



TalkFX and RYOS // MK PRO SDK Doc v0.2

Roccat Testlab*

Version History

October 15, 2013

- added TalkFX for other ROCCAT Products

September 26, 2013

- initial version

*contact: support@roccat.org

Contents

Introduction	1
1 TalkFX enabled Hardware	2
1.1 TalkFX Zones	2
1.2 TalkFX Protocol Info	2
2 Ryos Capabilities	3
2.1 Ryos Hardware	3
2.2 Ryos Protocol	3
2.3 Ryos LED Functionality	3
3 SDK Usage	5
3.1 Basics	5
Initializing TalkFX Connection • Colour Control • Ryos specific TalkFX Connection • Taking Control of the Ryos LEDs • Get the Ryos Lights Blinking	
3.2 Advanced Ryos	7
Best Performance	
3.3 Unobstrusive Integration	8
3.4 Example Integration	8
Preparation • Placing Init Code • Show Ammo and Health • Blink on Damage • Reflection	
3.5 Ryos CLI	10
4 Effect Design Considerations	11
5 APPENDIX: Roccat Talk API header	13

Introduction

This documentation explains how to access the LEDs of the Ryos MK PRO keyboard from within your program. It also covers the TalkFX functionality of other ROCCAT devices, which enables access to the RGB LEDs. Furthermore it covers a quick style-guide with recommendations for adding support for the Ryos and TalkFX in general into your Application.

Please read this document carefully, since using the SDK incorrectly, may cause unexpected (and unnecessary) slowdowns in your application.

We tried to keep the API slim and functional to suit all needs that could arise, if however you find that the API is missing a key feature, do not hesitate to contact us at support@roccat.org.

1. TalkFX enabled Hardware

Many Roccat devices are shipped with integrated RGB LED lighting, this includes currently the Kone[+], KoneXTD, Kone Pure, Kone Pure optical mice and the Isku FX keyboard.

With this SDK you can modify the colour and brightness of these LEDs in your software.

There is no need to worry about individual TalkFX capable products, all are connected with the Roccat Talk software on a single "colour-bus". So there is just a one method in the SDK you need to use for changing the colour of all connected TalkFX devices:

```
void Set_LED_RGB(BYTE bZone, BYTE bEffect, BYTE bSpeed, BYTE colorR, BYTE colorG, BYTE colorB);
```

You can call this method whenever you like inside your code, provided you have established connection to the TalkFX software in the background during your application startup.

1.1 TalkFX Zones

In TalkFX we differentiate between two "zones" you can access through the API, **Ambient** and **Event**.

Ambient represents slow changing atmospheric lighting, for example a Ambilight-style lighting effect. Or a slow green pulse when the player is poisoned in an adventure game.

Event represents short atmospheric blinks. For example a red flash, when being struck by an enemy, or a white flash when a checkpoint is reached.

Currently there is no ROCCAT device at retail that fully differentiates between these two effect areas. So if you are worried about compatibility then concentrate on the Event zone. However Ambient-zone support for already released ROCCAT hardware and upcoming ROCCAT hardware is in the works, and can be expected soon. Hence encourage you to implement effects for both zones so you can guarantee early support for cool lightFX in your software.

1.2 TalkFX Protocol Info

To communicate with the current TalkFX software, the native Windows Window-Message-Protocol is used in the background. Each change of the light configuration requires a window message to be sent.

2. Ryos Capabilities

In this section, we'll give you a quick explanation, on what is possible with the Ryos. Please note: the Ryos MK PRO is not able to change it's light colour or intensity by software.

2.1 Ryos Hardware

Inside the Ryos we used CherryMX switches with LED slots for single coloured LEDs.

The LEDs can be turned on and off by software, using this SDK.

Due to hardware and protocol limitations, the approximate latency for on/off events is currently about 20 to 30ms.

2.2 Ryos Protocol

To use any of the features from this SDK, the user needs to have the TalkFX software running in the background. TalkFX is the direct hardware communication interface used by most ROCCAT products.

Though in theory you could directly communicate with TalkFX directly from your application, we provide you with a much easier and less complicated way of doing so with this SDK.

2.3 Ryos LED Functionality

When using the SDK you should think of the Ryos as a simple (monochrome) bitmap display. Each key is addressed by an ID depending on it's position.

Starting from the top left corner the keys are numbered sequentially, so [Esc] is 0, [F1] is 2, [F2] is 3 and so on up to 110.

You can tell each key (more precisely: the LED inside the key's CherryMX switch) if it should be turned on or turned off. During SDK mode (more on that later), the keyboard behaves like a state-machine. This means it will do nothing on it's own and only change the light pattern when you tell it to.

This means a single command to light up - for example the [F10] key - can be send once during your program's runtime and [F10] will keep glowing until you send it the signal to turn itself off (or you deactivate the SDK mode).

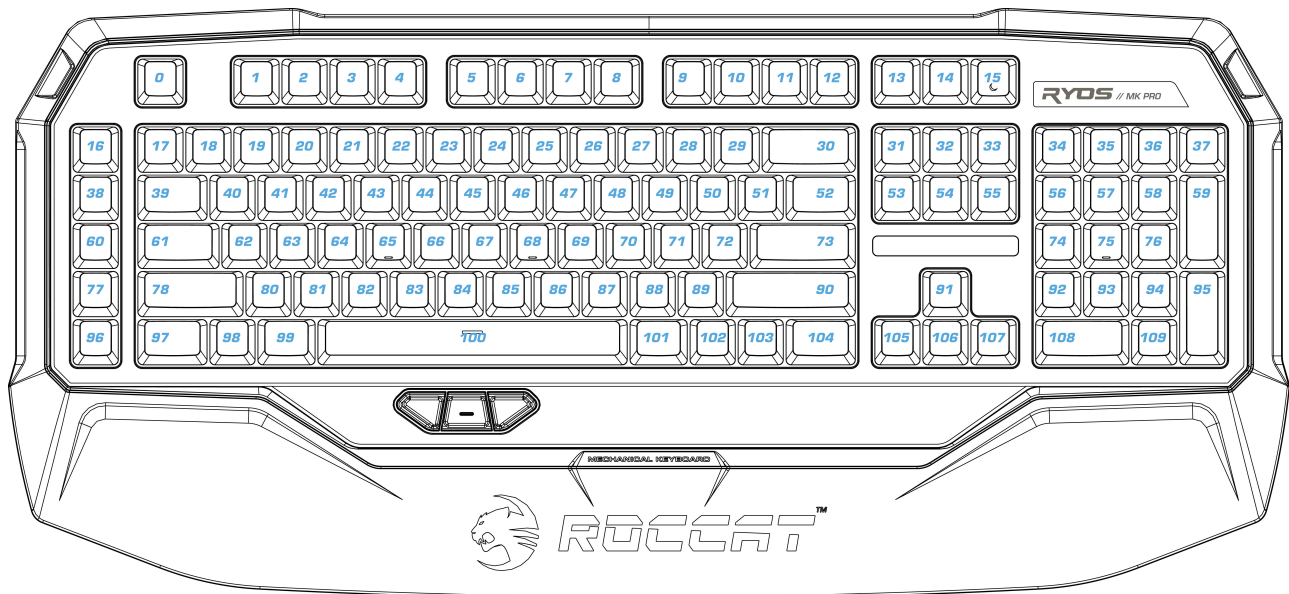


Figure 1. LED numbering on US-keylayout

If you prefer a frame-based model, you can also buffer the light data for all keys inside your program and send the whole data each time you want the lights updated. (We suggest you use the frame-based model, if you want to change the lights multiple times per second).

Note: Since keyboard layouts differ slightly across country zones, keep in mind that some keys might have different positions, or might not be present at all on certain layouts.

However the ID always corresponds to the physical switch on that particular physical location. So if you want to light up [Y], it will be 46 on an US keyboard, but on a german DE keyboard on that location will be [Z] and that key will light up.

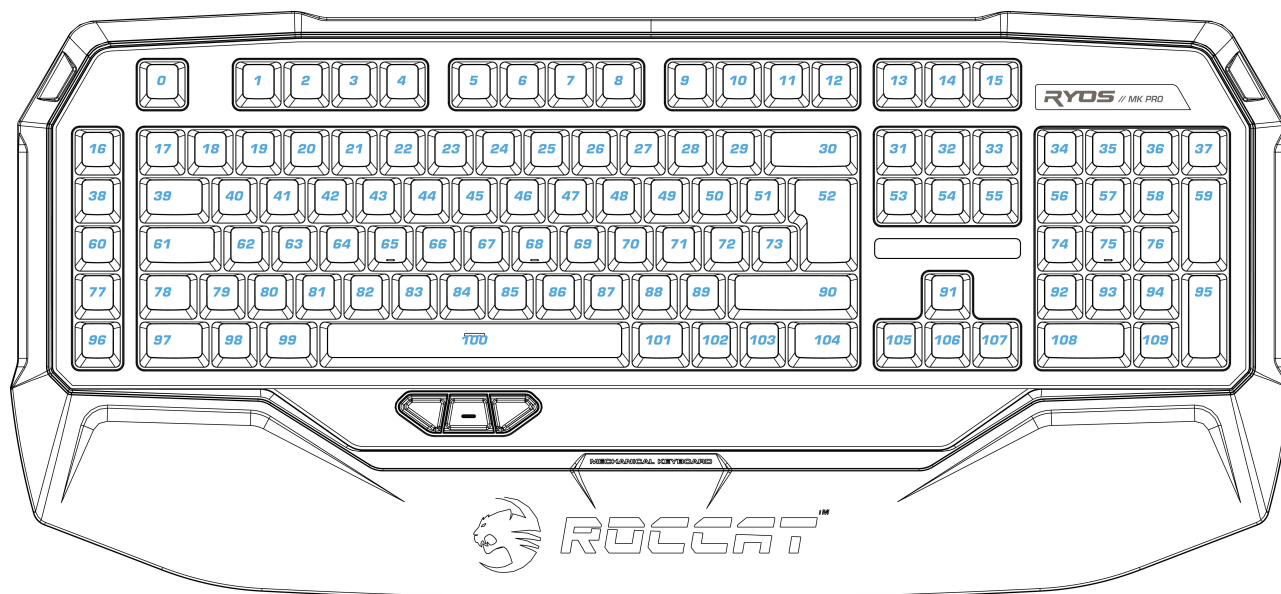


Figure 2. LED numbering on DE-keylayout

3.1 Basics

The TalkFX SDK comes with a set of three files:

Roccat_Talk.h The SDK header file. You always need to include this.

Roccat Talk SDK.dll The SDK DLL. If you prefer dynamic linking.

Roccat Talk SDK.lib The SDK Library file, for static linking.

There are more files in the SDK-Package, but only these three are required for implementation.

Simply include the header where you need it, and decide which library to you want to link against.

Keep in mind, that if you link static, any updates made to the SDK will have to be recompiled in your executable each time. If you link dynamic, you can most likely just replace the DLL without the need to recompile.

The SDK is designed to be used inside a C++ application, if your application is written in another programming language we suggest using the C/C++ native code binding capabilities of that particular language.

For example: JNI in Java, P/Invoke or DllImport in C#.NET, ctypes in Python or Win32API in Ruby. The SDK is not optimized for using it in non-C++ applications, and we currently do not support those officially. But if you manage to use the SDK in one of the languages above, or even another one, we would love to hear from you, so we can improve this documentation further.

If your programming language is not capable to include native DLLs due to language restrictions you can try using our CLI interface¹.

3.1.1 Initializing TalkFX Connection

By just initiating an instance of CROCCAT_Talk you have the required connection to the TalkFX software. No further connection required.

3.1.2 Colour Control

As already stated above: once you have created an instance of CROCCAT_Talk, you can start sending RGB codes to all Roccat devices connected to the current machine.

```
void Set_LED_RGB(BYTE bZone , BYTE bEffect , BYTE bSpeed , BYTE colorR , BYTE colorG , BYTE colorB );
```

bZone The effect zones are Ambient (0x00) and Event (0x01). Use Ambient when you have a low rate of updates. Use Event for fast paced updates. (When in doubt: use Event (0x01)).

bEffect Off (0x00), On (0x01), Blinking (0x02), Breathing (0x03), Heartbeat (0x04).

bSpeed no Change (0x00), Slow (0x01), Normal (0x02), Fast (0x03).

colorR simple RED value 0x00 to 0xFF.

colorG simple GREEN value 0x00 to 0xFF.

colorB simple BLUE value 0x00 to 0xFF.

Please remember to turn off the SDK mode when your application exits. To restore the user set colour to the TalkFX devices just use this method:

```
void RestoreLEDRGB ( ) ;
```

¹At the time this manual was written, the CLI interface was not available, so no further documentation on that topic is currently available.

3.1.3 Ryos specific TalkFX Connection

Before you can send any messages to the Ryos Keyboard, you first need to ensure that TalkFX is running and that a Ryos MK PRO is connected.

You can do so by simply doing this:

```
1 #include <stdio.h>
2
3 /* the following is necessary if windows.h is not already included */
4 #ifndef WIN32_LEAN_AND_MEAN
5 #define WIN32_LEAN_AND_MEAN
6 #endif
7 #include <Windows.h>
8
9 #include "ROCCAT_Talk.h"
10
11 int main(int argc, char** argv) {
12     CROCCAT_Talk roccat;
13     /* Try to connect to the Ryos */
14     if (!roccat.init_ryos_talk()) {
15         printf("Error: Could not connect to Ryos Keyboard!\n");
16     } else {
17         printf("Success: Connection established!\n");
18     }
19 }
```

3.1.4 Taking Control of the Ryos LEDs

Since the user might use the built-in LED control of the Ryos or a custom script for light effects, you need to take control of the LEDs before you can start switching them on and off.

You can do so by adding this to your code:

```
1 #include <stdio.h>
2 #include "ROCCAT_Talk.h"
3
4 /* the following is necessary if windows.h is not already included */
5 #ifndef WIN32_LEAN_AND_MEAN
6 #define WIN32_LEAN_AND_MEAN
7 #endif
8 #include <Windows.h>
9
10 int main(int argc, char** argv) {
11     CROCCAT_Talk roccat;
12     /* Try to connect to the Ryos */
13     if (!roccat.init_ryos_talk()) {
14         printf("Error: Could not connect to Ryos Keyboard!\n");
15     } else {
16         printf("Success: Connection established!\n");
17         /* Activate SDK mode */
18         roccat.set_ryos_kb.SDKmode(TRUE);
19
20         /* ... insert LED Control code here ... */
21
22         /* Give LED control back to system */
23         roccat.set_ryos_kb.SDKmode(FALSE);
24     }
25 }
```

IMPORTANT: Always remember to turn off the SDK mode when your application exits, otherwise the LEDs will keep their last state indefinitely.

3.1.5 Get the Ryos Lights Blinking

Once you have enabled the connection to TalkFX and the SDK mode, you can start to control the lights.

```
1 #include <stdio.h>
2 #include "ROCCAT_Talk.h"
3
4 /* the following is necessary if windows.h is not already included */
```

```

5 #ifndef WIN32_LEAN_AND_MEAN
6 #define WIN32_LEAN_AND_MEAN
7 #endif
8 #include <Windows.h>
9
10 int main(int argc, char** argv) {
11     CROCCAT_Talk roccat;
12     /* Try to connect to the Ryos */
13     if (!roccat.init_ryos_talk()) {
14         printf("Error: Could not connect to Ryos Keyboard!\n");
15     } else {
16         printf("Success: Connection established!\n");
17         /* Activate SDK mode */
18         roccat.set_ryos_kb_SDKmode(TRUE);
19
20         /* Ensure all LEDs off */
21         roccat.turn_off_all_LEDS();
22
23         /* Turn on a single LED */
24         roccat.set_LED_on(0); /* [Esc] */
25
26         /* Send a whole frame */
27         BYTE frame_data[110];
28         memset(frame_data, 0, 110); /* ensure 0 on all fields */
29         frame_data[1] = 1; /* [F1] */
30         frame_data[3] = 1; /* [F3] */
31         frame_data[5] = 1; /* [F5] */
32         frame_data[7] = 1; /* [F7] */
33         frame_data[11] = 1; /* [F11] */
34         roccat.Set_all_LEDS(frame_data); /* send frame to keyboard */
35
36         /* Give LED control back to system */
37         roccat.set_ryos_kb_SDKmode(FALSE);
38     }
39 }

```

The SDK provides you with these methods for LED control:

turn_on_all_LEDS() simply sets all LEDs to ON

turn_off_all_LEDS() simply sets all LEDs to OFF

set_LED_on(BYTE pos) sets a single LED at pos ON

set_LED_off(BYTE pos) sets a single LED at pos OFF

Set_all_LEDS(BYTE *pos) sets status for all 110 LEDs²

All_Key_Blinking(int dt, int count) Lets the whole keyboard blink at 'dt' interval (in ms) for 'count' times

3.2 Advanced Ryos

In the previous section you learned how easy it is to access the LED control of the ROCCAT Ryos MK PRO. In this section we give you additional advice on how to best utilize the SDK and how to easily integrate Ryos support in your existing applications.

3.2.1 Best Performance

Although we offer you some functions for convenience, like `set_LED_on()` and `All_Key_Blinking()`. You should only use those for debugging or for very simple use cases.

Use primarily `Set_all_LEDS(BYTE *pos)`, since with this method you can change the state of all LEDs at once with a single call. Every time you send an update to the Ryos a Window-Message is sent to TalkFX, this can affect performance when you send many of those updates (by setting single LEDs) in a single frame.

Also remember that the Ryos can only achieve slightly more than 30-40fps update rate. When you send updates faster than the Ryos can handle, those events will get buffered and you will experience visual lag on the keyboard.

It is vital to limit LED updates for optimal visual performance. So when using for example your `topmost.render_screen()` method, keep in mind that you cannot always safely assume that the framerate is limited by vertical retrace. (Even if you try to enforce vertical retrace, a user might have turned it off in the graphics driver).

²Attention: not all Keyboards have identical key layout, for example US and DE keymaps differ slightly. The SDK should map LED positions correctly, but if you set an LED to on that is not present on certain keyboard layouts, the user simply won't notice.

A wrapper class that keeps a static BYTE* framedata variable can help you to organize the updates. Let only that class send updates to the Ryos via the SDK. That way you can control when to send the updates, for example once every 2 frames (at 60Hz framerate). That way all updates to the framedata you perform from within other parts of your code are kept just inside your own application's memory until they are finally sent out by the wrapper.

Please note that we currently have no results for using the SDK in a multithreaded application. So do not assume the SDK to be thread safe and use all API calls from a single thread to avoid problems (for example your primary rendering thread).

3.3 Unobstrusive Integration

We at ROCCAT would love to hear that all your users have a Ryos MK PRO and all other Roccat TalkFX devices at home, but it is more likely that some users won't have a ROCCAT device at home.

Even ROCCAT device owners might not want your cool lightshow to be active all the time.

Use your wrapper-class to check if your light control should be on or off - a few if-statements should not pose an impact on performance.

So please add an option to your application settings to let the user turn the SDK mode on or off.

3.4 Example Integration

In this section we show you a small example on how we integrated the TalkFX SDK in a real game. We chose Cube by Wouter van Oortmerssen as our example, since it is open source and has nice, small code.

We are only describing the sourcecode changes we made, and we assume that you are familiar with whatever C++ compiler you are using.

3.4.1 Preparation

Make sure you can compile Cube from it's original codebase without any hassle.

Place ROCCAT_Talk.h into the include folder of the project.

Place Roccat Talk SDL.lib into the lib folder of the project.

Ensure the lib is properly recognized by your compiler and is linked to the final executable. (We used static linking in this example, you can go for dynamic if you want to.)

3.4.2 Placing Init Code

In cube.h add:

```
/* the following is necessary since windows.h is not already included */
#ifdef WIN32_LEAN_AND_MEAN
#define WIN32_LEAN_AND_MEAN
#endif
#include <Windows.h>

#include "ROCCAT_Talk.h"
```

And in main.c add:

```
...
int main(int argc, char **argv)
{
    // if (CROCCAT_Talk::s_ROCCATTALK == NULL) CROCCAT_Talk::s_ROCCATTALK = new CROCCAT_Talk();
    CROCCAT_Talk * m_ROCCATTALK = new CROCCAT_Talk();
    if(!m_ROCCATTALK->init_ryos_talk()) return -1;

    m_ROCCATTALK->set_ryos_kb_SDKmode(TRUE);
    m_ROCCATTALK->All_Key_Blinking(0,9);

    bool dedicated = false;
    int fs = SDL_FULLSCREEN, par = 0, uprate = 0, maxcl = 4;
    char *sdesc = "", *ip = "", *master = NULL, *passwd = "";
    ...
}
```

Note that you might need to add:

```
#pragma comment(lib, "Roccat Talk SDK.lib")
```


in the include section of main.c, depending on your compiler of choice.
And don't forget the cleanup code, so add:

```
...
void cleanup(char *msg)           // single program exit point;
{
    CROCCAT_Talk * m_ROCCATTALK = new CROCCAT_Talk();
    m_ROCCATTALK->set_ryos_kb_SDKmode(FALSE);
...

```

as well.

The SDK uses static variables for TalkFX communication, so you can either go for a single m_ROCCATTALK instance all the way through your code, or just initiate an instance whenever you need one.
This just just to give you an example on what is possible - the codebase of Cube is very compact compared to classic code styles, for production code we suggest to integrate the SDK in a more organized fashion to improve maintainability.

That's all you need for init and cleanup, next section features integration of the actual effects.

3.4.3 Show Ammo and Health

All you need to add is this code to renderextras.cpp:

```
...
CROCCAT_Talk *hud_talk = new CROCCAT_Talk();
void gl_drawhud(int w, int h, int curfps, int nquads, int curvert, bool underwater)
{
    // hud_talk->init_ryos_talk();
    // HERE BE HUD
    // player health
    int r_player_health = player1->health/10;
    static BYTE frame[110];
    memset(frame,0,110); // set all LEDS off
    for (int i=0; i<r_player_health; ++i) {
        frame[1+i]=1;
    }
    for (int i=17; i< (player1->ammo[player1->gunselect])+17 ; ++i) {
        frame[i]=1;
    }
    static int frameskipper = 0;
    if (frameskipper%3 == 0)hud_talk->Set_all_LEDS(frame);
    frameskipper++;
...

```

As you can see, we used a global CROCCAT_Talk instance here, you might want to do it a little more elegantly, but this will also work.

We assumed vertical retrace to be on, so at 60fps we only send updates to the keyboard every 3 frames. This is only a 48ms lag which is okay for a blinking keyboard.

3.4.4 Blink on Damage

To let the Ryos blink at each hit the player takes, you can simply insert this into weapon.cpp:

```
...
CROCCAT_Talk *damage_talk = new CROCCAT_Talk();
void hit(int target, int damage, dynent *d, dynent *at)
{
    if(d==player1) {
        selfdamage(damage, at==player1 ? -1 : -2, at);
        damage_talk->All_Key_Blinking(1,3);
    }
    else if(d->monsterstate) monsterpain(d, damage, at);
    else {
        addmsg(1, 4, SV_DAMAGE, target, damage, d->lifesequence); playsound(S_PAIN1+rnd(5), &d->o);
        damage_talk->All_Key_Blinking(1,3);
    };
    particle_splash(3, damage, 1000, d->o);
    demodamage(damage, d->o);
};
...

```

Note that we used the `All_Key_Blinking()` method here, which is considered harmful to application performance and timing, but it is an easy way to test out how effects are perceived by the player.

So that's it! All you need to insert for some nice FX with Ryos while playing Cube.

3.4.5 Reflection

Although the Cube sourcecode might differ a lot from your own personal application, we think it is safe to assume, that somewhere in your code (at least if your application is a computer game) equivalent methods to the ones discussed above are found.

Those are:

- application startup code (`int main()`)
- application shutdown code (`void cleanup()`)
- a method for HUD rendering (`void gl_drawhud()`)
- a method for damage calculation (`void hit()`)

The Ryos is not bound to the buffer swap of your rendering code, so you can basically insert LED code anywhere in your sourcecode. We highly recommend to make it a little more organized.

Also consider to add `#ifdef` statements around all Ryos SDK relevant code, so that you can check if any impact on application performance is generated by the SDK's methods you call.

If you think you are using the SDK correctly and still experience strong performance issues, do not hesitate to contact us.

3.5 Ryos CLI

The Ryos CLI is currently in development and should be released soon.



Figure 3. How gamer's will percieve the effects on the Ryos

When planning your desired effects for the Ryos, then try to think from the gamer's perspective. The keyboard will only be the in periphal field of vision, so if you show tiny effects, the gamer will most likely not notice them. Of course you are free to encourage the gamer to look at the keyboard during gameplay by design.

If you are not planning to add a true new gameplay element to your game with the Ryos, then stick with atmospheric effects, like flashing or a full-scale lifebar. Maybe even some kind of *where does the damage come from* effects.

Acknowledgments

Thanks go out to Wouter van Oortmerssen for writing the really nice Cubeengine, which we used here for our practical example. You can find Cube (along with sources) here: <http://cubeengine.com/cube.php4>

5. APPENDIX: Roccat Talk API header

As we are programmers ourselves, we know that sourcecode is often the best and quickest documentation and a good way to get a feeling of "how much work" implementing an API is.

So we included the only header file you need to #include right here:

```
1  /* ROCCAT_Talk.h */
2  #pragma once
3
4  class CROCCAT_Talk
5  {
6  public:
7      CROCCAT_Talk(void); /* default constructor */
8      ~CROCCAT_Talk(void); /* default destructor */
9
10     /* initiate connection to Ryos MK PRO keyboard and check if present */
11     BOOL    init_ryos_talk(void);
12
13     /* take control of a connected Ryos MK PRO keyboard */
14     BOOL    set_ryos_kb_SDKmode(BOOL state);
15
16     /* basic Ryos MK PRO LED control methods */
17     void    turn_off_all_LEDS(void);
18     void    turn_on_all_LEDS(void);
19
20     /* turn on/off a single LED at specified position */
21     void    set_LED_on(BYTE ucPos);
22     void    set_LED_off(BYTE ucPos);
23
24     /* send a whole array as a frame to the keyboard (manipulate multiple LEDs)*/
25     void    Set_all_LEDS(BYTE *ucLED);
26
27     /* simple blinking effect on Ryos MK PRO */
28     void    All_Key_Blinking(int DelayTime, int LoopTimes);
29
30     /* TALK FX method — set specified zone to effect and RGB colour */
31     void    Set_LED_RGB(BYTE bZone, BYTE bEffect, BYTE bSpeed, BYTE colorR, BYTE colorG, BYTE colorB);
32
33     /* TALK FX method — restore user LED colour at end of program */
34     void    RestoreLEDRGB();
35
36 protected:
37     HWND    m_hwnd;
38     UINT    m_uiMsgIDDiscover;
39
40     BYTE    GetKeyNo(BYTE cMapKey);
41
42     static HWND m_hTalkWnd;
43     static UINT m_uiMsgIDAttach, m_uiMsgIDControl;
44     static UINT m_uiMsgIDAttachForFX, m_uiMsgIDControlForFX;
45     static UINT m_uiMsgIDDiscoverForFX;
46     static BYTE bLedOnOff[15];
47     static LRESULT CALLBACK SDKWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
48 };
```