موتور 3 بعدی Irrlicht (نسخه 1.0)

(آشنایی و ساخت یک دموی تکنیکی)

بهترین اندازه دید : 1024 x 768

_____

نویسنده : علی کسایی

WWW.Persian-Designers.COM

Ali_nwdo@yahoo.com

این مقاله برای آشنایی شما دوستان عزیز با موتور 3 بعدی Irrlicht نوشته شده است و کوشیده شده تا در هنگام نوشتن این مقاله ، تمامی نکات مبهم در مورد کار با این موتور و نحوه استفاده از آن در محیط های مختلف ، توضیح داده شود. در عین حال با مطالعه این مقاله علاوه بر آشنایی با موتور Irrlicht ، تصوری هر چند جزیی در مورد اصول کار با API های 3 بعدی و نحوه برنامه نویسی در این گونه محیط ها پیدا خواهید کرد... تنها شرایط مورد نیاز برای استفاده از مطالب این مقاله ، آشنایی با زبان برنامه نویسی C++ ، داشتن موتور سورس باز Irrlicht بر روی هارد و فکر آزاد و خیال راحت است.

می دانم که مشتاقید هر چه زودتر به قسمتهای اصلی مقاله برسید... اما عجله نکنید. به خاطر داشته باشید که طراحی بازی های کامپیوتری از پر دردسر ترین اعمالی است که میشود با کامپیوتر انجام داد. بنابراین ابتدا فکرتان را آزاد کنید. یک فنجان چای آماده کنید و در حالی که به یک موسیقی ملایم گوش میکنید ، مطالعه مقاله را ادامه دهید ☺

Irrlicht چیست ؟

Irrlicht یک موتور گرافیکی است ، نه یک موتور بازی... به این معنی که هیچ تابعی برای کنترل شبکه یا صوت در آن پیدا نمی کنید... البته چند تابع تشخیص برخورد در آن تعبیه شده است... این موتور در واقع یک کتابخانه 3 بعدی برای زبان برنامه نویسی C++ است. بنابراین برای استفاده بهینه از این موتور و خلق جلوه های ویژه تصویری لازم است تا بر این زبان تسلط داشته باشید... اما یکی از مزیت های موتور Irrlicht که در آینده با آن بیشتر آشنا میشوید این است که با هر سطح مهارت در برنامه نویسی ، میتوانید با آن بازی بسازید. فقط کافیست تا اصول را بدانید...

چگونه از Irrlicht استفاده کنیم ؟

قبل از هر چیز به یک IDE یا Integrated Development Environment مثل Microsoft Visual C++ یا Dev C++ نیاز دارید. بعد از اینکار میبایست تمامی فایلهای کتابخانه ای و Header ها را به کامپایلری که از آن استفاده میکنید ، اضافه کرده و سپس شورع به نوشتن برنامه کنید.

آیا میشود از این موتور در محیط .NET. استفاده کرد ؟

بله و خیر... در واقع ورژن های جدید این موتور قابلیت پشتیبانی از تکنولوژی .NET. و برنامه نویسی Managed Code را دارا هستند. اما هنوز این قابلیت ها توسعه چندانی نیافته اند و در سطح ابتدایی باقی مانده اند... برای استفاده کامل از این تکنولوژی میبایست مدتی صبر کنید.

آیا میشود با Irrlicht برای کنسول بازی ساخت ؟

هنوز نه ، اما توسعه دهندگان موتور در حال مذاکره رسمی برای افزودن قابلیت طراحی بازی بر روی کنسول Xbox هستند...

من دراستفاده از این موتور با مشکلاتی مواجه شده ام. چکار کنم ؟

برخی از سوالات رایج در انتهای مقاله پاسخ داده شده اند. آنها را مطالعه کنید.

اطلاعاتی در مورد کتابخانه های دیگری که قابلیت اجتماع با Irrlicht را دارند :

از آنجایی که Irrlicht تنها یک کتابخانه 3 بعدی است ، بنابراین طراحی و توسعه یک بازی 3 بعدی به تنهایی در آن ممکن نیست... اما مشکلی در این بین وجود ندارد... همان گونه که Open GL هم تنها یک API گرافیکی خالص است ولی در بازیهای زیادی از آن استفاده میشود... در این قسمت به معرفی کتابخانه های جانبی می پردازیم که قدرت موتور Irrlicht در طراحی بازی های 3 بعدی را چند برابر میکنند...

API های صوت و موسیقی :

Fmod : http://www.fmod.org
OpenAL : http://www.openal.org

API های شبکه :

RakNet : http://www.rakkarsoft.com
Zoidcom : http://www.zoidcom.com
HawkNL : http://www.hawksoft.com/hawknl

کتابخانه های فیزیک :

ODE : http://ode.org
Tokamak : http://www.tokamakphysics.com
Novodex : http://www.novodex.com
Newton Game Dynamics : http://www.newtondynamics.com

کتابخانه های هوش مصنوعی (AI) :

Fear : http://fear.sourceforge.net
AI Steering Lib : http://opensteer.sourceforge.net
Mind : http://mind.sourceforge.net/cpp.html
Social AI : http://www.alicebot.org

ساخت اولین دموی تکنیکی 3 بعدی (نوشتن برنامه Hello World با Irrlicht)

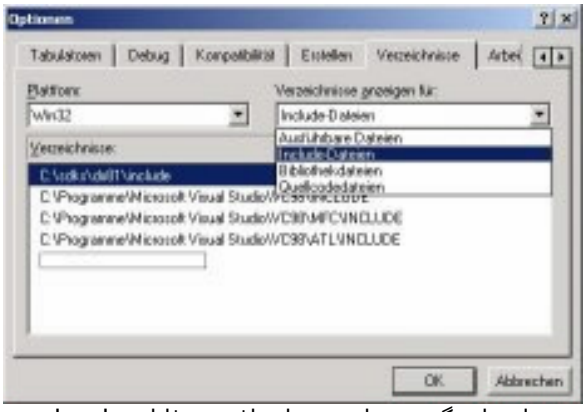
در ابتدای این مقاله می آموزید که چگونه محیط IDE خودتان را برای استفاده از Irrlicht آماده کنید و سپس می آموزید که چگونه برنامه Hello World را با استفاده از این موتور و توابع آن بنویسید... این مقاله نحوه استفاده از Video Driver ، GUIEnvironment و Scene Manager را بطور ساده به شما آموزش میدهد... نتیجه کار شما در انتها مطابق تصویر زیر خواهد بود :

آماده سازی کامپایلر :

برای استفاده از توابع موتور Irrlicht در محیط برنامه نویسی Visual C++ ، شما باید فایل Header این موتور با نام irrlicht.h را که در پوشه include قرار دارد را به لیست include های محیط برنامه نویسی تان اضافه کنید. توضیح این کار برای استفاده از موتور Irrlicht در محیط Visual Studio نسخه 6.0 را در زیر ملاحظه میکنید :

ابتدا به منوی Tools رفته و سپس گزینه Options را انتخاب کنید. از کادر محاوره باز شده روی زبانه Directories کلیک کنید تا لیست فهرست های موجود ظاهر شود. سپس از منوی Drop Down بالا ، گزینه Include Files را انتخاب کنید و در نهایت بر روی دکمه New کلیک کنید تا بتوانید مسیر فایل Header جدید را مشخص کنید. بعد از انجام این اعمال پنجره ها را ببندید. محیط برنامه نویسی شما اکنون آماده پذیرش توابع موتور Irrlicht میباشد.

تصاویر زیر در انجام اینکار به شما کمک خواهند کرد :



انتخاب گزینه مناسب برای افزودن فایل header

دادن مسیر مناسب برای فایل Header

شروع برنامه نویسی

بعد از اینکه فایل irrlicht.h را به کامپایلر افزودیم ، دیگر کامپایلر میداند که این فایل را در کجا بیابد و چگونه توابع فراخوانی شده آن را استفاده کند. بنابراین با نوشتن کد زیر ، این فایل Header را در ابتدای برنامه include میکنیم :

```
#include <irrlicht.h>
```

در هنگام برنامه نویسی با موتور Irrlicht باید به خاطر داشته باشید که همه توابع در فضای نام irr قرار دارند. بنابراین چنانچه شما قصد استفاده از یک کلاس خاص را دارید ، باید ابتدا از فضای نام irr استفاده کنید و به دنبال آن نام کلاس مورد نظرتان را بیاورید. برای مثال جهت استفاده از کلاس IrrlichtDevice باید از کد irr::IrrlichtDevice استفاده کنید.

چنانچه شما تجربه برنامه نویسی با زبان C++ را داشته باشید ، میدانید که این کار خسته کننده و مشکل است. بنابراین برای جلوگیری از آوردن name space در جلوی هر کلاس ، از کد زیر استفاده میکنیم تا به کامپایلر بگوییم تا قصد استفاده از این فضای نام برای کلاسها را داریم :

```
using namespace irr;
```

علاوه بر فضای نام اصلی که در بالا به آن اشاره شد ، 5 فضای نام فرعی نیز در موتور Irrlicht وجود دارند که همانند بالا به منظور جلوگیری از آوردن آنها در روبرو هر کلاس ، آنها را به برنامه می افزاییم :

```
using namespace core;
using namespace scene;
using namespace video;
using namespace io;
using namespace gui;
```

برای اینکه قادر باشیم تا از توابع موجود در فایل irrlicht.dll استفاده کنیم ، باید ابتدا لینکی به فایل کتابخانه Irrlicht,lib ایجاد کنیم. اگرچه ما میتوانیم این کار را در قسمت تنظیمات پروژه انجام بدهیم ، اما برای راحتی از کد زیر برای افزودن آن استفاده میکنیم :

```
#pragma comment(lib, "Irrlicht.lib")
```

بسیار خوب. اکنون نوبت استفاده از متود Main میرسد. در اینجا ما بسادگی از ()Main استفاده کرده ایم. اما بخاطر داشته باشید که چنانچه هدفتان تنها اجرای برنامه بر روی پلت فورم ویندوز است ، میتوانید از متود WinMain استفاده

کنید تا دیگر با پنجره مزاحم کنسول روبرو نشوید. با این وجود توجه داشته باشید که با استفاده از متود Main برنامه شما بر روی همه پلت فورم ها قابلیت اجرا پیدا خواهد کرد...

```
int main()
{
```

مهمترین قسمت در هنگام استفاده از موتور Irrlicht نحوه استفاده از تابع CreateDevice میباشد. تابع IrrlichtDevice که برای کنترل تمامی دستورات مربوط به اشیا در موتور میباشد ، توسط این تابع ساخته میشود. تابع createDevice دارای 7 پارامتر میباشد :

: DeviceType
برای مشخص کردن نوع سیستم رندرگر استفاده میشود. در این مثال از EDT_SOFTWARE استفاده شده است. اما میشود از انواع دیگر رندرگر نظیر EDT_NULL, EDT_DIRECTX8 or EDT_OPENGL هم استفاده کرد.

: windowSize
اندازه صفحه نمایش نهایی.

: Bits
عمق رنگ که میتواند بر روی 16 یا 32 تنظیم شود. هنگامی که از حالت windowed استفاده میشود ، این پارامتر غیر فعال میشود.

: Fullscreen
برای تنظیم صحنه نهایی بصورت fullscreen یا windowed استفاده میشود.

: Stencilbuffer
استفاده یا عدم استفاده از این سیستم در رسم سایه ها.

: Vsync
استفاده یا عدم استفاده از حالت vsync . این گزینه فقط در حالت fullscreen کاربرد دارد.

: eventReceiver
استفاده از یک شیئ برای دریافت رویداد ها. چون در این مثال از آن استفاده نمیکنیم آنرا بر روی 0 تنظیم میکنیم.

```
IrrlichtDevice *device =
               createDevice(EDT_SOFTWARE, dimension2d<s32>(512, 384), 16,
                       false, false, false, 0);
```

اکنون با استفاده از دستور زیر ، یک متن ثابت را بر روی نوار عنوان پنجره برنامه به نمایش در می آوریم. توجه داشته باشید که نقش حرف L که در جلوی متن آمده ، Wide کردن نوشته است :

```
device->setWindowCaption(L"Hello World! - Irrlicht Engine Demo");
```

در این مرحله با ایجاد و ذخیره 3 اشاره گر به video driver ، Scene Manager و Graphical user interface از تکرار مداوم کد های فراخوانی این توابع جلو گیری میکنیم :

```
IVideoDriver* driver = device->getVideoDriver();
ISceneManager* smgr = device->getSceneManager();
IGUIEnvironment* guienv = device->getGUIEnvironment();
```

اکنون با استفاده از دستور زیر یک نوشته ساکن را در گوشه بالای تصویر رسم میکنیم. توجه داشته باشید که چگونه از اشاره گر guienv در اینجا استفاده شده است :

```
guienv->addStaticText(L"Hello World! This is the Irrlicht Software engine!",
        rect<int>(10,10,200,22), true);
```

در واقع تا به اینجا برنامه ما تکمیل شده است. اما برای افزودن چند چیز جذاب به محیط ، از یک مدل بازی Quake 2 استفاده میکنیم. تنها چیزی که در این بین نیاز داریم این است که معادلات و مختصات مش ها را توسط تابع getMesh() استخراج کنیم و در نهایت با استفاده یک scene Node و تابع addAnimatedMeshSceneNode() آنرا بر روی صفحه نمایش دهیم. در ضمن این نکته را به خاطر داشته باشید که میوانید فایل های obj. مایا ، bsp. نقشه های quake و ms3d. مربوط به نرم افزار Milk Shape را نیز توسط این تابع بارگزاری کنید.

در اینجا ما از یک مدل بازی quake با نام Sydney.md2 استفاده کرده ایم. این فایل در پوشه Media وجود دارد.

```
IAnimatedMesh* mesh = smgr->getMesh("../../media/sydney.md2");
IAnimatedMeshSceneNode* node = smgr->addAnimatedMeshSceneNode( mesh );
```

برای اینکه مش ها اندکی زیبا تر و نرم تر بنظر برسند ، تغییرات کوچکی در Material های استفاده شده برای آن ایجاد میکنیم. قبل از هر کاری پارامتر نور را غیر فعال میکنیم. چون ما هیچ منبع نوری را در صحنه ایجاد نکردیم و در صورت فعال کردن این پارامتر ، مدل ما کاملا سیاه بنظر میرسد. سپس چرخه نمایش مدل را در فریم های 0 تا 310 تنظیم میکنیم و در نهایت از یک تکسچر برای پوشاندن مش ها استفاده میکنیم.

```
if (node)
{
        node->setMaterialFlag(EMF_LIGHTING, false);
        node->setFrameLoop(0, 310);
        node->setMaterialTexture( 0, driver-
>getTexture("../../media/sydney.bmp") );
}
```

برای اینکه بتوانیم به صورت 3 بعدی به این مدل نگاه کنیم ، یک دوربین را در مختصات (0, 10, 40-) قرار میدهیم.

```
smgr->addCameraSceneNode(0, vector3df(0,30,-40), vector3df(0,5,0));
```

بسیار خوب. تا به اینجا همه اجزا در محیط قرار داده شده اند و وقت آن است که همه آنها را در پنجره نهایی ترسیم کنیم. از این رو از یک حلقه while استفاده میکنیم تا برنامه تا ورود دستوری برای خاتمه ، بطور مداوم اجرا شود. برای این منظور از دستور زیر استفاده میکنیم :

```
while(device->run())
{
```

همه چیز باید در بین توابع beginScene() و endScene() قرار بگیرد تا قابلیت نمایش در صحنه نهایی را داشته باشد. تابع beginScene() صحنه را با یک رنگ دلخواه پر میکند و بافر عمق را با توجه به تنظیمات قبلی تنظیم میکند. دستور به شکل زیر استفاده میشود :

```
        driver->beginScene(true, true, SColor(0,200,200,200));

        smgr->drawAll();
        guienv->drawAll();
        driver->endScene();
}
```

بعد از این که تمامی این مراحل را انجام دادیم ، نوبت آن میرسد که تکلیف برنامه را بعد از بسته شدن مشخص کنیم. در واقع هدف این است که حافظه اشغال شده از سیستم را به آن برگردانیم. در غیر اینصورت تمامی عناصر حافظه در

سر جای خود باقی میمانند و این بدترین حالت قابل تصور در طراحی یک بازی سنگین 3 بعدی میباشد. بنابراین همواره این نکته را بخاطر داشته باشید که تمامی عناصر ایجاد شده در جریان بازی را در انتها از بین ببرید. برای این منظور از کد زیر استفاده میکنیم :

```
        device->drop();
 return 0;
    }
```

کار برنامه نویسی در اینجا به پایان رسیده است. کافیست که برنامه را کامپایل کرده و آنرا اجرا کنید.

عالی شده ، اینطور نیست ؟

امیدوارم که تا به اینجا از مطالعه این مقاله خسته نشده باشید. چنانچه اولین بار است که اقدام به ساخت یک محیط 3 بعدی میکنید ، ممکن است کدهای بکار رفته در این مقاله اندکی مشکل بنظر بیایند ، اما توجه داشته باشید که قسمت عمده این مقاله صرف توضیحات جانبی و دستورالعمل توابع شد و در واقع کل کد های بکار رفته در این دموی کوچک ، همین چند خط کد هستند :

```
#include <Irrlicht.h>

 using namespace irr;

 using namespace core;
 using namespace scene;
 using namespace video;
 using namespace io;
 using namespace gui;

 int main()
 {
        // start up the engine
        IrrlichtDevice *device = createDevice(video::EDT_DIRECTX8,
                core::dimension2d<s32>(640,480), false);

        video::IVideoDriver* driver = device->getVideoDriver();
        scene::ISceneManager* scenemgr = device->getSceneManager();

        device->setWindowCaption(L"Hello World!");

        // load and show quake2 .md2 model
        scene::ISceneNode* node = scenemgr->addAnimatedMeshSceneNode(
                scenemgr->getMesh("quake2model.md2"));

        // if everything worked, add a texture and disable lighting
        if (node)
        {
                node->setMaterialTexture(0, driver->getTexture("texture.bmp"));
                node->setMaterialFlag(video::EMF_LIGHTING, false);
        }

        // add a first person shooter style user controlled camera
        scenemgr->addCameraSceneNodeFPS();

        // draw everything
```

```
        while(device->run() && driver)
        {
                driver->beginScene(true, true, video::SColor(255,0,0,255));
                scenemgr->drawAll();
                driver->endScene();
        }

        // delete device
        device->drop();
        return 0;
}
```

خطایابی

چنانچه در مرحله ای از ساخت این دموی تکنیکی با مشکلی روبرو شده اید ، لطفا قبل از مطرح کردن آن در انجمن های تخصصی ، موارد زیر را بررسی کنید :

اگر در حین کامپایل مرحله ، پیغامهای خطای زیر را دریافت کردید :

```
fatal error C1083: Cannot open include file: 'irrlicht.h': No such file or directory
```

شما هنوز فایل های لازم را به درستی وارد محیط کامپایلر خود نکرده اید. مجددا قسمت ابتدایی مقاله را مطالعه کنید.

```
LINK : LNK6004: HelloWorld.exe not found or not built by the last incremental link; performing full link
LINK : fatal error LNK1104: cannot open file "Irrlicht.lib"
Error executing link.exe
```

شما فایل های لازم را به طور کامل وارد محیط کامپایلر خود نکرده اید. مجددا قسمت ابتدایی مقاله را مطالعه کنید.

```
This application has failed to start because Irrlicht.dll was not found. Re-installing the application may fix this problem
```

شما باید فایل مربوطه را در پوشه ای که برنامه در آن ذخیره شده است کپی کنید.

```
Could not load mesh, because file could not be opened.: ../media/sydney.md2

Could not open file of texture: stones.jpg
Could not load texture: stones.jpg
```

شما باید فایل های مربوطه را در پوشه ای که برنامه در آن ذخیره شده کپی کنید.

سایت طراحان ایرانی با هدف آموزش ساخت بازیهای کامپیوتری به زبان فارسی طراحی شده است و تا کنون مقالات متعددی در زمینه های مختلف برنامه نویسی و ساخت بازی در آن قرار گرفته است. مدیریت سایت از تمامی عزیزان علاقمند به بازی های کامپیوتری ، برنامه نویسان ، طراحان و سایر کسانی که به نحوی با بازی ها در ارتباطند ، دعوت به همکاری به عمل می آورد تا بدینوسیله یک پایگاه علمی و موثق در زمینه صنعت ساخت بازیهای کامپیوتری در ایران ایجاد گردد.

در ضمن بسیاری از نرم افزار های ساخت بازی های کامپیوتری که امروزه در سطح وسیع مورد استفاده قرار میگیرند ، در سایت جمع آوری شده است و با مبلغ بسیار ناچیزی در اختیار علاقمندان به طراحی بازی های کامپیوتری قرار داده شده است. استفاده از این نرم افزار ها در آغاز کار و به منظور آشنا شدن با اصول اولیه در طراحی بازیها بسیار موثر و مفید بوده و شما میتوانید تا با چند جستجوی ساده در این زمینه ، به صحت موضوع پی ببرید. لیست زیر برخی از نرم افزار هایی هستند که توسط فروشگاه الکترونیکی سایت به مشتاقان عرضه میشوند :

- Game Maker Version 5.0 – 5.1 – 5.2 – 5.3 – 6.0 (Registered)
- The Game Factory (Home – Professional) (registered)
- Xtereme 3D 1.0
- King Space 3D
- Genesis 3D V1.6
- 3D Game Studio 5.12
- 3D State (Morfit 3D) (Registered)
- Blender 3D
- Q3d (Unregistered)
- Alice 3D
- True Vision 3D V6.2
- DirectX 9.0 Complete SDK (Software Development Kit Package)

و موتور های سورس باز 3 بعدی زیر :

- Jolt3D (Third Person Maker) Demo & Source
- Boom3D (Demo & Source)
- PushTheLimits 3D Engine
- Mystica 3D
- Arfnold 3D (Source & Bin)
- TerraCresta 3D (Demo & Source)
- XEngine (Sample & Source)
- Muli3D (Sample & Source)
- Apocalyx (Source)
- Graden (Source)
- DXQuake (Source)
- 6DX
- CHAI 3D (Source)
- Axiom 3D
- syBR (Source)
- iRender 3D (SDK)
- Cube 3D (Source & SDK)
- Q Engine
- Hawk 3D Engine (Source & Bin)
- Neo Engine (Tools & Source & Tutorials)
- Aurora (installer & Tutorials)
- Soya 3D
- DexVT (Source)
- Jet 3D (Source & Bin)
- Traktor 3D SDK (Source & Bin)

- NemoX (Installer)
- Unreal 2 (SDK)
- Irrlicht 3D (SDK & Source)

و کامپایلر های

- Visual Basic V6.0
- Visual C++ V6.0

لینک فروشگاه الکترونیکی سایت طراحان ایرانی :

WWW.Persian-Designers.COM/index.php?pid=1

تجربه ای لذت بخش در زمینه ساخت بازیهای کامپیوتری را برایتان آرزومندیم