

# Eye Science SDK

## User's Guide

Last Revised 3 July 2000

Written For SDK Versions 3.1 or 3.2

<b><i>Content</i></b>	<b><i>Page</i></b>
<i>1.0 Introduction</i>	<i>2</i>
<i>2.0 SDK Contents</i>	<i>3</i>
<i>3.0 How to Use the SDK</i>	<i>4</i>
<i>4.0 Function Reference</i>	<i>6</i>

## 1.0 Introduction

Dear Developer,

EyeTech Digital Systems has been selling the Quick Glance eye gaze tracker since 1996. Although Quick Glance was developed initially as an input device for disabled people, many units have been sold to researchers and developers for other purposes. Some of these customers have suggested that we create a means for them to access data and control the Quick Glance system directly from their own software application. The Eye Science System Developer's Kit was created to meet these needs.

We hope you'll find this developer's Kit to be a useful tool. Please send us your suggestions for changes or additions you would like to see in future versions.

Sincerely,



Robert Chappell  
Technical Director  
EyeTech Digital Systems  
ets602@aol.com

## **2.0 SDK Contents**

To use the SDK you will need Quick Glance version 3.1 or higher. Read the Quick Glance user's guide for information on installing and calibrating the Quick Glance system. The Quick Glance Installation process will also install the Eye Science SDK software. After installation, we suggest that you run the sample program in "C:\Program Files\EyeTechDS\ SDKExample.exe". This program will demonstrate many of the features of the SDK. The source code for this program is found in "C:\Program Files\EyeTechDS\ Eye Science SDK\ "

There are three files you will receive which together comprise the SDK software. These three files are located in "C:\Program Files\EyeTechDS\ Eye Science SDK\ "

- 1) EyeScienceSDK.dll This is the dynamic link library which contains the executable code for the SDK functions. This file can reside in your Windows\System folder or you can place it in the folder "C:\Program Files\EyeTechDS" where the Quick Glance executable resides (in this case the program you write must also reside there). If you're writing your software in Visual Basic you will need to call the exported functions from this DLL.
- 2) EyeScienceSDK.lib This is a library which you can link to if you're writing your application using Microsoft Visual C++. By linking to the API functions through this library file you will not need to explicitly load functions from the DLL. A sample program "SDKExample" and its source code (see folder "C:\Program Files\EyeTechDS\ Eye Science SDK") has been included with your software. This sample program was written using Microsoft Visual C++ version 6.0.
- 3) EyeScienceSDK.h This is the include file which contains the SDK function prototypes. This is a text file you can read to get information on the functions and their arguments. If you're writing your software using Microsoft Visual C++ this will be an include file for your project.

## **3.0 How to Use the SDK**

### **3.1 Installation, Setup, and Calibration of the Quick Glance Software**

See the Quick Glance user's guide for instructions.

### **3.2 Setting Quick Glance Options**

Before your software can begin interacting with Quick Glance there are some options you will need to initialize in Quick Glance. These options will be valid for the current user, and each time that user is loaded the saved options will be loaded also.

First go to the dialog box "Setup\ More Options" and make the following changes to the settings:

- Check "Enable EyeScience SDK".
- Set the "Update Rate" to whatever rate you want to capture eye gaze data at. The maximum possible rate is 30 frames per second and is limited by the video rate of the camera. Your computer may limit you to something less than 30 frames per second if the processor is not fast enough.
- Set the "Smoothing Factor". If this is set to any value greater than 1, a median filter is applied to the gaze point positions across successive frames. This filtering gives better accuracy but results in a lag time. If you want no lag time and no filtering in your gaze point data, set this value to 1.
- Click on "Done" to exit the dialog box.

Now we will describe how to adjust the Quick Glance tracking to run at the maximum possible speed. If your computer is fast enough you may be able to achieve 30 frames per second without this adjustment. If you cannot achieve you're desired sampling rate then make the following adjustments in the dialog box "Setup\ Adjust Tracking"

- Select the "Manual" tracking method. The "Auto" tracking method does a better job of automatically compensating for different user's and for different lighting conditions, but requires about twice as much CPU time to process.
- Adjust the "Pupil Threshold" up and down until the system is tracking nicely. When Quick Glance is tracking properly you will see a large cross hair in the center of the pupil, small cross hairs on each of the two bright spots, and a tight ring of dots around the pupil. It is okay if there are also dots on other parts of the eye.
- You may have to return to this dialog box and make adjustments to "Pupil Threshold" if you're lighting conditions change. You will also need to make these adjustments individually for each user. They are saved with the user's files.
- Click on "Done" to exit the dialog box.

### 3.3 Activating the Eye Science Software

Before Quick Glance will respond to calls from the SDK's DLL and before you can use the Eye Science dialog box, you must activate the Eye Science software.

From the opening screen select "Eye Science". This should bring up the dialog box which displays the serial number for your eye tracking system and also allows you to enter a password. Enter the password you received when you purchased the Eye Science software. This should bring up the "Eye Science" dialog box. Select "Done". Your software is now activated and you can begin using the SDK.

### 3.4 Calling SDK Functions

Given below is a typical sequence of calls to the SDK functions. See the function reference section for details on how these routines work.

#### 1) Determine Status

```
GetSDKVersion();  
GetQGOnFlag();  
GetESEnabledFlag();
```

#### 2) Set the Size, Position, and State of the Quick Glance Window

```
DisableMouseInput();  
SetMWState(int MWState);  
MoveLeftRight()  
MoveUpDown()
```

#### 3) Perform a Calibration Update

```
CallCalUpdate()  
SaveAndQuitCalUpdate()
```

#### 4) Perform Data Collection

```
StartGPCollection(int CollectionMethod, int NumPointsRequested)  
StopGPCollection()  
GetNumGP()  
ReadGP(int &XPos, int &YPos, int &CapTime)
```

## **4.0 Function Reference**

### **4.1 Query Status Functions**

These functions should normally be the first called by your application to ascertain the status of Quick Glance and the SDK DLL.

double GetSDKVersion()

Retrieves the version number of the EyeScienceSDK.dll.

BOOL GetQGOnFlag()

This function returns TRUE if Quick Glance is running or FALSE if Quick Glance is not running.

BOOL GetESEnabledFlag()

This function returns TRUE if the Eye Science software is activated and enabled. Otherwise it returns FALSE.

### **4.2 Window Control Functions**

These functions control the size, position, and status of the Quick Glance window.

void DisableMouseInput()

Hides Quick Glance buttons. Call this function if you want to control Quick Glance exclusively through the SDK.

void EnableMouseInput()

Calling this function will configure Quick Glance so that a user can click on Quick Glance buttons.

void ExitQuickGlance()

Closes down the Quick Glance program.

void MoveLeftRight()

Moves the eye tools or small video window left/right.

void MoveUpDown()

Moves the eye tools or small video window up/down.

void ResetEnableEyeControl()

Stops the mouse cursor from following the user's gaze.

void SetEnableEyeControl()

Forces the mouse cursor to follow the user's gaze.

void ToggleLargelImage()

Displays/destroys the display of the video window at its maximum size. At the maximum size there is a pixel to pixel mapping of data collected and data displayed.

void SetMWState(int MWState)

Changes the Quick Glance main window state. The argument MWState can take these values listed in the table below. The values that can be used depend on whether or not mouse input has been disabled.

MWState Value	Mouse Input Enabled	Mouse Input Disabled
MWOpeningScreen	X	
MWEyeTools	X	
MWSmallVideo	X	X
MWOpeningScreenNoButtons		X
MWEyeToolsNoButtons		X
MWMinimized	X	X
MWRestored	X	X

### 4.3 Calibration Update Functions

The calibration update feature brings up a dialog box with a single target in the center. If SaveAndQuitCalUpdate() is called while the user is looking at the center of the target, the software will measure the error and subtract it from all subsequent gaze point measurements. There are two circumstances where it is important to use this feature:

- 1) if you want many users to operate the eyetracker without having to calibrate, you can pre-calibrate the system and then just perform the calibration update on each individual user. This will give very good accuracy in the center of the screen. Accuracy around the edges of the screen will vary depending on how different the user's eye is from the eye the system was calibrated on.
- 2) Even if a user has done a full calibration, the calibration update can compensate for changes in the user's eye - especially changes in pupil size.

void CallCalUpdate()

Brings up the calibration update dialog box.

void QuitCalUpdate()

Quits the calibration update dialog box without applying the update.

void SaveAndQuitCalUpdate()

Quits the calibration update dialog box and applies the update.

#### 4.4 Data Collection Functions

The four functions listed in this section control the collection and retrieval of gaze point data. This data is kept in a buffer internal to the Eye Science DLL. Each time a gaze point is placed in the buffer it remains there until it is retrieved with the ReadGP() function. The number of gaze points which can be collected before being retrieved is equal to GPDataSize. The value of GPDataSize is defined in EyeScienceSDK.h.

By periodically emptying the buffer with calls to ReadGP() data collection can continue indefinitely. If the buffer is allowed to completely fill up data collection will stop until re-started with a call to StartGPCollection().

void StartGPCollection(int CollectionMethod, int NumPointsRequested)

Calling this function tells Quick Glance to start placing gaze point data in the gaze point buffer. The fastest sample rates are achieved with the multimedia and loop methods. For the standard and multimedia methods, data collection will continue until StopGPCollection() is called. For the loop method, data collection will continue until StopGPCollection() is called or NumPointsRequested data points have been collected.

SCStandard

SCMultimedia

SCLoop

void StopGPCollection()

Calling this function tells Quick Glance to stop placing gaze point data in the gaze point buffer.

int GetNumGP()

Returns the number of unretrieved gaze points in the buffer. Call ReadGP() this many times to retrieve all the collected data.

void ReadGP(int &XPos, int &YPos, int &CapTime)

Retrieves one gaze point from the buffer. If no gaze points are available, the value returned for all arguments will be -1.

BOOL GetLoopCollectDoneFlag()

This function returns TRUE when a "Loop Method" data collection is complete.

void ResetLoopCollectDoneFlag()

Call this function before beginning a "Loop Method" data collection to reset the "Loop Collect Done Flag".