ISSUE 5

MarkUp

Martin Crownover:
Multiplayer Games

Plus...

Instant Collisions
Ammunition
Arrays & Lists
Clever AI
Registry

And Much More!

Interviews | Resources | Previews | Reviews | Tutorials

Teamwork Pt. 2

Welcome to Issue 5 of MarkUp, the only active GM magazine officially endorsed by YoYo Games ☺

Continuing from last month's discussion on last month's discussion on how to most effectively have a game development team (the concepts will work with most any team however) we go on to.....

Sharing work

When you are working in a geologically separated team, you need an efficient way to store and distribute work, and to list who will work on what.

Email can help with this to an extent, but there are a lot of good (and free) tools out there that can make the process a lot easier.

Basecamp

http://www.basecamphq.com/

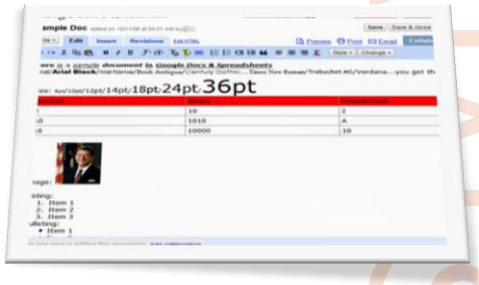
Basecamp is probably the most well known online collaboration tool. It lets you manage "projects" (the free version allows only one project, with unlimited participants), with events, deadlines and issues. It works like a specialized forum, allowing team members to track every detail of the project.

The free plan allows only one project, has no on-site file storage, and doesn't use SSL. You can get premium versions with more features starting at $12/month.

Google Docs & Spreadsheets

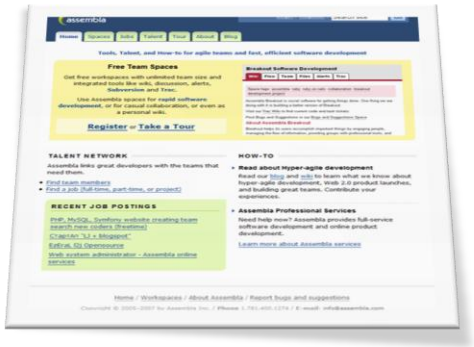
http://docs.google.com

Google's collaborative office tools let multiple people work on documents simultaneously. Perfect for working on team plans and the like.



SVN

SVN is the ultimate way to keep source code files synchronized between members of a team; it can work for just one person (to keep play-by-play backups) or for 10,000 people.



Best of all, you can get SVN hosting for free. Two popular sites to get said hosting are assembla.com and unfuddle.com.

See you next month!

Robin Monks ■

Contributors This Issue

Robin Monks	Sr. Editor
Eyas Sharaiha	Editor
Jason Stockton	Writer
Phil Gamble	Writer
Gregory	Writer
mr.gibblet	Writer
Bart Teunis	Writer
Michael Sharaiha	Writer
Sean Flanagan	Writer

Table of Contents

MarkUp is a gmking.org publication; please visit GMking for more free game development resources!

Photo © Freddy Thorvaldsen

GMR Featured Resources

One of the easiest ways to improve your knowledge of Game Maker and discover new possibilities is to use examples. Examples are a great way to include extra features in your game or to discover new ways of creating things. This month I am looking at some of the great places to find the examples you need.

GM Tutorials

http://gmtutorials.com/

Know what you're looking for? GM Tutorials has a group of creators who constantly add to the list of examples on the site. There are currently over 200 examples and tutorials ranging from RTS examples to fog or even a scroll bar example. Just go to the site and have a look.

The tutorials are handier for novices as they provide a bit of a walk through to go with any examples that they may include with the tutorial.

Simon Donkers

http://xrl.us/simondonkers

So what if you want complete game examples? Check Simon Donkers out. His site has a complete collection of full working game examples created in GM5 and 6 including breakout, snake, Pacman and more. There are also a few examples that aren't full game examples there as well.

GM Resources

http://xrl.us/gmres

No it's not the GM Resource, its GM Resources; the key difference being the plural on the end… Although this site doesn't have truck loads of examples it is slowly growing as it's a user based submission site that has many user submitted examples; many of those being 3D examples.

As the site is user submission based some examples may be lower quality than others so check any feedback before downloading (if the example has been reviewed or rated).

Conclusion

Still can't find what you want? GameMakerResource.com has links to plenty of other Game Maker sites with examples for you to download; so if you can't find what you want here, jump on there and browse the rest of the sites in our listing.

Remember as with all resources to give credit where credit is due. Creators deserve credit for their time and effort.

Jason Stockton,
GameMakerResource.com ■

Tinyweb

The tinyweb DLL is a web browser DLL that allows you to embed either local .html documents in your game, on even better, allow your game to open certain sites within. There are other browser DLLs out there, but this one is tiny, easy to use, and comes as an extension, library, and a bunch of scripts as well. The source is also included.

Get it now!
http://xrl.us/tinyweb

Arrays vs. Lists

The concept of arrays is basic and familiar to the majority of the programming languages out there. An array is a variable that is capable of storing multiple values, according to its index – which could be 1-dimensional or 2-dimensional in Game Maker.

However, Game Maker contains an excellent data structure – lists. A list is always 1-dimensional, and contains different values (sorted or not). Many other data structures exist in Game Maker, such as stacks and queues. What makes lists special is that it has direct-access, as opposed to their sequential access.

It might be easy to establish how arrays differ from regular variables, and how lists differ from other data structures. But the real question is: when to use lists, and when to use arrays – and which is better?

Understanding the differences

To be able to wisely choose which method to use for a particular case, the differences between lists and arrays must be established.

First, a clear advantage of arrays over lists is ability of using 2-dimensional arrays to form 'tables'. Lists particularly fail at this, and storing multiple variables in a "row" might lead to the need for string manipulations, or other tricks.

The use of CSV files to store data has been covered in the previous issue of MarkUp, and such an application is better implemented with arrays.

However, lists have their own advantages as well. First of all – according to Mark – handling lists is much faster and resource-friendly than arrays. So if you care about speed, lists have a good advantage. Not only that, but lists also can have data inserted to them in the middle, added to the end, sorted and re-sorted at any time, and more. Lists just fascinate me!

The major disadvantage of lists is that they are not saved with the game! So if you're game utilizes save/load, you'll either have to make your own save mechanism (like I did for one of my projects), or just make a script that loads the list after loading the game.

On the upside, though, Game Maker already provides reading and writing functions for lists, so you can easily save scripts to files and load them.

Where lists shine

I've mentioned the good points of lists in short before, but here's some info about what's really great about lists:

Copying Lists

A list could be copied from one id to another. This could hardly be achieved by arrays – but with lists, it only requires a single quick function. Copying a list is more useful than it seems, but its applications depend on the way you're using the lists in the first place.

For example, a list could be used to store multiple copies with multiple arrangements. Such an application allows a host of things to be done, including simple procedures such as finding the mode of values.

A more specific example is something like a "soccer tournament". We see in all FIFA games how teams are drawn up – such that teams in the quarterfinals, semi-finals, and finals are all displayed. This could be also done by copying a list of participants to other lists, and removing unqualified teams.

Another application is the use of "indexing", so that a long list is copied to different smaller lists, and all the list items except for those starting from a particular letter are removed. This results in having smaller lists, each having values starting with a single letter. When those are further sorted, they could be used in an index-like atmosphere, such as a dictionary.

Getting List Size

When using arrays, we almost always have to include another variable that acts as a "counter", so that the game only performs certain operations on the number of items that are there in an array. A list eliminates the need for such variable.

Easy Insertion of Values

Easy insertion of values to scripts is one of the things I LOVE about them! To properly insert a variable in an array, a loop should be performed to move all items occurring on or after the point of insertion one further step. This is all done automatically with lists, and not only that, but also much faster (in terms of execution speed).

Easy Removal of Values

Once a value is deleted from a list, all values fix their positions. Basically, it is the reverse of "Easy Insertion of Values", something that also isn't natively available for arrays.

Searching Capabilities

The searching capabilities for lists are

Arrays vs. Lists Cont.

extremely useful. You can find the value of a list row from its position, or get that list's ID by writing its value. Both give great possibilities, especially when used in conjunction with other list functions, such as sorting. You can find the n^{th} nearest instance from a particular object, etc.

Sorting

One of my favorites! You can sort all values in a script either ascending or descending. This also works for both numbers and text pretty well. This is pretty useful when using the list data structure to draw and actual list on the screen, which could become long.

Shuffling!

I'm in LOVE with shuffling. It has so many uses, ranging from the obvious "cards" to the AI for strategy games, or the order of events in a timeline for another.

Where Arrays Shine

Tables

Thought `variable[4]` arrays could be useful, something I really like about arrays is the use of two-dimensional arrays, like this `variable[6,3]`, to locate a value's position both on x and y. The use of two-dimensional arrays as tables is very interesting but will not be discussed in any further detail in this article.

Changing Values

Changing the fourth row of the array "variable" from 5 to 6 is pretty simple in arrays. A simple line of code:

```
variable[4]=6;
```

But in lists, a `ds_list_replace` function needs to be called, and the position of the row needs to be known (since with insertion and sorting, it's constantly changing) using yet another `ds_list_find_index` function. So it could be pretty painful at times.

Conclusion

Though lists could be used for everything, and arrays could be used for everything – some things are better handled by one mechanism than the other. Everything could be achieved, but when the results are the same, you might have to consider what is easier and quicker to write, and which is faster to execute.

Eyas Sharaiha ■

The MMORPG Makers

I'm not sure if I missed something when I joined the GMC. I think there were a set of rules for new members which I somehow didn't find. It's a great shame I haven't found them really. At least then I might be able to understand the more chaotic of people's decisions.

In almost every board on the GMC, it is possible to find a topic by someone wanting to create an MMORPG. This person usually would have joined in the last few weeks, or has been around for at least a year and a half. There's rarely someone in between those times.

The new members will be really enthusiastic about their project, expecting to have a game better than big names like RuneScape by the end of the week. The more experienced users will be coyer about their plan and then either decide not to follow it through or disappear for a few weeks as they work on a starting demo.

The truth is that creating a MMORPG takes an awful lot of work. You need to set up servers, understand all the online functions inside out and also create something that's actually worth playing. Nobody wants to look at their player standing in a room and not being able to move them.

I tried to make a MMO once. It was going to be a platform game where the user could design their character and then use it to explore the online world. When I got to making it multiplayer…it fell apart (though if anyone is interested in reviving it, contact me). The truth was, of course, that I knew nothing near enough about creating multiplayer games and I just wasn't up to it.

What I'm trying to say, I suppose, is that you have to know your limitations. Although you should always try and do your best, and make a game as well as you can, you shouldn't try and do something that you're incapable of. It just doesn't follow. Having said that, if you can do, you should go for it.

Gregory ■

AI: A Clever Opponent

The burden of creating artificial intelligence is that it is so hard to program. Professional developers can take months or even years just figuring out AI for a single enemy. Luckily, Game Maker comes with a feature that can help this problem, which is motion planning.

Motion planning is a simple yet effective way of doing 'search' AI. In this tutorial, you can learn how to make your own search AI and have an enemy chase after you. More information is provided in the GM Manual > GML > Gameplay > Motion Planning. This tutorial is for creating a basic. More can be done, though.

The first thing you need to do is set up a grid the MP will be worked out on. This is simple, and can be done with one function.

In the create event of your enemy, type:

```
{
mp_grid_create(left,top,hcells,
  vcells,cellwidth,cellheight);
}
```

Now, to set this up, we first have to specify the left and top of the room. This is obviously 0, 0. That part is easy. Next, we have to decide how many cells go horizontally. To do this, we must first figure out the width and height of our squares. You will usually want either 16x16 or 32x32. Let's say we're making a game where the AI searches on 16x16 squares. So, the cell width is 16, and the height is the same. Now, we can get back to figuring out how many should go horizontally and vertically. Note that in order for this to work, the room width and height must be a power of the width and height. So, for hcells and vcells, we instead type this finished piece of code:

```
{
mp_grid_create(0,0,room_width/16,
  room_height/16,16,16);
}
```

That creates our grid. But we want to be able to refer to it. So, we change it into:

```
{
grid_id = mp_grid_create(0,0,
  room_width/16,room_height/16,16
  ,16);
}
```

Now, we can refer to the grid by using the variable grid_id.

Well, we have created a grid, but that does not mean it does anything. It just sits there, soaking up your RAM and CPU process. So, now we have to make it do something.

Let's say that we have an object, obj_block, that we want our enemy to go around and try to catch the player. Here, we use the following piece of code:

```
{
mp_grid_add_instances(id,
  object,prec);
}
```

Let me explain this. The MP grid works on spaces that are available and unavailable. So, to make our enemy go around the unavailable spaces, we want to set obj_block as unavailable. We can also add the grid id, which we defined inside of grid_id. Prec is short for precision. You should usually switch precision on to create a better effect. So now, we can modify our code to this:

```
{
mp_grid_add_instances(grid_id,obj
_block,true);
}
```

Put this code in the create event, too.

And there we go. Now we have our grid all set up and ready, so the last thing we do is make the enemy follow the path.

Path? Did someone say path? Oh, that's right. Before you go on, make a completely empty path with nothing in it. Call it pth_move. This may sound funny to you, but you'll understand.

Now, here is the big finale. In the step event of the enemy, put the following:

```
{
mp_grid_path(grid_id,pth_move,x,y
,obj_player.x,obj_player.y,true);
path_start(pth_move,1,0,false);
}
```

I'll tell you what this does. mp_grid_path sets the path we selected (pth_move) and edits it so that it becomes a path between the starting point (3rd and 4th argument, enemy x and y) to the goal (5th and 6th argument, obj_player.x and obj_player.y), it sets the grid that we put this path on, (grid_id), and finally, the 7th argument, allows diagonal paths (if you want straight lines, set this to false.)

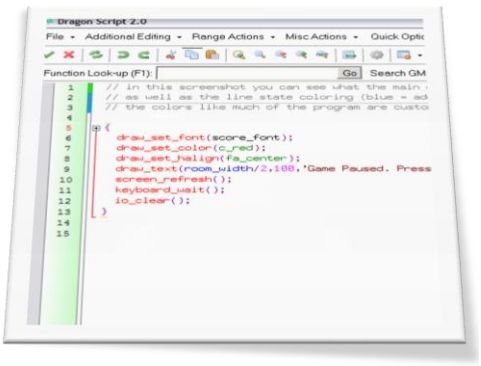
The next piece of code simply starts the path. It starts pth_move at a speed of 1 (this can be changed.) Warning: DO NOT CHANGE THE LAST TWO ARGUMENTS. This will make the path be un-relative and try to loop. You DO NOT want this!

Well, that's it, a guide to using search AI. This article only covers the basic functions, but you can look in The GM Manual > GML > Gameplay > Motion Planning for more information. I hope this guide will be of use to you as you progress in your AI career.

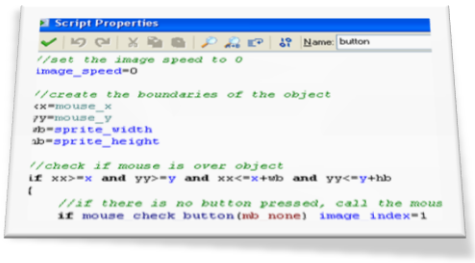
Sean Flanagan ■

Dragon Script vs. GM Editor

When it comes to coding, I'm very simplistic. I make websites using only Microsoft's Notepad and would happily code GML in that as well. So you can imagine how I felt when I took my first look at Dragon Script recently.



The only difference I find between Notepad and the built-in GML editor (below) is that the latter has colour coding and a quick reference at the bottom. This suits me very well – both of those features are really useful when coding. They help me to lay out my screen and remind me of all the arguments I need to know.



I think the thing which I really love about the built-in GML editor is its simplicity: it's very, very basic. Which is wonderful. There's nothing around to distract you, nothing there which you suddenly feel you should be using.

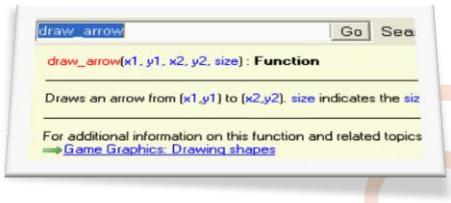
Dragon Script is very different. Its interface is covered in buttons, textboxes and drop-down boxes, all showing tools which could potentially be useful at some point but which seem to get in the way when you're trying to code. Although the main typing window is quite obvious, it is a bit overwhelming and difficult to figure out where to start.

I notice too, that Dragon Script seems very separate to GM. It seems like a completely separate program. Admittedly, it is, but the built-in editor feels like it's part of GM itself, whereas DS feels external, in the same way that using word to edit GML files would.

I think my real distrust stems from two things: Primarily, I'm not used to it so I'm wary and a bit confused. I want my old editor back. Secondly, the colour scheme is different. This may sound really picky but it's something I can't help but noticing. I'm extremely used to seeing my variables show up in light blue and constants in brown but here, that isn't the case. Fortunately the colour system, as in Game Maker, is fully customisable.

But I can't really spend this whole article saying that the built-in editor is better than Dragon Script because I would be prejudice and very unfair. Dragon Script has some great features which make coding so much easier. The auto-complete feature works similarly to the prompt at the bottom of the GML editor but also tells you what each prompt actually is (functions, constant, variable), which can be a great thing for new users of Game Maker Language who are not entirely sure what everything does.

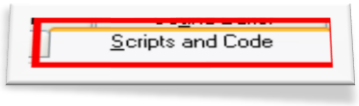
The other main special feature of Dragon Script for me is the Function Look-up which, surprisingly enough looks up functions. It gives a description of what the function does, explains it arguments and links you to the relevant article in the Game Maker help files.



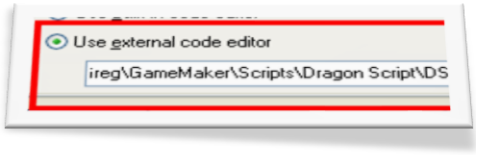
In conclusion, I think Dragon Script is very useful tool for people new to Game Maker Language. It has lots of tools to explain functions and keep everything on hand. However, I think more experienced users would find the built-in editor better as they don't need the fancy look-up tools but instead just a plain window to quickly edit their codes.

Switching to Dragon Script

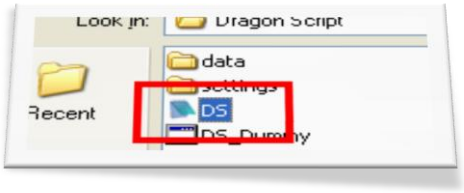
First, select the "Scripts and Code" tab in the Game Maker Preferences:



Select to "Use external code editor" and then enter the path for the Dragon Script exe:



You can click on the "…" button to find the program in your files:



Gregory ■

Script of the Month

Our favourite GML script resource – GMLscripts.com – has done it again, releasing over 100 GML scripts! One of the scripts that have really appealed to me is called `sprite_edit_begin` which initiates the editing of a sprite by taking advantage of a feature introduced in Game Maker 6.1: Surfaces.

Script #1: sprite_edit_begin

```
/*
**  Usage:
**      session = sprite_edit_begin(sprite)
**
**  Arguments:
**      sprite      sprite to edit
**
**  Returns:
**      session     an editing session ID
**
**  Notes:
**      Begins a sprite editing session. All draw commands are
directed to a
**      surface holding the a horizontal strip of each of the
frames in the
**      given sprite. When finished editing, call function
sprite_edit_end()
**      to implement the sprite changes and to reset the drawing
surface.
**      Resets blending mode to normal.
**
**  Dependencies:
**      sprite_edit_end()
**
**  GMLscripts.com
*/
{
    var sprite,a,w,h,n,xoff,yoff,surface,i;
    sprite = argument0;
    a = draw_get_alpha();
    w = sprite_get_width(sprite);
    h = sprite_get_height(sprite);
    n = sprite_get_number(sprite);
    xoff = sprite_get_xoffset(sprite);
    yoff = sprite_get_yoffset(sprite);
    surface = surface_create(w*n,h);
    surface_set_target(surface);
    draw_clear(c_black);
    draw_set_blend_mode_ext(bm_one,bm_zero);
    draw_set_alpha(1);
    for(i=0; i<n; i+=1) {
        draw_sprite(sprite,i,i*w+xoff,yoff);
    }
    draw_set_blend_mode(bm_normal);
    draw_set_alpha(a);
    return (string(surface)+':'+string(sprite));
}
```

Script #2: sprite_edit_end

```
/*
**  Usage:
**      sprite_edit_end(session)
**
**  Arguments:
**      session     the editing session ID provided by
sprite_edit_begin()
**
**  Returns:
**      nothing
**
**  Notes:
**      Ends the editing session started with
sprite_edit_begin, replacing
**      the old sprite with the edited sprite and freeing the
editing surface.
**
**  Dependencies:
**      sprite_edit_begin()
**
**  GMLscripts.com
*/
{
    var session, p, surface, sprite, w, h, n, prec, tran,
smth, load, xoff, yoff, temp, i;
    session = argument0
    p = string_pos(':',session);
    surface = real(string_copy(session,1,p-1));
    sprite = real(string_copy(session,p+1,10));
    w = sprite_get_width(sprite);
    h = sprite_get_height(sprite);
    n = sprite_get_number(sprite);
    prec = sprite_get_precise(sprite);
    tran = sprite_get_transparent(sprite);
    smth = sprite_get_smooth(sprite);
    load = sprite_get_preload(sprite);
    xoff = sprite_get_xoffset(sprite);
    yoff = sprite_get_yoffset(sprite);
    temp =
sprite_create_from_surface(surface,0,0,w,h,prec,tran,smth,load
,xoff,yoff);
    for(i=1; i<n; i+=1) {
sprite_add_from_surface(temp,surface,w*i,0,w,h);
    }
    sprite_assign(sprite,temp);
    sprite_delete(temp);
    surface_reset_target();
    surface_free(surface);
}
```

Use

Using this set of scripts is simple – all you have to do is begin with a `sprite_edit_begin()` script call, start using draw features **in the same event** and then save the sprite by calling the `sprite_edit_end()` script.

Such a procedure doesn't have to be put in the draw event, since after it is complete one time, the sprite will be saved like that, and could be used at any time.

Eyas Sharaiha ■

File Search Functions

File Search functions in Game Maker have a multitude of uses, and are crucial to several types of games. Compared to other functions, it is hard to make up creative uses for file search functions, since what they do is pretty straight forward – find files.

The Basics

Though many people think of file search as a limited mechanism to find a single file, it actually is capable of finding multiple files under the same conditions. To start finding files, the `file_find_first()` function is used. After that function, `file_find_next()` functions could be used to find the next file under the same conditions as those outlined in the first command.

To close the 'thread' of file-finding, a simple `file_find_close()` function is used. After such functions, "find next" functions cannot be used anymore.

Order of File Finding

Obviously, if multiple file possibilities existed for a search conditions, multiple files need to be returned. Also a well known fact is that Game Maker doesn't return multiple values – so only a single value will be returned for the first file finding function. Other files will be returned in the find next function.

The order of how the values are returned is ascending, so a file called "armor.csv" will be returned before "bike.ogg".

Arguments

The `file_find_first()` function requires two arguments: the mask and the attributions. While the find next function has no arguments, as it relies on the find first command.

The Mask

The mask for a file is a string that should match with both the name and location of the file to be found. The mask could include wildchars (*) which basically means any character or a group of characters.

So, using "*" as the mask would result in having all files being returned. Using a "*.txt" would return all text files in that given directory. Using "hello_*.gmk" would result in returning all folders that have the gmk extension *and* begin with "hello_".

The Attributes

The attributes argument is a (collection) of real-valued variables that could be added up together using regular addition "+" in Game Maker. For no attributes, the real value 0 is used. Here is a set of variables to be used with the attributes in the table below.

Variable	Description
`fa_readonly`	A read only file
`fa_hidden`	A hidden file
`fa_sysfile`	A system file
`fa_directory`	A directory
`fa_volumeid`	A volume-id file(which represents the volume of a drive)

To return files that are read-only, hidden, and system files, (`fa_readonly + fa_hidden + fa_systemfile`) is used.

Returned Values

If such a file exists, only the **name** of the file will be returned. This means that the path (location) of the file will not be returned, however, the file extension will be part of the name returned.

File Search Functions Cont.

If the file being searched for does not exist, an empty string ("") will be returned. This could be used to count the number of files satisfying a certain set of conditions by using a simple while loop that adds 1 to a counter each time `file_find_next()!=""`.

Applications

Copying All Folder Content

```
//Script: move_folder()
//move_folder(string folder1, string
//folder 2);
//Folders must not include final
//backslash courtesy of Eyas Sharaiha,
//featured on MarkUp
f1=string(argument0);
f2=string(argument1);
file=file_find_first(f1+"\*",0);
while(file!="")
{
  file_copy(f1+"\"+string(file),
f2+"\"+string(filename_name(file)));
  file_delete(f1+"\"+string(file));
  file=file_find_next();
}
file_find_close();
```

Counting files in a folder

```
//Script: count_folder()
//count_folder(string folder);
//Folder must not include final backslash
F0=string(argument0);
i=0;
file=file_find_first(F0+"\*",0);
while(file!="")
{
    i+=1;
    file=file_find_next();
}
file_find_close();
```

Conclusion

While finding folders might not have that many "different" uses – these uses still find their ways to multiple games, both with Game Maker on the GMC and other commercial games as well.

Finding files could be useful in multiple ways – either when searching for a particular file in multiple destinations, or even searching for a file which's exact name is not known. File finding functions could be used to create folder operations, or just as a method of storing settings, counting files, etc.

One good example I can now think of is counting the number of .sav files in a directory so that a game could display the names of all saved games in a certain directory.

All in all, the possibilities are endless!

Eyas Sharaiha ■

3D vs. 2D Games

The comparison between 2D and 3D games is, to most, striking. Whilst 3D games give you a sense of depth and three vertices to play with, 2D can only attempt to give a real-world feel…and tends to fail.

But why is it that 3D games are more popular to gamers of today? Looking at the top video game sales at Tesco's earlier this morning, only one of the twenty shown did not have a 3D environment. But said game was interface based and so was barely 2D either.

The main gaming difference between 2D and 3D is, in my eyes at least, is what you can do with each one. 3D provides a very extendible, multi-dimensional situation whilst 2D is preferred for simple, smaller games.

The third dimension allows producers of games to make various different extensions to simpler 2D gameplay. A first-person-shooter can give the player the opportunity to hide behind walls or above their opposition; something which 2D top-down games cannot offer so easily. In racing games too, the third dimension can give a z-axis, letting cars go off ramps, or really fly in collisions.

That isn't to say that 3D is the best arrangement to use though. Simple arcade or puzzle games work perfectly in 2D. There is absolutely no reason why someone would need to make a Pacman or space invaders game in 3D. If anything, this would take away the fun of it – the original, traditional gameplay element.

It is the extendibility of 3D games that make them so popular to the gaming audience. It's not often that a 2D game is made that has a great storyline and complex and dynamic gameplay. With 3D, that's a breeze. Thus, you can make a simple mini-game in 2D very successfully but, if you want to make something really vibrant and exciting, 3D is what should be used.

Gregory ■

Seiklus (17)

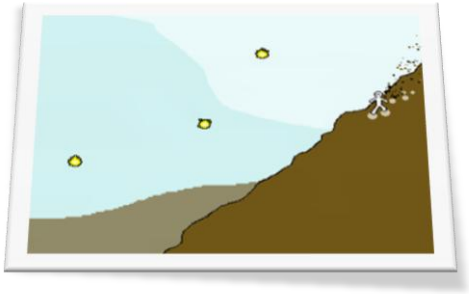
Where do I begin? Well maybe because this game is somewhat confusing, because this game has too many good and bad points that it is difficult to talk about but I'll do it anyway.

It is very addictive, well until you finish the game then you just throw it away, not that that's a bad thing because this is the case with every short term addictive game, and it's just a personal choice of the author.

Game play

A simple addictive game, I do think that after playing it I won't really play it again. But none the less it did achieve addictively even though short term, you know it's one of those games you have to play because it is worth it, but it all comes down to, I think, whether you want to play it again, and in this category it failed remarkably.

Also there are some aspects of the game that you don't know why they are there for, like the Piano – what is it for? Obviously you have to play a tune but there is no indication of that anywhere. Something should indicate that, as I said there are many aspects of the game that remain as I see vague and unanswered.



Controls

The simple up, down, left, right arrow keys technique, it never gets old, easy to use and above all everybody is used to it.

10/10 now really how can it get any other score, tell me?

Presentation

"Press 's'", that is how the welcome screen is, then "press '1' '2' '3'" to chose which of the three saves to continue, but is this really what you call nice, cause sure it is boring and lacks stimulus.

There is no indication about buttons, I replayed parts of the game once too many times over and over again because I did not know what it was, by the way it is "S".

Story line

The story begins by the player being blasted off a mountain away from his girlfriend/sister/wife, and then nothing gives indication of what is your main objectives, well at least until you get to the lights near the cave exit.

The story is, I don't know how to say this, short, ok the areas are not very predictable but there is no indication of the story other than in the beginning.

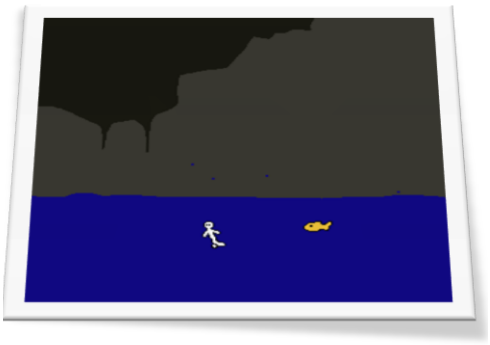
Graphics

Game graphics are those of platform games can't really judge them because that is how old school games should look, I do not think platform games have reached their full graphics potential, but that is no one's fault as much as those commercial developers, they got us

Seiklus (17) Cont.

used to the idea that platform games should look like this.

Only problem is the shape of the character he could have been done better i.e. not just a white, more than cartoonish ghost like man, try putting on some cloths on him.



Music

Turn off your speakers and on with your headphones! How could anyone stand that much repetition, but on there "plus" side there are silent areas were you get no music, so I urge you if you are playing this game put on your own playlists on your music player, the music at first is stimulating but after a few minutes of game play it starts to get annoying.

Sound effects

Well, there are very few sound effects in the game but they are generally ok, the game does need more especially during movement, walking of jumping not just when being kidnapped, taking the colored fire things, or playing the giant piano.

Conclusion

The game is addictive yes, but it gets boring after a while, in my opinion the game achieves greatness if the player wishes to play it more than once which this one has failed miserably.

It all starts here,
Michael Sharaiha ■

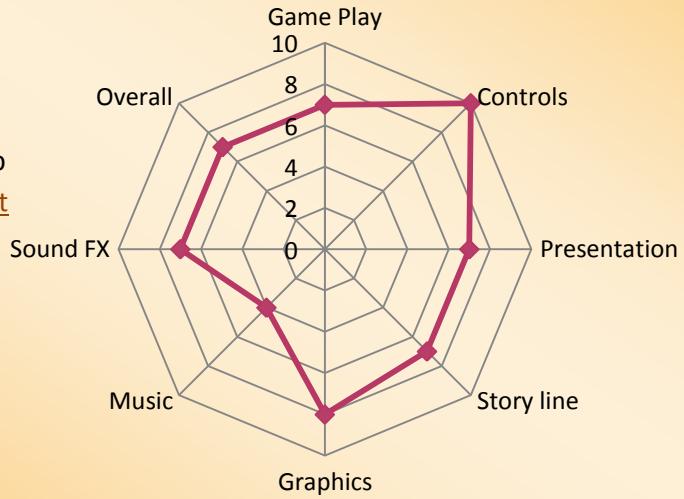
Conclusion

Pros: Enjoyable theme, gameplay, and overall feel.
Cons: Unpleasant to the eyes and ears.

Download Size: 2.6MB
Download Type: Game .zip
Author: clysm, autofish.net
Released: 8/15/03

Get it Now:
http://xrl.us/seiklus

Ratings



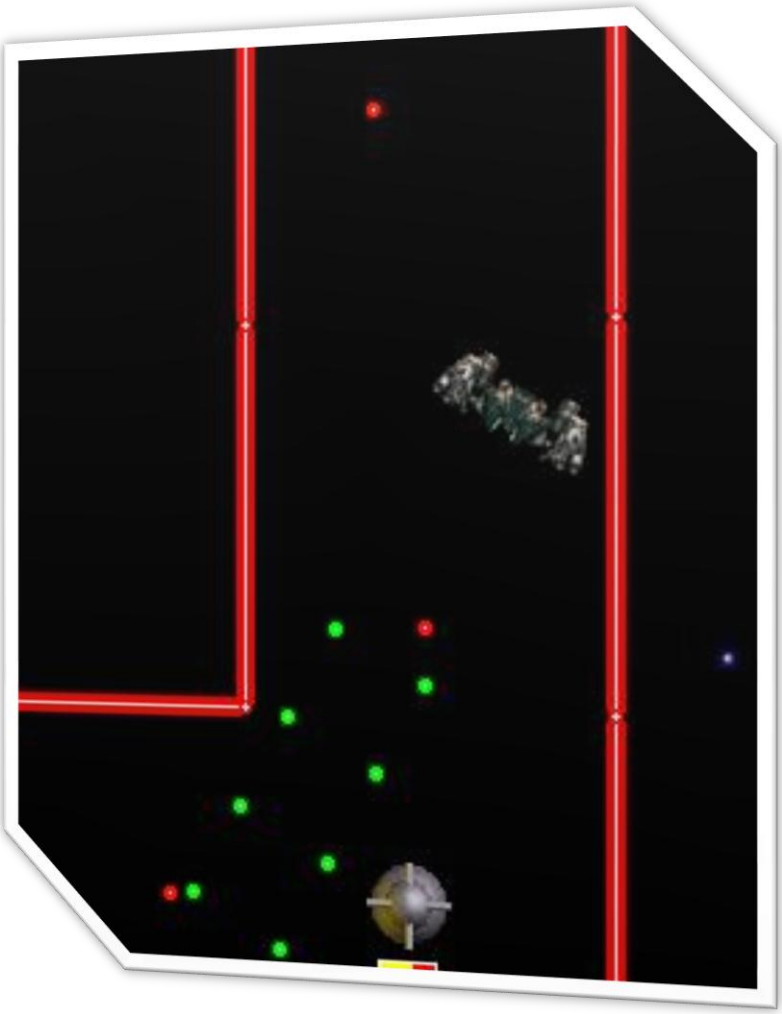
Arena

What they say

Arena is the two-player game where opponents battle within the confines of a random maze. With lots of options and multiple paths to victory, Arena is never the same game twice. This is a great rivalry game where two good friends can sit down and blast away at each other. This is my first original video game.

Review

Arena lives upto and exceeds my expectations. I first downloaded this game in 2003 and I am still playing it today. Games which have two players' controls contained within a single keyboard often spell disaster, however in Arena there are no key clashes, or awkward reaching across your opponent to the other side of the keyboard. With a vast array of in game customisation available and Arena's amazing random maze generator, each game really is different.



It is possible to win within seconds or games can last 20, 30 minutes or more - it all depends on how you choose to set up the game and your starting position in relation to sentries, bases, powerups and your opponent.

A key aspect of the game is customisation - there is so much about the game you can modify and this can change the game dramatically. You can control the availability of powerups, these consist of double fire toggles, bouncing bullet toggles, and coins which represent health. As well as setting the number of each kind of powerup to be randomly distributed around the maze, you also decide whether or not the powerups are replenished during the game or if once they are taken they are gone. You can even export your favourite configuration to be loaded next time you play Arena.

So, what is the aim of the game? There are a number of different ways you can win Arena - depending on the options you select. If your opponent dies before you, you win. It doesn't matter whether your opponent is killed by sentries, yourself or a combination of the two. If the option is selected you can win if at any stage during the game you have full health - of course for this to work players must not start with full health. There are also circular bases dotted around the maze, the number of which can be specified in game setup. If any one player controls all bases they win.

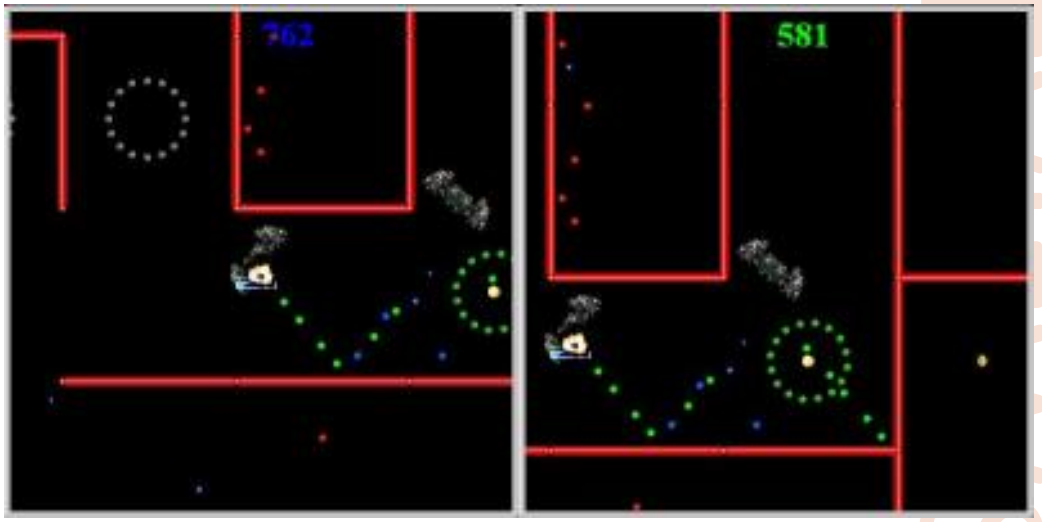
As well as your opponent you also have to watch out for the in-game sentries. These patrol certain areas of the maze in a straight line, constantly shooting out. If you get hit by a bullet you loose health, but beware sentries will also collect powerups to improve their bullets and increase their health if they are in their path. Sentries also collect bases if they come across one, so watch out!

As a two-player game Arena makes use of Game Maker's views function to enable a split screen, one side following each player around the maze. If you're quick you may be able to grab a sneaky glance of your rival's location, but likewise this makes you vulnerable too.

Arena

There are no notable bugs in the game, the maze generator only creates solveable mazes, so there is no chance of being isolated from half of the game. The only slight problem is that, if you try hard enough or are unlucky enough, you can get your ship stuck on the end of a section of maze wall - however this occurs very rarely.

There are no over-used sound effects, however if you are playing lots of games in a row the mission impossible theme tune on the menu and game setup screens may start to get irritating. The game does have background music but this does not distract from the gameplay. Perhaps the shooting sound effect will becoming irritating, especially if your opponent likes to hold down fire



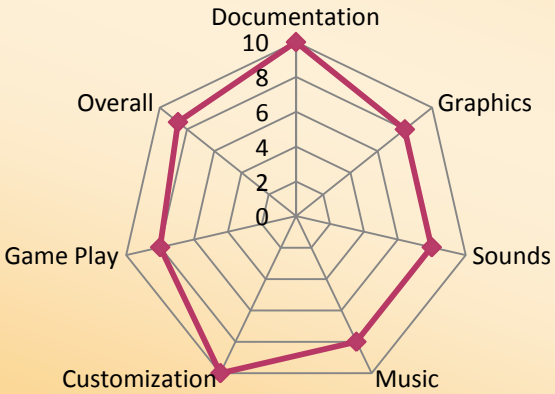
Arena's split screen follows both players around the maze.

The graphics and effects aren't amazing but in a game like this there is no need for professional quality graphics as this would just increase the file size. The sprites used are perfectly suited to the game and nothing looks out of place.

The game is accompanied with an in game help file, and the installation also includes a textual version of the documentation which covers all aspects of the game and explains what everything in the game does.

Conclusion

Ratings



Pros: Detailed instructions included with game. Virtually every aspect of the game can be changed.
Cons: Menu music may get annoying. Small issue of occasionally getting stuck.
Overall: An excellent two-player Game Maker game.

Download Size: 2.5MB
Download Type: Installer
Author: Dennis Toney, Delta 9 Games
Game Maker Version: 4.3
Released: 2/4/2003
Get it: xrl.us/arena

Phil Gamble, GameMakerBlog.com ■

CONCLUSION

Ammo

Creating an ammunition system might not be the hardest task on Game Maker for many, but for the new GM users out there – it's definitely challenging. I'll overview how to create a simple system that monitors the amount of bullet a certain weapon can shoot, and only allow the player to shoot that amount.

Designing the Basic System

Code needs to be added to the player object to:

1- Set the number of bullets the player starts with
2- Monitor number of bullets
3- Subtract a bullet each time the weapon fires
4- Reload the number of bullets when reload occurs

Initialization

To set a number of bullets the player starts with, a variable simply needs to be defined. Here, we will use the variable `ammo`:

```
ammo=50;
```

Display

To draw the current amount of bullets the player is allowed to fire, a simple `draw_text` function is used:

```
draw_text(X,Y,string(bullets));
//edit X and Y
```

Of course, when putting code in the draw event, the player sprite will now disappear, and will need to be drawn manually.

```
draw_sprite(sprite_index,image_index
,x,y);
```

If the player sprite is rotated using `image_angle` or manipulated in any other way, `draw_sprite_ext` should be used instead of `draw_sprite`, furthermore, you could use the `draw_self` script available from GMLscripts.com.

Reloading Bullets

Reloading bullets depends on the way you want bullets to be reloaded. Some games are made so that when the reload button is pressed, the number of bullets is incremented, so the existing bullets won't be lost.

Other games are made so that the remaining bullets are lost, and the number of bullets is then restored.

The code difference is fairly simple. In the first type, the ammo value will be incremented by a certain number:

```
ammo+=50;
```

The second way is what I consider the better way, and that is to set the number of bullets directly to the wanted value, so, if we wanted to set the number of bullets to 50, we use:

```
ammo=50;
```

In reality, that is what happens: the existing ammo in the gun is disposed; a new set of bullets is added to the gun/weapon.

Shooting the Bullets

```
if (ammo>0)
{
    SHOOT COMMAND
    ammo-=1;
}
```

The code is fairly simple – if a sufficient number of bullets exist, the gun is shot and the ammo variable is subtracted by 1.

The Counter Strike interface, showing ammunition

Ammo Cont.

Drawing ammo value in view

When you have a 2-dimensional game taking place in a large world, you will need to use the view feature in Game Maker so that only a part of that word is shown at any one moment on the screen.

Using regular x and y values are relative to the room itself, and not to the view. So, in order for us to correct this issue, the x and y coordinates of the view itself are obtained, and incremented to the X and Y values used for drawing the text.

In Game Maker 6.0 and above, `view_xview` corresponds to the view's X axis and `view_yview` corresponds to the view's Y axis. Before Game Maker 6.0, `view_left` used to correspond to the x axis, and `view_top` corresponded to the y axis.

This means that the previous draw event is now modified to the following in Game Maker 6:

```
draw_text(view_xview+X,view_yview+Y,
string(bullets)); //edit X and Y
```

Limiting the number of reloads

The player can obtain packages to increase amount of reloads in Falcon Squad, a GM game

Other than the fact that you run out of bullets and need to re-fill, one of the 'cool factors' of some games is that you also run out of reloads, and need to obtain certain 'packages' to increase the number of reloads you have. The analogy is simple, and only makes the game more realistic.

Initialization

For that same object, other than initializing the ammo itself, we should also initialize a variable used to count the number of reloads available, and since I'm a creative person, I'll call it `reloads`. You didn't see that coming did you?

```
reloads=5;
```

Reloading Ammo

In the reload event, some changes will now have to occur. We've talked about the single line of code that performs the actual reload above, this line of code will be referenced to as RELOAD, and will be surrounded with if conditions and another line of code to make it all work.

```
if (reloads>0)
{
    RELOAD
    reloads-=1;
}
```

Getting more reloads

A 'package' that gives the player some ammo could simply be an object that lies somewhere on the ground of that world. On the collision event for that object, the player should "take" the package – meaning the package object will have to be destroyed, and the number of reloads should be incremented.

```
with(other)
{
    instance_destroy();
}
reloads+=3; //EDIT VALUE
```

Using multiple weapons

What if you wanted multiple weapons to exist? How could you make bullets and reloads count individually for separate weapons?

Here comes one of the benefits of using **arrays**!

Concept

What will happen is that each variable will become an array, so `reloads[0]` represents the reloads for the first weapon, while `reload[1]` represents the reloads for the second weapon, and so on.

Dawn of the Infested, a Game Maker game, allows the player to choose between multiple weapons

Basically, each weapon will have its own index value, and a new variable will be added that stores the value of the current weapon being used.

Initialization

Initialization is now different; the

Ammo Cont.

number of bullets for several weapons must be initialized, as well as the new current weapon variable:

```
ammo[0]=50;
ammo[1]=50;
ammo[2]=20;
reloads[0]=5;
reloads[1]=10;
reloads[2]=2;
current_weapon=1;
```

Display

This time, only the number of bullets for the current weapon should be displayed. The value of the current weapon variable could be used to return the number of the bullets available for that weapon.

The way this is done is by replacing:

```
draw_text(view_xview+X,view_yview+Y,
string(bullets)); //edit X and Y
```

With:

```
draw_text(view_xview+X,view_yview+Y,
string(ammo[current_weapon]));
//edit X and Y
```

Reloading

The variable `ammo` would be replaced with `ammo[current_weapon]` instead. If you are also monitoring the number of reloads, the `reloads` variable will become `reloads[current_weapon]` as well.

Shooting the Bullets

Same as before, each `ammo` variable would be replaced with `ammo[current_weapon]`. This includes the 'if' conditions and the subtraction of number of bullets for that weapon.

Now, to choose for what type of bullet to be actually shot, a `switch` statement is used:

```
switch(current_weapon)
{
    case 0:
        SHOOT 1 COMMAND;
    break;
    case 1:
        SHOOT 2 COMMAND;
    break;
    case 2:
        SHOOT 3 COMMAND;
    break;
}
```

Getting more reloads

The actual reloading increases the number of bullets for the current weapon, but for increasing the number of reloads for a particular weapon, it becomes different.

Different packages exists for different weapons, and when a player collides with a particular package, this package only increases the number of reloads allowed for a particular weapon, whether or not it is the current weapon.

Multiple objects could be created for different packages, each supposedly for a different type of weapon. The `reloads` variable is then replaced by an array: `reloads[#]`. The "#" could be replaced with a fixed number (in this example 0, 1, or 2) corresponding the weapon this package is related to.

Switching between weapons

Switching the current weapon being used is very simple; all it requires is to change the value of the `current_weapon` variable.

Conclusion

Many – if not most – of the shooter games out there take advantage of such a system of ammunition. I hope this guide has made it easy for some of you to create your own shooter game.

Eyas Sharaiha ■

Martin "FredFredrickson" Crownover

For this issue, I decided to interview Martin Crownover, most known in the GMC as FredFredrickson, the creator of several awesome multiplayer games. Since this issue is more concentrated on multiplayer games, I thought having a developer-oriented discussion with someone with extensive GM-multiplayer experience would be beneficial to all of our readers.

Interview

Many newbie developers want to make MMORPG games, or any other type of multiplayer games. Any words for those people?

I don't like to tell people that making a successful MMO in GM is impossible, because it's technically not, but my advice for people who are beginners with GM and with high hopes for making such a game is to start small. Get comfortable with GM. Take the time to learn how to organize your projects and how to write efficient code, and then go on to make your masterpiece. It'll benefit you in the end, and even if you don't wind up using GM for your magnum opus, you'll at least come away from it with a good foundation for logical coding.

Many multiplayer games have features like chat, etc. What features does a multiplayer game need to have, and which are just 'bonuses'?

It really depends on the type of game. I think that if you're going to make a multiplayer game, you might as well get in all the features you want to get in, and then if you're planning on releasing a demo to the community or friends, get feedback on what works and what doesn't, and go from there. I think that as developers (both indie and commercial) start becoming more adept at creating engaging multiplayer experiences, and comfortable in that realm, we'll be seeing a lot of interesting implementations of it that challenge our beliefs of what is and what isn't a standard feature of multiplayer.

There are existing GM multiplayer capabilities, are they enough for making a good solid game?

To be honest, I haven't used them enough to answer this question with 100% accuracy, but from what I have seen and heard about the built-in Mplay functions, yes, I think it's very possible to create a good, solid multiplayer game using them.

What Game Maker resources do you find best for making a multiplayer game?

Obviously, I'd say that 39dll is one of the best resources out there for people to code online games with Game Maker. Enough people use it that you can get good help with it, and there are some pretty useful examples for it floating around as well. I'd recommend checking out just about all the open source engines I've seen that use it – everyone has their own unique way of programming multiplayer, and every multiplayer game is programmed somewhat differently from the next, so it's good to know a broad range of methods.

Lagging is a serious issue in multiplayer games, any tips on how to minimize this effect?

I'm always having trouble with this myself! But until Ethernet cables that can deliver instantaneous communication are developed, lag is something which will always be something we have to deal with. Basically, since it is always there in some form or another, the best we can do is

↺

GMMovie

This DLL extends the possibilities for playing both audio and video formats, though the DLL is specialized in video formats. The DLL uses MCI, which is available directly from Game Maker itself – but apart from being easier to use, its author says it might be faster. The DLL doesn't create any pauses and is ideal for playing intros and trailers inside the game.

Get it now!
http://xrl.us/2iai

Martin "FredFredrickson" Crownover Cont.

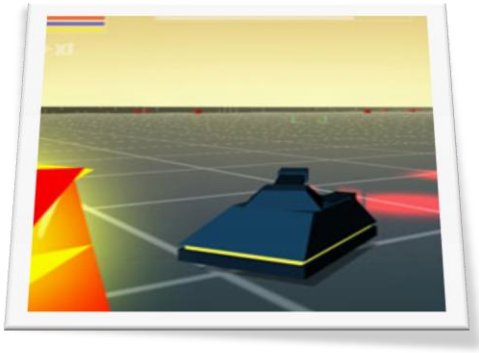
optimize the packets of information we send out, and then hide the rest of the problem. My advice is to go through your game, and figure out what is possible to compute on the client side, and what is not. Often, you can find things in your packets that can be taken out and computed client-side, as well as things that just don't need to be there. If you're clever about structuring your packets, and add in some sort of motion prediction / smoothing for movement (for action-oriented games), you're good to go.

Some multiplayer games create direct connections between players, while others depend on setting up servers, which do you think is best?

I think that as long as one player is the host, or a dedicated server application is used, the game should be fine. I've heard of people trying to set up connections where everyone is sending messages to everyone else, and I can't help but think that this would be a nightmare to control. Set up your game so that all the messages that players send and receive are routed through one central point. It may be hard to wrap your mind around this concept (it still is for me at times), but ultimately it will give you far greater control over what happens in your game, and how smooth of an experience it is for people.

Hover Tank 3D was the first multiplayer game I've seen you work on. How much have your games evolved since you started?

My games themselves have evolved out in different ways - each project I've done since HT3D has filled a gap for me, so to speak. I think that more than anything else, the code underneath the hood of my games has evolved as I've gone from project to project. There is a distinct difference between the way I coded HT3D back in 2005, and the way I would code it now. That's one of the reasons why I haven't re-opened the project to recompile it for GM7 / Vista - I don't want to look at the code and start tinkering with it! A progression in code practices and knowledge is good though, and I think that it will serve me well when I get around to working on the sequel to HT3D again.



Most of your multiplayer games have both single and multi-player modes. Why do you think this is necessary?

Even though most of the industry seems to be shifting to the multiplayer side of things, I think that it's still important to include the players with slow net connections, or who just wish to play alone. I can't say that my more recent work caters to this group as much as HT3D or Evolites have, but I am planning a pretty comprehensive single player experience for Falcon Squad, and I hope that people will play it and enjoy it.

GMC mods are always closing wish-list topics -- what is your GM wish list?

I try not to think too much about what I wish GM was, because I think I'd get too wrapped up in wishing it were different here and there, and then end up being disappointed with what it is. Don't get me wrong though, it's an awesome program, and I will continue using it until I run out of reasons to! As for my wish list, I would really like to see (as would many others, I am sure) a speed increase, more 3D functions (like extended model type support, better support for animations made in outside programs, etc.), and better security. Of those three, I think the one I would appreciate the most at this point would be the added 3D functionality. I know GM isn't really the best app for developing 3D games, but it'd still be nice, and I enjoy the challenge it presents with its speed and functional limitations. I imagine that in some ways, it's almost like developing for a console with limited specs.

Martin "FredFredrickson" Crownover Cont.

What makes a multiplayer game popular? Is it just quality?

I think quality is always big part of the success of a game, but for multiplayer games specifically, giving the players a fun way to interact with each other is key. As an example, most multiplayer games seem to revolve around players killing each other in some way or another, but if you give them an assortment of weapons to do it with, the game becomes more interesting.

I guess most of our readers already know about Reflect Account System. For those who don't what is it? Is it a member system, server system, client system, or what?

The Reflect Account System is an account system that was developed to make finding multiplayer games easier for people. Most people who know of the system know of it because they have used it to connect to other people in games like Aces High Over Verlor Island and Sapphire Tears… since it's based on a web server, it doesn't host these games itself, but it acts as a master server list with a lot of extra perks that the account system affords it.

Built off of some things I had originally put into Hover Tank 3D for multiplayer, it basically uses an account system to identify / authenticate players, and give developers a set of tools that are easy to use, but powerful as well. For example, once you've got the login system in place for your game, using the Reflect Account System, you can upload high scores with one script, and then download them and parse them with another. You can query the server for listings for your game, download nicknames and player colors from the server. It's a really handy system, and I think it can add a bit of professionalism to your game.

Not all Reflect-Enabled games are yours, yet you don't let anyone make a game taking advantage of the Reflect Account System. What criteria do you look for in a game that makes you decide whether or not will it make a good reflect-enabled game?

Really, I just look for people who are serious about making their game, and who have shown in their prior work to have the talent and drive to actually finish the project. I know I am guilty of it myself, but having a lot of unfinished games on the system isn't good, so having the ability to complete the game, and the will to polish it 'til it shines are probably the two most important things.

So, any plans on going commercial with (some) Reflect Games?

Eventually… you never know! I've been very busy with my work lately, but eventually, when I have more time to program with GM again, you might see me attempt a more lucrative release. Overall though, I'd like for the service, and the bulk of its games, to remain free.

Though I told you the interview will be developer-oriented, I can't help it but ask: is HT3D 2 coming down the pipe anytime soon? You haven't posted in the Reflect topic about that for a while.

Well, like I said, I have been quite busy lately. I recently decided I needed to overhaul Falcon Squad, and make it into something a little different than what it currently is, so maybe I will take the extra time to play around with some HT3D 2 stuff again. I am slightly interested in learning Xtreme 3D, or one of the other GM 3D DLL's, so who knows… maybe HT3D 2 will be even more graphically intense than I originally planned.

It's a wrap! Is there anything else you would like to add?

Not really, just thanks to you for the interview, and thanks to everyone who read it! I truly appreciate it.

Conclusion

So that was it – my interview with one of the most respected game makers out there, especially in the multiplayer area.

Be sure to check out Martin Crownover's games by visiting the Reflect Games website here: http://reflectgames.com/.

Evas Sharaiha ■

Game Maker and the Registry TUTORIAL

When creating games, there are a few ways to store small amounts of data. This can be done by storing the data in external files, but another way to do this is to use the registry to store data. The registry also has some advantages. This article will explain the use of the registry and how to access and change it with Game Maker.

The registry

Now what exactly is the registry? Actually, it's a database to store small data. This database is present on each computer that has Windows on it. The great thing is that there's a very easy way to access and view it. Windows comes with a handy tool: the registry editor. You can open the registry editor by typing "regedit.exe" in the command prompt. And there it is! You'll notice that the registry is subdivided in five main keys:

- HKEY_CLASSES_ROOT: this looks like it is a complicated key, but it's actually a very interesting one, as will be explained later in this article.
- HKEY_CURRENT_USER: as you might expect, this key contains data about the current user on a computer (related to user profiles).
- HKEY_LOCAL_MACHINE: this key looks like the above, but it contains data that applies to all users on a computer.
- HKEY_USERS: this key contains user profiles, not accessible with Game Maker, though
- HKEY_CURRENT_CONFIG: contains data about hardware profiles.

If you've checked out those keys, you will have noticed that the registry editor displays the keys in a tree structure. Keys can contain subkeys or just values (named variables with a value). This structure can be compared to the directory structure of Windows. Keys then are directories and values are files. Directories can contain subdirectories. And just like directories can contain multiple files, keys can contain multiple values. The values can be of different types. But since GM just supports two types (REG_SZ and REG_BINARY), these won't be explained further.

So the registry exists, but the registry contains data, and data is usually stored in some files. This is no other with the registry. The data from the HKEY_CURRENT_USER key, for example, can be found in "C:\Documents and Settings\<your username>" (if C: is the system drive, of course), in "NTUSER.DAT". It can't be opened, though. Windows doesn't let you do that. No registry editing that way.

The solution is simple: let's use Game Maker to do it.

GM's registry functionality

GM's registry functions allow for many possibilities. First of all, a remark: editing the registry might be dangerous if you don't know what you're doing. Here in this example, we will assume that we know what we're doing.

So let's look at GM's registry functions. GM has 11 registry-related functions. The first 5 allow for few possibilities. Why, you wonder? These functions only work with specific keys. Suppose you want to write a value to the registry and your game id is e.g. 782526. You could use a piece of code like this:

```
registry_write_string("A string ",
"string content ");
```

Then Game Maker will create the following key in the registry (you can check this in the registry editor):

```
HKEY_CURRENT_USER\Software\Game
Maker\782526
```

All strings and real values GM writes to the registry using the normal registry functions appear in that key. Reading values also reads values from that single key.

It is clear that these functions are very limited. You only get access to a single key in the registry. To get access to all keys of the registry, we will need to use the extended registry functions.

The extended registry functions almost have the same name as the previous ones, but you need to add "_ext" to the function. Also, these functions require another argument: the key. The previous functions didn't allow you to specify the key in which to write the values. These functions do. An example:

```
registry_write_real_ext("Software\My
Game ", "PlayTime ",6);
```

The function above writes the following key to the registry:

```
HKEY_CURRENT_USER\Software\MyGame
```

And, it writes the following data to that key:

```
PlayTime      6
```

You can again check this in the registry editor. When you open the Software

key, you'll notice the MyGame subkey, containing the real value (REG_BINARY) PlayTime set to 6.

Now there's still one thing that needs to be explained: setting the root. Up till now, we've only written values to the HKEY_CURRENT_USER key (which is the default used by Game Maker). This can be changed by using the `registry_set_root` function. This function requires only one argument, a real value indicating the registry root:

0. HKEY_CURRENT_USER
1. HKEY_LOCAL_MACHINE
2. HKEY_CLASSES_ROOT
3. HKEY_USERS

And now, you have full access to all keys of the registry. A small remark here: as you might've noticed in the examples, subkeys can be passed by using the "\" sign. That way, you can get the value of values in subkeys.

Reading values from the registry happens in exactly the same way.

Although these functions allow you to read from and write to the registry, they can't be used to e.g. get the number of subkeys in a key or the number of values in a key. Too bad, but the current functions are more than sufficient to add some nice functionality to your games and programs.

Tips 'n tricks

File association

By writing some data to the registry, you can associate files with your game or program. The registry key where all file associations are stored is the HKEY_CLASSES_ROOT key. You can check this in the registry editor: the HKEY_CLASSES_ROOT key starts with a huge list of file extensions, followed by an even longer list of file descriptions. Some keys and values are required:

- A key with the file extension is created e.g. "HKEY_CLASSES_ROOT\.gmf". Whenever a key is created, is always has a default value (which is actually <no value>).
- This default value is then set to e.g. "GameFile". Now what does that mean? It is the name of the file description that can be found lower in the list. Actually, the only thing this key does is refer to a key that contains the real file association information. This latter key also contains some subkeys.

So how can this be achieved by using GM's functions? Let's say you have a game that can open the ".gmf" file extension. The first thing you need to do is set the registry root to HKEY_CLASSES_ROOT:

```
registry_set_root(2);
```

From now on, all data will be written to HKEY_CLASSES_ROOT. The first key to write is the key named ".gmf":

```
registry_write_string_ext(".gmf","",
"GameFile");
```

This piece of code writes a new key "HKEY_CLASSES_ROOT\.gmf" to the registry and sets the value "GameFile" (you can choose this name yourself) to the default value .When you want to change the default value, you need to pass an empty string as the value name to the function.

Now the "GameFile" key needs to be created:

```
registry_write_string_ext("GameFile ",
" ", "A file type used by my game ");
```

The default value of this key is used as a description for your game file. Now subkeys need to be added to this key. The first one sets the icon for the file:

```
registry_write_string_ext("GameFile\De
faultIcon","","C:\Program
Files\MyGame\icon.ico");
```

"DefaultIcon" is the key where the icon file path is stored (you cannot choose this name yourself, it's predefined). Also note that it is again the default value that is used to store the filename of the icon file. Now the "open" key, that is the key that contains the information on how to open the file, needs to be created. The "open" key is a subkey of a key named "shell". To create it:

```
registry_write_string_ext("GameFile\sh
ell\open","","Open with &MyGame");
```

This piece of code creates the "open" key as a subkey in "shell", which is also created (you cannot choose these names yourself). As you can see, the default value is this time set to "Open with &MyGame". This text will be displayed as an item in the menu that is shown when the user right-clicks the file icon on his desktop. The "&" sign does something special. It indicates the letter to be used as a keyboard shortcut. In this case, it would be the M of "MyGame". So suppose the menu is open and the user does not use the mouse to select that menu item, then he can press M (not Shift+m) on his keyboard. That will give the same result.

GM and the Registry Cont.

Now there is one key that needs to be added:

```
registry_write_string_ext("GameFile\
shell\open\command","",'"C:\Program
Files\MyGame\mygame.exe" "%1"');
```

The key is called “command” and is a subkey of “open” (you can’t choose these names yourself). Note that 2 parameters are passed this time. These need to be put between double quotes. As a consequence, single quotes must be used in the piece of code. The first parameter is the link to the game’s executable. The second one is “%1”. This requires an explanation. At startup, some parameters can be passed to the game. These parameters all have an index. The “%1” value indicates that this parameter will have index 1. Voilà, that’s all for the file association. It might be necessary to log off and log on again for the changes to take effect.

Okay, so the file association is done. Windows knows which icon to display for the “.gmf” file extension and the path to the executable to open it. The problem is: the game doesn’t know yet that it needs to open that file on startup.

For that, we need to use another Game Maker function: `parameter_string(n)`

The `parameter_string` function returns the string of the n-th parameter that was passed to the game at startup. Since we gave this parameter index 1, it can be retrieved with:

```
filepath=parameter_string(1);
```

The filepath variable now contains the value of parameter 1. The value of that parameter is the path to the file the user opened. And now the problem is solved. Windows will open the file with your game and your game knows which file the user wants to open.

The Volatile Environment key

This key can be found in the `HKEY_CURRENT_USER` key:

```
HKEY_CURRENT_USER\Volatile
Environment
```

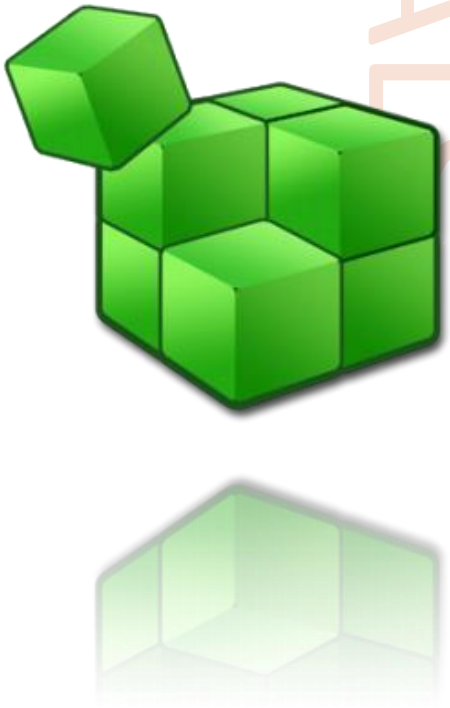
It contains some interesting values e.g. the current user’s application data directory (`APPDATA`), homedrive (`HOMEDRIVE`) and the user’s directory (`HOMEPATH`).You can view all these values in the registry editor.

The Environment key

This key can also be found in the `HKEY_CURRENT_USER` key:

```
HKEY_CURRENT_USER\Environment
```

Interesting values in this key are the user’s language (`LANG`) and the link to his temporary directory (`TEMP` or `TMP`).



Conclusion

The registry contains a lot of interesting information about everything on the computer. Unfortunately you need to know in which keys that information can be found. Game Maker’s registry-related functions can be used to get useful information from the registry, but also for writing to the registry e.g. file association and storing data from your game.

Bart Teunis ■

QUICK REVIEW

39DLL

39dll is a popular socket DLL which gives games that use it the power to access Windows Sockets and therefore multiplayer capabilities. Many popular multiplayer Game Maker games use the 39dll as a method of creation of servers and communication between different players of a game. Great DLL, constantly updated, and high quality – recommended.

Get it now!
http://xrl.us/2kmi

Instant Collisions

Hello, in this article, you will learn the common mistakes in 2D shooters.

Common Mistakes

Some of the most common mistakes made in these games are the bullets. Some people use the retro style bullets where the bullets move rather slow, but most people don't like waiting for that little black ball to move toward their target. This is a common mistake in games. This can easily be fixed with something called "rangefinding" it finds a range of pixels and returns true or false if the range ends before it hits its maximum distance. Most people do not know how to make this but it is actually very simple.

Starting the Script

Start with a script in your game, name the script: "instant_collide".

Now so it is user friendly and a lot like a function, we will add arguments to the script. Add this code into the script:

```
x1 = argument0; // Source X
y1 = argument1; // Source Y
dir = argument2; // Direction to shoot
EX: point_direction(x,y,mouse_x,mouse_y);
maxdist = argument3; //maximum
distance. so the game dosent crash if it
goes out of the room.
obj = argument4; // obj/flag to stop
at. i always enter "solid" (without
quotes)
dx = 0; // just to define destination
x,y
dy = 0;
col = 0;
var xx,yy,dist; // make the variables.
dist = 0;
```

Do not include the quotations. Now we have made our variables, and are ready to get into the good stuff.

That code simply defines the variables and chooses which entry will take place for the variables. Ex:

```
instant_collide(12,4,90,1000,solid);
```

That example would set the X1 to 12, Y1 to 4, DIR to 90,the max distance to 1000pixels, and the object to stop at to SOLID

The SOLID means the range will stop at a solid.

The Script's "Brain"

Now since we know what the code above does, we can now go into the real coding.

We will create a do{} to repeat the code until a certain point. This can be helpful in games.

Put this code into your game:

```
do {
    dist+=1;
```

The dist+=1; means that the script scans each pixel to find its destination, 4 works great but sometimes it goes through the blocks, change this around to change the performance of the game. Higher value = better FPS, Lower value = worse FPS.

Now we will add more code into that `do{}` event so it not only just adds on to a variable.

Add this code into your script:

```
    xx = x1+lengthdir_x(dist,dir);
// just the direction of fire and
stuff.
    yy = y1+lengthdir_y(dist,dir);
    dx = xx; // destination X
    dy = yy; // destination Y
```

The lengthdirs are just a directional virtual line, they are updated every time the `do{}` event is run so they add on pixels to the range.

The DX and DY are the return values; these are the Coordinates you will use later on in the script.

N_menu DLL

The N_menu DLL allows the user to create real Windows menus for inclusion in a game/tool. Many capabilities exist for adding images, submenu items, events, and more. The example is available only in .gmd, but could be easily converted to .gm6 and .gmk through Game Maker. A GEX extension is also available.

Get it now!

http://xrl.us/nmenu

Instant Collisions Cont.

Now we need to end the do event with a bracket and an `until();`

Insert this code into the script:

```
}
until(position_meeting(dx,dy,obj) or
dist>maxdist);
```

That checks if the position of the range has hit a solid object, or the object you preferred, and it also checks if it reached its max distance, so the game doesn't freeze if you don't hit anything.

Finishing the Script

Now what we need to do, is make the return variables and simply make an `if()` event to make it more user friendly to check if it has collided or not.

Add this code into your script:

```
if(position_meeting(dx,dy,obj)) { //
if the destination has collided with
something.
    col = 1; // yes it has.
} else {
    col = 0; // no it hasnt.
}
```

The code above checks whether it has collided or not. This can be useful.

Now we need the returns. These are what we need to even create a bullet.

Add this code into your script:

```
return dx; // return it, so we can use
to draw.
return dy;
```

And that's all of the script!

Your script should look like this:

```
x1 = argument0; // Source X
y1 = argument1; // Source Y
dir = argument2; // Direction to shoot
EX: point_direction(x,y,mouse_x,mouse_y);
maxdist = argument3; //maximum
distance. so the game dosent crash
if it goes out of the room.
obj = argument4; // obj/flag to stop
at. i always enter "solid" (without
quotes)
dx = 0; // just to define
destination x,y
dy = 0;
col = 0;
var xx,yy,dist; // make the
variables.
dist = 0;
do { // we use this do {} statement
to keep going each pixel.
    dist+=1; // add on to the
distance so we dont stop until we
hit something.
    xx = x1+lengthdir_x(dist,dir);
// just the direction of fire and
stuff.
    yy = y1+lengthdir_y(dist,dir);
    dx = xx; // destination X
    dy = yy; // destination Y
} until(position_meeting(dx,dy,obj)
or dist>maxdist);
if(position_meeting(dx,dy,obj)) { //
if the destination has collided with
something.
    col = 1; // yes it has.
} else {
    col = 0; // no it hasnt.
}
return dx; // return it, so we
can use to draw.
return dy;
```

Executing the Script in Game

Now we want to run the script in-game, make 3 sprites:

- spr_block
- spr_player
- spr_bullet (2x2) origin=1x1

Use the settings provided for the sprites.

You can set your player sprite to whatever you want. The same will happen with the block.

Now create the block object, and set its sprite to the spr_block.

Don't add any code to the block; just check the flag "Solid". Then create the bullet object. Set its sprite to spr_bullet.

Add a STEP event to the bullet obj and add a code block. Add this code into the Step event:

Instant Collisions Cont.


```
if(!place_free(x,y)) {
effect_create_below(ef_smoke,x,y,10,
c_gray);
    instance_destroy();
}
```

This code checks if it has collided with anything. If it has, it creates a puff of smoke, and destroys itself. Now make sure the bullet isn't solid.

Create the player object, and set the player object's sprite to spr_player.

You do not need any player physics for this, add it if you want though.

Add this code into the create event:

```
free=1;
shooting=0;
```

The free variable tells our player that the place is free. Otherwise it would say it isn't free before you shoot.

Shooting is set to 0 for later uses, it's just basic creation.

Now since we are done with the create event, add an End Step event and put this code in there:

```
if(variable_local_exists("col")) {
// see if we are shooting, so it dosent
cause an error.
    if(col=1) { // if col=1 (if the
line has collided with an object)
        free=false;
    } else { // if col=0 (the line
hasnt collided)
        free=true;
    }
}
```

That checks if we are shooting, and then sees if the col variable is true or false, so we can tell whether it has or hasn't.

That's all for the end event.

Now Create a Global Left Pressed event and add this code into it:

```
var dir;
dir=point_direction(x,y,mouse_x,mous
e_y); // set direction to point at
mouse.
shooting=1; // set to one so we can
get rid of the variable
instant_collide(x,y,dir,1000,solid);
instance_create(dx,dy,obj_bullet);
alarm[0]=1;
```

Make sure to set the 3rd argument for the instant_create to the bullet object of your game.

Also, you're probably thinking "What?! An alarm event??"

True, we will create it now.

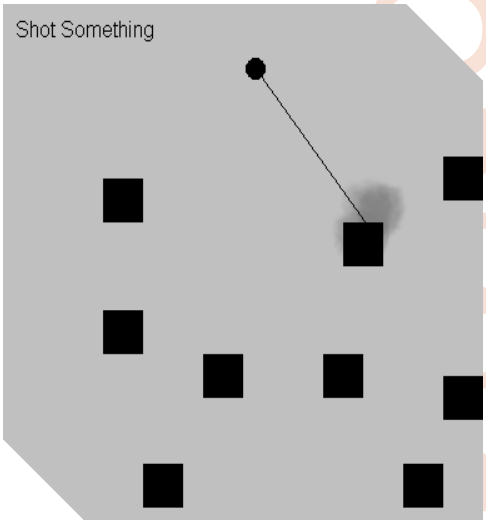
Create an alarm[0] event and add this code to it:

```
shooting=0;
```

That sets shooting to 0 so it doesn't keep drawing the line later on in this article. Now Finally!! Finishing the script, add a draw event into the player, and add this code into it:

```
if(free=false) { // if free is false
  draw_text(10,10,"Shot Something");
// say we have collided
} else { // if free is true
    draw_text(10,10,"Shot nothing");
// say we didnt collide
}
if(shooting=1) {
    if(variable_local_exists("dx")
&& variable_local_exists("dy")) {
// we check if the variables exist,
so we dont get an error.
        draw_line(x,y,dx,dy);
    }
}
draw_sprite(sprite_index,0,x,y);
//simply draw the sprite.
```

Now we are ready to set up the room! Create a new room and add the block objects all around and create a player. Your instant collisions are complete!



Conclusion

Instant collisions give many advantages over regular collision events, especially when it comes to bullets in top-down games.

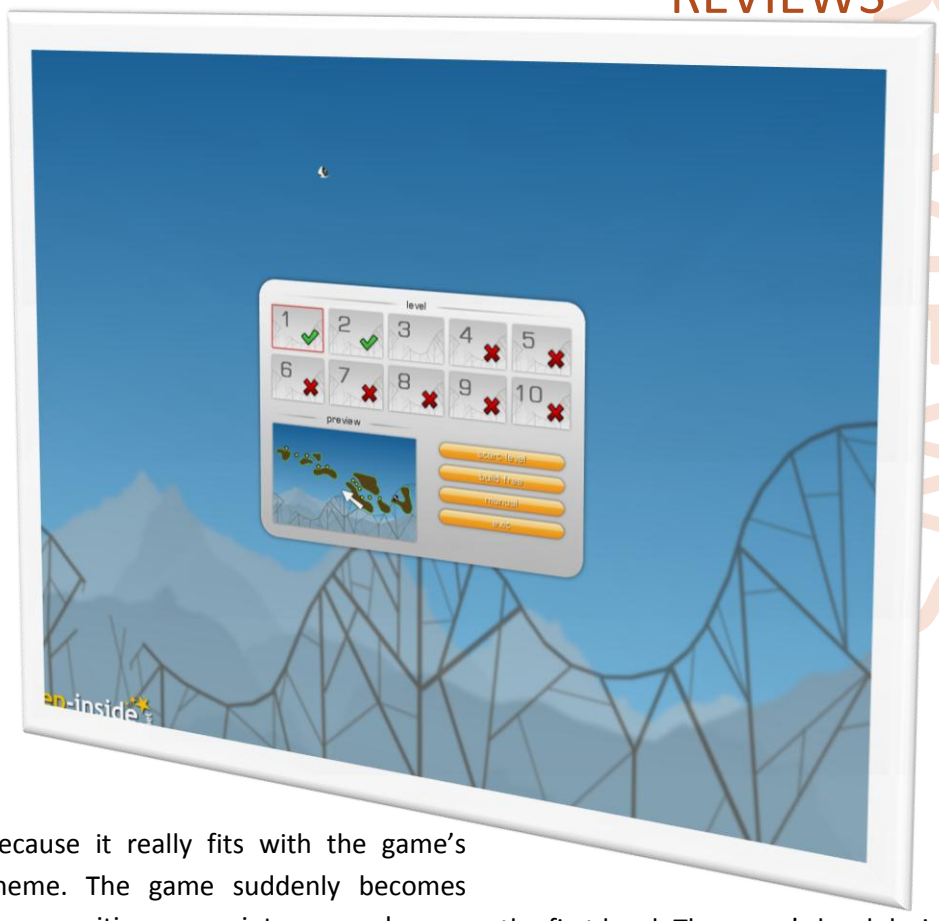
If you want a bullet with a realistic speed, chances are – the bullet will go through the body without even colliding with it!

In order for a bullet to collide with a body normally, it should be slowed down, considerably – this makes the game unrealistic, and sometimes – annoying; no one would really want to wait a couple of seconds for a bullet to kill the enemy – it should be done instantly; bullets are fast!

This script allows you to restore realism to your top down shooter games by making bullets much faster.

mr.gibblet■

Coaster Rider



CoasterRider is an excellent game by Bl@kSp@rk, as featured on the YoYo Games website. What is amazing to me is that such an excellent game never had a GMC topic, and virtually got no attention within the GMC.

Graphics

The game has **excellent** graphics! Whether it is the Pegasus itself, the trees, the smooth grass and pieces of "floating land", the starts that you need to collect, or the effects you get when you collect them. Amazing. I will be a fool if I gave it anything less than a 9.

Graphic Effects

The game has subtle graphic effects, only here and there. The good thing is not overused, and it is true what they say "less is more"… but that's too little! The game gets a 7/10 in Graphic Effects.

User Interface

Whether it is the menu, tools, or options – it functions so well, extremely easy to understand, and it's simple! The game has only few UI concepts, and it builds on them – it doesn't confuse the user with toolbars, menus, tooltips, windows, etc – all are just simple buttons with amazing graphics.

Even highlighting buttons and switching between different tools could become a rewarding experience! The game gets 10 out of 10, definitely!

Music

The game's music is soo good! There are 5 music items, all high-quality OGG files. The music makes the game more fun, because it really fits with the game's theme. The game suddenly becomes more exciting, more intense, and more capturing – excellent work! 9 out of 10.

Sound Effects

Too little sound effects, but when they are heard – it's great! Clicking buttons, colliding with stars or loops that speed you up, all that triggers sound effects. But I wanted more, like if the train bumps on the ground at high speeds, etc – I needed something that made the game feel more realistic, that something didn't exist. 7 out of 10.

Level Design

Levels are well-designed. The first level is easy, and then levels incrementally become harder. For me, level 3 is way too hard, but that's where great gamers are separated from the rest – good job!

However, Level 1 was not "all too easy"; I would've expected it to be easier for the first level. The game's level design is rated 9.

Overall

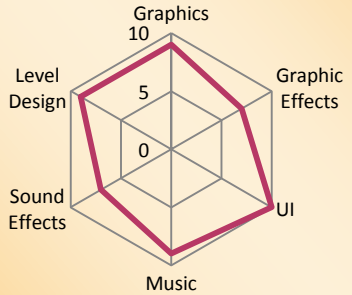
The game's overall score is 8.5 out of 10. That's a shame, because it could've gotten a lot better if Sound effects and Graphic effects are enhanced. Nevertheless, great work indeed!

Eyas Sharaiha ■

Conclusion

Get it now: http://xrl.us/coasterrider

Download Size: 12.3 MB



2P Shooter

What they say

Select from 6 heads, 8 bodies, and two guns to duel it out between two players. Now with 4 weapons, the old MK9's and M16, plus the new custom 'Riotgun' and a nice little sword!

Quick Review

In this split-screen two player game you and your friend each control a gray stick man inside a platform environment. Various weapons are spread throughout the room which can be collected and used to inflect harm on your opponent. The aim of the game is quite simply to kill your opponent before they destroy you.

Different types of weapons can be found around the room, each causing different amounts of damage and with their own unique features. For example individual weapons have different fire rates and swords can only be used close up but are quite destructive.

You can only shoot left and right, and once you've found a decent weapon you may as well stick with it as there are no limitations on the ammo you can use during the game.

The biggest criticisms expressed at YoYoGames are that the game should have a single player mode and something should be done to fix the jumping bug. I certainly agree with the second point – it is possible to get stuck on the side of platforms within the room. This is a pretty major bug and I'm surprised the game was released with this, especially seeing as this is the second version of the game to be released.

The support for a single player mode is something I don't necessarily agree with. The game works well as a multiplayer game – unless a decent and extensive single player mode can be added this will easily be outshone by the multiplayer option.

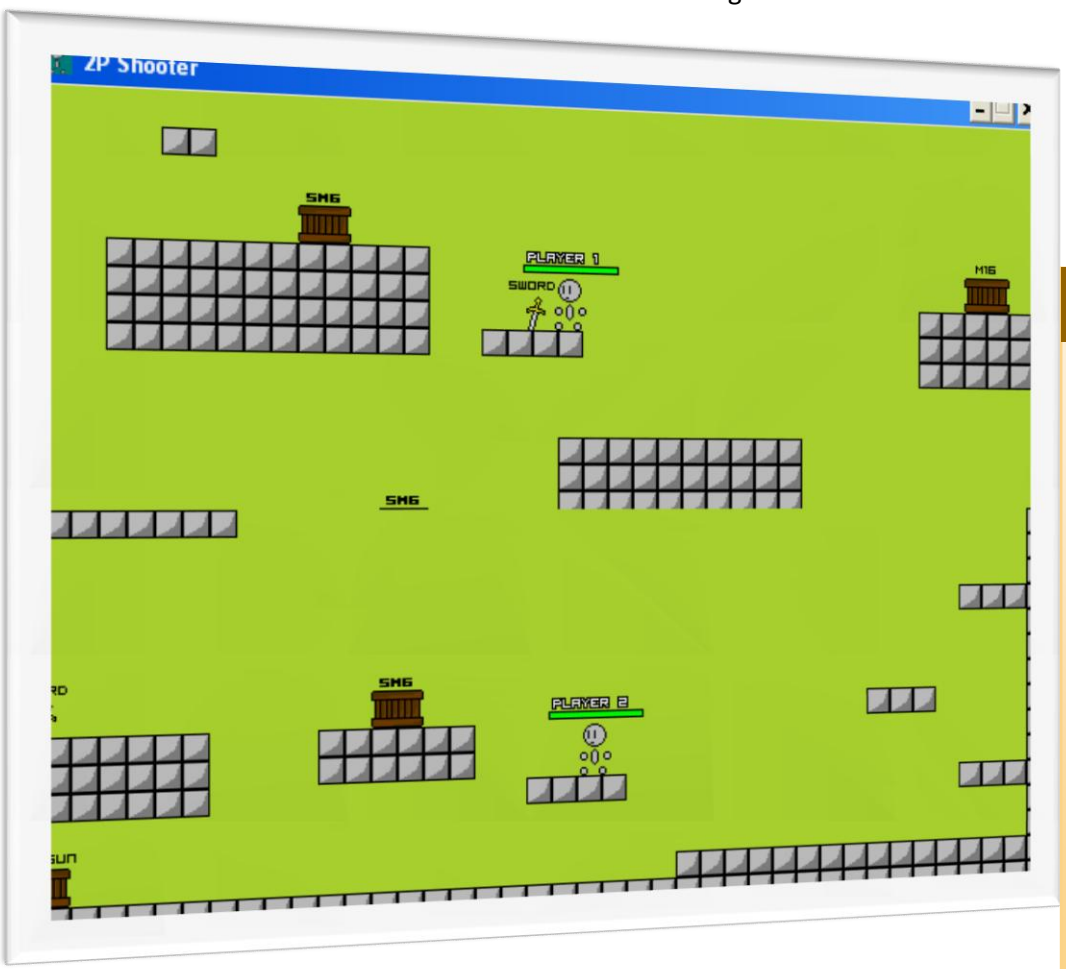
There are several options that can be customised including the appearance of both players; there are also two rooms to choose from. To be honest this doesn't make much difference at all to the game play but at least the options are there for you to choose from if you wish.

The game has suitable music and sound effects which complement the game well. I'm not sure if they are original but they do seem pretty generic. They are a nice touch and without them the game would be much duller.

Another criticism I have is the games menu. They consist of very small graphics which are hard to read and click. It should be simple to have a well designed and easy to use menu but it seems this has been overlooked in an attempt to improve the aesthetics.

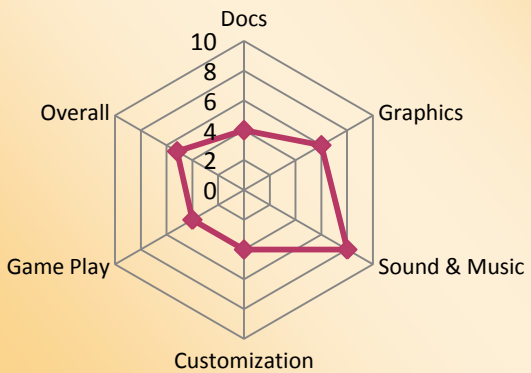
Phil Gamble,

GameMakerBlog.com ■



Conclusion

Get it now: http://xrl.us/2pshooter
Download Size: 1.3MB
Author: Jack Sanders
Released: 16 June 2007



Until Next Time!

And that was it! Issue 5 of MarkUp magazine and our first issue since the YoYo Games endorsement went public! We hope you see this issue as an improvement, as we have got a lot of support and feedback from the community. We truly hope this issue was satisfying in every way.

We started talking to YoYo Games and thinking about the endorsement since April this year, and it has been exciting so far! But more exciting than knowing what was about to happen, and waiting for the right time to announce it was the feedback we got from the entire community once the announcement took place.

There are many people we'd like to thank for all the hard work they have done for us. The YoYo Games staff is one of them, and all the MarkUp staff thank them for bearing with us, and for the wonderful job they carried out. We also appreciate the support that the GMC moderators have shown us.

Also a big 'thank you' to GameMakerBlog.**com**, GMLscripts.com, GameMakerResource.com, and hopefully soon – GMBase, they've been a great help. We also want to show our gratitude to everyone who has contributed to MarkUp, including our GMking.org 'Journalist' staff, and everyone else who has wrote for MarkUp.

But more importantly, we at GMking.org would really like to thank you – our readers, for giving us feedback, support, and comments; the help was truly appreciated!

If MarkUp is to grow, we'll need your help! You could help us by contributing anything to the magazine, whether it is articles, art, or any other form of help. To contribute to the magazine you could visit the MarkUp forum here, or by directly contacting either Robin or Eyas via e-mail (see MarkUp site) or PM.

What does the future hold for MarkUp? Well, as our readership increases, we expect to get more comments, feedback, and hopefully support. We hope our future issues will be of higher quality – and that could only happen by your much-valued contributions!

Once again, thanks for your support!
The MarkUp Staff

Be sure to check out...

It's a song! It's a sound clip! No, wait -- it's an audcast! That's right; we at GMking.org like to keep ourselves very busy! Since June, we started releasing our weekly netcast (we prefer AUDcast), talking about Game Maker under the name "GMPod".

The audcast is approximately 20-30 minutes long, and in it we discuss various issues occurring in the Game Maker world, and give advice to our listeners when it comes to using Game Maker.

We have received amazing support from the Game Maker Community! The great support has motivated us to continue working on the audcast and improve it. We'll be soon talking with some notable community members and we'll start adding some fun to the audcast as well!

Thank you for your amazing support so far, and please: keep those suggestions coming.

GMking.org Let them make games!

Markup is an open publication made possible by the contributions of people like you; please visit markup.gmking.org for information on how to contribute. Thank you for your support!