# Iris 2D API Reference Manual for Amethyst

## Introducition

Iris 2D is an open source game engine which is aimed to achieve high-efficency development and quick running based on DirectX 9.0. Its realization inside has related to the structure of RGSS(Ruby Game Scripting System), so game based on Iris 2D could be highly object-oriented and its code could be very simple and directly. And more, because of the mechanism of maintenance inside, it becomes unnecessary for users to think much about problems such as resource releasing — resource useless would be released at proper time for the auto-maintaining resource chain, which reduces the risk of resource overflow. User can easily do 2D-Programming just using the interface provided to simplify the complexity of vertex-operation in the tranditional 2D-Programming.

Iris 2D based on DirectX has a stable drawing speed, and the FPS could be very stable on many common machines under much complex condition. Without the out-of-date DirectDraw its performance can be ensured by hardware accelerationm, so Iris 2D is a high-efficency 2D drawing solution.

Iris 2D has 3 editions currently: origin(SC approach), library(API approach), ruby. Program’ source code of origin edition contains a window; library edition offers APIs in DLL outside and you should call them in your own program; and ruby edition is the Iris 2D embled with Ruby interpreter, this edition is under development.

## Use of Library Edition of Iris 2D

Firstly, we should download the library edition of Iris 2D from the project of Iris 2D’s hosting homepage, and we at least need these 4 files: fmodex.dll, Iris 2D Amethyst-Potato(API).dll, Iris 2D Amethyst-Potato(API).lib, Iris 2D Library.h(usually in the dll folder).

Before we use Iris 2D you need to create a new C++ project, and copy these 4 files to the main directory of your project to ensure the compiler environment can find them. If necessary, Iris 2D Library.lib should be pre-compiled(for Visual Studio users):

#include “Iris 2D Amethyst-Potato(API).h”

#pragma comment(lib, “Iris 2D Amethyst-Potato(API).lib”)

As the development of Iris 2D is based on window program, the structure of your program may look like as follow:

//…

IIrisApp\* app;

bool Display(){

//…

while(app->IsRunning()){

IrisGraphicsUpdate();

IrisInputUpdate();

// game logic、game render

}

return true;

}

// Windows program entry

int WINAPI WinMain(HINSTANCE hInstance,

HINSTANCE prevInstance,

PSTR cmdLine,

int showCmd){

app = GetIrisApp();

app->Init(hInstance, 800, 600, Display, L"My Iris App");

app->Run();

app->Release();}

}

Firstly, we get a pointer of object of IrisApp through function of GetIrisApp(), a unique singleton object that exists during the running of Iris 2D program and is used to control and visit any part of Iris 2D program. Any operation of calling of GetIrisApp() can just return the same pointer of a IrisApp object.

Then through the pointer of IrisApp gained, we call the Init function of it to initialize data of game’s window - particular parameters’ explanation are recorded in the API Documentation – and return false if oparation of initializing failed.

After initializing, we call Run() function enter the main loop of game.

The main loop, is the calling of the Display function transferred to Init() function as a parameter before whose returned value is type of bool and parameters are none. In word, the main loop of Iris 2D game is this Display function where all the game render, game logic proccess are handled. Further more, IrisApp can call the Display function only 1 time, Iris 2D will complete running and exit if we don’t set the “real loop” in Display function, so we are supposed to add a loop manually as follow:

while(app->IsRunning()){

IrisGraphicsUpdate();

IrisInputUpdate();

// game logic、game render

}

Particular explanation of IrisGraphicsUpdate function and IrisInputUpdate function are recorded in the API Documentation.

At last, we exit the main loop of game and call the Release function of app to release all the related resource.

Thus a complete Iris 2D program have been completed, then what we should do is the coding of game itself.

## API Documentation

APIs of Iris 2D in the mass can be divided into 2 classes – Object API part and Module API part. The Object API includes all the elements of 2D game development including sprite(IrisSprite), bitmap(IrisBitmap), viewport(IrisViewport) and so on. Essentially they are all C++ Classes. The Module API is a collection of functions which only need to offer operation related to Iris 2D itself, render, input/output without object-instantiation. Essentially they are all C++ Static Class Functions.

Notice: in class definition, all the parameters labeled as property can be accessed through accessor with SetXXX()/GetXXX().

The Object API includes classes as follow:

|  |  |
| --- | --- |
| Class | Function |
| IrisApp | Control all the key data and work of Iris 2D program including D3D Device, D3Dinput Device, operation of Initializing of Iris 2D program and so on. |
| IrisSprite | The sprite class of Iris 2D, is the core of Iris 2D, also the coer of 2D game. All the elements displayed on the screen is sprites. Every sprite in Iris 2D are maintained in Viewport. |
| IrisBitmap | The bitmap class of Iris 2D, achieved through GDI+ inside, while ahieving the render of IrisSprite combined with DirectX’s texture, can operate graphs in game in pixel-level. |
| IrisViewport | The viewport class of Iris 2D. What is called viewport is to render sprite in an area of window but do no influnce on other sprites. There can be a plurality of viewports in Iris 2D, and sprite will be set in the global viewport if it is not be set in an explicit viewport defaultly. And all the viewports in Iris 2D are maintained through a list. |
| IrisColor | The color class of Iris 2D which supports 32 bit color of ARGB format, is the structure data of Iris 2D’s color. |
| IrisFont | The font class of Iris 2D, is the structure data of Iris 2D’s font, used to draw string in IrisBitmap. |
| IrisPlane | The plane class of Iris 2D (under development). |
| IrisRect | The rect class of Iris 2D, is the data structure describing any area in window in Iris 2D. |
| IrisTilemap | The tilemap class of Iris 2D (under development). |
| IrisTone | The tone class of Iris 2D uesd in the tone balance of appearance. |
| IrisWindow | The basic class of Iris 2D used to draw window in game (under development). |

The Module API includes classes as follow:

|  |  |
| --- | --- |
| Module | Function |
| ModuleIrisGraphics | Offer operation as control the update of render, get window’s handle and so on in Iris 2D program. |
| ModuleIrisInput | Offer operation as contrl the update of input buffer, get the judgement information of former input and so on in Iris 2D program. |
| ModuleIrisAudio | Offer operation as control the audio playing in Iris 2D program. |

### Object APIs

#### IIrisApp

接口函数：

IIrisApp\* GetIrisApp()

功能：获取当前IrisApp的对象指针，注意这是一个单例对象，

返回值：当前IrisApp的对象指针。

void ReleaseIrisApp(IIrisApp\* app)

功能：释放IrisApp对象。

参数：

app：IrisApp指针。

类函数：

bool Init(HINSTANCE hInstance, int width, int height, bool(\*pf)(), wstring title)

功能：初始化Iris 2D程序。

参数：

hInstance：当前程序的实例句柄。

width/height：Iris 2D窗口的宽度、高度。

bool(\*pf)()：游戏主循环所在函数的函数指针。

title:窗口标题。

返回值：如果初始化成功为true否则为false。

void Run()

功能：初始化完毕，进入游戏循环。

void Release()

功能：程序执行完毕，释放Iris 2D中的所有资源。

bool IsRunning()

功能：返回当前Iris 2D的状态，如果正在运行返回true否则返回false。

bool IsQuited()

功能：返回当前Iris 2D的状态，如果已经退出返回true否则返回false。

float GetTimeDelta()

功能：返回两帧之间的真实时间。

返回值：返回两帧之间的真实时间，以毫秒为单位。

#### IIrisBitmap

接口函数：

IIrisBitmap\* GetIrisBitmap(wstring fileName)

功能：生成一个IrisBitmap对象。

参数：

fileName：磁盘上图片文件的名称，注意带上扩展名。

IIrisBitmap\* GetIrisBitmap(int width, int height)

功能：生成一个指定长宽的空白位图。

参数：

width/height：位图的长宽。

void ReleaseIrisBitmap(IIrisBitmap\* bmp)

功能：释放一个IrisBitmap对象 包括其所有资源。

参数：

bmp：IrisBitmap对象指针。

类函数：

void Dispose()

功能：释放Bitmap资源。建议不使用这个函数，这是为嵌入脚本语言提供的接口，如果要释放一个IrisBitmap对象，直接在C++中使用delete语句即可。

bool Disposed()

功能：如果IrisBitmap已经释放返回true否则返回false，用于对Dispose()函数进行确认的函数，不建议在C++中使用。

返回值：如果IrisBitmap已经释放返回true否则返回false。

bool Blt(int x, int y, IIrisBitmap \*srcIrisBitmap, IIrisRect \*rect, int opactiy)

功能：将src指定的IrisBitmap中的指定数据复制到当前IrisBitmap对象中。

参数：

x/y：目标Bitmap内存放位图的坐标。

srcIrisBitmap：源IrisBitmap。

rect：源IrisBitmap中的一个矩形区域，这个区域决定了复制到目标IrisBitmap的数据。

opacity：复制到目标IrisBitmap后的透明度，范围为0~255。

返回值：复制成功返回true否则反复false。

bool StretchBlt(IIrisRect \*destIrisRect, IIrisBitmap \*srcIrisBitmap, IIrisRect \*srcIrisRect, int opacity)

功能：允许拉伸缩放地从源IrisBitmap将指定数据复制到当前IrisBitmap对象中。

参数：

destIrisRect：目标IrisBitmap中用于接受拉伸缩放数据的区域的矩形。

srcIrisBitmap：源IrisBitmap。

srcIrisRect：源IrisBitmap中用于指定所需要操作的数据区域的矩形。

opacity：复制到目标IrisBitmap后的透明度，范围为0~255。

返回值：复制成功返回true否则返回false。

void FillRect(int x, int y, int width, int height, IIrisColor \*color)

void FillRect(IIrisRect \*rect, IIrisColor \*color)

功能：用指定颜色填充位图中指定的矩形区域。

参数：

x/y/width/height：矩形区域的左上角顶点坐标以及宽度和高度。

rect：矩形区域。

color：填充颜色。

void GradientFillRect(int x, int y, int width, int height, IIrisColor \*color1, IIrisColor \*color2, bool vertical = false)

void GradientFillRect(IIrisRect \*rect, IIrisColor \*color1, IIrisColor \*color2, bool vertical = false)

功能：用渐变填充位图中指定的矩形区域。（未实现）

参数：

x/y/width/height：矩形区域的左上角顶点坐标以及宽度和高度。

rect：矩形区域。

color1：渐变的起始颜色。

color2：渐变的终止颜色。

virtical：是否为垂直渐变，如果为false则默认水平渐变。

void Clear()

功能：用于清空IrisBitmap数据。

void ClearRect(int x, int y, int width, int height)

void ClearRect(IIrisRect\* rect)

功能：清空位图中指定区域内的数据。

参数：

x/y/width/height：矩形区域的左上角顶点坐标以及宽度和高度。

rect：矩形区域。

IIrisColor\* GetPixel(int x, int y)

功能：返回位图中指定位置的颜色值。

参数：

x/y：指定位置的坐标值。

返回值：IIrisColor指针，注意手动释放。

void SetPixel(int x, int y, IIrisColor \*color)

功能：设置位图中指定位置的颜色值。

参数：

x/y：指定位置的坐标值。

color：指定的颜色值。

void hueChange(int hue)

功能：改变位图的色调。

参数：

hue：色调值。

void blur()

功能：对位图进行模糊处理，注意该操作会比较耗时。（未实现）

void RadialBlur(float angle, int division)

功能：对位图进行放射性模糊，注意该操作会比较耗时。（未实现）

参数：

agnle：0~360的角度，角度越大，效果越圆润。

division：2~100的分界数，分解数越大，效果越平滑。

int TextSize(IIrisFont\* fontIris, wstring str)

功能：返回指定字符串在指定的IrisFont下绘制指定字符串时需要的区域（宽度）。

参数：

fontIris：指定的IrisFont指针，用于说明字体类型。

str：需要测量绘制区域的字符串。

void IrisDrawText(IIrisFont\* fontIris, int x, int y, int width, int height, wstring str, int align)

void IrisDrawText(IIrisFont\* fontIris, IIrisRect \*rect, wstring str, int align)

功能：在指定区域用指定的字体绘制字符串并设置其对齐格式。如果所需要绘制的字符串超出了指定区域将会自动裁剪。

参数：

fontIris：绘制用的字体，如果为NULL则使用默认字体。

x/y/width/height：指定区域的矩形的左上角坐标以及宽度高度。

rect：指定区域的矩形。

align：对齐方式，预设为居左对其，1为居中2为居右。此操作比较耗时，不建议每帧都进行字符串的绘制。

属性：

int Width：当前位图的宽度。

int Height：当前位图的高度。

IIrisFont\* Font：

#### IIrisColor

接口函数：

IIrisColor\* GetIrisColor(int red, int green, int blue, int alpha)

功能：生成一个IrisColor对象。

参数：

red/green/blue/alpha：颜色的RGBA值。

返回值：IrisColor类对象指针。

void ReleaseIrisColor(IIrisColor\* color)

功能：释放一个IrisColor对象。

参数：

color：IrisColor对象指针。

类函数：

void set(int red, int green, int blue)

void set(int red, int green, int blue, int alpha)

功能：设置IrisColor的颜色值，每个分量限制在0~255。如果没有设置alpha自动为255。

属性：

int Red/Green/Blue/Alpha：颜色的rgba值。

#### IIrisFont

接口函数：

IIrisFont\* GetIrisFont(wstring name, int size)

功能：同上。

参数：

name：同上。

size：字体大小。

返回值：IrisFont对象指针。

IIrisFont\* GetIrisFont(wstring name)

功能：获取一个IrisFont对象指针。此时字体大小为默认大小。

参数：

name：字体在系统中的名字。

返回值：IrisFont对象指针。

void ReleaseIrisFont(IIrisFont\* font)

功能：释放IrisFont类对象。

参数：IrisFont类对象指针。

属性：

wstring Name：字体在系统中的名字。

int Size：字体大小，以像素为单位。

bool Bold：设置是否为粗体。

bool Italic：设置是否为斜体。

bool Shadow：设置是否有阴影。

IIrisColor\* Color：字体颜色。

#### 5、IIrisPlane（未实现）

#### 6、IIrisRect

IIrisRect\* GetIrisRect(float x, float y, float width, float height)

功能：获取一个IrisRect对象指针。

参数：

x/y/width/height：矩形的左上角坐标值以及长宽。

返回值：IrisRect对象指针。

void ReleaseIrisRect(IIrisSprite \*rt)

功能：释放IrisRect对象。

参数：

rt：IrisRect对象指针。

void Set(float x, float y, float width, float height)

功能：设置矩形的左上角坐标，宽度和高度。

参数：x/y/width/height：矩形左上角坐标值，宽度和高度。

属性：

float X/Y/Width/Height：矩形的左上角坐标值，宽度和高度，注意宽度和高度虽然为float，但是如果用于位图的话那么会强制转换为int类型。

#### 7、IIrisSprite

接口函数：

IIrisSprite\* GetIrisSprite(IIrisViewport \*viewport)

功能：生成一个IrisSprite对象。

参数：

viewport：指定IrisSprite所在的视口的指针。

返回值：IrisSprite对象指针。

IIrisSprite\* GetIrisSprite()

功能：同上。此时视口自动设定为全局视口。

参数：

返回值：IrisSprite对象指针。

void ReleaseIrisSprite(IIrisSprite\* sp)

功能：释放IrisSprie对象。

参数：

sp：IrisSprite对象指针。

类函数：

void SetViewport(IIrisViewport \*tviewport)

功能：设定当前精灵的视口。

参数：

tviewport：目标视口。

IIrisViewport\* GetViewport()

功能：获取当前精灵的视口。

返回值：当前精灵的视口对象的指针。

void Dispose()

功能：释放当前精灵，注意不建议在C++中使用该函数，建议直接使用delete释放精灵，同时在释放精灵之前应该先释放精灵的位图和视口（如果需要），

bool Disposed()

功能：返回当前精灵是否释放。

返回值：true表示已经释放false表示没有释放。

void Flash(IIrisColor \*color, int duration)

功能：设定精灵的闪烁，不建议使用该函数，用处不大。

参数：

color：精灵闪烁时的最亮时候的颜色，如果设置为NULL则在闪烁的时候精灵消失。

duration：精灵闪烁的时间，以帧数为单位。

void Update()

功能：执行精灵的刷新，在没有特效时没必要调用这个函数。

int GetWidth()

功能：获取精灵的宽度。

返回值：精灵的位图区域的宽度。

int GetHeight()

功能：获取精灵的高度。

返回值：精灵的位图区域的高度。

void SetOpacity(int tOpacity)

功能：设置精灵的透明度。

参数：

tOpacity：0~255的值，值越大不透明度越大。

int GetOpacity()

功能：返回精灵的透明度。

返回值：精灵的透明度。

属性：

IIrisBitmap\* Bitmap：精灵的位图数据。

IIrisRect\* srcRect：限制精灵绘制的位图区域，超出部分将会自动修正。

bool Visible：精灵的可见性。

float X,Y：精灵的坐标值。

float Z：精灵在自身所在的视口内的高度，Z的值越大越靠前显示；如果Z相同则后生成的精灵靠前。

float Ox/Oy：精灵起始点的坐标值。

float ZoomX/ ZoomY：精灵的横向/纵向的缩放拉伸的值。

float Angle：精灵的旋转角度，以逆时针方向指定，单位为角度制，0~360。

float WaveAmp：定义波浪效果的幅度，单位为像素。波动效果是使用正弦曲线函数来绘制精灵，使精灵的横向位置比上次绘制的位置稍有不同。（未实现）

float WaveLength：定义波浪效果的频度，单位为像素。（未实现）

float WaveSpeed：定义波浪动画的速度，默认为360，值越大速度越快。（未实现）

float WavePhase：定义使用角度（最大360度）指定精灵顶行的周相。每次调用update方法会更新周相一次。一般情况下不需要使用此属性，除非需要同步化两个精灵的波浪效果。（未实现）

int BushDepth：精灵的草木繁茂处的深度。默认为0，值越高表示遮挡越高，可以实现如角色的腿被草丛半遮住等效果。（未实现）

int BushOpacity：精灵的草木繁茂处的不透明度。BushOpacity的值是建立在当前精灵的Opacity值之上的。（未实现）

int BlendType：精灵的合成方式。0：正常；1：加法；2：减法。（未实现）

IIrisColor\* Color：与精灵合成的颜色（Color 色彩类）色彩的 alpha 值作为合成的比例。此颜色与Flash效果的颜色分开处理。然而，alpha值较高的颜色在合成时，优先度较高。（未实现）

IIrisTone\* Tone：精灵的色调。

#### 8、IIrisTilemap（未实现）

#### 9、IIrisTone

void set(float red, float green, float blue)

void set(float red, float green, float blue, float alpha)

功能：设置色调的RGBA值。

参数：red/green/blue/alpha：色调的RGBA值。

属性：

Red/Green/Blue/Alpha：色调的RGBA值。

#### 10、IIrisViewport

IIrisViewport\* GetIrisViewport(float x, float y, float width, float height)

IIrisViewport\* GetIrisViewport(IIrisRect \*rect)

功能：生成一个IrisViewport对象。

参数：

x/y/width/height：视口的左上角坐标值与长宽。

rect：视口的矩形。

void ReleaseIrisViewport(IIrisViewport \*vp)

功能：释放IrisViewport对象。

参数：

vp：IrisViewport对象指针。

void Dispose()

功能：释放当前的视口，并将其从Iris 2D后台维护链中移除，但是不释放它的C++对象。该函数是提供给Ruby包装的，不建议在C++中使用该函数，用delete即可。

bool Disposed()

功能：判断当前视口是否释放。

返回值：已经释放返回true否则返回false。

void Flash(IIrisColor \*color, int duration)（未实现）

功能：开始视口的闪烁，类似于精灵的Flash函数。

参数：

color：视口闪烁时的最亮时候的颜色，如果设置为NULL则在闪烁的时候视口消失。

duration：闪烁的帧数。

void Update()

功能：刷新当前的视口包括其中所有的精灵，原则上没有必要调用这个函数。

float GetX()/GetY()

功能：获取当前视口的左上角坐标值。

返回值：当前视口左上角坐标值。

UINT GetWidth()/GetHeight()

功能：获取当前视口的宽度和高度。

返回值：当前视口的宽度和高度。

属性：

IIrisRect\* Rect：视口范围的矩形。

bool Visible：视口是否可见。

float Z：当前视口的高度。

float Ox,Oy：视口传送元原点的 X/Y 座标。根据该值变化进行滚动。

IIrisTone\* Tone：视口的色调。

IIrisColor\* Color：视口的颜色。（未实现）

#### 11、IIrisWindow（未实现）

### 2）Module APIs

#### 1、IrisFont

bool IrisFontExisted(wstring name)

功能：在系统中查找是否存在指定名字的字体。

参数：

name：字体名称。

返回值：如果存在返回true否则返回false。

wstring IrisFontGetDefaultName()

功能：获取默认的字体的名称。

返回值：默认的字体的名称。

int IrisFontGetDefaultSize()

功能：获取默认的字体的大小，单位为像素。

返回值：默认的字体的大小。

bool IrisFontGetDefaultBold()

功能：获取默认字体是否为粗体。

返回值：默认字体是否为粗体。

bool IrisFontGetDefaultItalic()

功能：获取默认字体是否为斜体。

返回值：默认字体是否为斜体。

bool IrisFontGetDefaultShadow()

功能：获取默认字体是否有阴影。

返回值：默认字体是否有阴影。

IIrisColor\* IrisFontGetDefaultColor()

功能：获取默认字体颜色。

返回值：默认字体颜色。

#### 2、IrisGraphics

HWND IrisGraphicsGetHwnd()

功能：获取当前Iris2D窗口句柄。

返回值：当前Iris2D窗口句柄。

int IrisGraphicsGetWidth()/IrisGraphicsGetHeight()

功能：获取当前游戏窗口宽度/高度。

返回值：当前游戏窗口宽度/高度。

void IrisGraphicsUpdate()

功能：刷新画面，原则上一帧调用一次，如果不调用则游戏画面不会跟进，建议在主循环中调用。

void IrisGraphicsWait(int duration)

功能：使画面等待指定帧数，在此期间什么都不做。等价于：

for(int i = 0; i < duration; i++) IrisGraphicsUpdate();

参数：

duration：等待的帧数。

void IrisGraphicsFadeOut(int duration)

功能：将画面淡出。

参数：

duration：淡出所需的帧数。

void IrisGraphicsFadeIn(int duration)

功能：将画面淡入。

参数：

duration：淡入所需的帧数。

void IrisGraphicsFreeze()

功能：将当前画面固定，准备渐变的执行。在调用 transition 方法之前，禁止一切画面重绘。

void IrisGraphicsTransition(int duration, wstring filename, int vague)

功能：执行由IrisGraphicsFreeze()所固定的画面到当前画面的渐变。

参数：

duration：渐变的画格数。省略时预设为8。

filename：指定的渐变图形文件名，未指定时则执行标准淡出。

vague: 设置图像的起始点和终结点之间边界的模糊度，数值愈大则愈模糊。省略时预设为 40。

void IrisGraphicsSnap2Bitmap(IIrisBitmap\* bitmap)

功能：将当前游戏画面截图存为位图对象。

参数：

bitmap：指定接受当前游戏画面的位图对象指针。

void IrisGraphicsFrameReset()

功能：将画面更新时间归零。在执行十分消耗时间的进程之后，调用此方法可以避免严重的画格跳跃现象。

void IrisGraphicsResizeScreen(int width, int height)

功能：修改画面的尺寸。（未实现）

参数：

width/height：画面的长宽。

int IrisGraphicsGetFrameRate()

功能：每秒钟刷新画面的次数(FPS)。

返回值：每秒钟刷新画面的次数。

int IrisGraphicsSetFrameRate(int fr)

功能：设置游戏的FPS。（未实现）

参数：

fr：游戏的FPS

unsigned long IrisGraphicsGetFrameCount()

功能：返回游戏进行的帧数。

返回值：画面刷新次数的计数。游戏开始时将此属性设为 0，游戏的时间（秒）可以由 此数值 / 刷新率 计算出来。

void IrisGraphicsSetFrameCount(int ct)

功能：设置游戏进行的帧数。

参数：

ct：游戏进行的帧数。

int IrisGraphicsGetBrightness()

功能：获取画面的亮度。

返回值：画面的亮度。

void IrisGraphicsSetBrightness(int bn)

功能：设置画面的亮度。

参数：

bn：画面的亮度，取 0～255 范围间的数值。在淡出、淡入和渐变方法中需要在内部修改此

数值。

#### 3、IrisInput

void IrisInputUpdate()

功能：刷新输入设备的缓冲区，并可以进行扫描检查。原则上该函数与IrisGraphicsUpdate()一起调用，如果不调用则无法进行输入检查。

bool IrisInputKeyBoardKeyPress(DWORD key)

功能：判断键盘上指定键位的按键当前时刻是否按下。

参数：

key：键值。

返回值：如果正在按下返回true否则返回false。

bool IrisInputKeyBoardKeyTrigger(DWORD key)

功能：判断键盘上指定键位的按键是否为触击，即按下即松开按键。

参数：

key：键值。

返回值：如果是返回true否则返回false。

bool IrisInputKeyBoardKeyRepeat(DWORD key)

功能：判断键盘上指定键位的按键是否双击。

参数：

key：键值。

返回值：如果是返回true否则返回false。

bool IrisInputMousePress(MouseKey botton)

功能：判断鼠标指定键位的按键当前时刻是否按下。

参数：

botton：键值。

返回值：如果正在按下返回true否则返回false。

bool IrisInputMouseTrigger(MouseKey botton)

功能：判断鼠标指定键位的按键是否为触击，即按下即松开按键。

参数：

botton：键值。

返回值：如果是返回true否则返回false。

bool IrisInputMouseRepeat(MouseKey botton)

功能：判断鼠标指定键位的按键是否双击。

参数：

botton：键值。

返回值：如果是返回true否则返回false。

long IrisInputMouseMiddleRoll()

功能：返回鼠标滚轮在两次调用IrisInputUpdate()的间隔时间中滚过的数值。

返回值：如果顺时针滚返回正值否则返回负值。

long IrisInputMouseXMove()/IrisInputMouseYMove()

功能：返回鼠标在两次调用IrisInputUpdate()的间隔时间中在横向纵向的移动大小。

返回值：横向纵向的移动大小，单位为像素。

int IrisInputGetCursorX()/IrisInputGetCursorY()

功能：返回当前鼠标在窗口中的位置。

返回值：当前鼠标在窗口中的位置。

bool IrisInputCursorInRect(IIrisRect\* rect)

参数：

rect：判断矩形。

功能：判断当前鼠标是否在一个矩形中。

返回值：如果是返回true否则返回false。

#### 4、IrisAudio

IRIS2DLIBRARY\_API bool IrisAudioBgmPlay(wstring filePath, int volume, int rate)

IRIS2DLIBRARY\_API bool IrisAudioBgsPlay(wstring filePath, int volume, int rate)

IRIS2DLIBRARY\_API bool IrisAudioSePlay(wstring filePath, int volume, int rate)

IRIS2DLIBRARY\_API bool IrisAudioMePlay(wstring filePath, int volume, int rate)

参数：

filePath：音频文件路径名。

volume：播放时的音量大小，100为音频文件的本身音量。

rate：播放时的频率，100为本身频率。

功能：播放一个BGM/BGS/SE/ME，注意BGM/BGS采用流方式播放而SE/ME采用Sound方式播放。

返回值：如果播放成功返回true否则返回false。

IRIS2DLIBRARY\_API bool IrisAudioBgmStop()

IRIS2DLIBRARY\_API bool IrisAudioBgsStop()

IRIS2DLIBRARY\_API bool IrisAudioSeStop()

IRIS2DLIBRARY\_API bool IrisAudioMeStop();

功能：停止当前播放中的BGM/BGS/SE/ME。

返回值：始终返回true。

IRIS2DLIBRARY\_API bool IrisAudioBgmFade(int fadeTime)

IRIS2DLIBRARY\_API bool IrisAudioBgsFade(int fadeTime)

参数：

功能：淡出当前的BGM/BGS，在指定时间范围内。该功能采用线程完成。

fadeTime：淡出的时间。

返回值：始终返回true。