A GMking.org Publication

MarkUp

April, 2007

Addictive Games

Tips on achieving every game makers dream! (pg. 14)

Interview with Vertigo Games

We talk with David Galindo about his past, present and future as a game maker (pg. 12)

Mr.Chubigans

Plus finding the treasures in free resource sites, exclusive review of Sapphire Tears, Game Manuals and More!

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!

Editor's Desk

Navigating Free Resources

With the overwhelming deluge of free resource sites, and the underwhelming dearth of truly good free resources, game creators are often left on a never ending merry-go-round of sites, and content.

There are some important things to remember before going on a resource scavenger-hunt; namely, you can't take everything you see. Copyright plays a big role in what resources you can use. You need to make sure the author of free sprites, fonts, etc will allow use of their materials in a game. Most authors of these kinds of resources will only allow them to be used in free games.

There are also a lot of free tools for game developers, IDEs like Game Maker for instance.

Here is an *extended* list of some of my favorite resource and tools sites.

Fonts

Font Freak
http://www.fontfreak.com/fonts-a.htm

Free Font Search Engine
http://www.searchfreefonts.com

Terragen

Backgrounds

stock.xchng
http://www.sxc.hu/

terragen
http://www.planetside.co.uk/terragen

Sprites

YoYo Games Resources
www.yoyogames.com/make/resources/

The People's Sprites
http://www.panelmonkey.org/

Open Clipart
http://openclipart.org

Music/Sounds

The GMC keep a excellent list of music and sound sites here:
gmc.yoyogames.com/?showtopic=22863

Graphics Editing Tools

GIMPShop
http://plasticbugs.com

Paint.NET
http://getpaint.net

Blender
http://www.blender.org/

Summary

You can get just about everything you'll need for your next game project for free, and you can also get some excellent non-game related freebies as well. Such as the http://openoffice.org office suite, http://getfirefox.com web browser and http://free.grisoft.com free anti-virus.

You can also find a wealth of free game resources in the gmc.yoyogames.com forums. And a lot of excellent free tools at sf.net and download.com.

See you next month!

Robin Monks

We're Flabbergasted!

We've been amazed by the response to our first two issues! Here are some of the great comments that we've received from our readers:

"Excellent in-depth going product with very nice GFX." --Daniel-Dane

"Cool magazine! I dunno how I missed the 1st issue of this." --XA-9

"The articles are very professional; I love the design. A bit short, but very good overall. Keep it up!" –benetonmovie

As always, we want to hear your feedback! Just sent it to devlinks @ gmail . com!

Our Contributors

Robin Monks	Editor
Eyas Sharaiha	Editor
Veeti Paananen	Writer
Bart Teunis	Writer
Stephan	Writer
Mike Connor	Writer
Phil Gamble	Writer

Table of Contents

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

C# Hello World Tutorial

I previously published a Hello World tutorial for C++, and I've decided to make another tutorial for the successor of C++: C#.

For this tutorial, you will need to have Microsoft Visual Studio 2005 installed for C# (Microsoft Visual C#). You can get the express version for free from the Microsoft here (xrl.us/MSVCBExp) Highly Recommended.

Starting a project

To teach you about code, it'll be best to start a completely empty project and see how it all falls in together. To start an empty project, choose "New Project…" from the File menu. Then choose the "Empty Project" icon (see screenshot below), and give it an appropriate name.

After you click "Ok", you will notice a "Solution Explorer" pain appear on the right side of the screen. Choose your solution name (project name) and right click it to see a list of different things you can do with it.

After right clicking it, choose the "Add" item from the menu, and from its submenu select "New Item".

The dialogue that will pop up will allow you to select different types of items. Choose "Code file".

Open it and you'll see they were nice enough to add a little bit of code to it. Blank the code file, since this tutorial aims at teaching you the cornerstones of C#.

Code file – initialization

Unlike C++, in C# you don't need initialize iostream, string handling functions, or any other type of header that you don't want to. The code is properly run-able as is.

However, most functions require adding some text before the actual function name, for example (System.) before input and output in consoles.

So, to make things easier, we use the "using" command:

```
using System;
```

This will save all the "System." when calling functions.

Initializing class and main function

Functions must belong to a certain **class**, so before making your main function – which includes all the code in it – you will need to create a class.

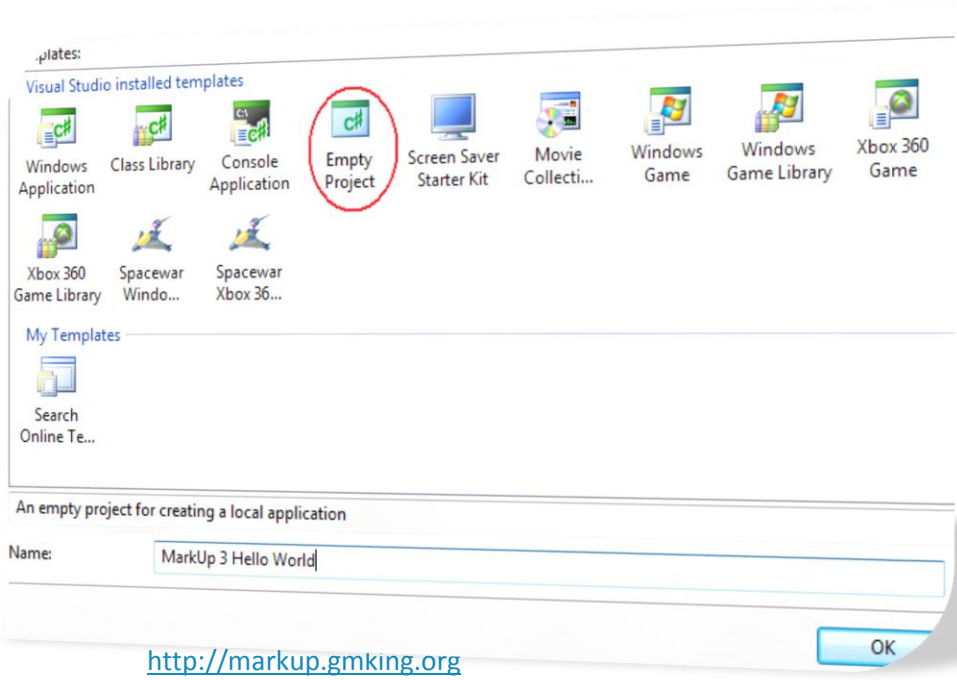
```
class GMkingTutorial
{
        ...
}
```

The class will contain the functions that we will need to use.

Now that you have a class, you can put the Main() function in it.

The Main() function

The main function acts as an entry point for the whole program. It is the single function that is automatically called by the program to begin execution. More advanced programs use the main functions to call the rest of the functions, but the main function itself needs to exist in order for a program to



C# Hello World Tutorial Cont.

execute.

To initialize the main function, you need to write the following:

```
static void Main()
{
        …
}
```

Right know, you really don't need to know what "static" and "void" represents, as you still (if you are indeed a newbie user, that is) have no or little knowledge about functions, etc.

The action's code

```
Console.WriteLine("Hello, world!");
```

The above line of code itself will write a "Hello, world!" line of text on the console. Note that if you hadn't wrote the "using System;" line at the beginning, you'd need to write:

```
System.Console.WriteLine("Hello,
world!");
```

Note that unlike C++, the "WrileLine" function automatically makes it an independent line, so that you wouldn't need to add a return character manually. There is, however, a Write function that does not do that automatically, if you for some reason need that.

The whole thing

```
using System;
class GMkingTutorial
{
    static void Main()
    {
        Console.WriteLine("Hello,
world!");
    }
}
```

Conclusion

C# is indeed a very interesting programming language, and if you're really good at C++, you might want to give it a shot. Of course, C# does have its own disadvantages, such as being tied up with the .NET framework, etc.

Eyas Sharaiha

TUNTIS' RANT

Running a Web (Browser) Game

Hello, I'm tuntis. I used to run a web browser game called "MarketCrash", which was quite a good experience. I, however, closed the game, and I'm going to tell you more about the game:

Ok, so I got the game source from a person that had ran the game in Finnish ("Pörssiromahdus", **Editor's note:** Yeah, we can't pronounce it either…) before.

He closed the project (being more successful than I was) to work on "Zezenia". I translated it to English and restarted the game.

I advertised literally everywhere I was allowed to and got a few (ten, maybe?) players. Ok, it went quite good for a few weeks or something. Then I started implementing new features, until I figured that I had started the economy and all in a bad way. It went somewhat like this:

1. User makes account.
2. User needs 4000€ to make a factory (business).
3. User goes to work (4) or takes a bank loan (4b).
4.
    a. User works until he has enough money to make his factory.
    b. User just goes to make the factory.
5. User makes a factory of specific type, sets prices and waits for somebody to do job for him, and therefore increase the number of products in stock.
6. Game runs.

Ok, obviously that couldn't have worked for a long time. So first I proposed that we restart the game with all good suggestions implemented, and have "game masters" to balance the going.

Unfortunately not really anybody was interested, so I even sent a mass e-mail to the high base of 104 users (counting banned ones). No, it didn't really work.

I closed the game shortly after, and offered the source to the best sounding person. I sent it to somebody; no further reply has been got.

Veeti Paananen,
http://tuntis.net

Game Maker Script of the Month

GMLscripts.com is an excellent resource for Game Maker scripts. I have chosen a great script from their site that demonstrates excellent methods of checking for collisions, etc.

This issue's script is "point_in_polygon" which checks whether or not is a point inside a polygon. It requires the points coordinates to be input and the id of the polygon.

point_in_polygon

```
/*
**  Usage:
**      point_in_polygon(x,y,polygon);
**
**  Arguments:
**      x,y         a test point, real
**      polygon     ds_list containing XY coordinate pairs
**                  defining the shape of a polygon
**
**  Returns:
**      TRUE if the point is inside the polygon,
**      FALSE otherwise
**
**  GMLscripts.com
*/
{
    var x0,y0,polygon,inside;
    var n,i,poly_x,poly_y,x1,x2,y1,y2,m,b,ix,iy;
    x0 = argument0;
    y0 = argument1;
    polygon = argument2;
    inside = false;
    n = ds_list_size(polygon) div 2;
    i = 0;
    repeat (n) {
        poly_x[i] = ds_list_find_value(polygon, 2*i);
        poly_y[i] = ds_list_find_value(polygon,
2*i+1);
        i += 1;
    }
    poly_x[n] = poly_x[0];
    poly_y[n] = poly_y[0];
    i = 0;
    repeat (n) {
        x1 = poly_x[i];
        y1 = poly_y[i];
        x2 = poly_x[i+1];
        y2 = poly_y[i+1];
        if (((y1 <= y0) && (y2 > y0)) || ((y1 > y0) &&
(y2 <= y0))) {
            if (x1 == x2) {
                if (x1 > x0) inside = !inside;
            }else{
                m = (y2 - y1) / (x2 - x1);
                b = y1 - m * x1;
                ix = (y0 - b) / m;
                iy = y0;
                if (ix > x0) inside = !inside;
            }
        }
        i += 1;
    }
    return inside;
}
```

Creating a polygon: polygon_create

I have created my own "polygon_create" script that could be used with the point_in_polygon script. Feel free to use it as well.

polygon_create

```
/*
**  Usage:
**      polygon = polygon_create(x1,y1,x2,y2,x3,y3,....);
**
**  Arguments:
**      x1,y1,etc. coordinates of polygon, in pairs
**
**  Returns:
**      ID of Polygon
**
**  Created by Eyas Sharaiha,
**   to be used for polygon_ functions
**   found at GMLscripts.com
**  MarkUp, Issue 3, May
*/
POLYID=ds_list_create();

i=0;
do
{
  ds_list_add(POLYID,argument[i]);
  ds_list_add(POLYID,argument[i+1]);
  i+=2;
}

until(variable_local_exists("argument"+string(i))
==false)


return POLYID;
```

Until next time,
Eyas Sharaiha

Sapphire Tears

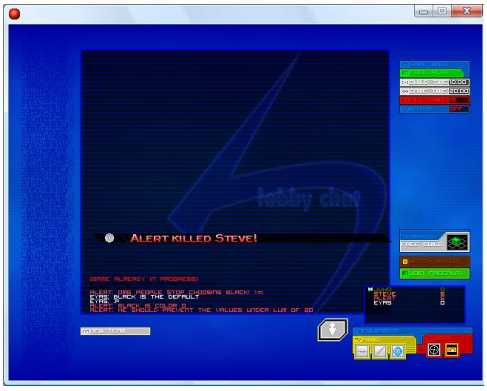
While on my mission to look for games that work on my operating system (Vista), I came across Sapphire Tears. Sapphire Tears is created by JakeX, well known for making cool multiplayer Game Maker games. I haven't played it much due to the lack of users on the server, but here's my judgment from what I've seen!

First of all, I should make it clear that the game is a Work In Progress, and not a complete Game Maker creation. But worry not, for I will judge it with that in mind!

User Interface Part 1: Pre-game UI

The game has an incredibly friendly user interface, even before starting to play, you are greeted with a Reflect Games System (will be later discussed) prompt for your account information. There's a great nice link at the button that opens a web-page for you to create a new account, in case you don't have one.

I couldn't help but compare it to AJAX-powered, Web 2.0-based applications that we now see all over the internet.



I like its design, look, and feel a lot! But on my ultra-high resolution and 17" Monitor, text is just too small! I know it's meant to be small and elegant; but that's just too much.

For all the above, my rating for the Pre-game UI is 9 on my 'metric' scale.

User Interface Part 2: *Gamer* Interface

What I mean by Gamer Interface is the environment of games in terms of options available, and the amount of control that this environment gives you over the game and player, and the ease of use and understanding of that interface.

When examining the interface, everything you need is really right in front of you! You have your health, ammo, map, team information, and other similar information. Once you get used to it, it's really easy to find what you need and use it.

With such amount of information available *all the time*, one would think the view is way too cluttered – but apparently: it isn't! It's simple and to-the-point, yet informative in every way you want it to be. I like it. But again: PLEASE! Use larger text!

Anyways, the game also gets 10 on Game Interface.

Gaming Controls

Before (or if) you ask, the answer is no: Gaming controls and gamer interface aren't the same. Controls include the use of input devices to control the player directly, while the Gamer Interface is about using the output to deduct what you should do with a player and do it.

So, the game uses WSAD for movement, space for jumping, left and right mouse

Sapphire Tears Cont.

buttons for controlling the weapons, mouse wheel to switch ammo, and some other keyboard buttons to give you all the kinds of information you should need.

Does the description above sound familiar? Well, it does to me, at least. WSAD movement, jumping, and the use of mouse in that way have been increasingly popular these days. I especially remember a similar set of controls for Counter Strike, and many others.

The reason such pattern and set of controls is popular is due to one reason: it works! It's easy, it's fast, and it is everything a gamer might need. So why not base yet another game off it, I say. The game controls get a 10.

Music

The game includes two full-length, custom made (as far as I can tell), and high quality mp3 tracks. First of all, in terms of the number of tracks, there are two ideas in conflict: less music files means less space, but on the other hand, more music files mean more versatility and fun.

Even if this is a work in progress, I would've expected some more music. True, the tracks are long and high-quality, which is a HUGE plus, but I just needed some more.

The first is really, really, good! It is exciting, fits with the mode of the game, and just makes you want to play! But then, there's the second one, which starts off way too calm and 'mysterious'… high quality? Yes. But that's just not what the game needs.

Due to lack of versatility in the music, along with my own (possibly inaccurate) opinion about the tracks, I'll give the music 7 out of 10.

Sound effects

There are two questions that I ask myself before determining the score of a game's sound effects: first, is the game giving me what it needs to give me? And then, is the game providing some extra effects that enrich the game experience?

Well, for the first question: the answer is definitely yes. The game is providing the needed sound effects, such as those for shooting, etc. That's a really good thing.

But then, the game is only providing what is required for it to give, there's not much of an 'enriched' user experience really. I like the sounds, they are well made, but that is as far as it goes. The game gets 8 out of 10 in sound effects.

User Experience

Dammit! The game's so exciting! No, I'm serious. When you start shooting people and stuff, there's this adrenaline rush that fills your body. The game is definitely fun. But there are a few annoyances here and there, such as the roughness of the stairs, and slight amount of bugs here and there. 8 out of 10. Period.

Sapphire Tears Cont.

Graphics Part 1: Models and textures

The game's models are pretty simple geometrical shapes, such as cones, spheres, and cuboids. Textures are soo simple, almost non-existing. But guess what? It fits the game. I enjoy the game's graphics and models, but when looking at it thoroughly, more should've been done. 7.5 out of 10 is a good score.

Graphics Part 2: Effects

If someone wants to look at the game in a shallow way, you'll only see one type of effects: explosions. I must admit they look really good, but I really wanted to see more of that.

Players are just simple models; their parts don't move or anything, they just move –as blocks – from one place to another.

One the other hand, I saw a nice effect that I liked: the reloading effect. It is those little things that just add up and deliver an excellent experience for the gamer. Unfortunately, it is only one "little thing". It just needed more.

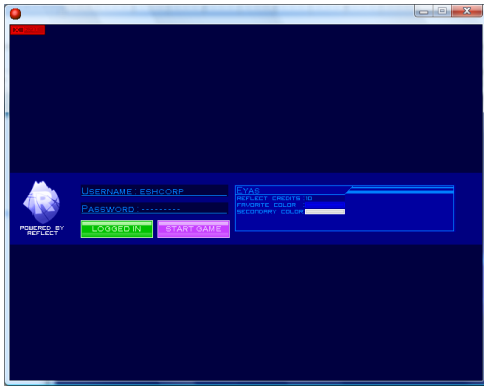
For all the reasons above, I give the game 7 out 10, as it is still a work in progress, and more will be hopefully be coming in time.

System

The actual game is probably coded entirely by JakeX, as I didn't see any reference to other systems being used. It was nice and fitted its purpose, but a few bugs appeared here and there.

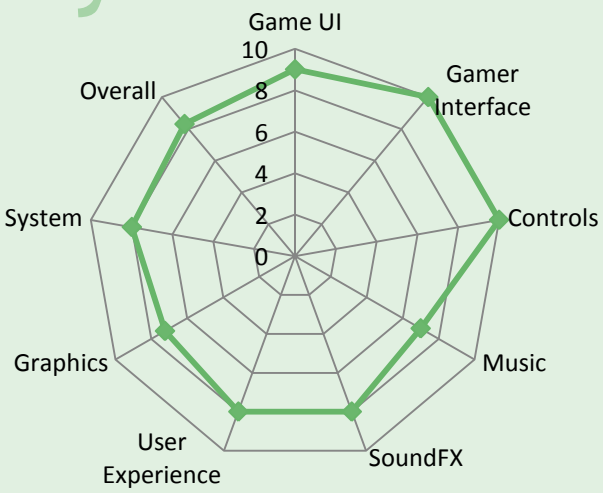
The game uses the Reflect Account system, which is an excellent system for multiplayer games, which uses DLLs to utilize excellent network and multiplayer gaming capabilities. FredFredrickson is to thank for the system.

For "system" the game gets 8 out of 10, mainly because of the bugs in the movement system, which should be ironed out in time for the public release.



Summary

Pros. The game is nice, and has a great interface! I like the controls and basic concept of the game and it seems to have a sufficient user-base, compared to a regular WIP game. The game also has a map editor, which I didn't mention. **Cons.** When you look at it, it is really simple -- not the good simple. I'd love to see a story line, better graphics and models, and more of everything. At its current stage, it is simply: too little.



Why I won't be judging on Storyline

I will not judge the game by its storyline, as it simply doesn't have – or need – a storyline. The game is more of a social one than it is a hardcore game. Though, I'm sure many people would appreciate a storyline.

Overall

The game gets an average score of 8.3 out of 10. That is really good, particularly to a work in progress game. JakeX have definitely done a huge amount of work on that game, but there's still a lot more to do.

8.3 is a really good score, but I'll be looking forward to review the whole thing, once it's complete.

Eyas Sharaiha

Get the latest version of Sapphire Tears for yourself at:
gmc.yoyogames.com/?showtopic=291170

GM's Binary File Functions

The binary file functions were added in Game Maker 5.2. They offer a better and more complete way to read and write data to and from files. Although it's not absolutely necessary that you use them, they will definitely be useful when you want to store data in more advanced file formats.

Binary?

To put it simple, binary means "consisting of ones and zeroes". When talking about binary data, we usually refer to computer data in its simplest form, that is, bytes.

A byte is the smallest amount of data that a computer works with. A byte is always 8 bits and each bit can be either 1 or 0. That gives a total of 256 values that can be stored in a byte, ranging from 0 to 255. Those values would be difficult to read and interpret, a solution is ASCII code.

ASCII code

In ASCII code, readable characters and some other characters, such as tab, line feed and carriage return, are represented by one of the values of a byte. The first 128 are defined in the ASCII standard and are the same on each computer; the latter 128 are extended ASCII codes and might be different depending on the computer you're working on. The ASCII table can be found at:

http://www.asciitable.com/.

Voilà, that's all for the theoretical part.

Reading a binary file

Now you can start to use GM to read and write bytes to files. To show how to read data from binary files, open Notepad and write "This is a binary file." (without the quotes), then save the file as "binary.txt". Now you might think: "Hey, aren't we going to create a binary file?". Well, you just did. Each text file is a binary file, but a text file contains readable characters (most of the time). So now you've created a binary file.

Open GM and add one object and one room and add the object to the room. Add the action "Execute a piece of code" to the create event as well. All code will go there. Save the editable in the same directory as the binary file.

Now it's time to add the code. The first thing you need to do is open the file:

```
file=file_bin_open("binary.txt",0);
```

Two arguments are passed to the function:

Argument	Value
filename	in this case "binary.txt"
mode	a real number indicating the mode: **0** reading **1** writing **2** both reading and writing

The function returns an integer id that must be used in all other binary file functions. This id is stored in the variable "file".

Next, the characters in the file need to be read. Enter the following code, and then run the game:

```
firstcharacter=file_bin_read_byte(file);
show_message(string(firstcharacter));
```

If you've tried this out, you'll notice that the message is "84". Normally, you would expect the message to be "T",

GM's Binary File Functions Cont.

since it's the first character in "This is a binary file." What is wrong here?

The file_bin_read_byte function, as the name implies, reads a byte from a file. Since a byte is a number (as seen before), this function returns a real value. Now we can see what's happening. The function doesn't return the character, but the ASCII code of the character. If you look up the ASCII code for the letter "T", you'll see it's 84.

To convert the ASCII code to a character, you can use GM's chr function:

```
chr(84);
```

The above code would return the string "T". That's exactly what we want. Now we can write the correct code:

```
firstcharacter=
          chr(file_bin_read_byte(file));
show_message(string(firstcharacter));
```

You'll notice that this piece of code correctly reads the byte and turns it into a string. Now there's an important remark here: the file_bin_read_byte function reads the byte, then goes to the next byte in the file. So the next time you use the function, it will return the next byte.

Now the first character has been read. But to read all of the characters in an easy way, we're going to use a loop (you can remove the above piece of code in the code editor).

A simple repeat loop will be sufficient. Only one question remains: how many times does the loop need to be repeated? It needs to be done for each byte in the file; this is the same as the file size.

The file_bin_size function can be used to retrieve the file size of a binary file. It returns the size, in bytes, of the file. That size is also the number of times the loop needs to be repeated. Add the following code in the code editor:

```
returnstr="";
repeat(file_bin_size(file))
{
returnstr+=chr(
          file_bin_read_byte(file));
}
show_message(returnstr);
```

Now if you run the game, you'll notice it reads the string correctly. The code does the following:

1. Initialize the string "returnstr"
2. For each byte in the file: read the byte, convert the ASCII code to a character and add it to the return string (remember that the file_bin_read_byte fucntion automatically jumps to the next position in the file when called).
3. Show a message that displays the string

That's it! Now the only thing that needs to be done is close the file. Use file_bin_close to do this:

```
file_bin_close(file);
```

Don't forget to call this function after you're done with writing to a file. If you don't, you'll soon get into trouble because GM only supports 32 files open at the same time.

Writing to a binary file

Writing data to a binary file is pretty much the same as reading data from it, but there are some important differences. First of all, the file needs to be opened (you can clear the code in the creation event of the object). Now add this code:

```
file=file_bin_open("test.txt",1);
```

Like the previous time, the file_bin_open function opens the file, but, as indicated by the 1, for writing. An important note: if you're using GM7 the file_bin_open function will create the file if it doesn't exist, if you're using GM6 it won't. To get around this, insert the following piece of code before the code you just entered:

```
fcreate=file_text_open_write("test.txt");
file_text_close(fcreate);
```

This creates the file by using the text file functions.

Now suppose you'd like to write the string "This is a file." to the file. For each character in the string, a byte needs to be written to the file. We will use a for loop to achieve this. Enter the following code in the code editor:

```
writestr="This is a file.";
for(i=1;i<string_length(writestr)+1;i+=1)
{
        file_bin_write_byte(file,ord(
        string_char_at(writestr,i)));
}
```

The piece of code does the following:

1. Set the string to write

GM's Binary File Functions Cont.

2. For each character in the string, write the corresponding byte to the file. Note that a new function is used here: ord. It does the opposite of chr. chr converts an ASCII code to a string character, ord converts a single string character to a real value, the ASCII code. Also note that the loop starts at i=1, because the first position in a string is 1.

And last, the file needs to be closed. Add the following code in the editor:

```
file_bin_close(file);
```

Now run the game. After the game is run, the file will have been created. If you open it in Notepad, you'll see that the text is there.

Advanced uses of the binary file functions

GM has a few other functions that deal with binary files. They are:

file_bin_rewrite: same use as file_bin_open, but clears the file if it has content. You don't need to specify the mode here, since it will always be write.

file_bin_position: in a binary file, you read or write at a certain position. That position can be retrieved with this function. The first position in the file is 0.

file_bin_seek: this function sets the position in the file. This position can, of course, never be bigger than the file size.

This tutorial covered reading and writing strings to binary files. To write real values to a file, you could convert them to a string, and then write that string to the file. Another, more advanced way to do this would be to make use of the fact that a single byte can contain 256 values. So integer numbers up to 255 (numbering starts at zero) can be put into a single byte. By using some math (**div** and **mod**), you can store very large integer numbers in just a few bytes.

Since all files are binary files, there is really no limit to which filetypes you can read and write. The problem, of course, is that you need to know the structure of the file. Examples would be: reading xml files, getting metadata from audio files,…

A clever use of the fact that a byte consists of 8 bits, either one or zero, allows for many new possibilities. 8 boolean values (true/false values) can be stored in a single byte. Or suppose you need storage for 16 possibilities. Then you need 4 bits, since 4 bits allow 2^4=16 combinations. The disadvantage is that there's a loss of 4 unused bits.

And this list goes on...

Bart Teunis

QUICK TIPS

Developing a Game

I have been studying the different ways that game developers create their games, and I am about to tell you what seems to be the most *efficient* process.

First of all, you must find an idea to base your game on, and then you should enhance the idea, until it is a story (unless its basketball or something without a story). Then you should check to see whether you are capable of making the game, because why start a game to find you can't continue?

After you have that figured out, it's usually good to make notes of anything about the game, including what code you may use, variables, maybe some concept art, etc.

Once you have planned it, begin by making some very simple placeholder sprites, and start on the engine.

Part of the way before you complete the engine, create a room in which you may test and debug features, and then continue on the engine. When the entire engine is done, you may want to add better sprites and sound.

It's important that you test the game yourself and see if you spot any errors, fixing them as you find them.

If you would care to release a beta of the game (to get suggestions and bug reports), then do so. After that is all done, distribute the game in whatever way you want (but be careful about copyrights).

Stephan

Game Developer Interview

David "Mr.Chubigans" Galindo 

I talked with David Galindo, better known in our community as "Mr.Chubigans", who is one of the top talented members of the Game Maker Community. He's probably best known for his 2004 "hit" game Sandbox of God, among others. Here's an interview with him that I hope you will enjoy as I did.

Eyas Sharaiha

Sandbox of God

1. First of all, since when did you start developing games and what was your first game?

I think it was around 2000 or so when I first started, all the way back with GM3. My first game was called Nocturnal Emmision, about a fish who tries to assassinate the president. E'gads was it awful! Mark Overmars thought the same thing (back when he actually took submissions for his website personally and sent back a thorough email of what he thought of the game...man, those were the days!) But, my second game called The Wal-Mart SUV Parking Lot Challenge fared much better.

2. What was the first game you think of as a success?

Probably the Sandbox of God was my biggest one. Other games like Acidbomb and Vivid Conceptions won awards on various sites, but personally SOG was much more of a success because of all the feedback I got...especially from Christian gamers.

I didn't intend the game to skew towards the religious side of things, which is why I kept it a bit light and funny...but I still got a lot of great feedback from people taking it their own way.

3. All your games have been popular, while many GMC members struggle to make one of their games popular. What, in your opinion, made your games unique?

I wouldn't say all my games. ☺ But really, I just build on the smallest of ideas and work my way to a game. Acidbomb was based off of a small minigame that was shoved in the extras of a game called Smartbomb on the PSP...and I thought, hey, if they modified the engine and added a whole bunch of stuff to that one mini-game, they could have had a whole game. Then I realized, hey, I could do that!

4. What is your point of strength? Design, programming, etc.

I am absolutely horrible at programming...most of my games are 80-90% drag and drop GM. I've had some people tell me that I should stop doing that, but really, why? That's what I love about Game Maker...I literally couldn't make games without it. So I would say design is by best attribute.

5. Where do you get ideas for your games?

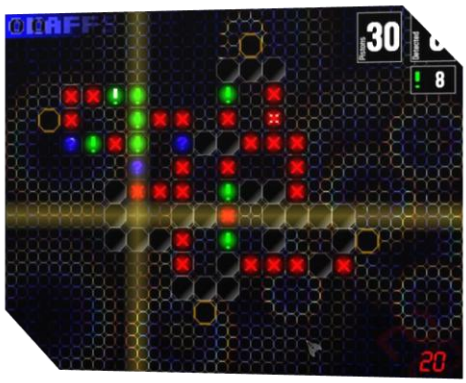
I just have to constantly be thinking about it. Most of the time it's influenced from other games that I thought had tons of potential but didn't quite get there, so I build on their core ideas and make it my own.

6. What made Ac!dbomb (1 and 2) popular in many sites outside the GMC? Was it your efforts or just the games?

It just kind of crawled outside of the GMC really...I have no idea how magazines and websites find them! I used to go through as many sites as

David "Mr.Chubigans" Galindo Cont.

possible and submit them, like I did with Ore no Ryomi 2. Now I find it more interesting to let the game progress on its own. If the game doesn't find a home outside my website, I know that I did something wrong with the game...as the case was with Vivid Conceptions. So much lost potential in that game...I really think I failed its core idea. Kinda depressing really.



Ac!dbomb 2

7. Vertigo Games has gone commercial, have you had any criticisms from within your own community?

I thought I would...in fact, I was dreading it immensely. But I haven't gotten a single negative response...in fact, it's mostly people congratulating me and asking if they could pitch in!

It was really remarkable, and it just makes me realize how great people can be sometimes, especially fellow GMers. I'm a very happy guy right now.

8. Which, from your existing games, will be released as commercial versions?

Mostly sequels on a case by case basis. Acidbomb 2 is the only exception, as I'm taking that game and just adding more new features to justify the price tag. All future games will be completely new, like Ore no Ryomi 3 (now called King of All Cooks) and possibly the Sandbox of God 2: Paradise.

Most won't have the "2" tag on them though. They'll either be called by their original title or a new one altogether, like King of All Cooks.

9. What new features should we expect in the commercial versions?

As much as I can think of. It's a bit different working commercially...and I'm a bit more limited. I can't say, "Well I think King of All Cooks should be $10." because all distributors start from $20, period. So I have to make a game and continually add on it until I can say, yeah, I'd pay $20 for this game.

It works for the game's benefit. There are a lot of new features going into Acidbomb 2 (now called ShellBlast in the retail world) that I couldn't have done any other way.

10. Your games will be published by Oberon Media. Where you the one who contacted them in the first place, or were it the other way around?

They contacted me, and I was very hesitant as you could imagine. After about two weeks from me not responding, they emailed me again and we set up a phone interview, and it was great. It should be going fully commercial about a month from now or so.

11. You still have a couple of non-commercial projects. Are you planning on completely shifting to commercial in time?

That's the plan, though I have a few games that wouldn't work at all in the commercial area (like Napalm of Valor: Helicopter Cacophony 3) that I'd like to pursue as freeware at some point this year.

Napalm of Valor: Helicopter Cacophony 3

12. When you look at the future, do you think Game Maker will still be the tool of your choice, or will you have to move on sometime?

As long as I'm making PC games, I'll be using Game Maker, no question.

13. Is there anything else you'd like to say?

It's a great time to be in the indie-game business, and all of us that use Game Maker are in a great position to be pioneers. I can't wait to see what the future of Game Maker and YoYo Games will bring us...it's an exciting time to be a game maker!

So that was it, my interview with Mr.Chubigans. You can visit his website at http://www.vertigogaming.net/.

Guide to Addictive Games

Many people often use the term "addictive" when describing a game and game developers who are familiar with this term try to achieve it in their games. But it's often more difficult than you might think.

An addictive game is a game which creates a need for the user to play the game again – an addiction. The idea is the game must have a good replay value that enables the player to think about playing the game in the first place.

Main points in "Addictivity"

I will later go into the details, but an "addictivity" of a game depends mainly on the following points:

1. Replay value
2. Game variety
3. Audience friendly

The third point is specifically important – in games, there's no such thing as "player friendly" or "user friendly" as there are all kinds of users: some like old-school games, some like action, some like strategy – you can never please them all! The idea is you should pick your target audience, and make your game appealing to that particular audience; no one expects you to make *everyone* love it.

Types of Addictive Games

A game can either achieve long term addictivity or short term addictivity. Short term addictivity is relatively easier to achieve, while long term addictivity is harder and usually requires more skill and creativity.

Short term addictivity

Short term addictivity means when someone is addicted to a certain game for a period of time, and then –for one reason or another– loses interest.

Loss of interest could be due to:

1. Little replay value for game
2. Game is a story; when it is complete, there's not much point playing it anymore

The former is an issue that only experienced game designers can overcome it, the second isn't really a problem – the game developer chooses to make a game with a certain story, levels, and winning conditions, and, when such winning conditions are met, it is to be expected that a player would stop playing the game.

When a game has little replay value, something in that game must be changed – exactly what needs to be changed depends on each particular case, sometimes ranges from adding little extras to a radical change in the game's actual concept.

Long term addictivity

If you want to achieve long term addictivity of a game, then you should first consider what type of game you're making. There are lots of types, but I'll only cover two general categories.

Guide to Addictive Games Cont.

Versatile games

Versatile games are games where the objectives, setting, and events of a game are in constant changing. This generally includes games like “Diablo”, “The Sims 2”, and “Guild Wars”.

What distinguishes a game with a story that only achieves short term addictivity with a game that also has a story but achieves long term addictivity? The answer is simple: a game that achieves long term addictivity is still addictive even after completing the game.

Certain games such as “Kingdom Hearts” are extremely great, however, they do not achieve long term addictivity; when you complete the game, there’s not much more to play. That’s okay of course, because this is how the game has been designed.

However, if you want to achieve long term addictivity, then the game should be playable after completing it, and have different challenges such as completing the game in a harder mode, or playing against other gamers, or just continuing spreading your power around the game’s “world”.

Also, games like “The Sims”, where no actual “ultimate objective” exists usually achieve long term addictivity. But of course, it all depends on good game design, and creativity.

Restricted Games

“Restricted games” is just a term I came up with when trying to describe that type of games. They are games like “Bejeweled” or “Zuma” which people just keep playing over and over.

The objective is the same, you have to collect things, build things, destroy them, etc.

What makes such a game addictive? Two (or one) words: **high scores**. People attempt to beat themselves and/or their friends and achieve a sufficient score.

Other classical examples include “Tetris” and “Pacman”.

Conclusion

There’s really no magic formula for making an addictive game, however, it always comes down to three things: skill, creativity, and hard work.

Eyas Sharaiha

XML vs. INI

In games – and, well, programming – everything is data; it's just stored in different ways! Graphics are data stored as images, functions and classes are, sounds are, and all of the variables in your game.

Certain settings would be better if stored outside the game's main executable since they might change all the time. We see two main formats used in storing such type of data and settings: XML and INI.

Simple Storage

Comparing the two isn't easy, since XML has its own advantages, and so does INI. When storing simple pieces of data, I believe INI would be the way to go. Here's why:

INI

```
[General]
Res=640x480
Name=John
Age=19
[Rabbit]
Speed=4
Difficulty=2
[Frog]
Speed=3
Difficulty=4
```

XML

The same information, when written in XML may look in different ways, according to how you want to write it, it could look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
<group name="general">
  <setting title="res">640x480</setting>
  <setting title="name">John</setting>
  <setting title="age">19</setting>
</group>
<group name="rabbit">
  <speed>4</setting>
  <difficulty>2</difficulty>
</group>
<group name="frog">
  <speed>3</setting>
  <difficulty>4</difficulty>
</group>
</settings>
```

Or even like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
 <general res="640x480" name="John" age="19"/>
 <rabbit speed="4" difficulty="2">
 <frog speed="3" difficulty="4"/>
</settings>
```

XML is much more complex for simple users, and requires more space what the INI file would occupy.

Complex Data

XML wasn't meant for storing simple settings and data; it was meant for more complex data, where multiple elements of the same kind are stored, read from, modified, and added to. Lists of objects with different properties such as their positions, etc, would be very useful by XML. If you're making your own script that saves your game's progress, for example, then XML should be an obvious choice.

Heck, even web pages are adopting the stricter XML-based markup language: XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h1>Title</h1>
<p style="text-align:center; font-weight: bold;">Hello, World!</p>
</body>
</html>
```

An example I like to share, is Wikipedia's example of XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="3" unit="cups">Flour</ingredient>
  <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
  <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together, and knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again, place in a tin, and then bake in the oven.</step>
  </instructions>
</recipe>
```

THIS is efficient for space, structured, and simple to handle – in terms of code.

Conclusion

Although I don't like saying it – no one can really say that XML is better than INI, because it depends on the usage. However, I believe that the **only** usage where INI *might* be considered superior is in simple settings and data.

In most methods, and practical applications on the usage of data storage techniques, XML will be – by far – superior.

Eyas Sharaiha

Gregg the Egg

Where do I begin with this game? As the author states, Gregg the Egg is all about a little creature that has just hatched from its egg.

Storyline

Your goal is to help it escape the dangerous world it's in. Encounter hungry monsters, deadly spikes and deep dungeons.

Luckily, Gregg evolves during its journey, giving you new skills and powers to continue your journey.

That hardly begins to describe the wonders that you will behold while playing this amazing game.

Gameplay

You will face many obstacles, some which require cleverness and cunning to overcome, whereas others take skill, timing, and a well thought out plan. No matter your playing style, Gregg the Egg promises to offer a variety of challenges along with great gameplay to ensure you the best gaming experience possible.

It's not your typical run-around-and-do-stuff kind of game. It's more of a freely flowing adventure, where you are trying to figure out how to save yourself from death and gain powers to aid you in your travels.

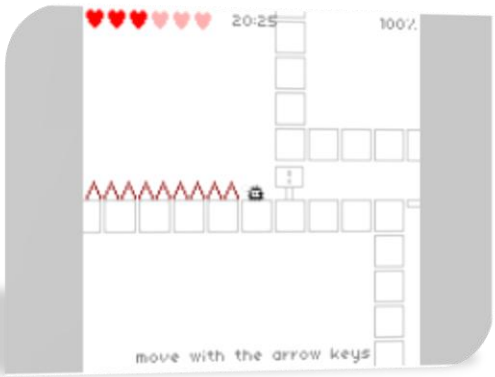
Right from the start, the controls are simple and basic. Signs will tell you what new actions you can perform, such as jumping, using launchers and teleporters, and downsmashing.

There are a number of goodies to pick up to enhance your score, such as gems and powerups, but you do not need to pick up everything to complete the game. It's up to you whether you wanna go on a treasure hunt and pick up everything you can, or just do the bare minimum and escape.

Some of the obstacles are very straightforward, such as jumping across a gap. Others, however, require a bit of skill/luck, such as maneuvering across multiple moving platforms while avoiding spikes of death.

Normal mode allows you to pick up health increases, making even the toughest parts still beatable without dying, although the more advanced players will certainly want to take as little damage as possible, hence requiring you to save less frequently, and end with a better time (time is equivalent to score).

Be advised that you might find yourself swearing a bit more than usual while playing this game, especially on hardcore mode. Don't worry; it's completely normal, especially if you forget to take your time!



Graphics

The graphics in this game absolutely rule. Why do I say that? Well let's look at a few examples of great games with simple graphics.

Who could ever forget SMB3? :

You can't say that you wish the graphics were "better". They helped make the game what it was. Even games like half life or system shock had bad graphics compared to anything today, but we loved it. It almost would have ruined the game had the graphics been amazing. It's just how it is. The same thing goes for this game.

The graphics don't suck, that's not what I'm saying. But they are simple. Nothing is overdone, and for the most part, it's a straightforward world of blocks and spikes, hearts and powerups. I don't think the simplicity of it all detracts from the game in any way shape or form.

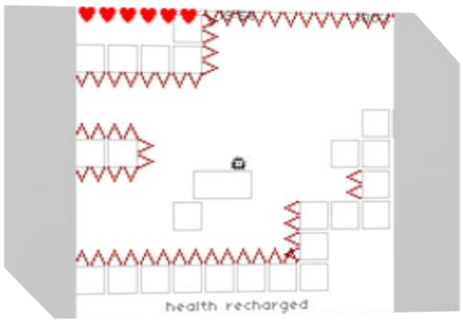
Music & Sound Effects

The music is good as well. Maybe not everybody's cup of tea- a 30 second looping midi which could possibly use a

Gregg the Egg Cont.

better drum section, maybe some double bass or heavy parts. But I find it nice to listen to, and having played this game for more than 15 hours total thus far, I haven't gotten annoyed with it yet. Sure after the first hour of playing it started getting to me, but then I realized it really fit the game well, and it has ceased bothering me since.

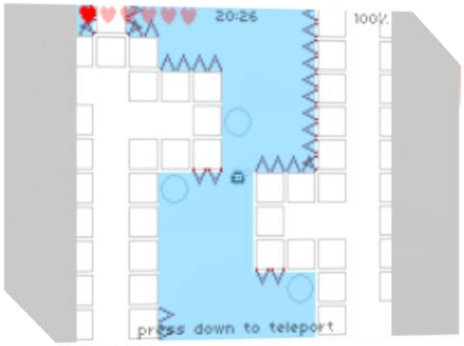
If you just can't tolerate it, feel free to switch the music off and play your own music, as this game has no sound effects. Strangely, the absence of sound effects did not bother me at all, as I would have expected it to. I really don't know how it would play with sound effects, but as it is, it plays just fine without.



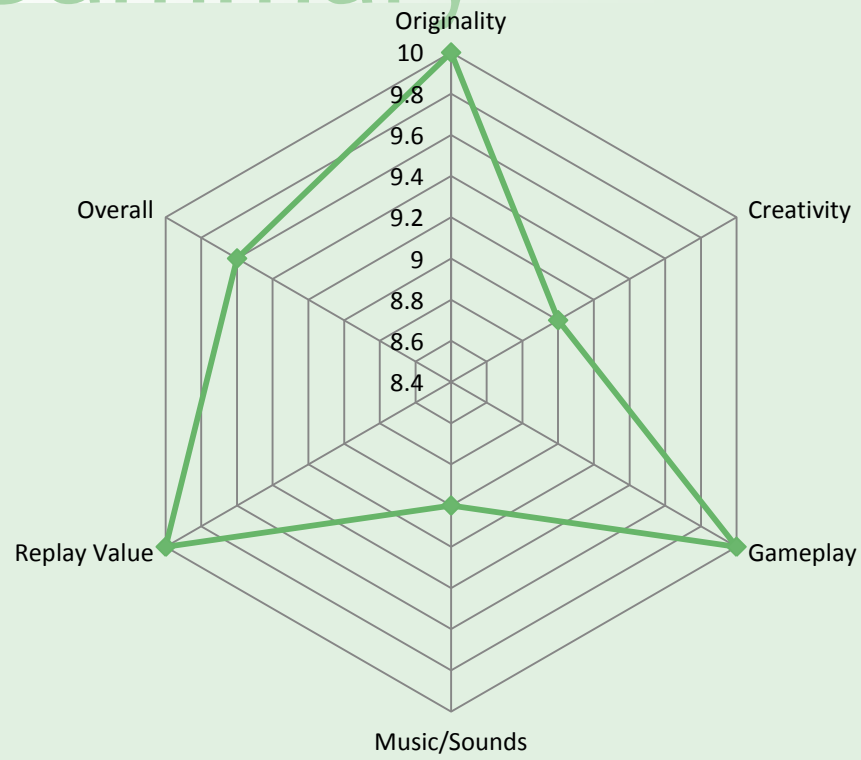
Controls

The controls aren't hard, and they do everything you need them to do. I didn't find any problems with controls while playing the game, except maybe the fact that some of the finer details of controls are not explained anywhere in the game.

For one, not only does letting go of the jump key keep you from jumping as high, but letting go of the jump key after using a bouncy-ball will also keep you from going as high. Aside from this little known fact, there are no real complaints.



Summary



Conclusion

This game rules from start to finish, being original, and creative, while also being totally awesome at the same time. It's not original for originality's sake, nor is it creative just to be ridiculously weird. It just happens to be different than anything you've ever played, or thought about playing, or even considered ever thinking about playing, in your entire life. While that may be a slight over statement, it still doesn't do this game justice.

But don't take my word for it. Download your very own copy today, and play through what America is calling "the best game since games were invented".

Mike Connor

Get the game

Gregg the Egg is available from:
http://gmc.yoyogames.com/index.php?showtopic=284271

Game Manuals

How many times have you started to play a new game before realizing that the programmers 'forgot' to include full instructions and a control map?

Finding yourself playing a game for the first time without knowing what you are meant to or how to do it is daunting; however there are a lot of games out there which are just like this.

Some people make use of the 'Game Information' screen which can be set up in Game Maker, and viewed in most games, even if it has been left blank by the authors, by pressing [F1]. This however has very limited capabilities and can display only basic text.

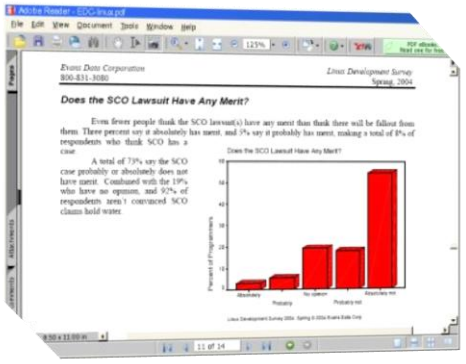
Of course, for people unfamiliar with Game Maker they may not even know that help is available by pressing [F1]. So it is always best to include an external manual or full instructions which are clearly accessible from your games main menu.

A well-designed manual can include screenshots, graphics, annotated diagrams and illustrations such as a keyboard map of controls. There will also be room for a full introduction to the game, and you can even include hints for players to get more out of your game.

Creating an external manual is easy. It can be done using the most basic of software – a Word Processor (**Editor's Note**: You can get a full featured, free word processor at abiword.com). Many games are distributed with a .txt readme file; however this usually contains just copyright information and is not suitable for a full professional-looking manual.

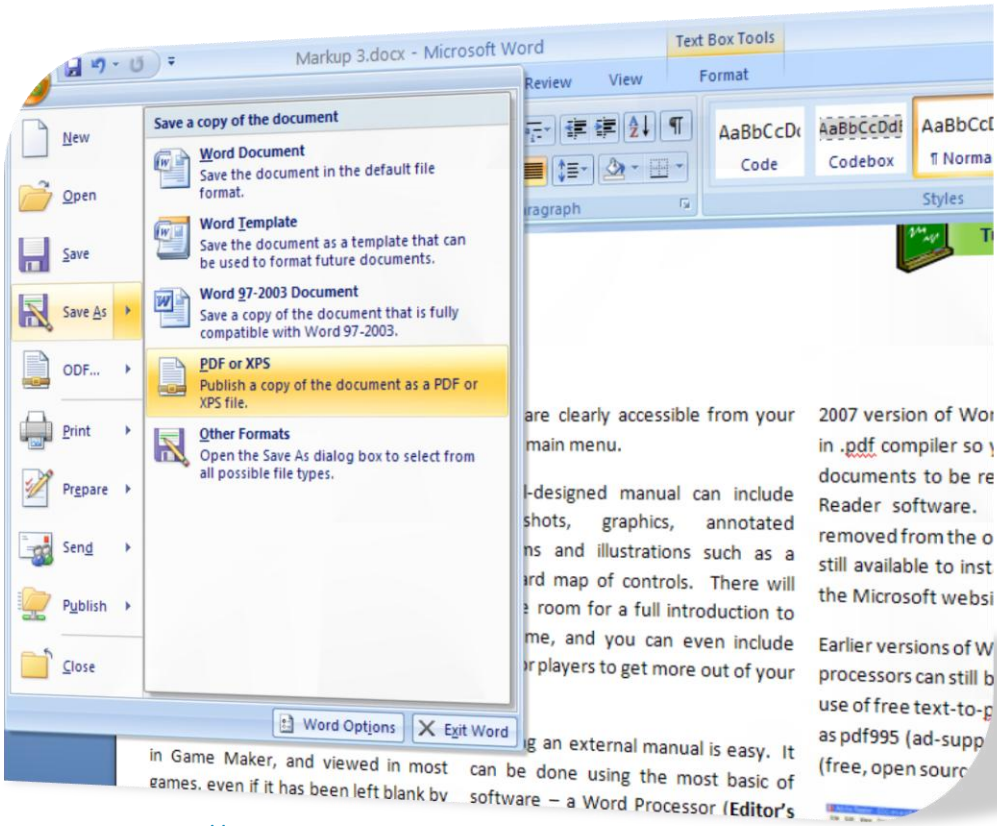
The beta version of Microsoft Office's 2007 version of Word did contain a built in .pdf compiler so you could save your documents to be read by Adobe's free Reader software. This however was removed from the official release, but is still available to install as a plug-in from the Microsoft website.

Earlier versions of Word or other word processors can still be used by making use of free text-to-pdf converters such as pdf995 (ad-supported) or PDFCreator (free, open source).



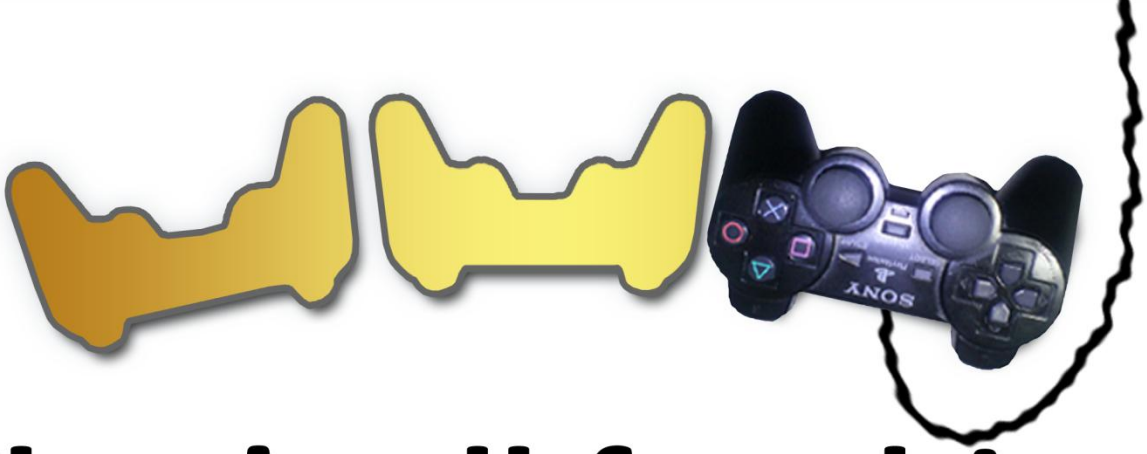
There is no excuse for not including good instructions with your game. They can make the difference between someone trying and failing to play your game, and someone playing and enjoying your game.

Phil Gamble,
http://gamemakerblog.com



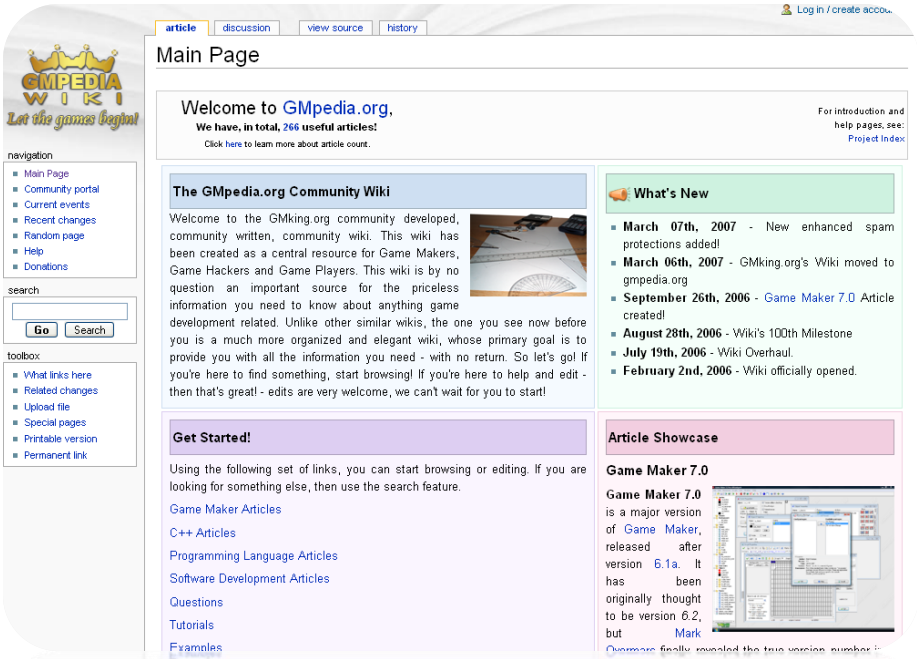
Links

Microsoft Office 2007 PDF/XPS plug-in, http://snipurl.com/1hk93
PDFCreator, http://pdfcreator.org
PDF995, http://pdf995.com
GMB: tutorial: Using pdf995 to create a .pdf file, http://snipurl.com/1hk96



That's all for this issue!

... but check our other online resources in the meantime...



gmpedia.org

forums.gmking.org

openload.info

ircbloopers.com

gmking.org

irc://irc.gamemaking.org/#gmking

GMking.org Let them make games!