

Getting Started with Game Maker Scripting

Mike Bailey

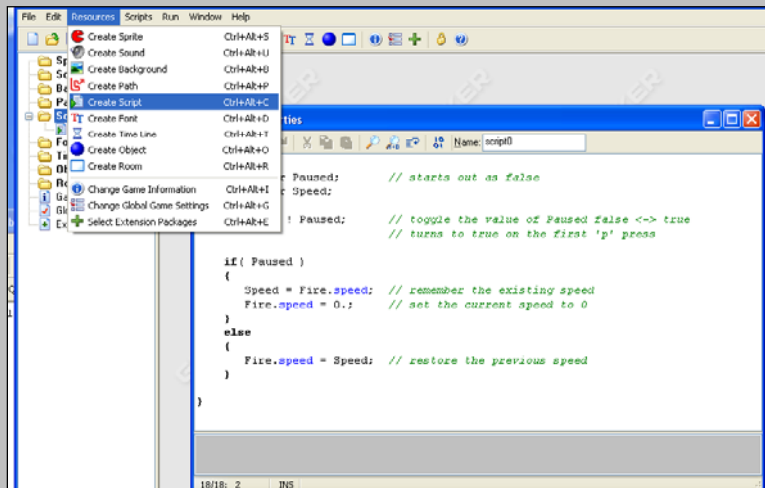
mjb@cs.oregonstate.edu

Oregon State University



Oregon State University
Computer Graphics

Scripts can be entered as a
"Resources → Create Script"

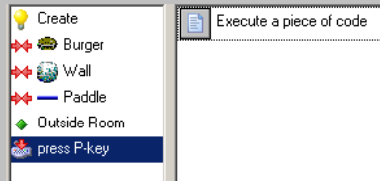


This gives you the chance to name the script, so you can use it in multiple objects

Computer Graphics

mjb - NoJanuary 27, 2009

Scripts can also be entered as an “Execute a Piece of Code” Action

A screenshot of the 'Execute Code' window in Game Maker. The window has a toolbar and a radio button for 'Applies To' set to 'Self'. The code is as follows:

```
{
  globalvar Paused; // starts out as false
  globalvar Speed;

  Paused = ! Paused; // toggle the value of Paused false <-> true
                    // turns to true on the first 'p' press

  if( Paused )
  {
    Speed = Fire.speed; // remember the existing speed
    Fire.speed = 0.; // set the current speed to 0
  }
  else
  {
    Fire.speed = Speed; // restore the previous speed
  }
}
```

OSU

Oregon
State
University

January 27, 2009

General Information

- Game Maker scripts look very much like programming in C, C++, and Java
- Scripts must begin with a left curly brace ({) and end with a right curly brace (})
- Statements end with a semi-colon (;)
- Variable names consist of letters, numbers, and the underscore (_)
- Variable names must begin with a letter
- Letters are case-sensitive, that is 'A' ≠ 'a'

OSU

Oregon State University
Computer Graphics

mjb - NoJanuary 27, 2009

General Information

- You can create conditional execution with an 'if-else' block

```
if( Paused )
{
    Speed = Fire.speed; // remember the existing speed
    Fire.speed = 0.; // set the current speed to 0
}
else
{
    Fire.speed = Speed; // restore the previous speed
}
```

- You can create a loop with a 'for' block

```
d3d_primitive_begin( pr_linestrip );
for( angle = 0.; angle <= 1440.; angle += 10. )
{
    radians = DegreesToRadians * angle;
    x = R * cos(radians);
    z = R * sin(radians);
    y = K * angle;
    d3d_vertex( x, y, z );
}
```

“for(initial-settings ; keep-going-if-this-is-true ; do-this-in-between-loops)”



Oregon State University
Computer Graphics

mjb - NoJanuary 27, 2009

The Structure of a Script

Scripts begin with a left curly brace {

Comments begin with a // and go to the end of the line

```
{
globalvar Paused; // Starts out as false
globalvar Speed;

Paused = ! Paused; // toggle the value of Paused false <-> true
// turns to true on the first 'p' press

if( Paused )
{
    Speed = Fire.speed; // remember the existing speed
    Fire.speed = 0.; // set the current speed to 0
}
else
{
    Fire.speed = Speed; // restore the previous speed
}
}
```

globalvar variables will keep their values in between calls to the script. (CS people call this "persistence".)

Scripts end with a right curly brace }



Oregon State University
Computer Graphics

mjb - NoJanuary 27, 2009

Note Game Maker's Automatic Color-coding when you Enter a Script – this helps prevent typos

```

{
  globalvar Paused; // starts out as false
  globalvar Speed;

  Paused = ! Paused; // toggle the value of Paused false <-> true
  // turns to true on the first 'p' press

  if( Paused )
  {
    Speed = Fire.speed; // remember the existing speed
    Fire.speed = 0.; // set the current speed to 0
  }
  else
  {
    Fire.speed = Speed; // restore the previous speed
  }
}
    
```

Comments begin with “//” and are green

Object names are purple

Property names are blue

Special constants are red

Special variables are blue

If these colors don't come up, then you've spelled something wrong!

Beware: names of things in scripts are all *case-sensitive*. That is, 'a' ≠ 'A'



Oregon State University
Computer Graphics

mjb – NoJanuary 27, 2009

Implementing a Pause feature with a Script

```

{
  globalvar Paused; // starts out as false
  globalvar Speed;

  Paused = ! Paused; // toggle the value of Paused false <-> true
  // turns to true on the first 'p' press

  if( Paused )
  {
    Speed = Fire.speed; // remember the existing speed
    Fire.speed = 0.; // set the current speed to 0
  }
  else
  {
    Fire.speed = Speed; // restore the previous speed
  }
}
    
```

The exclamation point means “not”. I.e., whatever **Paused** is now, change it to the other state.

The **if** statement causes something to happen if **Paused** is true. If it's not, then something **else** happens.

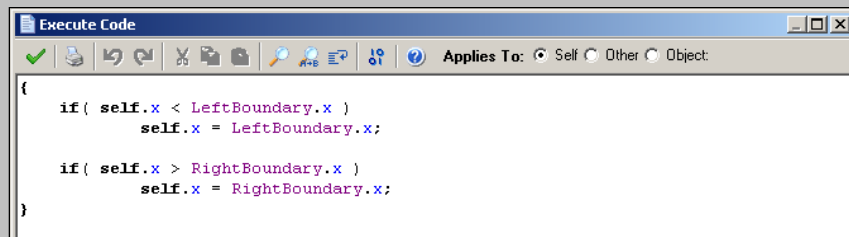
if and **else** statement bodies are delimited with curly braces.

The purpose of this script is to allow the 'p' key to pause the action to let you look at the state of your game. This is nice for development. When pausing, the script records the current Fire speed and sets the new Fire speed to 0. When un-pausing, the script restores the Fire speed to what it used to be.



Computer Graphics

Limiting Motion with a Script



The screenshot shows a window titled "Execute Code" with a toolbar and a text area containing the following script:

```
{
  if( self.x < LeftBoundary.x )
    self.x = LeftBoundary.x;

  if( self.x > RightBoundary.x )
    self.x = RightBoundary.x;
}
```

The window also includes a toolbar with various icons and a radio button menu labeled "Applies To:" with options "Self", "Other", and "Object".



Initializing the 3D Feature with a Script (Pro only)

```
{
  globalvar RotY;

  d3d_start();

  d3d_set_projection_ortho( 0., 0., room_width, room_height, 0. );
  d3d_set_perspective( true );

  RotY = 0.;
}
```



3D Drawing with a Script (Pro only)

```

{
  globalvar RotY;

  RotY += 10.;
  d3d_transform_set_rotation_y( RotY );
  d3d_transform_add_rotation_x( 20. );
  d3d_transform_add_translation( 200., 200., 0. );

  draw_set_color( c_green );
  d3d_primitive_begin( pr_linestrip );
    d3d_vertex( -30., -30., -30. );
    d3d_vertex( 30., -30., -30. );
    d3d_vertex( 30., 30., -30. );
    d3d_vertex( -30., 30., -30. );
    d3d_vertex( -30., -30., -30. );
    d3d_vertex( -30., -30., 30. );
    d3d_vertex( 30., -30., 30. );
    d3d_vertex( 30., 30., 30. );
    d3d_vertex( -30., 30., 30. );
    d3d_vertex( -30., -30., 30. );
  d3d_primitive_end();

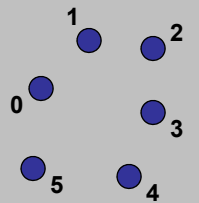
  d3d_primitive_begin( pr_linelist );
    d3d_vertex( 30., -30., -30. );
    d3d_vertex( 30., -30., 30. );
    d3d_vertex( 30., 30., -30. );
    d3d_vertex( 30., 30., 30. );
    d3d_vertex( -30., 30., -30. );
    d3d_vertex( -30., 30., 30. );
  d3d_primitive_end();
}

```

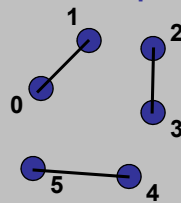


mjb - NoJanuary 27, 2009

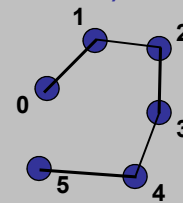
Geometric Primitive Topologies (Connections)



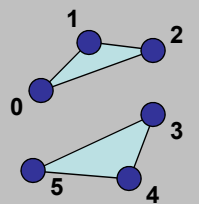
pr_pointlist



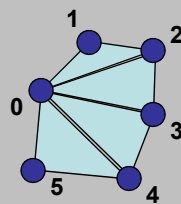
pr_linelist



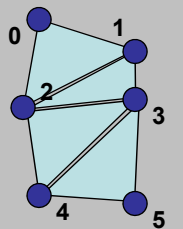
pr_linestrip



pr_trianglelist



pr_trianglefan



pr_trianglestrip



Oregon State University
Computer Graphics

```
d3d_primitive_begin( pr_linestrip );
```

mjb - NoJanuary 27, 2009

3D Transformations with a Script (Pro only)

```
globalvar RotY;  
  
RotY += 10.;  
d3d_transform_set_rotation_y( RotY );  
d3d_transform_add_rotation_x( 20. );  
d3d_transform_add_translation( 200., 200., 0. );
```



You Can Also Hook Transformations Up to the Mouse

```
d3d_transform_set_rotation_y( mouse_x );  
d3d_transform_add_rotation_x( mouse_y );  
d3d_transform_add_translation( 200., 200., 0. );
```

mouse_x and *mouse_y* are built-in Game Maker variables that tell you the current mouse position



Game Maker Web Links

General Game Maker Site:

<http://www.yoyogames.com>

These notes:

<http://cs.oregonstate.edu/~mjb/gamemaker>

276-page PDF Game Maker documentation:

<http://cs.oregonstate.edu/~mjb/gamemaker/gmaker.pdf>



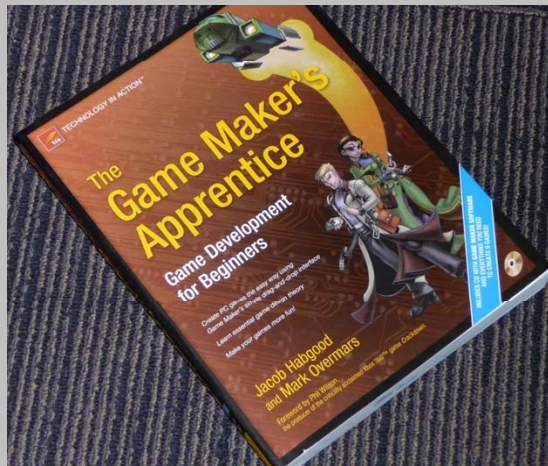
Oregon State University
Computer Graphics

mjb - NoJanuary 27, 2009

Reference Book

Jacob Habgood and Mark Overmars, *The Game Maker's Apprentice*, Apress, 2006.

(\$27 on Amazon)



Oregon State University
Computer Graphics

mjb - NoJanuary 27, 2009