| | [**Overview**](http://docs.google.com/index-overview-summary.html) | [**Project**](http://docs.google.com/project-summary.html) | **Class** | [**Tree**](http://docs.google.com/project-tree.html) | [**Deprecated**](http://docs.google.com/index-deprecated-list.html) | [**Index**](http://docs.google.com/index-all.html) | | --- | --- | --- | --- | --- | --- | | | ***CarnegieMellonGraphics*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**FRAMES**](http://docs.google.com/index.html)    [**NO FRAMES**](http://docs.google.com/CarnegieMellonGraphics2/Window.html) |
| SUMMARY:  INNER | FIELD | [CONSTR](#30j0zll) | [METHOD](#1fob9te) | DETAIL:  FIELD | [CONSTR](#3znysh7) | [METHOD](#17dp8vu) |  |

## **CarnegieMellonGraphics2**

Class Window

   in [CarnegieMellonGraphics.h](http://docs.google.com/CarnegieMellonGraphics.h.html)

**Direct Known Subclasses:** [FullScreenWindow](http://docs.google.com/CarnegieMellonGraphics2/FullScreenWindow.html)class **Window**

The Window class is used to represent that actual drawing surfaces. Currently on creating a new window object, a new window will open in user's operating environment. The background color is black by default.

Two different input paradigms have been provided. One is event based, and requires deriving a new Window class and then starting the CarnegieMellonGraphics event handler (described below). For this derived class to do something interesting, one or more of the "handle" member functions listed below must be overridden.

The other method of input works within the standard structured programming framework, and involves polling the "isQueueEmpty" functions to see if an event has occurred, removing the event from the queue, and processing it as desired.

Drawing into the window is then simply a matter of calling the appropriate member functions on the window.

**Updates:**

* A new method of input has been added: see [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown) and [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown).
* To go into fullscreen mode, specify this as an additional parameter to the constructor: [Window](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window).

If you're working on a game, you will probably be interested in investigating the following functions: [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip), [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown), and [Window](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window) (fullscreen mode).

| **Constructor Summary** | |
| --- | --- |
|  | [**Window**](#2et92p0)( const int width = 640, const int height = 480, const std::string& title = std::string("CarnegieMellonGraphics Window" ), bool fullscreen = false, int bitdepth = 32, int refreshrate = 60 )            Construct a new window with the specified size and title. |
|  | [**Window**](#3dy6vkm)( int xpos, int ypos, int width, int height, const std::string& title = std::string("CarnegieMellonGraphics Window" ), bool fullscreen = false, int bitdepth = 32, int refreshrate = 60 )            An alternate constructor where you get to choose the position of the window as well as all of the other usual parameters. |
| private | [**Window**](#1t3h5sf)( const [**Window**](#1t3h5sf)& window ) |
|  | virtual [**~Window**](#4d34og8)() |

| **Method Summary** | |
| --- | --- |
| void | [**addKeyboardEvent**](#3rdcrjn)( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& keyboard )            Functions to add events to the input queues. |
| void | [**addMouseEvent**](#lnxbz9)( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& mouse )            Functions to add events to the input queues. |
| void | [**addTimerEvent**](#1ksv4uv)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& timer )            Functions to add events to the input queues. |
| void | [**autoFlushKeyboardQueue**](#2jxsxqh)( bool enable )            Automatically lets you ignore the events in the keyboard message queue. |
| void | [**autoFlushMouseQueue**](#3j2qqm3)( bool enable )            Automatically lets you ignore the events in the mouse message queue. |
| void | [**autoFlushTimerQueue**](#4i7ojhp)( bool enable )            Automatically lets you ignore the events in the timer message queue. |
| void | [**copyRegion**](#1ci93xb)( const int x1, const int y1, const int x2, const int y2, const int width, const int height )            Copy the rectangular region at (x1, y1) to (x2, y2) |
| [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html) | [**createImage**](#2bn6wsx)( const int x, const int y, const int width, const int height )            Create an image from the rectangular region with an upper-left hand corner at (x, y) |
| void | [**disableAntialiasing**](#3as4poj)()            Renderer dependent capability. |
| void | [**disableAutoPageFlip**](#49x2ik5)()            By default each window will handle refreshing the graphics automatically for the user. |
| void | [**disableFullScreen**](#147n2zr)()  **Deprecated.** *Do not use. This works only on some systems. To ensure that you will always go into fullscreen mode, use the constructor* [*Window*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window)*.* |
| void | [**drawArc**](#23ckvvd)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end )            Draw an elliptical arc from angle start to end in degrees. |
| void | **drawBezierCurve**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords )            Draw a bezier curve using the specified style, and the two vectors specifying the control points |
| void | [**drawChordFilled**](#41mghml)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end )            Draw a filled elliptical chord with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box |
| void | [**drawChordOutline**](#vx1227)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end )            Draw an elliptical chord outline with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box |
| void | [**drawCircleFilled**](#1v1yuxt)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y, const int radius )            Draw a filled circle with style s and the specified radius at point (x, y) |
| void | [**drawCircleOutline**](#2u6wntf)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y, const int radius )            Draw a circle outline with style s and the specified radius at point (x, y) |
| void | [**drawEllipseFilled**](#3tbugp1)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 )            Draw a filled ellipse with inside the bounding box specfied by the two points (x1, y1) and (x2, y2) |
| void | [**drawEllipseOutline**](#nmf14n)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 )            Draw an ellipse outline with inside the bounding box specified by the two points (x1, y1) and (x2, y2) |
| void | [**drawImage**](#1mrcu09)( const [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)& image, const int x, const int y )            Draw the image at with its upper right-hand corner located at (x,y) |
| void | [**drawImage**](#2lwamvv)( const [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)& image, const int x, const int y, const [Transform](http://docs.google.com/CarnegieMellonGraphics2/Transform.html)& transform )            Draw the image at with its upper right-hand corner located at (x,y) using the specified transform |
| void | [**drawLine**](#111kx3o)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 )            Draw a line from (x1,y1) to (x2, y2) with style s |
| void | [**drawPixel**](#206ipza)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y )            Draw a pixel at (x,y) with style s. |
| void | **drawPolygonFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords )            Draw a filled polygon using the specified style and the two vectors to specifiy vertices |
| void | **drawPolygonOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords )            Draw a polygon outline using the specified style and the two vectors to specifiy vertices |
| void | **drawPolyLine**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords )            Draw a polyline using the specified style and the two vectors to specifiy vertices |
| void | [**drawRectangleFilled**](#1rvwp1q)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 )            Draw a filled rectangle with (x1, y1) as the upper left-hand corner, and (x2, y2) as the lower right-hand corner with style s |
| void | [**drawRectangleOutline**](#2r0uhxc)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 )            Draw a rectangle outline with (x1, y1) as the upper left-hand corner, and (x2, y2) as the lower right-hand corner with style s |
| void | [**drawText**](#3q5sasy)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const [Font](http://docs.google.com/CarnegieMellonGraphics2/Font.html)& f, const int x, const int y, const std::string& text )            Draw text in the specified font and style |
| void | [**drawText**](#kgcv8k)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const [Font](http://docs.google.com/CarnegieMellonGraphics2/Font.html)& f, const int x, const int y, const std::string& text, const [Transform](http://docs.google.com/CarnegieMellonGraphics2/Transform.html)& transform )            Draw text in the specified font and style, using the specified transform |
| void | [**drawTriangleFilled**](#34g0dwd)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3 )            Draw a filled triangle with vertices (x1, y1), (x2, y2), (x3, y3) with in style s |
| void | [**drawTriangleOutline**](#43ky6rz)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3 )            Draw a triangle outline with vertices (x1, y1), (x2, y2), (x3, y3) with in style s |
| void | [**drawWedgeFilled**](#xvir7l)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end )            Draw a filled elliptical wedge with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box |
| void | [**drawWedgeOutline**](#1x0gk37)( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end )            Draw an elliptical wedge outline with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box |
| void | [**enableAntialiasing**](#2w5ecyt)()            Renderer dependent capability. |
| void | [**enableAutoPageFlip**](#3vac5uf)()            Enabled by default; everything you draw will automatically appear on screen. |
| void | [**enableFullScreen**](#pkwqa1)( int bitdepth = 16, int refreshrate = 60 )  **Deprecated.** *Do not use. This works only on some systems. To ensure that you will always go into fullscreen mode, use the constructor* [*Window*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window)*.* |
| void | [**flipPage**](#1opuj5n)()            Make the changes that you've made appear on screen. |
| void | [**flushKeyboardQueue**](#2nusc19)()            Remove all pending messages in the keyboard event queue. |
| void | [**flushMouseQueue**](#3mzq4wv)()            Remove all pending messages in the mouse event queue. |
| void | [**flushTimerQueue**](#haapch)()            Remove all pending messages in the timer event queue. |
| int | [**getHeight**](#1gf8i83)() const            Return the height of the window. |
| [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) | [**getKeyboardEvent**](#2fk6b3p)()            Pull events off the front of the Keyboard event queue. |
| [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) | [**getKeyboardFilter**](#3ep43zb)()            Set filters on the input queues, useful in eliminating non-interesting events. |
| [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) | [**getMouseEvent**](#4du1wux)()            Pull events off the front of the Mouse event queue Use [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown) if you're interested in less precise input processing (i.e. when you're making a game). |
| [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) | [**getMouseFilter**](#184mhaj)()            Set filters on the input queues, useful in eliminating non-interesting events. |
| int | [**getMouseX**](#279ka65)() const            Return the last location of mouse X position. |
| int | [**getMouseY**](#36ei31r)() const            Return the last location of mouse Y position. |
| void | [**getPosition**](#45jfvxd)( int& x, int& y )            Retrieves the current position of the window. |
| int | [**getTimerCount**](#zu0gcz)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& te )            Get the number of times this timer event has occurred. |
| [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) | [**getTimerEvent**](#1yyy98l)()            Pull events off the front of the Timer event queue Use [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount) to get lower-overhead event processing. |
| [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) | [**getTimerFilter**](#2y3w247)()            Set filters on the input queues, useful in eliminating non-interesting events. |
| int | [**getWidth**](#3x8tuzt)() const            Return the width of the window. |
| virtual void | [**handleIdleEvent**](#rjefff)()            Event handlers for when in event handling mode. |
| virtual void | [**handleKeyboardEvent**](#1qoc8b1)( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& event )            Event handlers for when in event handling mode. |
| virtual void | [**handleMouseEvent**](#2pta16n)( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& event )            Event handlers for when in event handling mode. |
| virtual void | [**handleTimerEvent**](#3oy7u29)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& event )            Event handlers for when in event handling mode. |
| void | [**hide**](#j8sehv)()            Hide this window. |
| void | [**ignoreKeyRepeat**](#1idq7dh)( bool ignore )            Ignore repeated keystrokes sent to this window. |
| bool | [**isAutoPageFlipEnabled**](#2hio093)()            Will return whether or not auto page flipping is enabled. |
| bool | [**isButtonDown**](#3gnlt4p)( [MouseEvent::Button](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent..Button.html) b )            Tells whether a specified mouse button is down. |
| bool | [**isHidden**](#4fsjm0b)()            Tell if this window is hidden or not. |
| bool | [**isKeyboardQueueEmpty**](#1a346fx)()            Check whether there is input sitting in the Keyboard Queue. |
| bool | [**isKeyDown**](#2981zbj)( int key )            Tells whether the specified key is down. |
| bool | [**isModifierDown**](#38czs75)( [KeyModifiers](http://docs.google.com/CarnegieMellonGraphics2/KeyModifiers.html) km )            Tells whether the specified modifier was pressed during the last recorded mouse or keyboard event. |
| bool | [**isMouseQueueEmpty**](#47hxl2r)()            Check whether there is input sitting in the Mouse Queue. |
| bool | [**isTimerQueueEmpty**](#11si5id)()            Check whether there is input sitting in the Timer Queue. |
| static std::string | [**numberToString**](#20xfydz)( int n )            Convert the given number to a string. |
| static std::string | [**numberToString**](#302dr9l)( double d, int precision =-1 )            Convert the given double number to a string. |
| [Color](http://docs.google.com/CarnegieMellonGraphics2/Color.html) | [**readPixel**](#1f7o1he)( const int x, const int y )            Return the color at point (x, y) |
| [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) | [**registerNewTimerEvent**](#2eclud0)( const int milliseconds )  **Deprecated.** *This has unnecessary overhead. Use* [*startTimer*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer) *instead.* |
| void | [**resize**](#3dhjn8m)( int newwidth, int newheight )            Changes the size of the window after creation. |
| void | [**setKeyboardFilter**](#4cmhg48)( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& filter )            Change the filter for the keyboard queue. |
| void | [**setMouseFilter**](#16x20ju)( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& filter )            Change the filter for the mouse queue. |
| void | [**setPosition**](#261ztfg)( int x, int y )            Changes the position of the window after creation. |
| void | [**setTimerCount**](#356xmb2)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& te, int count = 0 )            Changes the count for a timer. |
| void | [**setTimerFilter**](#44bvf6o)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& filter )            Change the filter for the timer queue. |
| void | [**setTitle**](#ymfzma)( const std::string& title )            Changes the title of the window after creation. |
| void | [**show**](#1xrdshw)()            Show this window. |
| [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) | [**startTimer**](#2wwbldi)( int milliseconds )            Register a timer event to occur every specified interval until you stop it using [stopTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#stopTimer). |
| void | [**stopTimer**](#3w19e94)( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& timer )            Stop a timer that was started with [startTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer). |
| [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) | [**waitForKeyboardEvent**](#qbtyoq)()            Wait for a keyboard event to occur and then return the event. |
| [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) | [**waitForMouseEvent**](#1pgrrkc)()            Wait for a mouse event to occur and then return the event. |
| [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) | [**waitForTimerEvent**](#2olpkfy)()            Wait for a timer event to occur and then return the event. |

| **Constructor Detail** |
| --- |

### Window

public **Window**( const int width = 640, const int height = 480, const std::string& title = std::string("CarnegieMellonGraphics Window" ), bool fullscreen = false, int bitdepth = 32, int refreshrate = 60 );

Construct a new window with the specified size and title. (defaults to 640 by 480, with "CarnegieMellonGraphics Window" as the title)

This is the appropriate way to go into full screen mode. To create a regular windowed application, construct a window as follows:

Window myWindow();

or

Window myWindow(640,480,"my window"); // the last parameter defaults to false

To create a fullscreen window, pass true as the fourth parameter:

Window myFullScreenWindow(640,480,"my full screen window",true);

If the dimensions of the window are not a standard dimension or one that is not supported by your video card, the window will probably not be constructed and no error will be reported. Dimensions such as 320x200, 320x240, 400x300, 640x480, 800x600, 1024x768, etc. are recommended for full screen mode.

### Window

public **Window**( int xpos, int ypos, int width, int height, const std::string& title = std::string("CarnegieMellonGraphics Window" ), bool fullscreen = false, int bitdepth = 32, int refreshrate = 60 );

An alternate constructor where you get to choose the position of the window as well as all of the other usual parameters. **Since:** 2.1.5

### Window

private **Window**( const **Window**& window );

### ~Window

public virtual **~Window**();

| **Method Detail** |
| --- |

### addKeyboardEvent

public void **addKeyboardEvent**( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& keyboard );

Functions to add events to the input queues. These should generally not be used, but it is possible they may be useful in some circumstances.

**Note:**  calling this function will not set the key to be down

**See Also:** [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent)

### addMouseEvent

public void **addMouseEvent**( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& mouse );

Functions to add events to the input queues. These should generally not be used, but it is possible they may be useful in some circumstances.

**Note:**  calling this function will not set the button to be down

**See Also:** [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent)

### addTimerEvent

public void **addTimerEvent**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& timer );

Functions to add events to the input queues. These should generally not be used, but it is possible they may be useful in some circumstances. **See Also:** [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent)

### autoFlushKeyboardQueue

public void **autoFlushKeyboardQueue**( bool enable );

Automatically lets you ignore the events in the keyboard message queue. You should use this if you want to ignore the event queue altogether. Enabling/disabling this will not affect [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown). It is possible that the queue will at some point contain messages, but it will be flushed eventually. **Since:** 2.1.2 **See Also:** [autoFlushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushMouseQueue), [autoFlushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushTimerQueue), [flushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [isKeyboardQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyboardQueueEmpty), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown), [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown)

### autoFlushMouseQueue

public void **autoFlushMouseQueue**( bool enable );

Automatically lets you ignore the events in the mouse message queue. You should use this if you want to ignore the event queue altogether. Enabling/disabling this will not affect [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown). It is possible that the queue will at some point contain messages, but it will be flushed eventually. **Since:** 2.1.2 **See Also:** [autoFlushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushKeyboardQueue), [autoFlushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushTimerQueue), [flushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushMouseQueue), [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown), [isMouseQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isMouseQueueEmpty)

### autoFlushTimerQueue

public void **autoFlushTimerQueue**( bool enable );

Automatically lets you ignore the events in the timer message queue. You should use this if you want to ignore the event queue altogether. Enabling/disabling this will not affect [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount). It is possible that the queue will at some point contain messages, but it will be flushed eventually. **Since:** 2.1.2 **See Also:** [autoFlushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushKeyboardQueue), [autoFlushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushMouseQueue), [flushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushTimerQueue), [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent)

### copyRegion

public void **copyRegion**( const int x1, const int y1, const int x2, const int y2, const int width, const int height );

Copy the rectangular region at (x1, y1) to (x2, y2)

### createImage

public [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html) **createImage**( const int x, const int y, const int width, const int height );

Create an image from the rectangular region with an upper-left hand corner at (x, y) **See Also:** [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html), [drawImage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawImage)

### disableAntialiasing

public void **disableAntialiasing**();

Renderer dependent capability. If not supported by current renderer these will be ignored. Initial state is also dependent on the supplied renderer.

### disableAutoPageFlip

public void **disableAutoPageFlip**();

By default each window will handle refreshing the graphics automatically for the user. However in some circumstances it may be desirable to perform these refreshes manually. First call [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip) to disable automatic refreshes and then [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage) whenever you want to show what you've drawn. [enableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#enableAutoPageFlip) may then be used to enable automatic refreshes. **See Also:** [enableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#enableAutoPageFlip), [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage), [isAutoPageFlipEnabled](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isAutoPageFlipEnabled)

### disableFullScreen

public void **disableFullScreen**();

**Deprecated.** *Do not use. This works only on some systems. To ensure that you will always go into fullscreen mode, use the constructor* [*Window*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window)*.*

### drawArc

public void **drawArc**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end );

Draw an elliptical arc from angle start to end in degrees. The x and y coordinates specify the bounding box of the ellipse

### drawBezierCurve

public void **drawBezierCurve**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords );

Draw a bezier curve using the specified style, and the two vectors specifying the control points

### drawChordFilled

public void **drawChordFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end );

Draw a filled elliptical chord with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box

### drawChordOutline

public void **drawChordOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end );

Draw an elliptical chord outline with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box

### drawCircleFilled

public void **drawCircleFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y, const int radius );

Draw a filled circle with style s and the specified radius at point (x, y)

### drawCircleOutline

public void **drawCircleOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y, const int radius );

Draw a circle outline with style s and the specified radius at point (x, y)

### drawEllipseFilled

public void **drawEllipseFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 );

Draw a filled ellipse with inside the bounding box specfied by the two points (x1, y1) and (x2, y2)

### drawEllipseOutline

public void **drawEllipseOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 );

Draw an ellipse outline with inside the bounding box specified by the two points (x1, y1) and (x2, y2)

### drawImage

public void **drawImage**( const [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)& image, const int x, const int y );

Draw the image at with its upper right-hand corner located at (x,y) **See Also:** [drawImage(constImage&,constint,constint,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawImage(constImage&,constint,constint,constTransform&)), [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)

### drawImage

public void **drawImage**( const [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)& image, const int x, const int y, const [Transform](http://docs.google.com/CarnegieMellonGraphics2/Transform.html)& transform );

Draw the image at with its upper right-hand corner located at (x,y) using the specified transform **Since:** 2.10 **See Also:** [drawImage(constImage&,constint,constint)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawImage(constImage&,constint,constint)), [Image](http://docs.google.com/CarnegieMellonGraphics2/Image.html)

### drawLine

public void **drawLine**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 );

Draw a line from (x1,y1) to (x2, y2) with style s

### drawPixel

public void **drawPixel**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x, const int y );

Draw a pixel at (x,y) with style s. If you want to do a lot of bit manipulation, you should use [EditableImage](http://docs.google.com/CarnegieMellonGraphics2/EditableImage.html). drawPixel can be slow if you do a lot of pixel-level editing. **See Also:** [EditableImage](http://docs.google.com/CarnegieMellonGraphics2/EditableImage.html)

### drawPolygonFilled

public void **drawPolygonFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords );

Draw a filled polygon using the specified style and the two vectors to specifiy vertices

### drawPolygonOutline

public void **drawPolygonOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords );

Draw a polygon outline using the specified style and the two vectors to specifiy vertices

### drawPolyLine

public void **drawPolyLine**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const std::vector< int >& xcoords, const std::vector< int >& ycoords );

Draw a polyline using the specified style and the two vectors to specifiy vertices

### drawRectangleFilled

public void **drawRectangleFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 );

Draw a filled rectangle with (x1, y1) as the upper left-hand corner, and (x2, y2) as the lower right-hand corner with style s

### drawRectangleOutline

public void **drawRectangleOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2 );

Draw a rectangle outline with (x1, y1) as the upper left-hand corner, and (x2, y2) as the lower right-hand corner with style s

### drawText

public void **drawText**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const [Font](http://docs.google.com/CarnegieMellonGraphics2/Font.html)& f, const int x, const int y, const std::string& text );

Draw text in the specified font and style **See Also:** [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [numberToString(int)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(int)), [numberToString(double)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(double))

### drawText

public void **drawText**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const [Font](http://docs.google.com/CarnegieMellonGraphics2/Font.html)& f, const int x, const int y, const std::string& text, const [Transform](http://docs.google.com/CarnegieMellonGraphics2/Transform.html)& transform );

Draw text in the specified font and style, using the specified transform **Since:** 2.10 **See Also:** [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [numberToString(int)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(int)), [numberToString(double)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(double))

### drawTriangleFilled

public void **drawTriangleFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3 );

Draw a filled triangle with vertices (x1, y1), (x2, y2), (x3, y3) with in style s

### drawTriangleOutline

public void **drawTriangleOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3 );

Draw a triangle outline with vertices (x1, y1), (x2, y2), (x3, y3) with in style s

### drawWedgeFilled

public void **drawWedgeFilled**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end );

Draw a filled elliptical wedge with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box

### drawWedgeOutline

public void **drawWedgeOutline**( const [Style](http://docs.google.com/CarnegieMellonGraphics2/Style.html)& s, const int x1, const int y1, const int x2, const int y2, const double start, const double end );

Draw an elliptical wedge outline with from start to end degrees The two points (x1, y1) and (x2, y2) specify the bounding box

### enableAntialiasing

public void **enableAntialiasing**();

Renderer dependent capability. If not supported by current renderer these will be ignored. Initial state is also dependent on the supplied renderer.

### enableAutoPageFlip

public void **enableAutoPageFlip**();

Enabled by default; everything you draw will automatically appear on screen. To get much better performance, use [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage) and [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip). **See Also:** [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip), [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage), [isAutoPageFlipEnabled](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isAutoPageFlipEnabled)

### enableFullScreen

public void **enableFullScreen**( int bitdepth = 16, int refreshrate = 60 );

**Deprecated.** *Do not use. This works only on some systems. To ensure that you will always go into fullscreen mode, use the constructor* [*Window*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#Window)*.*

### flipPage

public void **flipPage**();

Make the changes that you've made appear on screen. All drawing is done off screen, this will show the user the off-screen page. **See Also:** [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip), [enableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#enableAutoPageFlip), [isAutoPageFlipEnabled](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isAutoPageFlipEnabled)

### flushKeyboardQueue

public void **flushKeyboardQueue**();

Remove all pending messages in the keyboard event queue. **See Also:** [autoFlushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [isKeyboardQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyboardQueueEmpty), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown), [waitForKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForKeyboardEvent)

### flushMouseQueue

public void **flushMouseQueue**();

Remove all pending messages in the mouse event queue. **See Also:** [autoFlushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushMouseQueue), [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isMouseQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isMouseQueueEmpty), [waitForMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForMouseEvent)

### flushTimerQueue

public void **flushTimerQueue**();

Remove all pending messages in the timer event queue. **See Also:** [autoFlushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushTimerQueue), [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent), [waitForTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForTimerEvent)

### getHeight

public int **getHeight**() const;

Return the height of the window. **See Also:** [getWidth](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getWidth)

### getKeyboardEvent

public [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) **getKeyboardEvent**();

Pull events off the front of the Keyboard event queue. Use [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown) if you're interested in less precise input processing (i.e. when you're making a game). **See Also:** [flushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushKeyboardQueue), [ignoreKeyRepeat](http://docs.google.com/CarnegieMellonGraphics2/Window.html#ignoreKeyRepeat), [isKeyboardQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyboardQueueEmpty), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown), [waitForKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForKeyboardEvent)

### getKeyboardFilter

public [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) **getKeyboardFilter**();

Set filters on the input queues, useful in eliminating non-interesting events. Intended for advanced users only. If set to NULL, no filtering is performed. If set to some value, only events that do not match the event pattern will be placed into the queue. There is no default keyboard filter. **See Also:** [setKeyboardFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setKeyboardFilter)

### getMouseEvent

public [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) **getMouseEvent**();

Pull events off the front of the Mouse event queue Use [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown) if you're interested in less precise input processing (i.e. when you're making a game). **See Also:** [flushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushMouseQueue), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isMouseQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isMouseQueueEmpty), [waitForMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForMouseEvent)

### getMouseFilter

public [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) **getMouseFilter**();

Set filters on the input queues, useful in eliminating non-interesting events. Intended for advanced users only. If set to NULL, no filtering is performed. If set to some value, only events that do not match the event pattern will be placed into the queue. By default, when not using the CarnegieMellonGraphics event handler, MOUSE\_MOVE events are filtered out to prevent unnecessary events from cluttering the queue. **See Also:** [setMouseFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setMouseFilter)

### getMouseX

public int **getMouseX**() const;

Return the last location of mouse X position. **See Also:** [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [getMouseY](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseY), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown)

### getMouseY

public int **getMouseY**() const;

Return the last location of mouse Y position. **See Also:** [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [getMouseX](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseX), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown)

### getPosition

public void **getPosition**( int& x, int& y );

Retrieves the current position of the window. **Parameters:** x - [out] x position of the window y - [out] y position of the window **Since:** 2.1.5 **See Also:** [resize](http://docs.google.com/CarnegieMellonGraphics2/Window.html#resize), [setTitle](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTitle), [setPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setPosition)

### getTimerCount

public int **getTimerCount**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& te );

Get the number of times this timer event has occurred. Generally used in conjunction with @setTimerCount, this is very good for making sure that your program keeps a time (like maintaining a framerate). **Since:** 2.1.2 **See Also:** [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent), [setTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTimerCount)

### getTimerEvent

public [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) **getTimerEvent**();

Pull events off the front of the Timer event queue Use [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount) to get lower-overhead event processing. **See Also:** [flushTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty), [waitForTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent)

### getTimerFilter

public [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) **getTimerFilter**();

Set filters on the input queues, useful in eliminating non-interesting events. Intended for advanced users only. If set to NULL, no filtering is performed. If set to some value, only events that do not match the event pattern will be placed into the queue. There is no default timer filter. **See Also:** [setTimerFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTimerFilter)

### getWidth

public int **getWidth**() const;

Return the width of the window. **See Also:** [getHeight](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getHeight)

### handleIdleEvent

public virtual void **handleIdleEvent**();

Event handlers for when in event handling mode. To be useful you must derive your own version of Window that overides these functions **See Also:** [WindowManager](http://docs.google.com/CarnegieMellonGraphics2/WindowManager.html)

### handleKeyboardEvent

public virtual void **handleKeyboardEvent**( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& event );

Event handlers for when in event handling mode. To be useful you must derive your own version of Window that overides these functions **See Also:** [WindowManager](http://docs.google.com/CarnegieMellonGraphics2/WindowManager.html)

### handleMouseEvent

public virtual void **handleMouseEvent**( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& event );

Event handlers for when in event handling mode. To be useful you must derive your own version of Window that overides these functions **See Also:** [WindowManager](http://docs.google.com/CarnegieMellonGraphics2/WindowManager.html)

### handleTimerEvent

public virtual void **handleTimerEvent**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& event );

Event handlers for when in event handling mode. To be useful you must derive your own version of Window that overides these functions **See Also:** [WindowManager](http://docs.google.com/CarnegieMellonGraphics2/WindowManager.html)

### hide

public void **hide**();

Hide this window. **See Also:** [isHidden](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isHidden), [show](http://docs.google.com/CarnegieMellonGraphics2/Window.html#show)

### ignoreKeyRepeat

public void **ignoreKeyRepeat**( bool ignore );

Ignore repeated keystrokes sent to this window. If this is disabled, keystrokes will occur repeatedly which might be unwanted for games using the keyboard. **Since:** 2.1.2 **See Also:** [autoFlushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown)

### isAutoPageFlipEnabled

public bool **isAutoPageFlipEnabled**();

Will return whether or not auto page flipping is enabled. **Since:** 2.1.5 **See Also:** [disableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#disableAutoPageFlip), [enableAutoPageFlip](http://docs.google.com/CarnegieMellonGraphics2/Window.html#enableAutoPageFlip), [flipPage](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flipPage)

### isButtonDown

public bool **isButtonDown**( [MouseEvent::Button](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent..Button.html) b );

Tells whether a specified mouse button is down. You can check for "chording" of mouse buttons in this fashion. For example:

if (isButtonDown(MouseEvent::LEFT\_BUTTON) && isButtonDown(MouseEvent::RIGHT\_BUTTON))   
 // do something if both buttons are down

This is the recommended method of input if you are writing a game. If you need absolute precision, then do not use this function. There is no guarantee that if a button is pressed checking it in this fashion will ever show that it has been pressed.

This should probably be used in conjuction with [autoFlushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushMouseQueue).

**Since:** 2.1.2 **See Also:** [autoFlushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushMouseQueue), [flushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushMouseQueue), [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown), [isMouseQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isMouseQueueEmpty), [waitForMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForMouseEvent)

### isHidden

public bool **isHidden**();

Tell if this window is hidden or not. **See Also:** [hide](http://docs.google.com/CarnegieMellonGraphics2/Window.html#hide), [show](http://docs.google.com/CarnegieMellonGraphics2/Window.html#show)

### isKeyboardQueueEmpty

public bool **isKeyboardQueueEmpty**();

Check whether there is input sitting in the Keyboard Queue. **See Also:** [flushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [waitForKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForKeyboardEvent)

### isKeyDown

public bool **isKeyDown**( int key );

Tells whether the specified key is down. key can either be an ascii character, like 'a' or a [NamedKey](http://docs.google.com/CarnegieMellonGraphics2/NamedKey.html), like [NamedKey::ESCAPE](http://docs.google.com/CarnegieMellonGraphics2/NamedKey.html#ESCAPE). isKeyDown will always return the same value for lowercase and uppercase version of the same letter. This is because there are ways to have a key be pressed in its lowercase form and then be raised in its uppercase form (e.g. press 'a', then press shift, then release 'a'). You can use [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown) to test if a modifier was pressed at the time that the character was. **Since:** 2.1.2 **See Also:** [autoFlushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#autoFlushKeyboardQueue), [flushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [ignoreKeyRepeat](http://docs.google.com/CarnegieMellonGraphics2/Window.html#ignoreKeyRepeat), [isKeyboardQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyboardQueueEmpty), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown), [isModifierDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isModifierDown), [waitForKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForKeyboardEvent)

### isModifierDown

public bool **isModifierDown**( [KeyModifiers](http://docs.google.com/CarnegieMellonGraphics2/KeyModifiers.html) km );

Tells whether the specified modifier was pressed during the last recorded mouse or keyboard event. This will be accurate if there is a lot of keyboard/mouse input going on, but will probably be innaccurate in other instances. It is recommended that if you need precision to use the eventqueues. This is an approximate way of getting whether a modifier is down. **Since:** 2.1.2 **See Also:** [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown)

### isMouseQueueEmpty

public bool **isMouseQueueEmpty**();

Check whether there is input sitting in the Mouse Queue. **See Also:** [flushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushMouseQueue), [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [waitForMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForMouseEvent)

### isTimerQueueEmpty

public bool **isTimerQueueEmpty**();

Check whether there is input sitting in the Timer Queue. **See Also:** [flushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushTimerQueue), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [waitForTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#waitForTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent)

### numberToString

public static std::string **numberToString**( int n );

Convert the given number to a string. Recommended for use with drawText. **Since:** 2.1.4a **See Also:** [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [int)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(double,)

### numberToString

public static std::string **numberToString**( double d, int precision =-1 );

Convert the given double number to a string. Recommended for use with drawText. **Parameters:** d - [in] number to convert to string precision - [in, optional] number of digits of precision (numbers after decimal point) **Since:** 2.1.4a **See Also:** [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#drawText(constStyle&,constFont&,constint,constint,conststring&,constTransform&)), [numberToString(int)](http://docs.google.com/CarnegieMellonGraphics2/Window.html#numberToString(int))

### readPixel

public [Color](http://docs.google.com/CarnegieMellonGraphics2/Color.html) **readPixel**( const int x, const int y );

Return the color at point (x, y)

### registerNewTimerEvent

public [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) **registerNewTimerEvent**( const int milliseconds );

**Deprecated.** *This has unnecessary overhead. Use* [*startTimer*](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer) *instead.*

Register a timer event to occur after the specified number of milliseconds. Only causes the timer to go off once. If you want repeated occurrences, you must re-register when you recieve the event.

**See Also:** [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [setTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTimerCount), [startTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer), [stopTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#stopTimer)

### resize

public void **resize**( int newwidth, int newheight );

Changes the size of the window after creation. **Since:** 2.1.5 **See Also:** [getPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getPosition), [setPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setPosition), [setTitle](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTitle)

### setKeyboardFilter

public void **setKeyboardFilter**( const [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html)& filter );

Change the filter for the keyboard queue. **See Also:** [getKeyboardFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardFilter)

### setMouseFilter

public void **setMouseFilter**( const [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html)& filter );

Change the filter for the mouse queue. **See Also:** [getMouseFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseFilter)

### setPosition

public void **setPosition**( int x, int y );

Changes the position of the window after creation. **Since:** 2.1.5 **See Also:** [getPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getPosition), [resize](http://docs.google.com/CarnegieMellonGraphics2/Window.html#resize), [setTitle](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTitle)

### setTimerCount

public void **setTimerCount**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& te, int count = 0 );

Changes the count for a timer. This is useful for resetting a timer's count once you've checked it. **Since:** 2.1.2 **See Also:** [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent)

### setTimerFilter

public void **setTimerFilter**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& filter );

Change the filter for the timer queue. **See Also:** [getTimerFilter](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerFilter)

### setTitle

public void **setTitle**( const std::string& title );

Changes the title of the window after creation. **Since:** 2.1.5 **See Also:** [getPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getPosition), [setPosition](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setPosition), [resize](http://docs.google.com/CarnegieMellonGraphics2/Window.html#resize)

### show

public void **show**();

Show this window. **See Also:** [hide](http://docs.google.com/CarnegieMellonGraphics2/Window.html#hide), [isHidden](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isHidden)

### startTimer

public [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) **startTimer**( int milliseconds );

Register a timer event to occur every specified interval until you stop it using [stopTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#stopTimer). **Since:** 2.1.5 **See Also:** [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty), [setTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTimerCount), [stopTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#stopTimer)

### stopTimer

public void **stopTimer**( const [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html)& timer );

Stop a timer that was started with [startTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer). **Since:** 2.1.5 **See Also:** [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty), [setTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#setTimerCount), [startTimer](http://docs.google.com/CarnegieMellonGraphics2/Window.html#startTimer)

### waitForKeyboardEvent

public [KeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/KeyboardEvent.html) **waitForKeyboardEvent**();

Wait for a keyboard event to occur and then return the event. If there are already events in the queue the function will return immediately. **See Also:** [flushKeyboardQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushKeyboardQueue), [getKeyboardEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getKeyboardEvent), [ignoreKeyRepeat](http://docs.google.com/CarnegieMellonGraphics2/Window.html#ignoreKeyRepeat), [isKeyboardQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyboardQueueEmpty), [isKeyDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isKeyDown)

### waitForMouseEvent

public [MouseEvent](http://docs.google.com/CarnegieMellonGraphics2/MouseEvent.html) **waitForMouseEvent**();

Wait for a mouse event to occur and then return the event. If there are already events in the queue the function will return immediately. **See Also:** [flushMouseQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushMouseQueue), [getMouseEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getMouseEvent), [isButtonDown](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isButtonDown), [isMouseQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isMouseQueueEmpty)

### waitForTimerEvent

public [TimerEvent](http://docs.google.com/CarnegieMellonGraphics2/TimerEvent.html) **waitForTimerEvent**();

Wait for a timer event to occur and then return the event. If there are already events in the queue the function will return immediately. **See Also:** [flushTimerQueue](http://docs.google.com/CarnegieMellonGraphics2/Window.html#flushTimerQueue), [getTimerCount](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerCount), [getTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#getTimerEvent), [registerNewTimerEvent](http://docs.google.com/CarnegieMellonGraphics2/Window.html#registerNewTimerEvent), [isTimerQueueEmpty](http://docs.google.com/CarnegieMellonGraphics2/Window.html#isTimerQueueEmpty)

| | [**Overview**](http://docs.google.com/index-overview-summary.html) | [**Project**](http://docs.google.com/project-summary.html) | **Class** | [**Tree**](http://docs.google.com/project-tree.html) | [**Deprecated**](http://docs.google.com/index-deprecated-list.html) | [**Index**](http://docs.google.com/index-all.html) | | --- | --- | --- | --- | --- | --- | | | ***CarnegieMellonGraphics*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**FRAMES**](http://docs.google.com/index.html)    [**NO FRAMES**](http://docs.google.com/CarnegieMellonGraphics2/Window.html) |
| SUMMARY:  INNER | FIELD | [CONSTR](#30j0zll) | [METHOD](#1fob9te) | DETAIL:  FIELD | [CONSTR](#3znysh7) | [METHOD](#17dp8vu) |  |