



Intel[®] Laptop Gaming TDK User Guide

Version 1.0

Revision 0.7

August, 2006



Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The **Intel Laptop Gaming TDK** may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting [Intel's Web Site](#).

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Copyright © 2006, Intel Corporation. All rights reserved.

* Other brands and names may be claimed as the property of others.



Contents

1	Introduction	5
	1.1 About the Intel Laptop Gaming TDK.....	5
	1.2 Key Features.....	5
	1.3 Requirements.....	5
	1.4 Product License	6
2	Getting Started	9
	2.1 Installing the Intel Laptop Gaming TDK	9
3	Using the Laptop Gaming TDK.....	12
	3.1 Description of Files	12
	3.2 Setting Up Visual Studio for Use with the Intel Laptop Gaming TDK	12
	3.3 Initiating the Laptop Gaming TDK Interface.....	13
	3.4 Calling Modes	13
	3.4.1 Polling Mode	14
	3.4.2 Threaded Event-Based Mode.....	14
	3.5 Showing Power Status.....	16
	3.6 Reducing Power Consumption by Keeping Track of Network Adapter Information	
4	Recommendations	22
	4.1 Troubleshooting	22
	4.2 Recommendations For Increasing System Battery Life*	22
	4.2.1 Detecting Power Source Changes.....	22
	4.2.2 Detecting CPU Utilization.....	23
	4.2.3 Monitoring Connectivity Status	23
	4.2.4 Changing Display Resolution.....	23
A	Appendix: API Specifications	24
	A.1 General APIs.....	24
	A.2 Power	25
	A.3 Network	26
	A.4 Display	27
	A.5 Processor.....	29
	A.6 Bandwidth	30

Table of Illustrations

Figure 1	Interface with Enable Mobility Thread enabled.....	14
Sample 1	Polling mode example.....	15
Sample 2	Receiving and handling TDK-generated events.....	16
Sample 3	Registering an event ID to the Event Observer.....	16
Figure 2	Battery meter showing remaining charge and battery charge time	17
Figure 3	Battery meter showing AC power and remaining battery power.....	17
Sample 4	UpdateBatteryData method used as a single entry point.....	18
Sample 5	HUD updated with stored data.....	19
Sample 6	Setting the number of frames to show the battery meter.....	20
Sample 7	Determining if the laptop is connected.....	21
Sample 8	Identifying if the connection is wired or wireless.....	21
Sample 9	Confirming that a wireless connection is disabled.....	22



Revision History

Revision Number	Description	Date
0.1	<ul style="list-style-type: none">Initial release.	April 2006
0.5	<ul style="list-style-type: none">Updated to incorporate single sample application and usage examples.	June 2006
0.6	<ul style="list-style-type: none">Updated to incorporate additional sample application information.	June 2006
0.7	<ul style="list-style-type: none">Updated to reflect changes correlating to initial software release.	August 2006

1 *Introduction*

1.1 About the Intel Laptop Gaming TDK

The Intel Laptop Gaming Technology Development Kit (TDK) enables game developers and game engine developers to design and develop laptop-aware games. The Intel Laptop Gaming TDK does this by providing functionalities that dynamically detect changes in system-state information, such as Power, Network and Processor. Using this information, game developers can alter game behavior to maximize game play on laptop computers using WiFi technology. Game developers can also use the Intel Laptop Gaming TDK to receive notifications of platform events to gain recommendations on how to modify the complexity of game elements to achieve maximum gameplay. In this context, “maximum game play” means utilizing mobility features in a game in a way that reduces power consumption, which in turn allows for maximum battery life while playing the game.

1.2 Key Features

- **Enhanced Battery Power Management:** The Intel Laptop Gaming TDK allows game developers to adapt software performance based on whether the laptop computer is running on AC or battery power. If the laptop is using battery power, the Intel Laptop Gaming TDK provides for maximum power efficiency by enabling a range of features such as identifying the number of batteries being used, their power levels and estimated time remaining. The Intel Laptop Gaming TDK also makes it possible to identify the laptop’s existing power scheme, and to change that scheme based on the power source.
- **Processor-Specific Settings Capability:** The Intel Laptop Gaming TDK allows for games to identify several characteristics of a laptop’s processor, including the number of CPU cores, the current and maximum frequencies, the level of CPU utilization, and the number of running threads.
- **Network Support:** The Intel Laptop Gaming TDK supports both Ethernet and WiFi adapters that conform to 802.11a/b/g standards. It provides game developers with the ability to build games that identify the available network connections and their IP addresses, to enable or disable any given network adapter, and to determine available network bandwidth and WiFi signal strength.
- **High Performance with Minimal Footprint:** The Intel Laptop Gaming TDK provides game developers the ability to design games for a wide variety of laptop models and usage scenarios without placing an unreasonable demand on system resources. The Intel Laptop Gaming TDK should not require more than five percent of system memory or CPU processing capacity.

1.3 Requirements

The following are requirements for installing and using the Intel Laptop Gaming TDK (as tested). Other configurations may also work, but have not been verified.

Hardware:

- Intel® Core™ Duo Processor or faster recommended
- Minimum 512 MB Random Access Memory (RAM)
- Minimum 128 MB Video RAM, using AGP 4x or PCI Express Bus



- 802.11 a/b/g Wireless Local Area Network (WLAN), or Fast Ethernet Local Area Network (LAN) connection

Software:

- Microsoft® Windows® XP, Professional/Home/Media Center Edition
- DirectX® 9 SDK Dec 2005 or later

1.4 Product License

IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING.

Do not copy, install, or use the Materials provided under this license agreement ("Agreement"), until you have carefully read the following terms and conditions.

By copying, installing, or otherwise using the Materials, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, do not copy, install, or use the Materials.

End User License Agreement for the Intel Laptop Gaming Technology Development Kit

1. LICENSE DEFINITIONS:

- A. "Materials" are defined as the software, documentation and other materials, including any updates and upgrade thereto, that are provided to you under this Agreement.
- B. "Sample Source" is the source code file(s) that: (i) demonstrate(s) certain limited functions that may be developed to utilize the Intel Laptop Gaming Technology Development Kit product; (ii) are identified as sample source code; and (iii) are included in the technology development kit but are separate and independent from your copy of the Intel Laptop Gaming product.

2. LICENSE GRANT:

- A. Subject to all of the terms and conditions of this Agreement, Intel Corporation ("Intel") grants to you a non-exclusive, non-assignable, copyright license to use the Materials for your internal product development purposes only.
- B. The Materials are under development by Intel and may include software, specifications, prototypes, or other materials. Title to the Materials remains with Intel or its suppliers. You shall not mortgage, pledge or encumber the Materials in any way. You shall return all Materials, keeping no copies, upon termination or expiration of this Agreement.
- C. You may modify and redistribute the Materials or any portion of the Materials thereof, but you cannot redistribute it as the Intel Laptop Gaming TDK.
- D. Subject to all of the terms and conditions of this Agreement, Intel grants to you a non-exclusive, non-assignable copyright license to modify the Sample Source, or any portions thereof, and distribute the Sample Source, or any portions thereof, as part of the product or application you developed using the Sample Source.
- E. EXCEPT AS PROVIDED HEREIN, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED HEREIN.

3. **OTHER LICENSE RESTRICTIONS:**

- A. If you receive your first copy of the Materials electronically, and a second copy on media, or visa versa, then you may use the second copy only in accordance with your applicable license stated in this Agreement, or for backup or archival purposes. You may not provide the second copy to another user.
 - B. You may NOT: (i) use or copy the Materials except as provided in this Agreement; (ii) rent or lease the Materials to any third party; (iii) assign this Agreement or transfer the Materials without the express written consent of Intel; (iv) adapt, or translate the Materials in whole or in part; (vi) attempt to modify or tamper with the normal function of a license manager that regulates usage of the Materials; or (v) sublicense or transfer the Materials and derivatives thereof to any third party.
 - C. This Agreement provides you with a single user license. This means you as an individual may install and use the Materials on an unlimited number of computers provided that you are the only individual using the Materials and only one copy of the Materials is in use at any one time. A separate license is required for each additional use and/or individual user in all other cases. If you are an entity, Intel grants you the right to designate one individual within your organization to have the sole right to use the Materials in the manner provided above.
 - D. You shall not make any statement that your product is "certified", or that its performance is guaranteed, by Intel, by virtue of this license grant. You shall not use Intel's name or trademarks to market your product without written permission from Intel. You shall prohibit disassembly and reverse engineering of the Material and you shall not publish reviews of Materials designated as beta without written permission by Intel. You shall indemnify, hold harmless, and defend Intel and its suppliers from and against any claims or lawsuits, including attorney's fees, that arise or result from your distribution of any product.
4. **COPYRIGHT:** Title to the Materials and all copies thereof remain with Intel or its suppliers. The Materials are copyrighted and are protected by United States copyright laws and international treaty provisions. You will not remove any copyright notice from the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided herein, Intel does not grant any express or implied right to you under Intel patents, copyrights, trademarks, or trade secret information.
5. **NO WARRANTY AND LIMITED REPLACEMENT:** THE MATERIALS AND INFORMATION ARE PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. If the media on which the Materials are furnished are found to be defective in material or workmanship under normal use for a period of ninety (90) days from the date of receipt, Intel's entire liability and your exclusive remedy shall be the replacement of the media. This offer is void if the media defect results from accident, abuse, or misapplication.
6. **LIMITATION OF LIABILITY:** THE ABOVE REPLACEMENT PROVISION IS THE ONLY WARRANTY OF ANY KIND. INTEL OFFERS NO OTHER WARRANTY EITHER EXPRESS OR IMPLIED INCLUDING THOSE OF MERCHANTABILITY, NON-INFRINGEMENT OF THIRD- PARTY INTELLECTUAL PROPERTY OR FITNESS



FOR A PARTICULAR PURPOSE. NEITHER INTEL NOR ITS SUPPLIERS SHALL BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

7. **UNAUTHORIZED USE:** THE MATERIALS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE IN ANY TYPE OF SYSTEM OR APPLICATION IN WHICH THE FAILURE OF THE MATERIALS COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR (E.G MEDICAL SYSTEMS, LIFE SUSTAINING OR LIFE SAVING SYSTEMS). Should the buyer purchase or use the Materials for any such unintended or unauthorized use, the buyer shall indemnify and hold Intel and its officers, subsidiaries and affiliates harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of product liability, personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Intel was negligent regarding the design or manufacture of the part.
8. **USER SUBMISSIONS:** You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this Agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes.
9. **TERMINATION OF THIS LICENSE:** This Agreement becomes effective on the date you accept this Agreement and will continue until terminated as provided for in this Agreement. Intel may terminate this license at any time if you are in breach of any of its terms and conditions. Upon termination, you will immediately return to Intel or destroy the Materials and all copies thereof.
10. **U.S. GOVERNMENT RESTRICTED RIGHTS:** The Materials are provided with "RESTRICTED RIGHTS". Use, duplication or disclosure by the Government is subject to restrictions set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Materials by the Government constitutes acknowledgment of Intel's rights in them.
11. **APPLICABLE LAWS:** Any claim arising under or relating to this Agreement shall be governed by the internal substantive laws of the State of Delaware, without regard to principles of conflict of laws. You may not export the Materials in violation of applicable export laws.

* Other names and brands may be claimed as the property of others

2 Getting Started

2.1 Installing the Intel Laptop Gaming TDK

Follow the steps below to install the Intel Laptop Gaming TDK:

1. Run the program setup executable file. The following screen displays:



2. Click on the **Next** button. The license agreement screen displays.



3. Read the license information. If you agree with the license terms and want to continue the installation, select the **I accept the terms in the license agreement** option and click on the **Next** button.



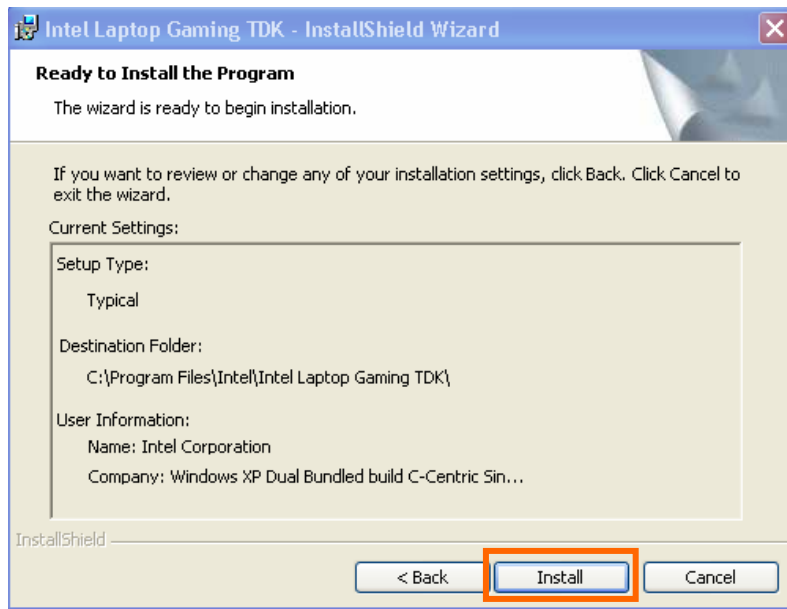
Note: If you do not accept the license terms, the program installation will not continue.

4. Choose a drive and directory on your computer to install the software. You may accept the default path, or click on the **Change** button if you want to install to a different drive or directory. When you have chosen where to install the software, click on the **Next** button.

Note: For the purpose of the rest of this document, the directory where the TDK is installed will be referred to as the <TDK_INSTALL_DIR>. The default value of the <TDK_INSTALL_DIR> will be C:/Program Files/Intel/Intel Laptop Gaming TDK/.

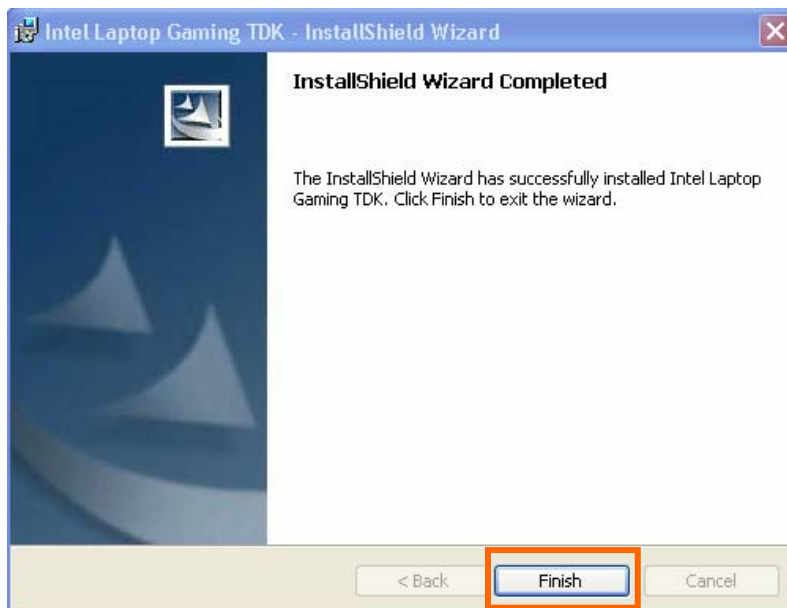


5. The screen image on the following page shows the installation settings that the installer will use to install the software. If you want to change any of these settings, click the **Back** button. If you find the settings acceptable, click on the **Install** button. The installation program will install the software.



6. When the installation program has completed installing the software, click on the **Finish** button. The Intel Laptop Gaming TDK is now installed. The program release notes will automatically display when the installation program exits.

Note: If after installing the software you need to re-install the Intel Laptop Gaming TDK or want to change any of the original installation settings, re-run the installation program.





3 Using the Laptop Gaming TDK

3.1 Description of Files

The files in the Intel LaptopGamingTDK that are relevant to users of the TDK are as follows:

Filename	Location	Description
GamingTDK-Provider.sln	<TDK_INSTALL_DIR>/Source Code\Laptop Gaming DLL\GamingTDK-Provider	Microsoft Visual Studio* v7.1 solution file that contains the TDK code. The TDK library can be built by compiling this solution file.
IntelGamingTDKAPI.h	<TDK_INSTALL_DIR>/Source Code\Laptop Gaming DLL\GamingTDK-Provider	This file contains the interface class definition, and is the ONLY file that the user of the TDK needs to include in the client application.
ILGDefines.h	<TDK_INSTALL_DIR>/Source Code\Laptop Gaming DLL\GamingTDK-Provider	This file contains the definitions for the event names.
CommandClient.sln	<TDK_INSTALL_DIR>/Source Code/Samples	This is a Visual Studio v7.1 command line application that shows the barebones usage of the TDK APIs.
BasicPhysics_2003.sln	<TDK_INSTALL_DIR>/Source Code/Samples/	This is a Visual Studio v7.1 solution file that contains a DirectX sample application showcasing a use of TDK.
BasicPhysics_2005.sln	<TDK_INSTALL_DIR>/Source Code/Samples	This is a visual studio version 8.0 solution file that contains a DirectX sample application showcasing use of TDK.

3.2 Setting Up Visual Studio* for Use with the Intel Laptop Gaming TDK

Follow the steps below to setup the Visual Studio environment to build the client application with the Intel Laptop Gaming TDK:

1. Add the following directory to the VC++ Include path:
`<TDK_INSTALL_DIR>/Source Code/Laptop Gaming DLL/GamingTDK-Provider`
2. Add the following directory to the VC++ library path:

<TDK_INSTALL_DIR>/Source Code/Laptop Gaming DLL/GamingTDK-Provider/Bin/Lib Files/

Note: If the libraries provided with the Intel Laptop Gaming TDK are not compatible with the version of Visual Studio that the client application uses, then build the TDK solution file to obtain compatible libraries.

3.3 Initiating the Laptop Gaming TDK Interface

The Intel Laptop Gaming TDK interface class provides the complete range of features of the TDK, implemented as a single class. To get the single instance of the TDK interface, the developer can use the following code snippet:

```
#include <IntelGamingTDKAPI.h>

IntelLaptopGamingTDKInterface *IGT =
    IntelLaptopGamingTDKInterface::GetTDKInterface();
```

3.4 Calling Modes

There are two calling modes for the APIs of the Intel Laptop Gaming TDK: **polling mode** and **thread event-based mode**. The Intel Laptop Gaming TDK allows both these modes to coexist in the client code. This section describes how a game developer can implement these two modes, with appropriate code snippets from a sample application. The sample application has a checkbox “Enable Mobility Thread” that when checked, will run the application in a threaded event-based mode.

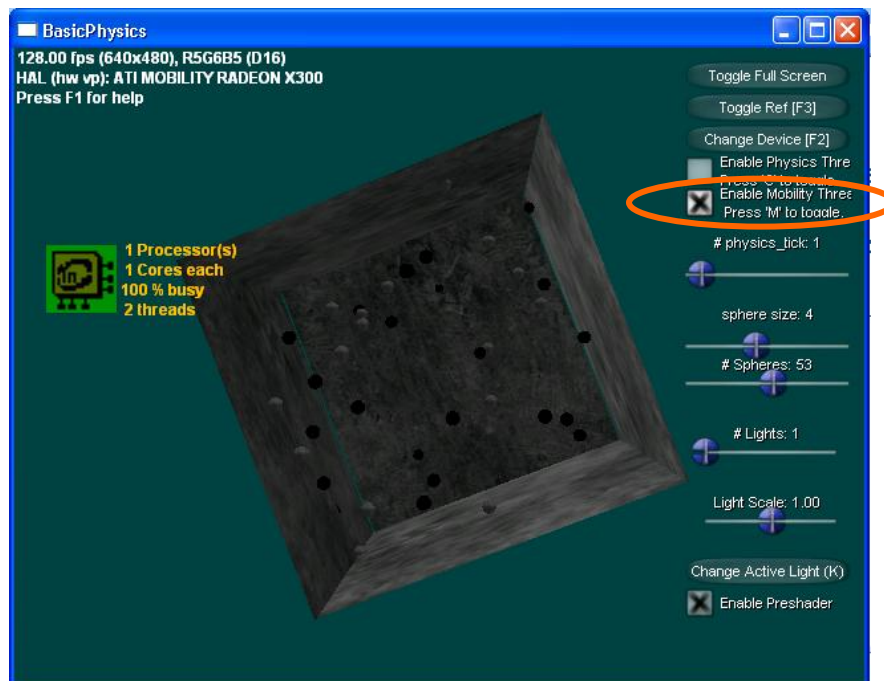


Figure 1: Interface with the Enable Mobility Thread checkbox enabled



3.4.1 Polling Mode

In polling mode, the game loop continuously polls for the data using the Intel Laptop Gaming TDK interface APIs. In this mode, the TDK APIs can be issued as follows:

```
void game_render_loop() {
    // Do game stuff
    . . .

    static int pwrSource = 0;
    // Assuming 60fps, we will poll every 5 secs.
    static int pollingFrequency = 300;
    if (pollingFrequency == 0) {
        // Get the Power Source
        pwrSource = IGT->GetPowerSrc();
        // Reset polling frequency
        pollingFrequency = 300;
    } else {
        pollingFrequency--;
    }
    if (pwrSource == AC_Power) {
        // Max physics
        // Max effects
    } else if (pwrSource == Battery_Power) {
        // Min physics
        // Min effects
    }
}
```

Sample 1: Polling mode example

3.4.2 Threaded Event-Based Mode

When the game developer wants to keep polling for the platform characteristics separate from the game code, choose to register for events from the TDK. Registering for an event causes the TDK to spawn a thread for monitoring that event. Optionally, the game developer can also provide a threshold value that will ensure that the user-defined event handler will be called when the threshold is reached. Depending on the type of event, it could be a lower limit threshold or an upper limit threshold.

In order to receive and handle the events generated by the TDK, follow these steps:

Create an event observer by deriving a class from the **Notifiable** class and implementing the method **notify()**, as shown below:

```
// Class for receiving event notification in threaded mode.
class CGamingTDKObserver
: public Notifiable
{
public:
    CGamingTDKObserver(IDirect3DDevice9* device)
        : Notifiable(), pD3DDevice(device)
    {
    }
    virtual void notify(int event, long value = 0);
private:
    IDirect3DDevice9* pD3DDevice;
};

void CGamingTDKObserver::notify(int event, long value)
{
    CRITICAL_SECTION cs;
    InitializeCriticalSection(&cs);
    EnterCriticalSection(&cs);
    if (event == BATTERY_LIFE_PRCNT_CHANGED || event == POWER_SRC_CHANGED) {
        // Handle event here
    }
    if (event == CURRENT_CPU_UTIL) {
        // Handle event here
    }
    LeaveCriticalSection(&cs);
    DeleteCriticalSection(&cs);
}

CGamingTDKObserver observer;
```

Sample 2: Receiving and handling TDK-generated events

To perform an action defined by the notify() method of the observer, register the event observer to the event Id using the

IntelLaptopGamingTDKInterface::RegisterEventObserver() method. An example from the sample application is shown below:

```
IGT->RegisterEventObserver(LOW_BATTERY_REACHED, &observer);
IGT->RegisterEventObserver(POWER_SRC_CHANGED, &observer);
IGT->RegisterEventObserver(CURRENT_CPU_UTIL, &observer);
```

Sample 3: Registering an event ID to the Event Observer

3.5 Showing Power Status

One of the primary usage models is to show the status of the platform power in the game HUD (Heads-Up Display). The sample application below shows a sample HUD with power meter. The power meter detects whether the laptop is running on battery or AC power, and measures the percentage of battery charge remaining. It also shows the estimated remaining time left before the battery runs out of charge.

The power meter can be toggled on or off by pressing the **b** key on the keyboard.



Figure 2: Battery meter showing remaining charge and battery charge time

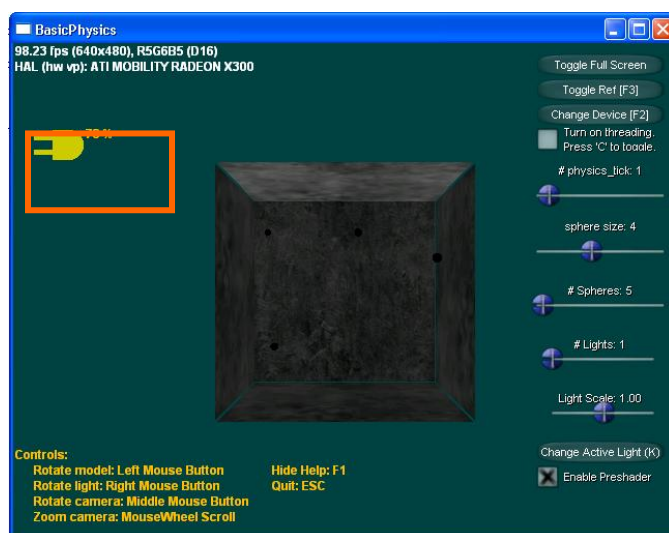


Figure 3: Battery meter showing AC power and remaining battery power

When the laptop is using AC power, the battery meter displays the remaining battery charge. When the laptop is using battery power, then in addition to displaying the remaining charge the battery meter also shows the time remaining before the battery runs out of charge.

The key TDK interface methods used to build the battery meter in the sample application are shown below, with some code snippets from the sample application.

The **UpdateBatteryData** method is used as a single entry point to get the latest power information. This function is called both in the polling mode and in the threaded mode. The key methods used in this function are:

- IntelLaptopGamingTDKInterface::GetPowerSrc
- IntelLaptopGamingTDKInterface::GetPercentBatteryLife
- IntelLaptopGamingTDKInterface::GetBatteryLifeTimeRemaining

```
int powerSource = AC_Power;
unsigned long batteryLifeTimeRemaining;
int batteryChargeRemaining;
...
CMeter* UpdateBatteryData(IDirect3DDevice9* pD3DDevice)
{
    // Get the power source
    powerSource = IGT->GetPowerSrc();

    // Get the powerMeter depending on whether we need battery
    meter or acpower meter
    if (powerSource == AC_POWER) {
        // Show battery meter
        batteryMode = false;
    } else if (powerSource == BATTERY_POWER) {
        batteryMode = true;
    } else {
        return powerMeter;
    }
    batteryChargeRemaining = IGT->GetPercentBatteryLife();

    if (batteryMode) {
        batteryLifeTimeRemaining = IGT->GetBatteryLifeTimeRemaining();
    }
    return powerMeter;
}
```

Sample 4: UpdateBatteryData method used as a single entry point

The code snippet on the next page shows the use of these functions in the sample application. The render loop of the game updates the HUD with the data stored in the variables shown in bold.



```
void render_loop(IDirect3DDevice9* pD3DDevice)
{
    ...
    while (1) {
        // game rendering
        ...

        // render the battery meter
        CMeter* meter = UpdateBatteryData(pD3DDevice);
        if (meter) RenderMeter(meter);

        // Show the text for the battery charge remaining.
        CDXUTTextHelper hlpr (pFont, pSprite, 10);
        hlpr.Begin();
        ...
        wchar_t* text1 = new wchar_t[256];
        swprintf(text1,L"%2d %%",batteryChargeRemaining);
        hlpr.DrawTextLine(text1);
        ...
        hlpr.End();
        delete[] text1;

        // Convert the battery life time remaining to H:M:S format.
        if (batteryMode) {
            // Show the text for the battery life time remaining.
            int hours = batteryLifeTimeRemaining/3600;
            int mins = (batteryLifeTimeRemaining%3600)/60;
            int secs = (batteryLifeTimeRemaining%3600)%60;

            wchar_t* text2 = new wchar_t[256];
            hlpr.Begin();
            ...
            swprintf(text2,L"%2d:%2d:%2d",hours, mins, secs);
            hlpr.DrawTextLine(text2);
            ...
            hlpr.End();
            delete[] text2;
        }
    }
}
```

Sample 5: HUD updated with stored data

If the game is using the Intel Laptop Gaming TDK in a threaded mode, depending on the type of event, the game developer can update the value from the second parameter for the notify method.

In the code snippet below, it is set to show the battery meter for 300 frames since the last time that the power source or the battery percent changed.

```
void CGamingTDKObserver::notify(int event, long value)
{
    EnterCriticalSection(&cs);
    if (event == POWER_SRC_CHANGED) {
        batteryMode = (IGT->GetPowerSrc() == Battery_Power);
    }
    if (event == BATTERY_LIFE_PRCNT_CHANGED) {
        batteryChargeRemaining = value;

        if (batteryMode) {
            batteryLifeTimeRemaining =
                IGT->GetSecBatteryLifeTimeRemaining();
        }
    }
    LeaveCriticalSection(&cs);
}
```

Sample 6: Setting the number of frames to show the battery meter

3.6 Reducing Power Consumption by Keeping Track of Network Adapter Information

Wireless radio and adapters use substantial power. If the laptop is being powered by a battery, and the game is being played in single-player mode with no use of the network connection by other applications on the system, it would save power and prolong gameplay to turn off the wireless adapter and radio.

In the following sample code, the Intel Laptop Gaming TDK detects if the wireless receiver is enabled while there is a wired connection. In this case, the application alerts the user to turn off the wireless adapter to save power.

First determine if the laptop is connected. This is accomplished by checking the number of active IP addresses using the **GetIPAddresses()** API.



```
int numIPs;
char** ipaddresses = IGT->GetIPAddresses(&numIPs);
bool bConnected = false;
if (numIPs == 0) {
    // Not connected
    bConnected = false;
    bVPN = false;
} else if (numIPs > 1) {
    // Possibly in VPN
    bConnected = true;
    bVPN = true;
} else {
    if (strcmp(ipaddresses[0], "0.0.0.0") == 0) {
        // Single connection.
        bConnected = false;
        bVPN = false;
    } else {
        // Single connection.
        bConnected = true;
        bVPN = false;
    }
}
```

Next, determine if the connection is a wireless or a wired connection using the `IsWirelessAdapterConnected()` method.

```
// if not wireless connection, the connection must be wired
if (bConnected) {
    if (IGT->IsWirelessAdapterConnected()) {
        bWiredConnection = false;
    } else {
        bWiredConnection = true;
    }
}
```

Sample 8: Identifying whether the connection is wired or wireless

Finally, alert the user to turn off wireless if the laptop is connected through a wired connection and is running on battery power.

Different OEM laptops handle enabling or disabling of wireless in different ways. For example, on the IBM* (or Lenovo*) Thinkpad* T42 and T43 models, using the Fn+F5 to turn off the wireless turns only the radio off; it does not disable the wireless adapter. In other OEM laptops, a similar operation could disable the wireless adapter. To determine consistently if the laptop wireless is indeed disabled, the game developer can check for the following:

```
if (!IGT->GetWirelessEnabled() ||  
    ((IGT->GetWirelessEnabled() &&  
      (IGT->GetWirelessSignalStrength() == -1))  
    ) {  
    // Wireless is indeed disabled  
}
```

Sample 9: Confirming that a wireless connection is disabled



4 Recommendations

4.1 Troubleshooting

The following are possible issues that may arise when using the Intel Laptop Gaming TDK, and suggested solutions:

- If the wireless radio is turned-off on an IBM Thinkpad T42, **IsWirelessAdapterEnabled()** will still return true since from the OS point, the hardware is enabled but the OEM provided another layer to turn-off the transmission. Use this in combination with signal strength to determine if the wireless card is actually disabled.
- To handle the event notifications, the game developer will have to derive a class from the **Notifiable** class and implement the **notify()** method. If the developer wants to use the **Direct3D*** device in this method, then make sure that the device was created for multi-threaded use (provide the device setting: **D3DCREATE_MULTITHREADED**). If this is not used, it will crash when the events are registered. To work around this problem, either use the **D3DCREATE_MULTITHREAD** setting or update a thread safe variable specifying that the battery meter display needs updating. The main thread (for the game loop) will decide to update the battery meter if that variable is set. This way, only the main thread needs to use the **D3D** device.
- **GetDisplayScreenBrightness()** and **SetDisplayScreenBrightness()** APIs change the gamma factor and may not be available because it succeeds only for devices with drivers that support downloadable gamma ramps in hardware.

4.2 Recommendations For Increasing System Battery Life*

*Note that the experiments below have been conducted on specific system configurations and a few production games as well as sample applications. The behavior may vary for each game depending on implementation details as well as system configuration.

4.2.1 Detecting Power Source Changes

Using the **POWER_SRC_CHANGED** event or polling for the **GetPowerSrc()** offered by the Intel Laptop Gaming TDK in the game loop, game can detect when the power source is changed from AC to battery. Alternately, the game could also set some policies to track battery life/time remaining, such as watching for 40% battery life left using the **LOW_BATTERY_REACHED** event or **GetPercentBatteryLife()** API in polling mode. This information can then be used to alter some of the game settings/behaviour in order to save power leading to more play time.

- Initial experimentation with some game titles has shown that when the frame rate is capped to a lower value, it may potentially increase the battery savings. When running on Intel Integrated Graphics (IIG), this may reduce the CPU load which can lead to CPU and platform power savings. When running on discrete graphics, capping the frame rate to a lower value can potentially reduce computations on the GPU which can lead to reduced power consumption. This could result in increasing battery life of the system.

- If a game offers options to switch to low video settings such as ‘Shader’, ‘Shadow Effect’, ‘Water Effect’, ‘Texture’ etc, these could potentially help save power as compared to running with high display settings. Lowering the video options may decrease computations on CPU/GPU and lead to reduced power consumption, also increasing battery life
- Experiments have indicated that accessing a DVD drive could increase power consumption during spin-up and read operations on the drive. Some games could be loading movie files and other such media into memory as needed directly from the DVD disk. The battery life could potentially increase if such operations are minimized when the system is battery powered.

4.2.2 Detecting CPU Utilization

With use of Intel SpeedStep Technology, the processor can change operating frequency based on CPU usage of the system in a portable/laptop power scheme. Experiments have indicated that platform power consumption is lower when operating in a lower frequency state. The Intel Laptop Gaming TDK offers `GetNumProcessors()` and `GetCurrentCPUUtilization(CPUNum)` to determine the number of processors/cores available and the current CPU usage for each core. With the help of these APIs, the game developer can potentially determine if any imbalance exists among cores/processors and then may be able to multi-thread the code to get balanced utilization across the processors/cores. Using multi-threading may reduce effective CPU usage as well, causing the system to run in a lower frequency state resulting in reduced platform power consumption and increasing battery life.

4.2.3 Monitoring Connectivity Status

Experiments done with enabling and disabling wireless adapter has indicated that system power consumption increases when wireless is enabled. If a game is running in single-player mode and any other application is not using the wireless connection, the radio adapter can be disabled leading to power savings. The TDK offers **`GetConnectedAdapter()`** API to determine current network connection. If current network connection is wired and wireless radio is enabled (can be obtained by calling **`IsWirelessAdapterEnabled()`** and **`IsWirelessAdapterConnected()`** APIs), the game can prompt user to turn off/disable the radio potentially leading to increased battery life.

4.2.4 Changing Display Resolution

Experiments have indicated that lowering the display resolution of a game can potentially help reduce power consumption. The benefits of this technique vary considerably for each game, based on the implementation. The Intel Laptop Gaming TDK offers **`SetDisplayResolution()`** API to change the resolution. Based on dynamic state change in system power status, this API can be used to alter resolution.



A Appendix: API Specifications

The following are API specifications used with the Intel Laptop Gaming TDK. The usage of the API and events is demonstrated in Command-Client application included with the TDK. See CommandClient.cpp for detailed usage.

A.1 General APIs

Function Name	Input	Output	Description
GetTDKInterface	None	None	This is a static function which returns pointer to IntelLaptopGamingTDKInterface class. This pointer is then used to invoke functionality offered by the TDK. This pointer needs to be deleted after the use.
RegisterEventObserver*	Int EventID, Notifiable* pNotif Int EventParam = -1 Int UpdateInterval = 2000	None	This function controls beginning of the event notification model (threaded model). It takes in eventID to watch for. See event tables below to get individual event IDs. The second parameter is pointer to derived notifiable class where the TDK code will call back when specified event is triggered. The third parameter is a control parameter for a specific event. This is an optional parameter and set to -1. e.g. for BATTERY_LIFE_PRCNT_CHANGED this parameter indicated the battery life value to watch for. The last parameter indicated update interval which is set to 2000ms by default. This can be overridden by user specific value.
UnregisterEventObserver*	Int EventID	None	This function control halting of event notification model (threaded model). It takes in EventID as parameter to halt corresponding running thread.

* These functions provide an alternative to polling methods to get various device properties. Developers can choose to use polling method as needed.

A.2 Power

Function Name	Input	Output	Description
IsLaptop	None	Bool	This function returns true if the current system is a laptop, false otherwise. This is determined by checking for the availability of a system battery.
GetPwrSrc	None	Type: Int Possible Values: AC_POWER 1 BATTERY_POWER 0 UNDEFINED 255	<p>This function returns the type of power source currently being used by the mobile platform, as well as the percentage of battery life remaining. When the laptop is powered using an AC adapter, this function will return 1; if the laptop is powered using battery, this function will return 0.</p> <p>This function should ordinarily be called only when running on battery power.</p>
GetPercentBatteryLife	None	Type: Int	This function returns the battery usage in percent. It will return the battery usage independent of whether the laptop is powered by an AC adapter or not. It will return the average battery lifetime among all the available batteries. Returns -1 when function call fails.
GetSecBatteryLifeTime Remaining	None	Type: unsigned long	This function returns the estimated time (in seconds) remaining before the battery runs out of charge, or will return -1 if remaining seconds are unknown. It will return an accumulated battery lifetime if there is more than one battery.
GetCurrentPowerScheme	None	Type: enum PowerScheme	This function returns current power scheme set for the system. Return values are Throttle_None, Throttle_Constant, Throttle_degrade and Throttle_Adaptive.
SetCurrentPowerScheme	Enum PowerScheme	Bool	This function takes a new power scheme as input and changes the current power scheme to the new one. Returns true when function succeeds; false otherwise.



Event Name	Description
POWER_SRC_CHANGED	This event is triggered when change in power source occurs between AC and battery. This is an alternative to using GetPowerSrc() API described above in polling mode. Developers can use this event to detect a change in power source and to alter game behavior accordingly in order to increase battery life.
LOW_BATTERY_REACHED	This event is triggered when user specified battery life remaining is reached. The developers can register to watch for x% battery life and take action when the specified battery life is reached.
BATTERY_LIFE_PRCNT_CHANGED	This event is triggered when each time change in battery life percent occurs.

A.3 Network

Function Name	Input	Output	Description
Get80211SignalStrength	None	Type: int	If the laptop has an 802.11 wireless adapter and if the wireless adapter is not disabled, then this function returns the signal strength of the active wireless adapter from 0-100. Returns -1 when wireless adapter is disabled.
IsWirelessAdapterConnected	None	Type: bool	If the laptop has a wireless adapter and if it connects to the network using the wireless adapter, then this function returns true. Otherwise, it returns false.
IsWirelessAdapterEnabled	None	Type: bool	If the laptop has a wireless adapter and if the adapter is enabled to receive wireless signal, then this function returns true. Otherwise, it returns false.
GetAllNetworkAdapters	int*	Type: Char**	This function returns list of all network adapters present on the system. It takes in pointer to int to return the number of network adapters present by reference.
GetConnectedAdapter	None	Type: Unsigned char*	This function retrieves the name of the currently connected adapter. If the laptop is connected via VPN, it may return name of VPN adapter based on whether VPN gets registered as physical adapters and if it comes first

			in the list. If there is no active network connection, this function returns 'No Adapter Connected.'
GetIPAddresses	Int*	Type: Char**	This function returns an array of IP addresses. Multiple IPAddresses may be present in case when laptop is connected via VPN. This function also returns the number of IPAddresses via reference using input parameter int*. When no active network is present, this returns 0.0.0.0.

Event Name	Description
WIRELESS_SIGNAL_STRENGTH	This event is triggered every sampling interval (default 2 seconds) to get the current wireless signal strength.
CONNECTIVITY_STATUS	This event is triggered when system connectivity status changes. (i.e. when at least one network connection becomes available and when the last active network connection is dropped).

A.4 Display

Function Name	Input	Output	Description
GetDisplayColorDepth	None	Type: Int	This function returns the current color depth settings (i.e., 16-bit/32-bit). Returns -1 when the function fails.
GetDisplayResolution	None	Type: Struct Resolution	This function returns struct 'Resolution' with height and width members. If the function call fails, both members are set to -1.
GetDisplayScreenBrightness	None	Type: Float	This function returns current screen brightness (gamma value). Valid values are between 0 – 100. Returns -1 when the function fails. May not work for all systems (succeeds only for devices with drivers that support downloadable gamma ramps in hardware.)



SetDisplayColorDepth	Int	Type: Bool	This function takes a new depth buffer value and changes the current value with the new one. If the change is successful, this functions returns true. False otherwise.
SetDisplayResolution	Struct Resolution	Type: Bool	This function takes in a new resolution structure and changes the current resolution with the new one. Returns true when function is successful, false otherwise.
SetDisplayScreenBrightness	Float	Bool	This function takes in a new brightness value (gamma value) between 0-100 and changes the current gamma with the new value. Returns true when function is successful, false otherwise. May not work for all systems (succeeds only for devices with drivers that support downloadable gamma ramps in hardware).
RestoreOriginalBrightness	None	Type: Bool	This function lets the user restore the brightness to the value it was set before calling SetDisplayBrightness() function. Returns true when function is successful, false otherwise. May not work for all systems. (succeeds only for devices with drivers that support downloadable gamma ramps in hardware).

A.5 Processor

Function Name	Input	Output	Description
GetNumCoresPerProcessor	None	Type: Unsigned int	This function returns the number of cores available on the system per physical processor.
GetNumProcessors	None	Type: Unsigned int	This function returns the number of processors available on the system including physical and logical processors.
GetCurrentCPUFrequency	None	Type: Unsigned long	This function returns the current frequency at which processor is running in MHz.
GetMaxCPUFrequency	None	Type: Unsigned long	This function returns the max frequency at which CPU is running in MHz.
GetCurrentCPUUtilization	Int CpuNum = -1	Type: int	This function returns the current CPU utilization. If there are multiple processors, it will return average CPU utilization across all CPUs. If no input parameter provided, it returns average CPU utilization. By using the input parameter as CPU num, it returns utilization for that CPU. This function returns -1 if COM initialization fails.
GetNumActiveThreadsForProcess	Const char*	Int	This function takes a process name (no .exe, just the name) and returns number of active threads for that process. If NULL is passed in as input paramater, this function returns number of threads associated with the



			current process. This function returns -1 if it fails and -2 if input parameter is invalid.
--	--	--	---

Event Name	Description
CURRENT_CPU_UTIL	This event is triggered every sampling interval (default 2 seconds) to get the current CPU utilization. A parameter can be specified to get utilization across each core. The default mode is to get average utilization across the system.
CURRENT_CPU_FREQ	This event is triggered every sampling interval (default 2 seconds) to get current CPU frequency.

A.6 Bandwidth

Function Name	Input	Output	Description
GetCurrentTxRate	None	Unsigned long	This function returns the current system transmit rate in bytes/sec. If failed, it returns 4294967295
GetCurrentRxRate	None	Unsigned long	This function returns the current system receive rate in bytes/sec. If failed, it returns 4294967295
GetTotalTxRxRate	None	Unsigned long	This function returns the current total (Tx + Rx) system rate in bytes/sec. If failed, it returns 4294967295

Event Name	Description
CURRENT_TX_RATE	This event is triggered every sampling interval (default 2 seconds) to get current system transmit rate.
CURRENT_RX_RATE	This event is triggered every sampling interval (default 2 seconds) to get current system receive rate.
CURRENT_TOTAL_BW	This event is triggered every sampling interval (default 2 seconds) to get the current system total (Tx + Rx) rate.