بنام مالک یوم الدین

جلسه هشتم : آشنایی با زبان برنامه نویسی GML

سطح آموزش : سخته !

پیش نیاز : یاد داشتن یک زبان برنامه نویسی مفیده!

میزان زمان لازم برای یادگیری : ۵ ساعت

داشتن زبان برنامه نویسی مستقل در گیم میکر یکی از ویژگی های مهم اون هست که بعضی امکاناتی که توی اکشن ها نیستن رو بتونیم پیاده سازی و ازش استفاده کنیم.

یادگیری هر زبان برنامه نویسی معمولا دارای یه پروسه هست که به صورت زیر فهرست شده است :

۱- تاریخچه زبان برنامه نویسی

۲- آشنایی با قواعد نوشتاری

۳- آشنایی با عملگرها و متغیرها و انواع آن ها

۴- آشنایی با ساختار های کنترلی زبان برنامه نویسی

۵- برنامه نویسی ساخت یافته در زبان برنامه نویسی

۶- برنامه نویسی شی گرا در زبان برنامه نویسی

در این آموزش ما به مورد های ۱ و ۲ و ۳ و ۴ میپردازیم, در ادامه هم چندتا تابع بدرد بخور رو معرفی میکنیم.

تاریخچه و معرفی GML

زبان GML در سال 1999 توسط شرکت یویو گیمز (YOYO Games) ساخته شد. این زبان ترکیبی از زبان های برنامه نویسی C++ و جاوا و دلفی هست. داشتن زبان برنامه نویسی مخصوص برای یک موتور بازی سازی دارای فواید و معایبی هست. از فواید آن میتوان به منحصر شدن زبان برنامه نویسی برای بازی سازی و داشتن توابع مفید برای بازی سازی اشاره کرد. عیب این زبان فقط این است که به قدرتمندی زبان های شی گرا مثل C++ , C# و جاوا نیست.

قواعد کدنویسی در GML

برای کدنویسی به زبان GML پنجره یک شی را باز کنین و در قسمت اکشن ها از سربرگ control قسمت Code اکشن قطعه کد را درگ دراپ کنین. حالا می تونیم کد GML بنویسیم. درحین کدنویسی باید نکات زیر رو مد نظر داشت :

۱- انتهای هر دستور بهتر است به (;) ختم شود.

۲- زبان GML به حروف بزرگ و کوچک حساس است.

۲- برای گذاشتن کامنت و یادداشت برای اینکه مثلا عملکرد کدها رو توضیح بدین ۲ تا راه حل وجود داره که در زیر آن را مشاهده می کنین:

```
// first comment
/*
second comment
*/
```

آشنایی با متغیر ها و عملگرها در GML

در ابتدا باید با مفهوم متغیر آشنا بشیم و اون موقع وارد GML میشیم. متغیرها در اصل چند خونه از حافظه (Ram) هستند که ما داخلش طبق نیازمون مقدار میریزیم و باهاش کار می کنیم. هر متغیری اسم داره که خودمون تعیین می کنیم و با اسمش می تونیم به خونه حافظه اش دسترسی پیدا کنیم. توی GML چهار نوع داده ای داریم که وقتی یک متغیر رو پر می کنیم طبق مقداری تعیین شده نوع داده ایش در نظر گرفته میشود.

۱- عدد صحیح :

```
speed_person = 10;
```

۲- عدد اعشاری :

```
speed_missile = 5.23;
```

۳- رشته کاراکتری :

```
message = "Hello! Wellcome";
```

۴- مقدار منطقی :

```
isVissible = true; // or false
```

برای نامگذاری متغیرها نکات زیر را باید رعایت کنیم :

۱- نام متغیر حتما باید با حرف شروع شود.

۲- برای نامگذاری متغیرها می توانید از حروف کوچک و بزرگ و اعداد و کاراکتر (_) استفاده کنید.

۳- نام متغیرها نباید با اسم اسپرایت ها , صدا ها , اشیا , اتاق ها و ... یکی باشد.

۴- از کلمات کلیدی زبان GML نباید به عنوان نام متغیر استفاده شود.

در زبان GML می توانیم از عملگر های +(جمع کننده و چسباننده) -(تفریق) *(ضرب) /(تقسیم) استفاده کنیم.

کاربردهای عملگر + :

```
a = 2;
b = 3;
sum = a + b; // 5
//---------------------------
text1 = "Hello ";
text2 = "Ali";
text3 = text1 + text2; //Hello Ali
```

یک کاربرد جالب عملگرها در ساده سازی دستورات است :

```
sum = sum + 50; // sum += 50;
```

محدوده تعریف شده متغیرها : هر متغیری که تعریف می کنین در همان محدوده ای که تعریف شده است قابل استفاده هستند. مثلا در کد های بالا متغیر text3 فقط در اکشن قطعه کد مربوطه قابل استفاده است و خارج از آن تعریف نشده است. برای اینکه به یک متغیر در تمام آبجکت ها دسترسی داشته باشیم از کلمه کلیدی global به صورت زیر استفاده می کنیم :

```
global.health = 100;
global.health += 20;
```

آشنایی با ساختار های شرطی و تکرار در GML

ساختار شرطی if

ساختار کلی به شکل زیر میباشد :

```
if ( شرط یا شروط )
{
دستورات
}else{ دستورات۲ }
```

گفتار فارسی : اگر (شرط یا شروط برقرار بود) آنگاه { دستورات } را اجرا کن در غیر اینصورت { دستورات۲ } را اجرا کن

* قسمت else الزامی نیست و با توجه به نیاز ممکن هست در ساختار باشد یا نباشد.

چه جوری شرط و شروط بذاریم؟

شرط و شروط گذاشتن توی هر زبان برنامه نویسی به شکل مقایسه مقداری ۲ عدد یا مقایسه منطقی یک متغیر هست.

مقایسه مقداری ۲ عدد :

برای مقایسه عملگر داریم : > کوچکتر => کوچکتر مساوی < بزرگتر <= بزرگتر مساوی == مساوی بودن =! مساوی نبودن

مثال :

```
if ( x > 2 ) {…}
if ( y <= 20 ){…}
if ( player.speed == 5 ){…}
if ( a != 20 ){…}
```

مقایسه منطقی یک متغیر:

هر جا حرف از متغیر منطقی , مقایسه منطقی و عملگر منطقی داریم یعنی منظورمون اینه که میخواهیم با متغیر هایی کار کنیم که مقدار داخلشون true هست یا false .

برای مقایسه منطقی یک عملگر داریم : ! برعکس کردن (یعنی اگر true میشه false و برعکس)

مثال :

```
a = true;
b = false;
if ( a ){…}
if ( a == true ){…}
if ( !b ){…}
```

نوشتن چند شرط :

برای نوشتن چند شرط برای if ما ۲ تا عملگر داریم که ارتباط بین شروط رو نشون میده و با توجه به نیاز بین شرط هامون باید بذاریم.

عملگر && که معنای " وَ " رو میده و عملگر || معنای " یا " رو میده.

مثال :

اگر a درست باشد و y هم کمتر از ۲۰ باشد آنگاه فلان دستورات را اجرا کن.

```
if ( a == true && y < 20 ){…}
```

اگر سرعت player بیشتر از ۲۰ باشد یا مشخصه x آن کمتر مساوی ۱- باشد آنگاه فلان دستورات را اجرا کن.

```
if ( player.speed > 20 || player.x <= -1 ){…}
```

ساختار switch :

```
switch ( متغیر ){
case مقدار :  دستورات
      break;
case مقدار :  دستورات
      break;
case مقدار :  دستورات
      break;
default :  دستورات
}
```

مثال :

```
switch(n)
{
case  1 : room_goto(1);
  break;
case  2 : room_goto(2);
  break;
case  3 : room_goto(3);
  break;
case  4 : room_goto(4);
 break;
case  5 : room_goto(5);
  break;
default :  game_end();
}
```

در مثال بالا روی متغیر n سوییچ انجام دادیم و با توجه به مقادیری که می تواند n بگیرد به جای چند if از سوییچ استفاده کردیم. همون طور که می بینید برای مقدار های ۱ تا ۵ دستورات نوشتیم اما پیش بینی آنکه اگر n هیچکدام از مقادیر ۱ تا ۵ نباشد هم کردیم با کمک default .

ساختار حلقه شرطی while :

```
while ( شرط یا شروط ){
دستورات
};
```


گفتار فارسی : تا زمانیکه (شرط یا شروط برقرار است) { دستورات } را اجرا کن.

قسمت شرط و شروط مثل ساختار if هست.

مثال :

```
while (z < 20){
z = z + 1 ;
};
```

حتما از حلقه طوری استفاده کنید که در حالتی شکسته شود و گرنه یک حلقه بی نهایت داریم و اون موقع سیستم عامل یا گیم میکر جلوی شما را میگیرد یا بازی هنگ میکند.

ساختار حلقه شرطی do :

```
do{
دستورات
}until( شرط یا شروط );
```

تفاوت این حلقه با while در این هست که در حلقه do دستورات حداقل یکبار اجرا میشوند.

ساختار حلقه شرطی repeat :

```
repeat( تعداد تکرار ){
دستورات
}
```

مثال :

```
repeat( 3 ){
	player.speed += 1;
}
```

ساختار حلقه شرطی for :

```
for( شماره شروع ; شرط پایان حلقه ; گام افزایش یا کاهش ){
دستورات
}
```

مثال :

```
for( i = 0 ; i < 5 ; i += 1 ){…}
```

حلقه بالا از i = 0 شروع میکنه و با افزایش یک گام در هر مرحله (i += 1) با شرط i < 5 متوقف میشود , در کل دستورات ۵ بار اجرا میشوند.

ساختار تعریف شده With توسط GML :

```
with( آبجکت یا کلمات کلیدی ){
دستورات
}
```

مثال:

```
with (obj_enemy){
instance_destroy();
}
```

دستور بالا میگه برای تمامی آبجکت هایی به نام obj_enemy دستور ()instance_destroy رو اجرا کن.

جای obj_enemy میتونیم یه سری کلمات کلیدی که GML برامون گذاشته رو بزاریم و استفاده کنیم:

all : تمامی اشیای داخل اتاق

self : خود شی ای که این دستورات داخلش نوشته میشه

other : جسم دیگری (کاربردش برای اونت Collision هست)

solid : اشیایی که خاصیت solid رو دارن.

id : یک شماره شش رقمی برای هر آبجکتی که توی اتاق وجود داره اختصاص پیدا میکنه که منحصر بفرده. (کاربردش وقتیه که بخواهیم یک شی خاص یک سری دستورات رو اجرا کنه)

دستور exit : باعث ادامه ندادن خط های کد میشود.

```
a = true;
if ( a == true ){
	exit;
}
instance_destroy(); // don't run
```

دستور break : باعث شکستن حلقه های تکرار و دستورات سوییچ میشود.

نکات مفید

۱- یک سری متغیر داریم که برای هر شی به صوزت اتومات وجود دارد و طبق حالت ها و حرکت های شی مقدار آنها تغییر میکند. برای مثال میتونیم به متغیر های زیر اشاره کنیم :

ویژگی	توضیحات
x	مشخصه طولی جسم ,که نشان دهنده فاصله جسم تا محور طول ها در یک اتاق هست
y	مشخصه عرضی جسم ,که فاصله جسم تا محور عرض ها در یک اتاق درونش ذخیره میشود
speed	سرعت شی که هم اکنون دارد درونش ریخته میشود
direction	جهت حرکت شی که هم اکنون دارد درونش ریخته میشود.
gravity	مقدار جاذبه جسم , اگر در بازی از جاذبه استفاده کرده باشیم کاربرد دارد
hspeed	سرعت جسم در جهت افقی
vspeed	سرعت جسم در جهت عمودی
sprite_index	نشان دهنده این است که جسم چه اسپرایتی را دارد و می توان آن را تنظیم کرد
sprite_width	مقدار عرض اسپرایت را در خودش ذخیره می کند
sprite_height	مقدار طول اسپرایت را در خودش ذخیره می کند
image_alpha	مقدار شفافیت تصویر جسم را در خودش ذخیره دارد و می توان آن را تغییر داد
image_angle	زاویه ای که تصویر در آن حالت است را در خود نگه می دارد و می توان آن را تغییر داد
image_index	شماره تصویری از اسپرایت که دارد نمایش داده می شود را در خودش دارد
image_number	تعداد تصویر های اسپرایت را در خودش دارد
image_speed	سرعت پخش تصویر های اسپرایت را دارد و می توان آن را تغییر داد
image_xscale	مقدار پیش فرض آن یک است با تغییر آن می توان تصویر را نسبت به محور y آیینه کرد و تصویر را از نظر مشخصه x کشید یا فشرده کرد
image_yscale	مقدار پیش فرض آن یک است با تغییر آن می توان تصویر را نسبت به محور x آیینه کرد و تصویر را از نظر مشخصه y کشید یا فشرده کرد

دسترسی به متغیرهای بالا به شکل (objectName.speed) هست.

۲- یه سری تابع در مورد جسم و instance ها داریم :

تابع	توضیحات
instance_create(x, y, obj)	شی obj را در مختصات x,y می سازد
instance_destroy()	شی را نابود می کند (حذف می کند)
instance_exists(obj)	آیا شی obj وجود دارد؟
instance_find(obj, n)	از شی obj به تعداد n پیدا می کند
instance_nearest(x, y, obj)	نزدیک ترین شی obj نسبت به مختصات x,y را پیدا می کند
instance_furthest(x, y, obj)	دور ترین شی obj نسبت به مختصات x,y را پیدا می کند
instance_copy(perf)	شی را کپی می کند. اگر pref مساوی true باشد شی را create می کند
instance_change(obj, perf)	شی کنونی را به شی obj تغییر می دهد. اگر pref مساوی true باشد شی کنونی را destroy و شی obj را create می کند

تابع چیه؟

تابع یه دستوره که داده های ورودی رو میگیره و طبق اون ها خروجی را میسازه. بعضی تابع ها ورودی ندارن و بعضی هاشون هم خروجی ندارن ولی یه تغییری در سیستم رو انجام میدن. برای تابع instance_exists(obj) شی obj ورودی تابع هست و خروجی آن true هست یا false .

۳- توابعی در رابطه با حرکت و برخورد داریم :

تابع	توضیحات
distance_to_object(obj)	فاصله تا شی obj را محاسبه می کند
distance_to_point(x, y)	فاصله تا نقطه به مختصات x,y را محاسبه می کند
move_towards_point(x, y, sp)	شی به سمت مختصات x,y با سرعت sp حرکت می کند
place_empty(x, y)	آیا مکان x,y خالیست؟
place_meeting(x, y, ob)	آیا شی در نقطه x,y به شی ob برخورد می کند؟
point_distance(x1, y1, x2, y2)	فاصله نقطه x1,y1 تا نقطه x2,y2 را محاسبه می کند
point_direction(x1, y1, x2, y2)	زاویه خط متصل بین دو نقطه x1,y1 و x2,y2 را محاسبه می کند

۴- توابع و متغیر هایی در ارتباط با اتاق ها داریم:

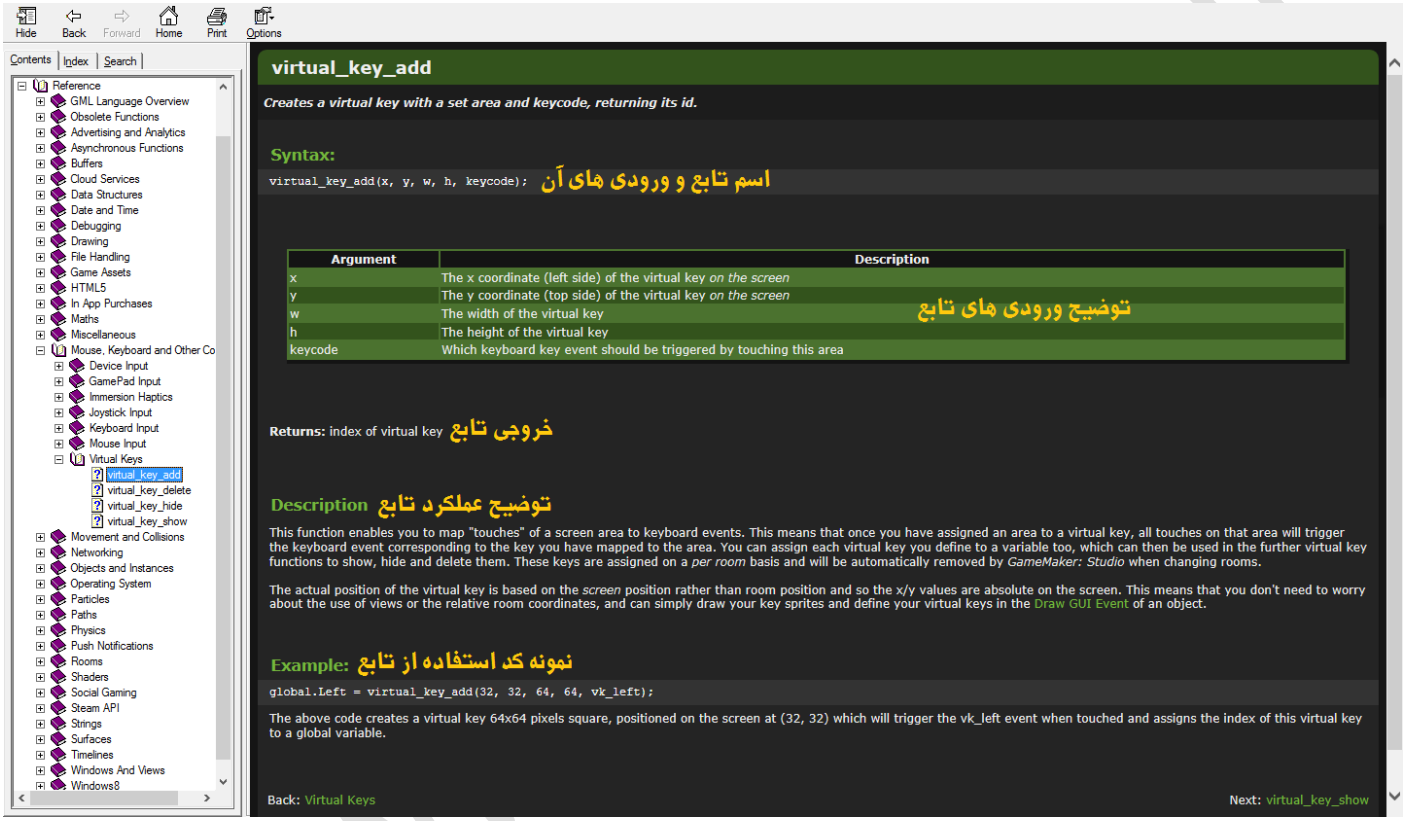
تابع یا متغیر	توضیحات
room_goto(numb)	رفتن به اتاق شماره numb
room_goto_next()	رفتن به اتاق بعدی
room_goto_previous()	رفتن به اتاق قبلی
room_restart()	ری استارت کردن اتاق کنونی
room_exists(index)	آیا اتاقی با شماره index وجود دارد؟
room_next(numb)	اتاق بعد از شماره numb را محاسبه می کند
room_previous(numb)	اتاق قبل از شماره numb را محاسبه می کند
room_get_name(index)	نام اتاق با شماره index را محاسبه می کند
room_first	متغیری که شماره اولین اتاق را نگه میدارد
room_last	متغیری که شماره آخرین اتاق را نگه میدارد
room_height	مقدار طول اتاق را نگه میدارد و میتوان آن را تغییر داد
room_width	مقدار عرض اتاق را نگه میدارد و میتوان آن را تغییر داد
room_caption	نام اتاق که دوست داریم در نوار ویندوز نمایش داده شود را نگه میدارد

۵- توابعی در ارتباط با مدیریت کیبورد و ماوس داریم :

تابع	توضیحات
keyboard_check(key)	آیا کلید key فشرده و نگه داشته شده است؟
keyboard_check_pressed(key)	آیا کلید key فشرده شده است؟
keyboard_check_released(key)	آیا کلید key رها شده است؟
keyboard_key_press(key)	کلید key فشرده شود
keyboard_key_release(key)	کلید key رها شود
keyboard_set_map(key1, key2)	عملکرد کلید key1 و key2 یکسان میشود
keyboard_get_map(key)	کلید یکسان key را محاسبه می کند
virtual_key_add(x, y, w, h, keycode)	ساخت کلید مجازی در مختصات x,y با اندازه w,h که کلید keycode را شبیه سازی می کنند
virtual_key_hide(index)	پنهان کردن کلید مجازی به شماره ی index
virtual_key_show(index)	نمایان کردن کلید مجازی به شماره ی index
mouse_check_button(numb)	آیا کلید numb ماوس فشرده و نگه داشته شده است؟
mouse_check_button_pressed(numb)	آیا کلید numb ماوس فشرده شده است؟
mouse_check_button_released(numb)	آیا کلید numb ماوس رها شده است؟
mouse_wheel_down()	آیا اسکرول ماوس به سمت پایین چرخانده شده است؟
mouse_wheel_up()	آیا اسکرول ماوس به سمت بالا چرخانده شده است؟

mouse_button	آخرین کلید ماوس که فشرده شده است را نگه میدارد
mouse_x	مقدار x که هم اکنون اشاره گر ماوس دارد را نگه میدارد
mouse_y	مقدار y که هم اکنون اشاره گر ماوس دارد را نگه میدارد

۶- توابع GML بسیار زیاد و برای کاربرد های مختلف, از حرکت اجسام تا دانلود و آپلود فایل روی سرور وجود دارد و تا اینجا آن هایی که مهم بوده است را معرفی کردیم برای مطالعه بیشتر باید به فایل help که در محل نصب گیم میکر هست مراجعه کنید. حالا میخواهیم به شما ماهیگیری در زمینه یادگیری این توابع یاد بدیم . هر تابع را که بخواهید مطالعه کنید از چند بخش تشکیل شده است که روی شکل مشخص شده اند :



بالاخره این قسمت تموم شد :)

امام زمان (عج) : هیچ چیز مثل نماز، بینی شیطان را به خاك نمی‌مالد، پس نماز بخوان و بینی شیطان را به خاك بمال. بحارالأنوار، ج ۵۳، ص ۱۸۲