify GLUT_ALPHA. The same applies to GLUT_LUMINANCE.

---

glutInitWindowPosition and glutInitWindowSize set the initial window position and size respectively.

Usage

```
void glutInitWindowSize(int width, int height);
void glutInitWindowPosition(int x, int y);
```

width
: Width in pixels.

height
: Height in pixels.

x
: Window X location in pixels.

y
: Window Y location in pixels.

Description

Windows created by glutCreateWindow will be requested to be created with the current initial window position and size.

The initial value of the initial window position GLUT state is -1 and -1. If either the X or Y component to the initial window position is negative, the actual window position is left to the window system to determine. The initial value of the initial window size GLUT state is 300 by 300. The initial window size components must be greater than zero.

The intent of the initial window position and size values is to provide a suggestion to the window system for a window's initial size and position. The window system is not obligated to use this information. Therefore, GLUT programs should not assume the window was created at the specified size or position. A GLUT program should use the window's reshape callback to determine the true size of the window.

---

glutCreateWindow creates a top-level window.

Usage

```
int glutCreateWindow(char *name);
```

name
ASCII character string for use as window name.

Description glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

Implicitly, the current window is set to the newly created window.

Each created window has a unique associated OpenGL context. State changes to a window's associated OpenGL context can be done immediately after the window is created.

The display state of a window is initially for the window to be shown. But the window's display state is not actually acted upon until glutMainLoop is entered. This means until glutMainLoop is called, rendering to a created window is ineffective because the window can not yet be displayed.

The value returned is a unique small integer identifier for the window. The range of allocated identifiers starts at one. This window identifier can be used when calling glutSetWindow.

---

glutDisplayFunc sets the display callback for the current window.

Usage

```
void glutDisplayFunc(void (*func)(void));
```

func
The new display callback function.

Description

glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and (if no overlay display callback is registered) the layer in use is set to the normal plane. The display callback is called

with no parameters. The entire normal plane region should be redisplayed in response to the callback (this includes ancillary buffers if your program depends on their state).

GLUT determines when the display callback should be triggered based on the window's redisplay state. The redisplay state for a window can be either set explicitly by calling glutPostRedisplay or implicitly as the result of window damage reported by the window system. Multiple posted redisplays for a window are coalesced by GLUT to minimize the number of display callbacks called.

When an overlay is established for a window, but there is no overlay display callback registered, the display callback is used for redisplaying both the overlay and normal plane (that is, it will be called if either the redisplay state or overlay redisplay state is set). In this case, the layer in use is notimplicitly changed on entry to the display callback.

See glutOverlayDisplayFunc to understand how distinct callbacks for the overlay and normal plane of a window may be established.

When a window is created, no display callback exists for the window. It is the responsibility of the programmer to install a display callback for the window before the window is shown. A display callback must be registered for any window that is shown. If a window becomes displayed without a display callback being registered, a fatal error occurs. Passing NULL to glutDisplayFunc is illegal as of GLUT 3.0; there is no way to ``deregister'' a display callback (though another callback routine can always be registered).

Upon return from the display callback, the normal damaged state of the window (returned by calling glutLayerGet(GLUT_NORMAL_DAMAGED) is cleared. If there is no overlay display callback registered the overlay damaged state of the window (returned by calling glutLayerGet(GLUT_OVERLAY_DAMAGED) is also cleared.

---

glutReshapeFunc sets the reshape callback for the current window.

Usage

```
void glutReshapeFunc(void (*func)(int width, int height));
```

func
: The new reshape callback function.

Description

glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or

whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped.

If a reshape callback is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered callback), the default reshape callback is used. This default callback will simply call glViewport(0,0,width,height) on the normal plane (and on the overlay if one exists).

If an overlay is established for the window, a single reshape callback is generated. It is the callback's responsibility to update both the normal plane and overlay for the window (changing the layer in use as necessary).

When a top-level window is reshaped, subwindows are not reshaped. It is up to the GLUT program to manage the size and positions of subwindows within a top-level window. Still, reshape callbacks will be triggered for subwindows when their size is changed using glutReshapeWindow.

---

glutIdleFunc sets the global idle callback.

Usage

```
void glutIdleFunc(void (*func)(void));
```

Description

glutIdleFunc sets the global idle callback to be func so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received. If enabled, the idle callback is continuously called when events are not being received. The callback routine has no parameters. The current window and current menu will not be changed before the idle callback. Programs with multiple windows and/or menus should explicitly set the current window and/or current menu and not rely on its current setting.

The amount of computation and rendering done in an idle callback should be minimized to avoid affecting the program's interactive response. In general, not more than a single frame of rendering should be done in an idle callback.

Passing NULL to glutIdleFunc disables the generation of the idle callback.

---

glutMouseFunc sets the mouse callback for the current window.

Usage

```
void glutMouseFunc(void (*func)(int button, int state,
                                int x, int y));
```

func
: The new mouse callback function.

Description

glutMouseFunc sets the mouse callback for the current window. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON. For systems with only two mouse buttons, it may not be possible to generate GLUT_MIDDLE_BUTTON callback. For systems with a single mouse button, it may be possible to generate only a GLUT_LEFT_BUTTON callback. The state parameter is either GLUT_UP or GLUT_DOWN indicating whether the callback was due to a release or press respectively. The x and y callback parameters indicate the window relative coordinates when the mouse button state changed. If a GLUT_DOWN callback for a specific button is triggered, the program can assume a GLUT_UP callback for the same button will be generated (assuming the window still has a mouse callback registered) when the mouse button is released even if the mouse has moved outside the window.

If a menu is attached to a button for a window, mouse callbacks will not be generated for that button.

During a mouse callback, glutGetModifiers may be called to determine the state of modifier keys when the mouse event generating the callback occurred.

Passing NULL to glutMouseFunc disables the generation of mouse callbacks.

---

glutMotionFunc and glutPassiveMotionFunc set the motion and passive motion callbacks respectively for the current window.

Usage

```
void glutMotionFunc(void (*func)(int x, int y));
void glutPassiveMotionFunc(void (*func)(int x, int y));
```

func
: The new motion or passive motion callback function.

Description

glutMotionFunc and glutPassiveMotionFunc set the motion and passive motion callback respectively for the current window. The motion callback for a window is called when the mouse moves within the window while one or more mouse buttons

are pressed. The passive motion callback for a window is called when the mouse moves within the window while no mouse buttons are pressed.

The x and y callback parameters indicate the mouse location in window relative coordinates.

Passing NULL to glutMotionFunc or glutPassiveMotionFunc disables the generation of the mouse or passive motion callback respectively.

---

glutKeyboardFunc sets the keyboard callback for the current window.

Usage

```
void glutKeyboardFunc(void (*func)(unsigned char key,
                                   int x, int y));
```

func
: The new keyboard callback function.

Description

glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

During a keyboard callback, glutGetModifiers may be called to determine the state of modifier keys when the keystroke generating the callback occurred.

Also, see glutSpecialFunc for a means to detect non-ASCII key strokes.

---

glutMainLoop enters the GLUT event processing loop.

Usage

```
void glutMainLoop(void);
```

Description

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.