

به نام خداوند بخشنده مهربان

gamestudio A7

آموزش شبیه ساز فیزیک در 3D Game Studio a7

نویسنده: آرش فواجویی نژاد ARASHDJ71

ورژن نرم افزاری مورد نیاز: 3d game studio a7

پیش نیاز: مطالعه مقاله برنامه نویسی در گیم استودیو.

سطح مقاله: متوسط , پیشرفته

بهترین حالت مشاهده: مد اقل 600*800

ایمیل: arashdj72@yahoo.com

این مقاله تنها از سایت **طراحان پارسی** قابل دریافت است:

www.persian-Designers.COM

.....

مقدمه:

open dynamics 3d game studio دارای یک سیستم زیر مجموعه ی فیزیکی است که تا مدودی بر پایه موتور فیزیکی engine نوشته شده و کمک میکند تا مرکات مدلها در بازی واقع گرایانه تر شود. به طور کلی این بخش از rigid bodies (شبیه سازی مدلهای سه بعدی بدون بد شکل شدن و زشت شدن در هنگام حرکت) این بخش به اندازه water bodies کاملاً پویا عمل میکند.

چون این موتور فیزیکی فشار زیادی به CPU وارد میکند بطور default یا از پیش تعریف شده این بخش غیر فعال است. زمانی که ما میخواهیم از این فیزیک استفاده کرده و آن را برای بازی خود فعال کنیم باید entities یا موجودیت ها را ثبت کرده و آنها را برای شبیه سازی فیزیکی فراخوانی کنیم. این به این معنیست که شما مثل یک نماینده ی انحصاری کنترل این آبجکتها و اشیاء رو بدست میگیرید (به وسیله موتور فیزیکی).

اگر برای مدت طولانی نمیخواهیم از این سیستم فیزیکی استفاده کنیم باید جهت بهینه شدن بازی entities را غیر فعال کرده و به طور کلی در زمان مورد نظر سیستم فیزیکی را از کار بپندازیم.

شاید برای شما وسوسه انگیز باشه که در همه جای بازی از این موتور فیزیکی استفاده کنید و یا شاید به خاطر سادگی کار اصلاً آن را غیر فعال نکنید ولی پیشنهاد ما به شما این است که برای پردازش و نمایش رانتر کار این سیستم را فقط در مواقع نیاز فراخوانی (فعال) کنید.

با فعال کردن اولیه این شبیه سازی مرکتی در موتور فیزیکی برای اشیاء برای شما شروع یک تمول فواید بود زیرا در ابتدا شبیه سازی فیزیکی و اعمال آن روی اجسام بصورت بسیار نا مطلوب انجام فواید گرفت ولی با فراخوانی توابع و پیش نیازهایی که در ادامه به آنها اشاره میکنیم شما میتوانید این چاشنی را به بازی خود افزوده و به بازکن اجازه بدهید که در دنیای غیر واقعی که شما خلق کردید شناور شده و مرکت کند.

شما در physics examples های این نرم افزار میتونید پروژه مرکت یک اتومبیل روی یک terrain یا یک جرقیل که بوسیله آن میتوانید آبجکتهای روی زمین را بلند کرده و جابه جا کنید رو پیدا کرده و سورس این پروژه ها را مورد بررسی و تجزیه و تحلیل و قرار دهید.

توابع:

در اینجا ۳ تابع که در فراخوانی و فعال کردن سیستم فیزیکی نقش اساسی دارند را مورد بررسی قرار میدهم.

- **ph** جزء توابع کلی global در تغیر رفتار و مرکات اجسام و آبجکتها تاثیر دارد.
- **Phent** فراخوانی توابع فیزیکی بوسیله این کلمه صورت میگیرد.
- **Phcon** برای تعیین ممدوده ای خاص برای اشاره گرها موجود مورد استفاده قرار میگیرد.

توپه:

فقط مدلهای سه بعدی با فرمت wmb یا mdl میتوانند در شبیه سازی فیزیکی مورد استفاده قرار گیرند. در زیر مرامل یک شبیه سازی فیزیکی ساده را مرور میکنیم:

- ۱- ایجاد یک جسم در محیط به وسیله دستور ent_create یا بوسیله wed.
- ۲- جابه جایی مدل مورد نظر تا در وضعیت مناسب از مرملة قرار گیرند.
- ۳- فراخوانی phent_settype برای سپردن کنترل جسم به دست شبیه ساز فیزیک.
- ۴- مشخص کردن خواص و ویژگی های جسم به وسیله کلمات کلیدی (phent_setmass, phent_setgroup, phent_setdamping)
- ۵- تنظیم کردن جاذبه به وسیله ph_setgravity.
- ۶- فعال سازی شبیه ساز فیزیک نرم افزار به وسیله ph_selectgroup.

مال به افتصار در مورد کلمات کلیدی به کار رفته در جملات بالا توضیح میدهم:

: Ent_create

تابعی جهت ایجاد جسم در محیط بازی. سافتار کلی این تابع به شکل زیر است:

ent_create(STRING* filename, VECTOR* position, function);

file name: نام مدل مورد نظر که میتواند از فرمت‌های wmp, mdl, tga, pcx, bmp, or dds format یا

کلمه null برای ایجاد به جسم تعریف نشده استفاده کرد.

Position: مشخص کردن وضعیت مدل در آغاز ایجاد شدن.

Function: فراخوانی اکشن و رویدادی برای مدل و یا استفاده از کلمه null برای بدون اکشن بودن مدل.

مثال ent_creat

you = ent_create ("flash.pcx", temp, flash_prog);

: phent_settype

با فراخوانی این تابع شما در مقیقت موتور فیزیکی را فعال کرده و سایر متغیر ها و توابع را بدست آن میسپارید (اگر

به طور صمیم از درون [ph_selectgroup](#) فراخوانی شده باشند). و با غیر فعال کردن این کلمه موتور فیزیکی از کار

میفتد و سایر اعمال بدست توابع و متغیر استاندارد گیم استدیو واگذار میشود.

شکل کای این کلمه به صورت زیر است:

phent_settype (ENTITY*, var type, var hull);

entity: موجودیت اشاره کننده به فعال سازی یا غیر فعال سازی موتور فیزیکی.

Var tybe: از دو بخش PH_RIGID برای شروع کار موتور جهت یک استفوانبندی کاراکتری و PH_WAVE برای

شروع کار موتور برای شبیه سازی مثلا به dynamic water surface (شبیه سازی آب در سطح یک terrain) مورد

استفاده قرار میگیرد.

اگر مقدار var tybe صفر شود موتور فیزیکی خاموش میشود(همان کاری که باید در انتهای فراخوانی موتور انجام

دهیم).

Var hull: شکل برافورد یا تصادم در این آبجکت که میتوان از:

PH_BOX, PH_SPHERE, PH_CYLINDER or PH_POLY.

در این متغیر استفاده نمود.

شکل کلی فراخوانی و روشن کرد موتور فیزیکی با phent_settype :

```
ENTITY* myCrate;
// ...
// on startup
myCrate = ent_create( "crate.mdl", nullvector, any_function );
phent_settype( myCrate, PH_RIGID, 0 );
phent_settype( myCrate, 0, 0 );
// ...
// during gameplay let's have fun with a crate:
var position;
phent_settype( myCrate, PH_RIGID, PH_BOX );
position = myCrate.x; // this will give you an approximate position
//myCrate.x = position + 10; // this won't work!
phent_settype( myCrate, 0, 0 );
myCrate.x = position + 10; // unregistered - now we can change the position
phent_settype( myCrate, PH_RIGID, PH_BOX ); // restart from new position
```

توجه: کدهایی با رنگ قرمز فقط توضیح میباشند و در عمل کاری انجام نمیدهند.

: phent_setmass

میزان جرم جسم (تأثیر مستقیم بر سرعتش دارد) بوسیله این تابع مشخص میشود به وسیله این تابع میتوانید اجسام را مثلاً وادار به گردش کرده و یا مانع از گردش آنها شوید.

شکل کلی این کلمه ی کلیدی به شکل زیر است:

phent_setmass (ENTITY* entity, var mass, var hulltype);

entity : اشاره گر مورد نظر معمولاً از my استفاده میشود.

Var mass : متغیر جرم با مقدار دادن به جسم جرم داده و با 0 قرار دادن موجب ثابت ماندن جرم جسم میشود.

var hulltype : نوع حرکت و یا پرفرش (رفتار) جسم که میتوان از متغیرهای از پیش تعریف شده ای مثل:

PH_SPHERE, PH_BOX, PH_CYLINDER, or PH_POLY.

استفاده نمود. (مقدار این متغیر باید برابر با مقدار متغیر فراخوانی شده در بخش var hull در phent_settype باشد.)

متغیر mass بر مسب کیلو گرم میباشد و فاکتور اصلی آن روی زمین پند است که برابر با 0.45 کیلو گرم است. کمترین جرمی که میتوان به آبجکت فود بدهید 0.001 kg است و کمتر از آن 0 است و با صفر قرار دادن متغیر var mass جسم در هیچ زاویه ای حرکت پویا نخواهد داشت.

مثال phent_setmass :

phent_setmass(my, 0, PH_SPHERE);

: phent_setdamping

این تابع برای شبیه سازی اصطمکاک و تأثیر آن بر سرعت اجسام کاربرد دارد که مقدارش بین ۰ تا ۱۰۰ متغیر است. شکل کلی آن بصورت زیر است:

phent_setdamping (entity, var linear, var angular);

entity : یک مومودیت یا آبجکت مثلاً همان آبجکت mycrate :

به شکل زیر تعریف میکنیم.

ENTITY* myCrate;

جسم مورد نظر رو برابر با یک اشاره گر قرار میدهیم!

بعد بطور مثال در اکشن جسم مورد نظر آن را برابر با جسم قرار میدهیم به شکل زیر:

```
action object1{
    mrcrate=me;
}
```

و یا مثل مثال قبلی به شکل زیر به جسم رو از درون sed فراخوانی میکنیم.

ENTITY* myCrate;myCrate = ent_create("crate.mdl", nullvector, any_function);

Var linear :متغیر خطی یعنی مقدار اصطمکاک برافوردی آبجکت بر مسب خط مثلاً برافورد چرخ اتومبیل با زمین بین 0.00 تا 100.00 متغیر است.

var Angular : متغیر زاویه ای میزان برافورد گوشه و یا زوایای جسم به اطراف و میزان اصطمکاک آن که بین 0.00 تا 100.00 متغیر است

مثال توضیحات بالا برای phent_setdamping :
phent_setdamping (mycrate,50,70);

: phent_setgroup

برای اینکه موجودیت یا آبجکتهای خود را در گروه های مشخصی قرار دهیم مورد استفاده قرار میگیرد و شکل آن بدین گونه است.

phent_setgroup (entity, var groupID);

entinty : آبجکت مورد نظر(قبلا در مورد ایجاد این آبجکتها در sed و یا در wed توضیح کافی داده شده) مثلا:

player

Var groupid : شماره و یا همان آیدی شناسایی آبجکت در گروه!

توجه: این متغیر امبارا باید ۲ یا بالاتر از ۲ انتخاب گردد!

بطور مثال برای قرار دادن په آبجکت در گروهی خاص:

// player collides with bot and forcefield, but bot can walk through the forcefield

phent_setgroup(player, 2); *// قرار گرفته ۲*

phent_setgroup(forcefield, 4); *// قرار گرفته ۴*

phent_setgroup(patrolling_bot, 4);

while (1) {

if (player_inside_building) { *// انجام یک عمل در دستور*

شرطی مثلا خارج شدن پلیر از ساختمان در اینجا مثال زده شده.

ph_selectgroup(1+2+4); *// activate group 1+2+4*

} else {

ph_selectgroup(1+2); *// disable group 4 cause player has left the building*

}

: ph_setgravity

این تابع شامل شبیه ساز نیروی جاذبه و تعیین میزان آن روی یک محور است.

گاهی اوقات روی محور **Y** به شکل **vecgravity.y**

گاهی اوقات روی محور **X** به شکل **vecgravity.x**

گاهی اوقات روی محور **Z** به شکل **vecgravity.z**

اگر بخواهیم از کمتر از مقدار **X.Y** یا **Z** استفاده کنیم(مقدار منفی این بردارها)

مرف تعریف آن بشکل زیر میشود:

Vecgravity. -x یا -y و یا -z .

مثلا اگر میخواهید باد را در یک تونل شبیه سازی کنید برای جهت باد باید از منفی و مثبت بردار **X** استفاده کنید.

نیروی جاذبه در فیزیک نقش مهمی را ایفا میکند مثلا شبیه سازی جاذبه زمین که برابر با $1, -9.81 \text{ m/s}^2$ ca گاهی

اوقات این کمیت به **۱** اینج هم میرسد که اندازه **2.54 cm** سانتی متر است. پس برای مثال نیروی جاذبه استاندارد

زمین مول محور **X** را به شکل زیر مینویسیم.

$\text{vecGravity.z} = (-9.81 * 100 / 2.54) = -386$

و یا برای فضا که فلاء کامل است باید مقدار تمامی این متغیر ها را ۰ قرار داد مثل شکل زیر:

vecGravity to (0,0,0)

و دیگر مرکات از جمله اصطمکاک نسبت به جرم ماده بستگی به نیروی جاذبه وارد بر ماده یا وزن آن از سوی زمین

دارد.

شکل کلی این تابع برای تعریف به شکل های زیر است:

```
var earthgravity[3] = 0,0, -386;
var jupiterGravity[3];
ph_setgravity( earthgravity );
sleep(1); // 1 second of ordinary gravity

ph_setgravity( nullvector );
sleep(1); // floating in space for another second

vec_set( jupiterGravity, earthGravity );
vec_scale( jupiterGravity, 2.5 );
ph_setgravity( jupiterGravity );
sleep(1); // things will become really heavy now..
```

توجه: شماره بعد از متغیر ها که در یک پرانتز قرار دارد نشان دهنده میزان ورودی است که این متغیر در خود قرار

میدهد مثلا با نوشتن شماره ۳ یعنی ۳ ورودی به این متغیر میتوان نسبت داد.

- در خط اول ph_setgravity برابر با متغیر earth gravity قرار دارد که مقدار 0,0,-389 به آن داده شده است.

- در خط دوم نوعی فلاء ایجاد شده:

```
ph_setgravity( nullvector );
```

nullvector به معنی همان 0,0,0 ایست که قبلا تعریف کردیم پس استفاده از هر دو اینها صمیم است به شکل زیر
مثلا:

```
ph_setgravity(0,0,0);
```

یا به شکل ۳ عمل میکنیم و ابتدا مقدار jupiterGravity را برابر با earthGravity قرار داده و سپس در خط بعد مقدار عددی jupitergravity که در سافتار vec_scale متما باید از ۳ مقدار تشکیل شده باشد را در فاکتور مثلا ۲,۵ ضرب میکنیم مثلا اگر ما داشته باشیم:

```
Vec_set(my.x,your.x)
////////////////////////////////
vec_set( jupiterGravity, 0.0.2 );
////////////////////////////////
jupitergravity[3]=0,0,2;
////////////////////////////////
vec_scale(VECTOR* vector, var factor);
////////////////////////////////
var v[3] = { 10, 20, 30 };
vec_scale(v,2.5); // v now contains 25, 50, 75;
////////////////////////////////
```

و اینگونه است که ما داریم:

```
ph_setgravity( jupiterGravity );
```

یعنی همان:

```
ph_setgravity(25,50,75);
```

ph_selectgroup :

همه ی موجودیت ها باید برای شبیه سازی فیزیک در گروههای مشخصی قرار گیرند(قبلا چگونگی این کار توضیح داده شده)

وقتی یک کاراکتر بطور مثال از دید دوربین خارج میشود ما دیگر آنرا نخواهیم دید و در ادامه بازی تاثیر ندارد پس با کمک این تابع گروه این کاراکتر را پس از خارج شدن از view خود تغیر میدهیم و باعث میشویم مانند یه ذره فنتی عمل کند و دیگر نیازی به انجام شبیه سازی بیهوده برای این کاراکتر نداریم.

شکل کلی این تابع اینگونه است:

ph_selectgroup (var bitmask);

1	0	1	0		<- Bitmask (6), first four bits of 21 shown	فعال باشد؟
0	0	0	1		<- Ent1 group ID (1)	no
0	0	1	0		<- Ent2 group ID (2)	Yes
0	1	0	0		<- Ent3 group ID (4)	No
1	0	0	0		<- Ent4 and Ent5 group ID (8)	Yes

به طور کلی اگر به کد پایین نگاه کنید متوجه خواهید شد که:

```
phent_setgroup( ent1, 1 );
phent_setgroup( ent2, 2 );
phent_setgroup( ent3, 4 );
phent_setgroup( ent4, 8 );
phent_setgroup( ent5, 8 );
// activate ent1, all others are paused
ph_selectgroup( 1 );
// activate ent2 & ent3
ph_selectgroup( 6 );
// activate ent4 & ent5
ph_selectgroup( 8 );
// run them all
ph_selectgroup( 15 );
```

توضیح:

اول هر ent را بصورت یک گروه در آورديم.

بعد گروه ها را بر مسب جمع مقادیرشان انتخاب کردیم.

مثلا (6) ph_selectgroup یعنی گروه هایی رو انتخاب کن که مجموع مقادیرشان ۶ شود. پس اینجا ent2 و ent4 انتخاب میشوند. همنظور برای سایر گروهها.

پایان قسمت اول...

کلیه مطالب این مقاله بوسیله نویسنده مقاله تهیه و تنظیم شده و
حاصل زحمات شبانه روزی نویسنده آن است پس لطفا بدون ذکر منبع و
نام نویسنده از مطالب درج شده در آن استفاده نکنید.

هرگونه سوال در مورد این مقاله را در تالار گیم استادیو سایت طراحان
پارسی مطرح کنید.

هفته های پایانی مرداد 87

با آرزوی موفقیت و خدا نگهدار...

Www.PERSIAN-DESIGNERS.com