Pacificsys Games
presents

مروری بر نحوه طراحی و ساخت بازی های دو بعدی ایزومتریک

با الهام از بازی محبوب `Streets of rage 3`

نویسنده :

ستاره الوند پور

ناشر :

Www.Persian-Designers.Com
همراه با دموی آموزشی

فصل اول :

آشنایی با بازیهای دو بعدی ایزومتریک

سلام دوستان

نمیدونم آیا شما هم جزو کسانی بودید که کودکی خود را با رویای بازی های کامپیوتری سپری کرده یا نه ؟ آیا شما هم مدام در حال سروکله زدن با پدر و مادر خود برای تعویض کنسول های جدید بودید ؟

Atari , Atari 2600 , Nintendo , Nintendo DS , Sega , Mega Drive , Sega Genesis , ...

هر کدام از این کنسول ها ، جذابیتهای مخصوص خودش را داشت . مطمئنا بازی محبوب آتاری ، یعنی هواپیمای آنرا هر کسی برای یکبار هم بازی کرده باشد هرگز فراموش نخواهد کرد . همانطور که شما اکنون با دیدن تصویر زیر ، یاد دوران خوش قدیم افتادید :

تصویر بالا ، قسمتی از بازی محبوب Streets of rage 3 است که در ایران با نام شورش در خیابان آنرا میشناسند .
دقیقا یادم نیست چند هفته یا چند ماه وقت صرف اجرای این بازی کردم . همیشه دلم میخواست تا مرحله آخر را بازی کنم و ببینم محیط های جدید چگونه است ، دشمنان جدید چه موجوداتی هستند و اینکه آخر داستان به کجا می رسد ...

در آن زمان که سن و سال کمی داشتم ، ساخت چنین بازی زیبایی که هنوز هم زیبایی خود را بین هزاران بازی سه بعدی امروزی حفظ کرده ، برای من یک معجزه بود ...

این بازی تاامروز برای من یک الگو بود . و امروز ما آنرا الگوی آموزشی خود کردیم تا ببینیم این معجزه دوران خوش کودکی واقعا چگونه ساخته شده ... و اینکه آیا ما هم میتوانیم معجزه کنیم ...؟!

فصل اول :

آشنایی با بازیهای دو بعدی ایزومتریک

بسیار خوب .

همینکه یک نظری به گذشته کردیم ، آمادگی لازم را برای کسب تجارب جدید در دنیای بازیسازی بدست آوردیم .

بنابراین الگوی ما در این سلسله مقالات ، بازی Streets of Rage 3 است که میتوانید آنرا روی کامپیوتر شخصی خود و تنها با نصب شبیه ساز دستگاه سگا که یک برنامه نرم افزاری بسیار کوچک است اجرا کنید . این شبیه ساز را میتوانید به راحتی در اینترنت و با جستجوی واژه ای مثل Sega genesis emulator پیدا و به رایگان دانلود کنید . همچنین تمام بازیهای سگا از جمله همین شورش در خیابان را میتوانید از سایت های متعدد که ROM اینگونه بازیها را به رایگان منتشر میکنند دانلود کنید از جمله سایت های Romworld.com , Coolrom.com و یا جستجوی نام کامل بازی در Google.com

همچنین میتوانید کد های cheat (نسوز کننده بازی) پیدا و دانلود کنید و در قسمت cheat نرم افزار شبیه ساز سگا وارد و فعال کنید تا بازی را بدون نگرانی از باختن اجرا و تمام کنید و تمام قسمتهای آنرا مشاهده کنید .البته در پایان این مقاله ، کد های نسوز کننده ، اطلاعاتی در مورد شبیه ساز قدرتمند Gens و غیره را میتوانید مشاهده نمائید .

برگردیم به مبحث آشنایی با بازی های ایزومتریک ...

من خیلی راحت این واژه را معنی میکنم . دو تصویر زیر ، هردو بازی های دو بعدی هستند . اما تصویر سمت راست ، بازی دو بعدی از نوع Platform است و تصویر سمت چپ یک نوع بازی از نوع ایزومتریک:

بنابراین بازیهای ایزومتریک ، مایه های طراحی سه بعدی نیز دارد ، هرچند که این تصاویر همه از نوع Sprite و دو بعدی هستند ولی چیزهایی مثل چگونگی قرار گرفتن کاراکتر ها در صحنه و همچنین اشیایی مثل جعبه ها در بازی سمت چپ ، سه بعدی طراحی شده اند . در واقع زاویه دید دوربین بازی ، فرق بین بازی های دو بعدی Platform با بازی های دو بعدی ایزومتریک است .

فصل دوم :

مقدمه ای در طراحی کاراکتر های ایزومتریک

بسیار خب . میدونم که منتظر شروع بحث بصورت جدی هستید . پس شروع میکنیم ..

کاراکتر ها :

کاراکتر ها یا همان شخصیت های اینگونه بازی ها ، نقش بسیار مهمی را در خوب جلوه دادن بازی ارائه میکنند . کاراکتر باید حرکاتی مناسب ، طراحی دقیق و زیبا و اندازه ای متناسب داشته باشد .

ما از نرم افزار های زیر برای ساخت شخصیتهای بازی خود استفاده میکنیم :

Adobe Photoshop (نسخه ۸ یا بالاتر)
MetaCreations Poser (نسخه ۴ یا بالاتر)

نرم افزارهای ذکر شده در بالا را یا اکثرا دارید یا به راحتی میتوانید در هر نرم افزار فروشی پیدا کنید .
نرم افزار Poser به این لحاظ انتخاب شد که میتواند در عرض چند دقیقه نمونه اولیه خوبی از شخصیت را به ما بدهد . البته این در صورتی است که عموما بازی شما با انسانها سروکار داشته باشد . ترسیم فریم به فریم موجودات عجیب و غریب بر میگردد به ذات هنری شما و اینکه چقدر در طراحی موفق هستید .
فعلا چون پروژه های موجود در این مقالات آموزشی هستند و میخواهیم روی طراحی بازی و نه طراحی کاراکتر ها کار کنیم ، عجالتا از Poser استفاده میکنیم .

من در Poser یک چنین شخصیتی آماده کرده ام :

آماده کردن چنین شخصیتی در Poser تنها به ۳ یا ۴ دقیقه وقت نیاز دارد . در صورتی که بخواهید خودتان آنرا نقاشی کنید ، تصور میکنید چند ساعت باید صرف آن کنید . پس میبینید که Poser فعلا چیز خوبی برای خلق تصاویر با کیفیت ایزومتریک است .

مقدمه ای در طراحی کاراکتر های ایزومتریک

Panel ها در 3D Gamestudio :

همانطور که میدانید ، برای استفاده از یک تصویر دو بعدی در محیط 3DGS ، نیاز است تا آنرا به عنوان یک Panel معرفی کنیم :

```
BMAP  my_bitmap = <my.bmp>;

PANEL  my_bitmap_panel
{
    bmap  = my_bitmap;
    layer = 2;
    pos_x = 10;
    pos_y = 10;
    flags = overlay,transparent,visible;
    alpha = 100;
}

PANEL*  my_panel = my_bitmap_panel;
```

اینها دستوراتی هستند که تصویر my.bmp را در مختصات ۱۰و۱۰ مانیتور نمایش میدهند .

همانطور که میدانید ، ذکر پارامتر overlay در جلوی عبارت flags ، این مفهوم را دارد که می خواهیم نقاط با رنگ سیاه (R=0 , B=0 , G=0) نمایش داده نشوند . در واقع نقاط سیاه رنگ در عکس my.bmp را شفاف کردیم . برای این منظور ، به عکس زیر دقت کنید :

این همان تصویر قبلی ماست که اینبار Background شخصیت را با کد رنگ سیاه پر کردیم . حالا اگر این تصویر را در 3DGS توسط پنلی که overlay باشد نمایش دهید ، قسمتهای سیاه رنگ دیده نخواهند شد و بجای آنها ، Background بازی را مشاهده خواهید کرد .

فصل دوم :

مقدمه ای در طراحی کاراکتر های ایزومتریک

بسیار خب . پس کار ما با مجموعه ای از عکس های ثابت با زمینه سیاه است .
حالا میخواهیم عمل راه رفتن را در این شخصیت تولید کنیم . برای این منظور ، به Poser بر میگردیم و با استفاده از قسمت Poses و زیر قسمت Walk Designer ، یک عمل راه رفتن را به شخصیت اعمال میکنیم . سپس از تمام این فریم ها به کمک گزینه Make Movie عکس تهیه میکنیم .

اکنون برای هر فریم ، یک عکس داریم . بعدا خواهید دید که چگونه با استفاده از دستورات C-Script میتوانیم به شخصیت خود حالت انیمیشن بدهیم .

در فصل بعد ، با نسخه اول دموی آموزشی این مجموعه مقالات آشنا خواهید شد .

فصل سوم :

آشنایی و کاربرد دموی آموزشی (نسخه ۱)

تصویر بالا ، نسخه اول دموی آموزشی این مجموعه است که میتوانید آنرا از طریق آدرس ذکر شده در تاپیک " **همه چیز درباره مقالات ایزومتریک** " در انجمن ساخت بازی با 3D Gamestudio دانلود کنید .تاپیک فوق را میتوانید در لینک زیر پیدا کنید:

http://www.persian-designers.com/forum/viewforum.php?f=37

در نسخه اول این دمو ، بازیگر تنها راه میرود ، از اشیاء عبور میکند و در کل تنها چند نکته حرفه ای در بازی وجود دارد . در نسخه های آتی که در شماره های بعدی این مقالات به بررسی آنها میپردازیم ، ویژگیهایی مهمتری از جمله : وجود دشمنان و هوش مصنوعی ، قابلیت مبارزه (مشت و لگد و پرش) و Health به آن افزوده خواهد شد . ولی برای شروع از تکنیک های بسیار مهم Scrolling و همچنین Map Browsing استفاده میکنیم . همچنین یک تکنیک ابتکاری در مواجهه با مشکل Layer ها یا همان لایه ها نیز شرح داده خواهد شد .

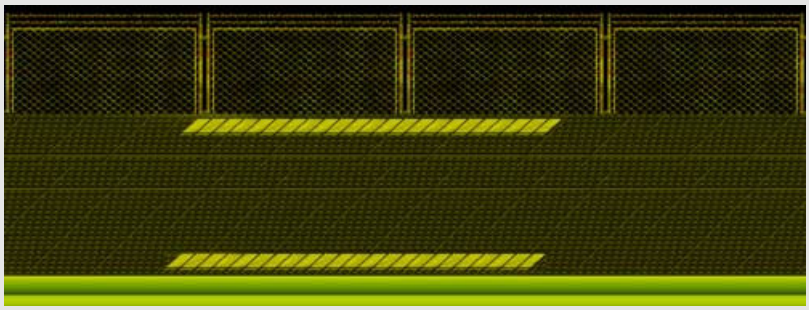
برای ادامه توصیه میکنم ابتدا فایل دمو را دانلود و اجرا کنید تا مفاهیمی که در ادامه بحث توضیح داده میشوند قابل درک باشند .

background Scrolling (1)

حرکت دادن به Background توسط روش Scroll-Loop :

Background یا پس زمینه این دمو تنها از دو تصویر زیر تشکیل شده است :

تصویر بالا ، تصویری از شهر است که در بسته دمو با نام city1.bmp با اندازه 800 x 600 پیکسل ایجاد شده است .



این تصویر هم در واقع جاده بازی و به نوعی پل است و با نام road1.bmp و اندازه 800 x 300 در بسته دمو قرار دارد .

همانطور که میبینید ، هر دو تصاویر قابلیت این را دارند تا بصورت حلقه وار و نامتناهی Scroll شوند . وقتی دستور حرکت بازیکن صادر میشود ، بصورت عمومی تصویر جاده یا پل به سمت مخالف جهت بازیکن Scroll میشود و همزمان با آن ، تصویر شهر با نصف سرعت حرکت پل به همان جهت مخالف حرکت میکند . اگر دمو را اجرا کنید ، میبینید که حرکتی بسیار نرم و نزدیک به واقعیت ایجاد میشود .

background Scrolling (2)

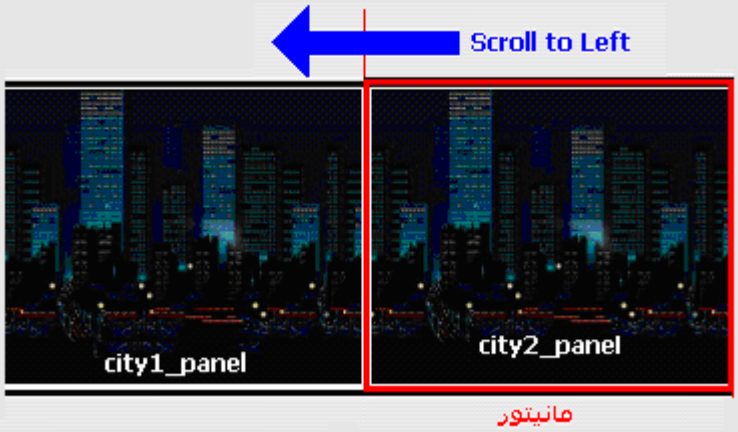
حرکت تصاویر پس زمینه رابطه نزدیکی با حرکات و حالات بازیکن دارد . همانند بازی اصلی ، وقتی بازیکن قسمتی از مسیر را طی کرد ، دیگر نمیتواند به عقب برگردد و در منتها علیه سمت چپ صفحه نمایش متوقف میشود .
در همین حال ، بازیکن باید حداقل تا نصف صفحه نمایش ، خودش به جلو حرکت کند تا پس زمینه Scroll شود .

ممکن است کمی گیج کننده باشد ، ولی با اجرای دمو ، دقیقا همین حالت را مشاهده خواهید کرد .

بررسی تئوری حرکت دهنده پس زمینه :

ابتدا بهتر است به تئوری اینکار بپردازیم . به تصویر زیر دقت کنید :

برای حرکت نامتناهی ، به دو تصویر یکسان نیاز داریم . تصویر اول به طور کامل روی صفحه نمایش قرار دارد و تصویر دوم دقیقا بعد از تصویر اول .
با فرض اینکه اندازه صفحه نمایش 800 x 600 پیکسل باشد ، الگوریتم کار به این صورت است که اگر مختصات X تصویر دوم برابر (۰) شد ، به این معناست که تصویر دوم به محل تصویر اول رسیده ، بنابراین چون اسکرول ادامه پیدا میکند ، باید تصویر اول را به محل قبلی تصویر دوم منتقل کنیم :



اینکار برای مختصات X تصویر اول هم چک میشود و در صورتی که این مختصات به (۰) برسد ، مختصات X تصویر دوم به ۸۰۰ تغییر میکند .
در واقع یک جابجایی ساده اما حساب شده باعث میشود که تا ابد این تصاویر با هم جابجا شود و چون نوع طراحی آنها به گونه ای است که میتوانند پشت سر هم قرار بگیرند ، کاربر عملا متوجه ایت تغییرات نمیشود . او فقط میبیند که در یک شهر در حال عبور است .

همین تئوری برای تصویر پل یا جاده هم صدق میکند .

اکنون با هم کد C-Script این الگوریتم را میبینیم . بعد از کد ، من توضیحات لازم را میدهم :

```
bmap back01_bmap=<city1.bmp>;

panel background_city1 {
	bmap = back01_bmap;
	layer=4;
	pos_x=0;
	pos_y=0;
	flags=visible;

}
panel* city1=background_city1;
panel background_city2 {
	bmap = back01_bmap;
	layer=4;
	pos_x=800;
	pos_y=0;
	flags=visible;

}
panel* city2=background_city2;
function scroll_city_front()
{
	if(city2.pos_x==-0.5){city1.pos_x=800;}
	if(city1.pos_x==-0.5){city2.pos_x=800;}
	city1.pos_x-=0.5;
	city2.pos_x-=0.5;

	wait(1);
}
```

قطعه برنامه بالا ، دقیقا همین تکنیک را پیاده میکند . در ساختار panel چون نیازی به شفافیت نقاط تاریک تصویر شهر نداشتیم ، از عبارت overlay در جلوی =flags خبری نیست .

در ادامه یک اشاره گر برای پس زمینه شهر ایجاد کردیم .

در داخل تابع ، سرعت حرکت پس زمینه شهر برابر 0.5 حرکت است که این سرعت را برای برای جاده یا پل برابر 1 قرار داده ایم . زیرا همانطور که گفتیم ، سرعت حرکت پس زمینه شهر نصف سرعت حرکت تصویر جلویی یا همان جاده است . علت هم مشخص است : شهر دور تر از دید ما قرار دارد و جاده به ما نزدیکتر است .

کد مربوط به حرکت جاده یا پل هم مشابه همین کد است . ببینید :

background Scrolling (4)

```
bmap back02_bmap=<road1.bmp>;

panel background_road1 {
      bmap = back02_bmap;
      layer=5;
      pos_x=0;
      pos_y=300;
      flags=visible,overlay;

}
panel* road1=background_road1;

panel background_road2 {
      bmap = back02_bmap;
      layer=5;
      pos_x=800;
      pos_y=300;
      flags=visible,overlay;

}
panel* road2=background_road2;

function scroll_road_front()
{
      if(road2.pos_x==0){road1.pos_x=800;}
      if(road1.pos_x==0){road2.pos_x=800;}
      road1.pos_x-=1;
      road2.pos_x-=1;
      if(boxg1_status==1){boxg1_panel.pos_x-=1;}
      if(bar1_status==1){bar1_panel.pos_x-=1;}
      if(stp1_status==1){stp1_panel.pos_x-=1;stp2.pos_x-=1;}
      wait(1);
}
```

فعلا به خطوط قرمز رنگ کاری نداشته باشید و آنها را از الگوریتم Scrolling جدا کنید . همین خطوط قرمز رنگ مربوط به تکنیک Map Browsing است که در فصل بعد به آنها میپردازیم .

اکنون هر بار که توابع `scroll_road_front()` و `scroll_city_front()` اجرا شوند ، جاده و شهر به مقدار واحد های تعریف شده به سمت چپ حرکت میکنند .

در فصل بعد ، به مفاهیم بیشتری از تکنیک اسکرولینگ و همچنین Map Browsing میپردازیم .

فصل چهارم :

Map Browser (1)

Map Browser به معنای لغوی یعنی مرورگر نقشه . این نام تکنیکی است که نقشه دو بعدی ما را واقعا خلق میکند . البته منظورم از نقشه ، همان محیط خیابانی است که از آن عبور میکنیم .

وظیفه اصلی این تکنیک ، ایجاد و مدیریت اشیایی است که بازیکن در طول بازی با آنها مواجه میشود .

برای درک بهتر ، توضیحات بیشتری را روی تصویر دمو میدهم :

همانطور که در تصویر مشاهده میکنید ، علاوه بر بازیکن ، دو لایه دیگر هم وجود دارند که از تصاویر پس زمینه جدا هستند و کاملا مستقل عمل میکنند . اما وقتی دمو را اجرا میکنید ، این دولایه که توسط تابع Map Browser اداره میشوند ، چسبیده و همراه با پل حرکت میکنند . در نگاه اول ممکن است اینطور به نظر رسد که طراحی آنها همراه با پل بوده ، ولی این دو تصویر کاملا جدا هستند . مثلا شکل شئی مانع که در وسط پل قرار گرفته ، ممکن است خیلی جلوتر باشد و میتوان به راحتی با تغییر متغییرهایی خواص ، جای آن را بر روی پل به راحتی عوض کرد .

برای این منظور ، لازم است تا بدانیم کجای نقشه قرار گرفتیم تا به موقع شئی مورد نظر را جلوی بازیکن ایجاد کنیم .

متغییر **map_pos** ، همواره محل فعلی خیابان را با استفاده از یک عدد صحیح به ما نشان میدهد . مقدار این متغییر در دمو ، همانطور که در تصویر میبینید ، در گوشه بالا و سمت چپ صفحه نمایش دیده میشود . اشیایی مثل مانع در وسط پل و همچنین ستون فلزی موجود در بازی و جعبه ها که همگی را در تصویر بالا میبینید ، توسط مقدار همین متغییر ظاهر شده اند .

اساس کار مقدار دهی به این متغییر به این صورت است که با هر Scroll پس زمینه به جلو ، به مقدار این متغییر افزوده میشود و در واقع برنامه نویس دقیقا میداند که بازیگر تا کجا در نقشه جلو رفته است.

Map Browser (2)

من توضیح میدهم که چگونه تکنیک Map Browser میتواند این کار را انجام دهد .

برای این کار از شئی مانع استفاده میکنیم :

این شئی ، در واقع تصویری به شکل زیر است و با نام stp1.bmp در بسته دمو قرار دارد :

کد های زیر ، Panel های لازم را برای این تصویر ایجاد میکنند :

```
panel stp1_panel {
        bmap = back05_bmap;
        layer=9;// back of player layer
        pos_x=750;
        pos_y=350;
        flags=visible,overlay;
}
panel* stp1=stp1_panel;
panel stp2_panel {
        bmap = back05_bmap;
        layer=11;// forward to player layer
        pos_x=750;
        pos_y=350;
        flags=visible,overlay;
}
panel* stp2=stp2_panel;
```

همانطور که میبینید ، ما دو پنل برای این تصویر و در واقع شئی ایجاد کردیم . فکر میکنید علت اینکه دو پنل برای اینکار لازم است ، چیست ؟

مساله لایه ها :

همانطور که میدانید ، در قانون پنل ها ، آن پنلی در جلو قرار میگیرد که عدد پارامتر Layer آن بزرگتر باشد . اکنون برای درک بیشتر این مساله ، به دو تصویر زیر دقت کنید :

Map Browser (3)

همانطور که میبینید ، طبق قانون طبیعی هر بازی ، اگر بازیکن جلوی شئی قرار بگیرد ، شئی در پشت آن دیده میشود و اگر بازیکن در پشت شئی برود ، آن شئی در جلوی آن قرار میگیرد .

شاید اگر میشد در زمان Runtime (زمان اجرای بازی) پارامتر Layer از یک Panel را تغییر داد ، این مشکل به سادگی رفع میشد . اما میدانید که این کار امکان ندارد و ما مجبوریم دو پنل از یک شکل را دقیقا روی هم قرار دهیم .

با این تفاوت که پارامتر Visible یکی از آنها همواره خاموش یا Off باشد . پنل اول در لایه پشتی بازیکن قرار دارد و پنل دوم در لایه جلویی . هنگامی که بازیکن طبق مختصات خودش و شئی مورد نظر ، جلوی آن قرار میگیرد ، پنلی که لایه آن پائین تر از بازیکن است روشن میشود و بلعکس ، وقتی بازیکن در پشت شئی قرار میگیرد ، پنلی که لایه آن بالاتر از بازیکن است روشن میشود .

بنابراین مساله لایه ها را میتوان به این ترتیب حل کرد .
چک کردن اینکه بازیکن در کجای شئی قرار دارد ، برعهده خود شئی است . من تابعی برای stp1.bmp نوشتم که مدام این موضوع را چک میکند و الگوریتم بالا را اجرا میکند .

نام این تابع stp1_browser است که به صورت نامتناهی اجرا میشود . این تابع باید درست هنگام شروع بازی اجرا شود .

کد های این تابع را در صفحه بعد ببینید تا توضیحات تکمیلی را بدهم :

Map Browser (4)

```
var stp1_status=0;

function stp1_browser()
{
	while(1)
	{
		if(map_pos>0 && map_pos<1000)
		{
			stp1_status=1;// move to left

			} else {
			stp1_status=0;// stopped
		}

		if(player.pos_y<stp1.pos_y-80)
		{
			stp2.visible=on;
			stp1.visible=off;wait(0);
		}
		if(player.pos_y>stp1.pos_y-80)
		{
			stp1.visible=on;
			stp2.visible=off;wait(0);
		}
		wait(0);
	}
	wait(0);
}
}
```

همانطور که میبینید ، باز هم قسمتهای قرمزرنگی از تابع وجود دارند که مربوط به مبحث map_pos میشوند که در ادامه توضیح خواهم داد .
قسمتهای سبز رنگ ، مسئول رسیدگی به مساله لایه ها هستند . همانطور که میبینید ، باروشن و خاموش کردن به موقع پنل های stp1 و stp2 ، بازیکن میتواند به پشت و جلوی شئی مانع حرکت کند .

بسیار خب . برگردیم به مبحث اصلی که همان Map Browser بود .
قسمتهای قرمز رنگ تابع فوق ، یک متغییر عمومی به نام stp1_status را مقدار دهی میکنند . این مقدار دهی بر اساس مقدار فعلی متغییر map_pos که موقعیت نقشه خیابان را مشخص میکند ، مقدار دهی میشود . در طول برنامه ، متوجه میشوید که هر گاه متغیر stp1_status برابر 1 بود ، شئی stp1 همراه با اسکرول پل حرکت میکند و هرگاه مقداری برابر 0 داشت ، از حرکت باز می ایستد . سرعت حرکت stp1 نیز باید برابر با سرعت حرکت پل یا جاده باشد چون اصولا شئی ثابتی است و از خود حرکتی ندارد .

اگر به دستور IF قسمت قرمز رنگ دقت کنید ، میبینید که محدوده حرکتی 0 تا 1000 برای شئی stp1 در نظر گرفته شده است . این عبارت به آن معناست که شئی stp1 تنها زمانی مجوز حرکت دریافت میکند که نشانگر نقشه خیابان عددی بین 0 تا 1000 باشد .

Map Browser (5)

در واقع هنگامی که بازیکن به مقدار 1000 واحد map_pos در خیابان به جلو حرکت کند ، شئی stp1 همراه با حرکت پل از صفحه نمایش خارج میشود و چون دیگر نیازی به حرکت آن نیست ، متغییر حرکتی آن یعنی str1_status برابر با 0 میشود .

اما ببینیم متغییر stp1_status چگونه شئی stp1 حرکت میدهد یا از حرکت آن جلوگیری میکند ؟

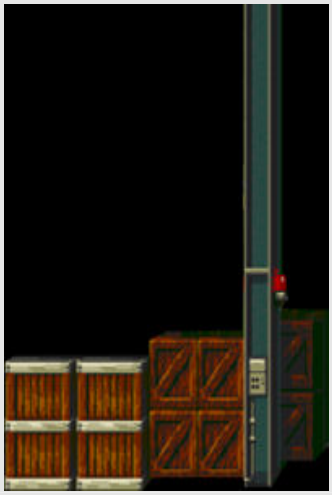
گفتیم که شئی stp1 همراه با پل یا جاده حرکت میکند . پس بازمیگردیم به تابع اسکرول کننده پل و دوباره آنرا مرور میکنیم :

```
function scroll_road_front()
{
	if(road2.pos_x==0){road1.pos_x=800;}
	if(road1.pos_x==0){road2.pos_x=800;}
	road1.pos_x-=1;
	road2.pos_x-=1;
	if(boxg1_status==1){boxg1_panel.pos_x-=1;}
	if(bar1_status==1){bar1_panel.pos_x-=1;}
	if(stp1_status==1){stp1_panel.pos_x-=1;stp2.pos_x-=1;}
	wait(1);
}
```

همانطور که قبلا هم گفتم ، قسمتهای قرمز رنگ مربوط به حرکت اشیاء روی پل یا جاده هستند . خط قرمز سوم که مشخص تر از بقیه است ، مجوز حرکتی شئی stp1 را چک میکند و در صورتی که اجازه حرکت داشت ، انرا همراه با اسکرول پل ، حرکت میدهد . البته هم stp1 و هم stp2 را چون همانطور که دیدیم ایندو پنل هر دو درواقع یکی هستند و باید همیشه مختصات آنها دقیقا برابر هم باشند تا مساله لایه ها را به خوبی حل کنند .

بقیه خطوط قرمز هم دقیقا کاری مشابه با خط سوم را انجام میدهند ، منتها با اشیاء دیگر .

مثلا خط قرمز اول ، مجوز حرکتی شئی boxg1 را چک میکند که همان شکل زیر است :



Map Browser (6)

یا خط دوم که ستون جلوتر در خیابان را مدیریت میکند :

میتوانید در یک جستجوی خیلی ساده در متن کد برنامه دمو ، توابع bar1_browser و boxg1_browser که برای کنترل و مرور این اشیاء نوشته شده اند را پیدا کنید .

در اینجا ، مبحث Scrolling را به پایان میرسانیم . در فصل بعد به چگونگی انیمیشن کردن بازیکن میپردازیم .

فصل پنجم :

Character Animation (1)

اگر دمو را اجرا کنید ، میبینید که حرکات راه رفتن بازیکن ، نرم و طبیعی است . به لطف وجود نرم افزاری مثل poser و سایر نرم افزار های سه بعدی که میتوان از آنها عکس تهیه کرد ، دیگر ساخت کاراکتر های دو بعدی انیمیشنی چندان سخت نیست . اما مشکل وقت گیر بودن آنها همچنان پابرجاست .

در ابتدای این مقاله گفتم که کاراکتر این دمو با نرم افزار Poser ساخته شده است . کار با این نرم افزار بسیار ساده و نحوه استفاده از آن را میتوان در عرض چند ساعت فرا گرفت .

مدل کاراکتر ما ، با استفاده از مجموع 34 فریم و عکس حرکت میکند . اگر زیاد به نرم راه رفتن کاراکتر اهمیت نمیدهید ، میتوانید از تعداد فریم های راه رفتن آن کم کنید و همه چیز را با 10 فریم جمع و جور کنید .

برای انیمیشن کردن فریم ها ، ابتدا تمام فریم ها را با استفاده از دستورا ت تعریفی BMAP تعریف میکنیم . در زیر بخشی از این تعاریف که در دمو وجود دارد را آورده ام :

```
bmap wr0001=<wr_0001.bmp>;// walk to right frames
bmap wr0002=<wr_0002.bmp>;
bmap wr0004=<wr_0004.bmp>;
bmap wr0006=<wr_0006.bmp>;
...
bmap wl0001=<wl_0001.bmp>;// walk to left frames
bmap wl0002=<wl_0002.bmp>;
bmap wl0004=<wl_0004.bmp>;
bmap wl0006=<wl_0006.bmp>;
...
```

بعد از تعریف فریم ها ، نوبت به توابعی میرسد که در حین حرکت بازیکن به جلو یا عقب ، این فریم ها را با سرعت مناسب عوض کند و حالت انیمیشن را بوجود آورد .

تابع player_walk_right() تابعی است که بازیکن را همراه با انیمیشن به سمت راست صفحه نمایش که همان جلو در خود بازی است ، حرکت میدهد . این تابع را میتوانید در کد برنامه دمو پیدا و مشاهده کنید .(به علت طولانی بودن نسبی ، در این مقاله متن کامل آن آورده نشده است) .

در داخل این تابع ، متغیری به نام player_pos_status مقدار دهی میشود که حالت راست یا چپ بودن بازیکن را نگه میدارد . این در زمانی مورد استفاده پیدا میکند که بازیکن بخواهد با سمت بالا یا پایین حرکت کند . چون ما فریمی برای حالا بالا و پائین نداریم ، پس در هنگام بالا رفتن یا پایین آمدن ، باید بدانیم که بازیکن در چه سمتی قرار گرفته . راست یا چپ ؟
مثلا اگر بازیکن در حال حرکت به راست است و کاربر بخواهد در قسمتی از خیابان به سمت بالا برود ، چون متغیر player_pos_status برابر 1 یعنی راست است ، بنابراین تابع بالا رفتن بازیکن از فریمهای حرکت به سمت راست بازیکن استفاده میکند و بازیکن را به بالا هدایت میکند .

بطور خلاصه متغیر player_pos_status آخرین وضعیت سمت چپ یا راست بازیکن را برای استفاده در توابع player_walk_up() و player_walk_down() نگه داری میکند که کاربردی کاملا موثر و واقعی دارد .

دو متغیر داخلی f و ff نیز وجود دارند که به ترتیب شماره فریم و سرعت انیمیشن را مدیریت میکنند .
در ادامه تابع میبینید که عباراتی مثل زیر ، فریم هارا در مورد پنل player که پنل اصلی بازیکن است تعویض میکنند . این تعویض ها بر اساس مقدار فعلی شماره فریم یا همان متغیر f انجام میشود :

Character Animation (2)

```
//- walking frame changes :
if(f==1){player.bmap=wr0001;}
if(f==2){player.bmap=wr0002;}
if(f==3){player.bmap=wr0004;}
if(f==4){player.bmap=wr0006;}
if(f==5){player.bmap=wr0008;}
if(f==6){player.bmap=wr0010;}
if(f==7){player.bmap=wr0012;}
……
….
..
```

همانطور که میبینید ، پارامتر BMAP از پنل player جای خود را به فریم های جدید میدهد و حالت انیمشن پدید می آید .

از همین روش میتوان برای انیمشن کردن هر چیزی که در بازی مورد استفاده قرار میگیرد استفاده کرد و به علت اینکه فقط از تصاویر و پنل ها استفاده میکنید ، تغییری در سرعت بازی ایجاد نمیشود و میتوان بر خلاف بازی های سه بعدی که کثرت اشیای سه بعدی باعث افت سرعت بازی میشود ، در این نوع بازیها به تعداد زیادی شئی انیمیشن دو بعدی ایجاد کرد .

بسیار خب دوستان .

در این شماره از مقاله ، به برخی از مفاهیم اینگونه بازیها پرداختیم . دیدیم که نیاز نیست برای یک بازی مثل STREETS Of RAGE حتما نقشه های طولانی کشید و خیابانهای عظیمی طراحی کرد تا بازیکن به یک شئی برسد زیرا با استفاده از تکنیک Map Browsing اینکار عملا منتفی است .
همچنین دیدیم که چطور با استفاده از دو پنل بجای یک پنل ، مساله لایه ها را در بازیهای ایزومتریک حل کرد و کاری کرد تا بازیکن دور یک شئی را بصورت طبیعی طی کند .
همینطور به روش ساده انیمیت کردن کاراکتر ها با استفاده از فریمهای ثابت پرداختیم .
در شماره های آتی از این سلسله مقالات ، می آموزیم که چگونه برخورد بازیکن با یک شئی را در خیابان ثبت کنیم و برای آن تابعی بنویسیم . همچنین به دشمنان در بازی های خیابانی و هوش مصنوعی ساده بکار رفته در آنها میپردازیم و میبینیم که چطور میشود با دشمنان دو بعدی ، جنگهای تن به تن داشت .
اصول مشت زنی ، پرش ، کتک خوردن و غیره ... از نکاتی است که الگوریتم و تکنیکهای آنها در شماره آینده این سلسله مقالات به شما معرفی میشوند .

دوستان میتوانند از طریق لینک زیر و ورود به تاپیک " **همه چیز در مورد مقالات ایزومتریک** " اقدام به دانلود دموی آموزشی این مقاله کنند :

http://www.persian-designers.com/forum/viewforum.php?f=37

همچنین در این تاپیک میتوانید راهنمای انگلیسی کاملی در مورد Panel ها را دانلود کنید .

موفق و پاینده باشید
ستاره الوند پور
Pacificsys Games

APPENDIX (1)

Pacificsys Games Products Review

Pacificsys Games Products: (For PC)

JigsawLands 1.0 SE (Build 6.2)

پازل های معروف Jigsaw اینبار با روایتی نو!
دارای ۸ پازل داخلی و قابل توسعه تا بینهایت بازی . توسعه مراحل رایگان.
30 قطعه و مناسب برای کودکان ۶ سال به بالا . مجهز به امکاناتی نظیر جان ، نقشه پازل ، مرورگر قطعات پازل ، کلید کمکی ، نمایش دهنده میزان شفافیت تصویر اصلی برای کمک به کودک ، زمانسنج بازی و ...
قابلیت ایجاد پازل های دلخواه به رایگان توسط Pacificsys Games یا خود کاربران همراه با آموزش رایگان.
نصب سریع و آسان به زبان فارسی .
قیمت برای ارسال به سراسر ایران فقط :
۲۰۰۰ تومان !
امکان خرید از طریق تماس تلفنی

STARWARS : The Start Wave

(BETA)

سری جدید مبارزه های Invaders را با محصول جدید Pacificsys games تجربه کنید !
اینبار جنگ های ستاره ای بر سر دزدیدن گوشت از زمینی ها توسط خفاشهای فضایی است .
یک بازی واقعا سرگرم کننده با مراحل متعدد و فانتزی ، با طراحی محکم و رعایت استاندارد های بازیسازی جهانی که میتواند شما را مدتها پشت کامپیوتر نگه دارد .

بزودی منتشر میشود ...

Pacificsys-Games-Shopping@yahoo.com

APPENDIX (2)

How to play SEGA GENESIS games ?

دوستانی که مایل به اجرای بازی معروف سگا ، Streets of rage 3 هستند میتوانند از طریق سایت زیر ، برنامه شبیه ساز Sega genesis را دانلود و نصب کنند . سپس از طریق همین سایت میتوانند ROM بازی Streets Of rage 3 را پیدا و دانلود کنند .

Http://www.CoolRom.com

بازی های SEGA Genesis با پسوند .BIN هستند که به راحتی توسط شبیه ساز اجرا میشوند . پس از اجرای شبیه ساز میتوانید در صورتی که جوی استیک دارید ، آنرا بازی کنید وگرنه توسط امکانات خود شبیه ساز ، جوی استیک را روی کیبورد کامپیوتر خود به راحتی تعریف و بازی کنید .

در زیر ، تصویری از اجرای بازی شورش در خیابان را در برنامه شبیه ساز سگا میبینید :

APPENDIX (3)

SEGA : Streets Of Rage 3 Cheat Codes !

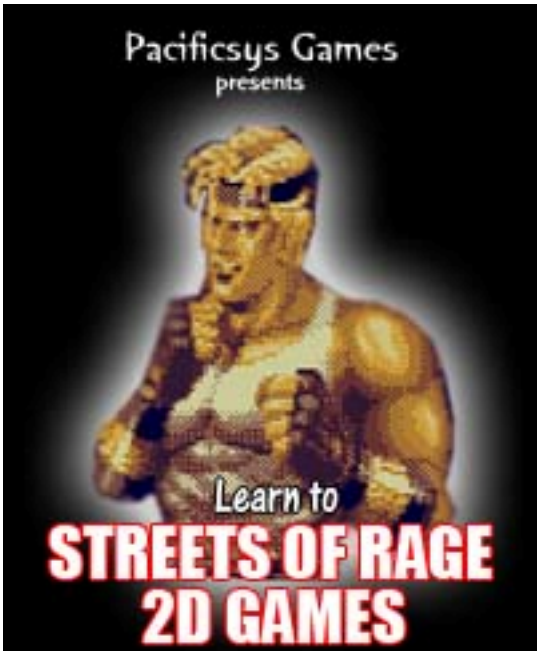
با استفاده از کد های نسوز کننده زیر ، میتوانید با خیال راحت بازی را تا انتها اجرا کنید !

CODE : **AF9A-APTW**
NAME : **damin mance**

CODE : **AF9A-APTW**
NAME : **echo-pix**

این کد ها را میتوانید در بخش **File > Game Genie** وارد و فعال کنید تا بازی نسوز شود .

APPENDIX (4)
About



عنوان انگلیسی :
Learn to Streets of rage 2d games
عنوان فارسی :
آموزش ساخت بازی های دو بعدی ایزومتریک
(با الهام از بازی Streets of rage 3)
پایه آموزشی : متوسط
جلد : اول
تعداد صفحه : ۲۳ صفحه همراه با تصاویر رنگی
نویسنده : ستاره الوند پور
صاحب امتیاز : Pacificsys Games
ناشر : Persian-Designers.Com
نوع فایل کتاب : Adobe PDF format
نوع امتیاز : رایگان همراه با ذکر Credist
سال انتشار : ۱۳۸۵ / 2006

همراه با فایل دموی آموزشی