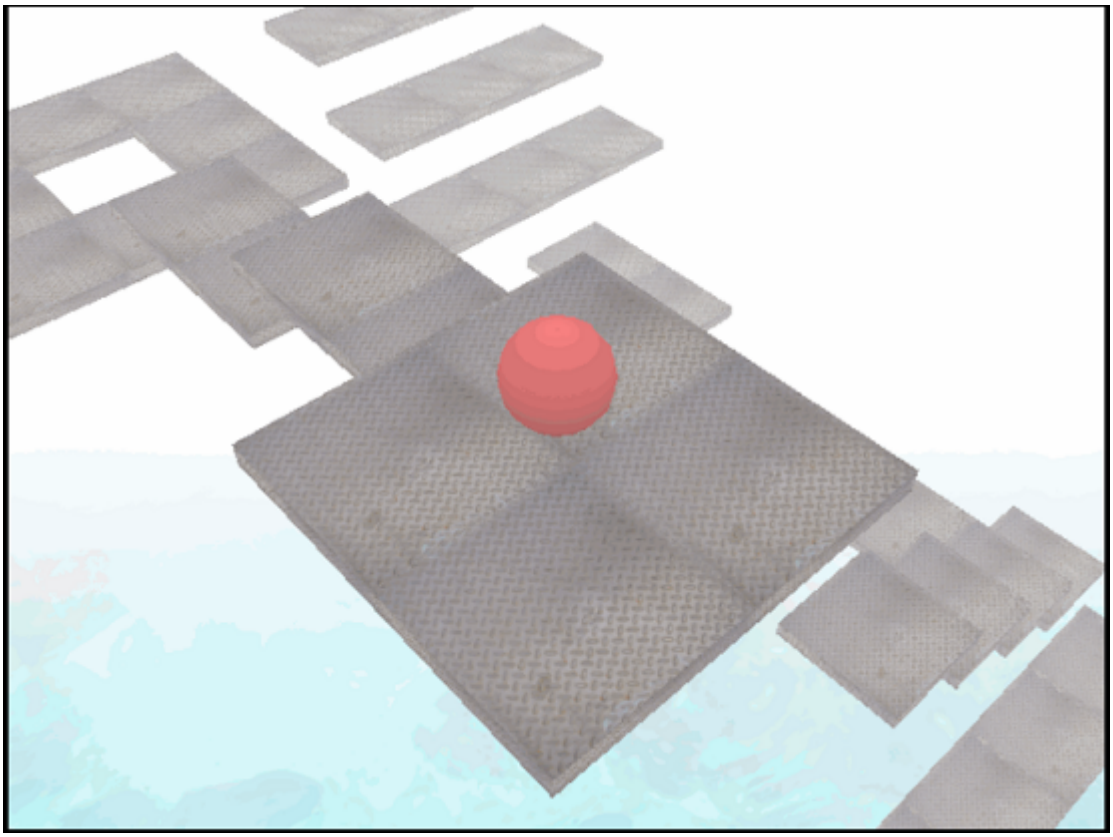
آموزش های سه بعدی GM6

جلسه پنجم : ساخت platform ها



نویسنده : Prince_Of_Persia

مطالبی که در این جلسه می آموزید :

- در رابطه با بازیهای platform
- استفاده از جاذبه
- طراحی مراحل ساده
- استفاده از دید سوم شخص
- انیمشین سازی با استفاده از تغییر شکل
- ایجاد حرکت ، پرش ، سقوط کاراکتر

P30World

در رابطه با بازیهای Platform

در آموزش های قبلی ما دیدیم که چگونه می توانیم مقدار z را تعریف نمایید و از آن بصورت interactive استفاده نماییم. ما از کد های مانند زیر استفاده کردیم :

```
//set speed
if z>0 then speed*=.99;
```

بازیهای platform معمولا بازیهای هستند که از یک سطح افقی صاف در آن استفاده می شود که کاراکتر می تواند روی آن حرکت کند ، بپرد یا با مشکلات گوناگون دیگری برخورد داشته باشد. ارتفاع در این گونه بازیها نقش اساسی را بازی می کند.

داشتن یک بازی بدون هیچگونه تفاوت در ارتفاع هیچ حسی از یک بازی platform را نشان نمی دهد. اگر شما تفاوت ارتفاع در بازیهای platform خود نداشته باشید ، تنها یک سطح صاف وجود خواهد داشت. باید مکانهایی وجود داشته باشد که امکان سقوط از سطح به مکان عمیق تری را فراهم نمایند یا راهی برای رفتن با مکانهایی با ارتفاع بالاتر وجود داشته باشد.

اگر شما یک کاراکتر را داشته باشید که بر روی یک سطح حرکت می کند ، این کاراکتر باید امکان سقوط یا پرش به سطح های دیگر را داشته باشد. پرش و سقوط در ارتفاع عملی است که در موقعیت z تعریف و تشریح می شود.

استفاده از جاذبه
نیروی جاذبه چیزی است که باعث کار کردن بازی های سبک platform می شود. بدون جاذبه مکنست شما یک کاراکتر داشته باشید که می توانید در وسط آسمان حرکت کند.

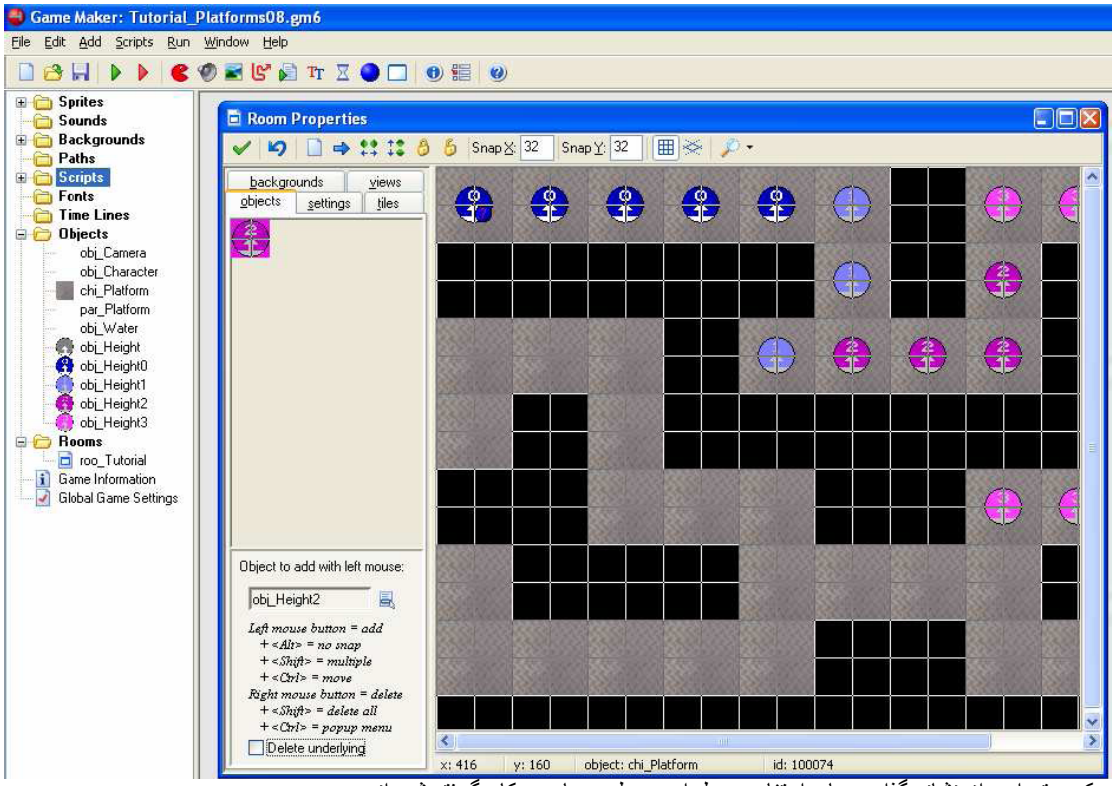
برای این آموزش ما به دو بافت یا texture نیاز داریم : یک سطح فلزی و یک سطح آب ، زیرا آنها تنها چیزهایی هستند که ما نیاز داریم. ما می خواهیم که کارکاتر بروی سطح های فلزی راه برود یا اینکه در سطح آب سقوط کند. کاراکتر ما به آب حساسیت دارد و باعث مرگ کاراکتر می شود.

در دنیای واقعی نیروی جاذبه همیشه عمل می کند. در دنیای مجازی بازیهای platform ، جاذبه می تواند فعال و غیر فعال شود. این بدان معنی است که ما می توانیم جاذبه را برای کاراکتر هنگامی که روی سطح جامد است غیر فعال کرد و هنگامی که بر روی آن نیست فعال کرد.

در بخش های بعدی این آموزش شما می توانید مشاهده کنید که از جاذبه برای راه رفتن ، سقوط ، پرش و دویدن کاراکتر استفاده می شود.

طراحی مراحل ساده
بدلیل ساده تر کردن کار ما سطح های ساده ای را ایجاد می کنیم و از سطح های روی هم افتاده و همپوشان (سطح های که بر بالای یکدیگر قرار دارند) استفاده نمی کنیم. سطح های همپوشان می توانند از سطح بالا و پایین پیموده شوند. سطح های ساده پهلو به پهلوی منتها با ارتفاع های متفاوت قرار دارند.

ما باید مرحله خود را در ویرایشگر room نرم افزار GM طراحی کنیم تا سطوح را در مکان و ارتفاع های متفاوت قرار دهیم. برای انجام این کار ما از نشانه گذارهای ارتفاع استفاده می کنیم که در طراحی مرحله قرار می دهیم.این بدان معنی است که می توانیم یک مرحله را هم در ظاهر هم در ارتفاع طراحی کنیم.



عکس :تعدادی از نشانه گذاری های ارتفاع در طراحی سطوح ساده به کار گرفته شده اند.

اگر شما نگاهی به اسکریپت scr_Marker فایل سورس این آموزش بندازید می بینید که برای هر بخش از سطوح ما یک مقدار z را نسبت داده ایم.

```
//make the height
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height0,false,false) then
z=0*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height1,false,false) then
z=1*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height2,false,false) then
z=2*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height3,false,false) then
z=3*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height4,false,false) then
z=4*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height5,false,false) then
z=5*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height6,false,false) then
z=6*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height7,false,false) then
z=7*global.Grid;
```

```
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height8,false,false) then
z=8*global.Grid;
else
if collision_rectangle(x-32,y-32,x+32,y+32,obj_Height9,false,false) then
z=9*global.Grid;
```

کد بالا ارتفاع برای هر کاشی از سطح را ایجاد می کند (به هر حال مقدار global.Grid ارتفاع شبکه را مشخص می کند). کاشی های واقعی در رویداد Draw آبجکت par_Child طراحی می شوند (الگوی والد و فرزند Parent_child در این مثال استفاده شده است). کدی که برای طراحی کاشی استفاده می شود بصورت زیر است :

```
//draw a platform tile
d3d_draw_block(x-32,y-32,z-8,
x+32,y+32,z,background_get_texture(bac_Metal),1,1);
```

هنگامی مراحل خود را طراحی می کنید بهتر است با قرار دادن کاشی های سطوح کار خود را آغاز کنید ، سپس نشانه گذارهای ارتفاع و در آخر آبجکت دوربین را قرار دهید (که آبجکت دوربین به نوبه خود آبجکت کاراکتر را ایجاد می کند).

استفاده از دید سوم شخص

البته ما به یک دوربین برای مشاهده صحنه احتیاج داریم. این عمل با استفاده از کد دوربین بصورت زیر انجام پذیر است :

```
//draw what camera sees
//from x,y,z
xf=obj_Character.x-sin(-degtorad)
obj_Character.direction+90))*global.Zoom
yf=obj_Character.y+cos(-degtorad)
obj_Character.direction+90))*global.Zoom
zf=obj_Character.z+global.Zoom

//to x,y,z
xt=obj_Character.x
yt=obj_Character.y
zt=obj_Character.z

d3d_set_projection (xf,yf,zf,xt,yt,zt, 0,0,1);
```

اولین بخش نقطه ای را که دید از آنجا شروع می شود (xf,yf,zf) را مشخص می کند. این مکان بسته به موقعیت و جهت کاراکتر و همچنین بزرگنمایی و مقدار زوم دوربین می باشد. سینوس و کسینوس به طریق خاصی استفاده می شوند تا دوربین را طوری بچرخانند که در پشت کاراکتر قرار بگیرد.

دوربین به مکان آبجکت (xt,yt,zt) نگاه می کند. در آخر پروجکشن همانگونه که انتظار می رفت در خط آخر کد تنظیم می شود.

انیمشین سازی با استفاده از تغییر شکل

یک انمیشن ساده پرش برای آشکار کردن هنگامی که بازیکن پرش یا سقوط می کند با هنگامی که اینکار را نمی کند اضافه شده است. توپ قرمز که کاراکتر اصلی ما می باشد بسته به سرعت خود تغییر شکل می دهد. کد زیر گویای کار است :

```
//animation (deform)
if abs(zspeed)<8 then deform=abs(zspeed);
//draw ball
d3d_draw_ellipsoid(x-16,y-16,z-deform,     x+16,y+16,z+32+deform,
background_get_texture(bac_Character),1,1,16);
```

ایجاد حرکت ، پرش ، سقوط کاراکتر

کدی که کار پرش ، راه رفتن و سقوط کاراکتر را انجام می دهد نسبتا پیچیده است. این کد شامل منطق ریاضی است. در رویداد Step کاراکتر (obj_Character) ما کد scr_Gravity را قرار می دهیم. این کد بررسی می کند که کاراکتر روی سطح پلاتفورم است یا خیر. این کد بصورت زیر است :

```
//check if jumping
if jump=1 then exit;

//check if falling
if fall=1 then exit;

//define gravity
if collision_point(x,y,chi_Platform,false,false) then
{
znear=instance_nearest(x,y,chi_Platform).z;
if z=znear then zgravity=0;
else {fall=1;speed+=.2;zgravity=-.5;}
}
else {fall=1;speed+=.2;zgravity=-.5;}
```

در همان رویداد Step کد حرکت کاراکتر در راستای z را قرار می دهیم :

```
//move in z direction
z+=zspeed;
zspeed+=zgravity;

//drowned
if z<-512-128 then game_restart();
```

بررسی به زمین نشستن کاراکتر نیز در همین رویداد انجام می پذیرد. کد آن به شکل زیر است :

```
//check if not falling yet
if sign(zspeed)>=0 then exit;

//check if falling
if sign(zspeed)<0 then fall=1;

//land if falling
if collision_point(x,y,chi_Platform,false,false) then
{
znear=instance_nearest(x,y,chi_Platform).z;
if z=znear ||
```

```
 (z>znear && z<znear+global.Grid/4)
then {jump=0;fall=0;z=znear; zspeed=0; zgravity=0}
}
```

ما به جزییات اسکریپ های بالا نمی پردازیم زیرا تنها نمونه های ساده ای هستند که می گویند اگر (if) اتفاق مورد نظر ما افتاد آنگاه (then) واکنش مورد نظر ما را انجام دهد. این کد باعث می شود که کاراکتر بر روی سطوح به زمین بنشیند.

اسکریپت پایانی ما اسکریپت پرش می باشد (درررویداد فشردن Space از obj_Character می تواند یافت شود) که نام آن scr_Jump است. این کد بصورت زیر است :

```
/check if jumping
if jump=1 then exit;

//check if falling
if fall=1 then exit;

//check if you can jump
znear=instance_nearest(x,y,chi_Platform).z;
if z=znear then {jump=1;zspeed=8;zgravity=-.5}
```

دوباره ما بررسی های زیادی را در این کد انجام دادیم. این کد بررسی می کند که اگر کاراکتر دقیقا بروی سطوح پلاتفورم است باعث پرش کاراکتر شود.

*****