Using Game Maker 8: GML Scripting

Mike Bailey

mjb@cs.oregonstate.edu

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

Oregon State University

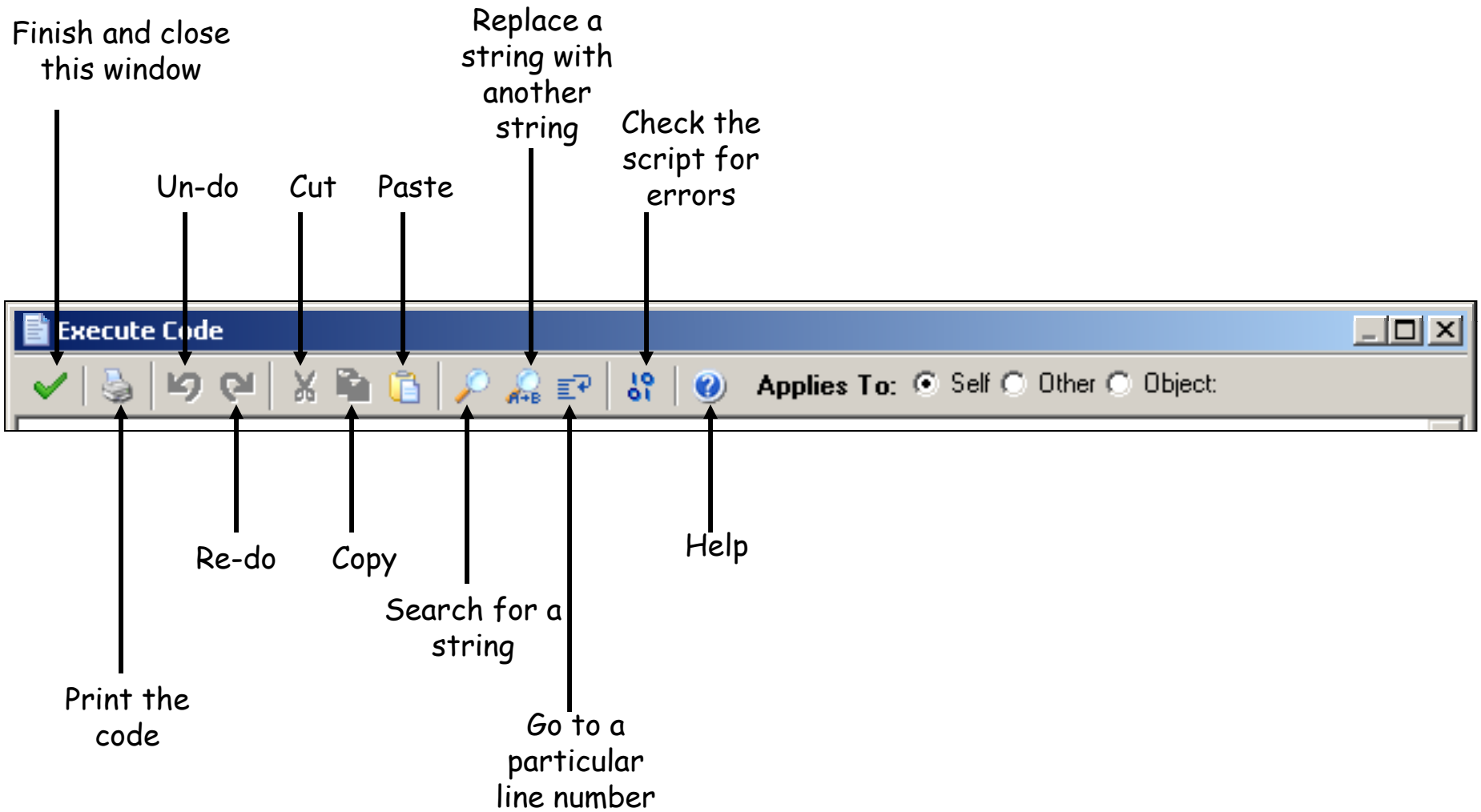
Scripting using the Game Maker Language (GML)

There are two neat things about using GML:

1. It allows your game to do things that the drag-and-drop features can't do by themselves

2. It looks very much like C++ and Java programming !

Scripting User Interface



The Structure of a Script Action

Scripts begin with a left curly brace {

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

Execute Code

Applies To: Self Other Object:

```
{
    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
    }

}
```

Scripts end with a right curly brace }

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

Execute Code

Applies To: Self Other Object:

```
{
    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
    }

}
```

Comment: green

Object name: purple

Property name: 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'

Implementing a Pause feature with a Script

Execute Code

Applies To: Self Other Object:

```
{
    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
    }

}
```

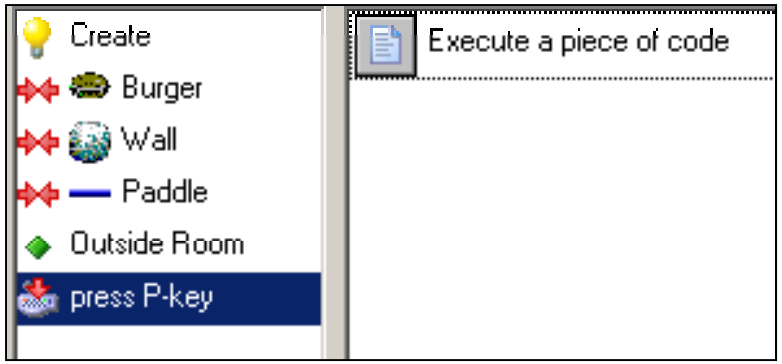
The exclamation point means "not". I.e., whatever ***Paused*** is now, change it to the other state.

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.

Define the Fire Object's Events



1. control→Execute Code

Execute Code

Applies To: Self Other Object:

```
{
    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
    }

}
```

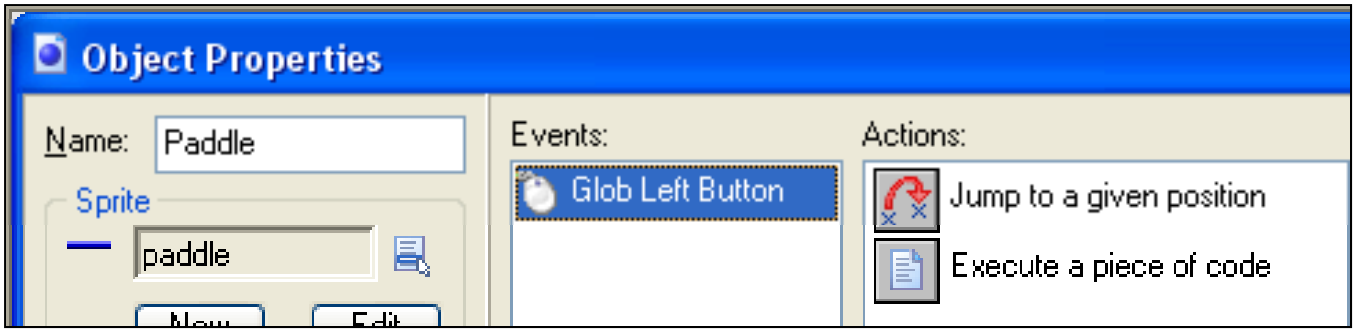
Limiting Motion with a Script

Execute Code

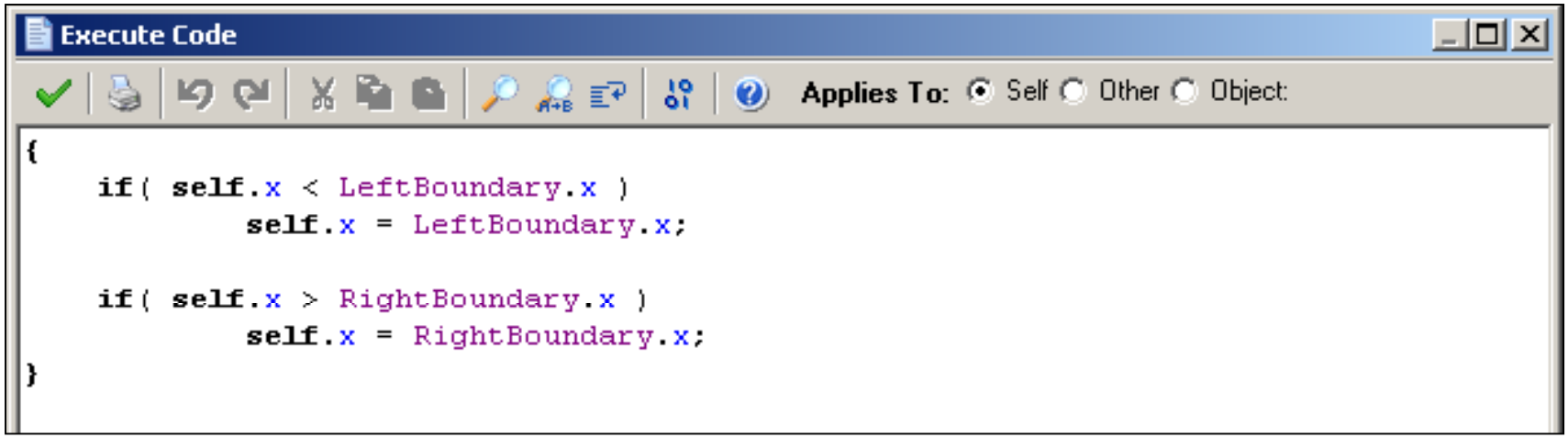
Applies To: Self Other Object:

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

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

Define the Paddle Object's Events



1. control→Execute Code




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

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

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

File Edit Resources Scripts Run Window Help

- Create Sprite Ctrl+Alt+S
- Create Sound Ctrl+Alt+U
- Create Background Ctrl+Alt+B
- Create Path Ctrl+Alt+P
- Create Script Ctrl+Alt+C
- Create Font Ctrl+Alt+D
- Create Time Line Ctrl+Alt+T
- Create Object Ctrl+Alt+O
- Create Room Ctrl+Alt+R
- Change Game Information Ctrl+Alt+I
- Change Global Game Settings Ctrl+Alt+G
- Select Extension Packages Ctrl+Alt+E

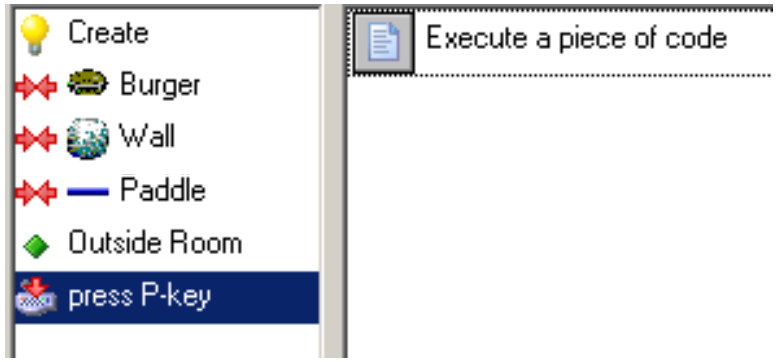
Name: script0

```
r Paused;        // starts out as false
r Speed;

! 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
    }

}
```

18/18: 2 INS

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

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



Execute Code

Applies To: Self Other Object:

```
{
    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
    }

}
```

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’

Functions you can use in Game Maker Scripts

abs(f)	Absolute value of a number
arccos(c)	Arc whose cosine is c
arcsin(s)	Arc whose sine is s
arctan(y_over_x)	Arc whose tangent is y_over_x
arctan2(y, x)	Arc whose tangent is y/x, taking signs into account
ceil(f)	Next highest whole number
cos(f)	Cosine oif r
degtorad(d)	Turn d into radians
exp(f)	e (2.71828…) raised to the f power
floor(f)	Next lowest whole number
frac(f)	Fractional (non-whole number) part of f
ln(f)	Log to the base e (2.71828…) of f
log2(f)	Log to the base 2 of f
log10(f)	Log to the base 10 od f
radtodeg(r)	Turn r into degrees
random(f)	A random number between 0. and f
round(f)	Round f to the nearest whole number
sign(f)	The sign of f (-1. or +1.)
sin(r)	The sin of r
sqr(f)	The square of f
sqrt(f)	The square root of f
tan(r)	The tangent of r

. . . and lots more . . .

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) “

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

Execute Code

Applies To: Self Other Object:

```
{
    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'