April 2008 Issue 13 [it's all here]

markup

Artificial Intelligence: *a practical approach!*

New to MarkUp Magazine: David Perritte's Music Composition Tutorials!

SCRIPT OF THE MONTH

DS_GRID_SET_GRID_REGION: Fixed

DEVELOPMENT JOURNAL

God of Rock

STEP-BY-STEP TUTORIALS

AI Command Stack

GAME MAKER INSIGHT

Persistent vs. Global

AND MUCH MORE!

Welcome

Since first appearing on computers a year and a month ago, MarkUp has found its place in the hearts of many Game Maker users.

But, now I feel I must apologize for our antics with the fake launch of our Issue 13. From the bottom of our hearts here at MarkUp, we'd like to say:

"We got ya!"

Cloud Gaming

While we don't hear the term "Cloud Gaming" specifically frequently, it is impossible to identify that this trend exists in the gaming world as well. When it comes to software in general, there's a general shift towards the concept of Cloud Computing or 'computing in the cloud' as they call it. Applications like Google Docs and Live Mesh are emerging, as well as attempts to create web "operating systems" that are accessed from within a browser.

The general purpose of cloud computing is to make applications, etc. accessible from the web and executed directly from there rather than have such process occur locally.

When it comes to gaming, such trend is more complicated. While games like Flash games are (from a user standpoint) completely located in the web and is run after being 'buffered', when it comes to 'serious' commercial gaming, such trend is much harder to achieve. The reason for this is that the main trend in commercial gaming is realism, which counters cloud gaming, since it results in huge amounts of data that cannot be *only* stored in the cloud and retrieved by the computer.

Instead, the furthest current-day commercial gaming can go into cloud gaming is with MMORPGs such as Guild Wars and World of Warcraft, where *user interaction* and the entire *gameplay* both replies on and occurs in the cloud, the main files, models, resources, and executables responsible for letting such thing happen exist locally.

It'll be interesting to see how gaming develops as cloud computing develops as well, who knows if such predictions will come true entirely or turn out to be completely mistaken.

Eyas Sharaiha

Contributors

Robin Monks	Sr. Editor
Eyas Sharaiha	Editor
Bart Teunis	Writer
Philip Gamble	Writer
Matthew Malone	Writer
David Perritte	Writer
Joshua Smith	Writer
Darragh Tobin	Writer
icuurd12b42	Writer
Veeti Paananen	Writer
Zach Hext	Writer

Table of Contents

YoYo Games and the Community

Philip Gamble

Since appearing on the Game Maker scene at the start of 2007 YoYo Games have faced criticism from certain members of the community.

Most of this criticism is about poor customer service and problems with the Digital Rights Management that is used to protect Game Maker 7.

YoYo Games' relationship with users, in particular members of the GMC, has until recently been poor. Their infrequently updated glog did not address the issues that mattered to members and instead seemed to be looking far into the future.

Recently though that has all begun to change; it started with a long overdue upgrade of the Invision Power Board software used to power The Game Maker Community. Along too came a new server, effectively eliminating the MySQL overload errors that had in the past plagued the forum at popular times of day. A skin change made the forum look warmer and more welcoming, a reintroduction of post count and an increase in the size of Personal Mail inbox in response to member demands have made things better.

YoYo Games staff, Mark Overmars and in particular Sandy Duncan have also begun to post more actively at the GMC, something which ideally should have happened right from the takeover.

The dual communities of the YoYo Games forum and GMC don't make sense, as tuntis wrote on Game Maker Blog, this is "decentralizing the community". The YoYo Games forum appears to serve no purpose other than as a general chit chat area for what appears to be the less mature Game Maker users. I believe the official word on this one is that the YoYo Games forum is for gamers where as the GMC is for developers.

In February Sandy quietly mentioned on his blog that YoYo plan to invest $5M into the Game Maker software and their website in the coming years. Many members are disappointed at the lack of apparent progress with the development of Game Maker which has seen no new releases for nearly a year, despite the emergence of a game decompiler. It is also not specified how much of this funding will be invested into the Game Maker software as opposed to developing and promoting the YoYo Games website.

An attempt to take Game Maker cross-platform with work starting on a Mac version of Game Maker was also met with a mixed response with some users suggesting that YoYo Games' time would be better spent improving the PC version instead of targeting what remains a very small market.

One universally popular move has been the two Game Maker competitions that YoYo have run, which with the results of the ancient civilization competition announced in early May will see YoYo having handed out $3,500 of cash prizes to users of the software. No other Game Maker competitions have come close on the number of entries received and value of prizes awarded.

Sandy has also made some frank admissions about YoYo Games writing on the GMC "I think, at times, that the DRM sucks and I think, always, that our customer support is terrible." Many community members would have set that many months back, let's hope that now YoYo Games are sure of their problems something gets done about this quickly.

Decompilers

Philip Gamble

An exclusive from GameMakerBlog.com

You may have noticed that despite all the talk of decompilers elsewhere recently Game Maker Blog has remained a decompiler-free zone.

What is the decompiler?

At the end of January a program was released that enables anyone who has it to decompile executable files created with Game Maker versions 5.3a, 6, 6.1 and 7. The original resources, scripts and objects programmed into the game are retrieved and organised as they would be if you were editing the game. In effect this is an .exe to .gmd/.gm6/.gmk converter.

Why do some people see this as a problem?

Many people weren't too happy at the release of the decompiler. YoYo Games weren't happy because it shows that the security on their product has been breached and the authors of the decompiler have breached their EULA. Some users of Game Maker fear that their games will be decompiled, their resources and code stolen and that the game could possibly be passed off by someone else as their own work. The chances of this happening to your game are very slim, I am not aware of any games that have been released where the authors have been accused of decompiling a previously made game.

Can anything be done to protect my game?

At the GMC many people [1, 2, 3] have come up with programs which they claim will protect your executable Game Maker files. So far all of them have not been able to live up to their claims, as most of them simply compress the .exe and uncompress it (to its unprotected form) before running it.

Will my game ever be 100% safe?

The simple answer is No. Now hackers have seen that it is

possible to decompile Game Maker games chances are they will develop a solution to any anti-decompiling measures that are introduced.

What are YoYo Games doing?

Good question. Apart from "*Just don't do it or you'll get into trouble*" and a threat to take action against those who wrote the decompiler it doesn't look like YYG are doing much. They promised "*we do our best to make sure this can't happen with future releases of Game Maker AND the website.*", yet no patch has been released almost 2 months after the decompiler was released.

What should I do?

- Firstly don't panic. It is unlikely that someone will steal your program.
- Don't believe these so-called 'decompiler protectors'. All released so far do not work.
- If you are concerned contact YoYo Games and ask them why they still haven't released a patch or a new version of Game Maker, 2 months after the decompiler was released.

Initial source of news: Decompiler announcement at gmnews

Script of the Month

Eyas Sharaiha

An exclusive from GMLscripts.com

Game Maker's ds_grid_set_grid_region function has the potential to be very beneficial to game developers, however, the function appears to be buggy in Game Maker 7 and does not behave expectedly. Therefore, GMLscripts.com comes to the rescue again, to provide a replacement script with the same name to fix the functionality in Game Maker 7!

'Fixing' Functions

This function provides an excellent example of how we can use scripts to temporarily replace buggy functions in Game Maker; when a script is created with the same name as the function, the script will basically override the function and therefore using the function's name will result in the calling of the script instead. If the function is fixed in a later version of Game Maker (yet it has the same format), then you can easily convert your source to the latest version of Game Maker and delete the fixing script, then, all the code you used to refer to the script will automatically refer to the function of the same name instead, thus creating little or no issues in keeping your source code updated.

GMLscripts.com owner "xot" comments:

> *"One of the interesting ideas that this demonstrate is that GM's built-in functions can be replaced with custom scripts by giving the script the same name. Working that way lets you cleanly handle bugs in the Game Maker engine. When the bug is one day fixed, all you have to do is remove your script and everything else will work the same."*

What the Script Does

The script basically copies a certain region of a certain list into another. The grid which the values are *copied to* is the first argument, id, and the source is the grid which the values are copied *from*. The x1,y1,x2,y2 arguments get the coordinates of the region you want to be copied from the source. The xpos,ypos arguments indicate the x and y positions where the grid region will be pasted, i.e. where the "x1,y1" item will be located.

The Script

Usage

```
ds_grid_set_grid_region(id,source,x1,y1,x2,y2,xpos,ypos)
```

Arguments

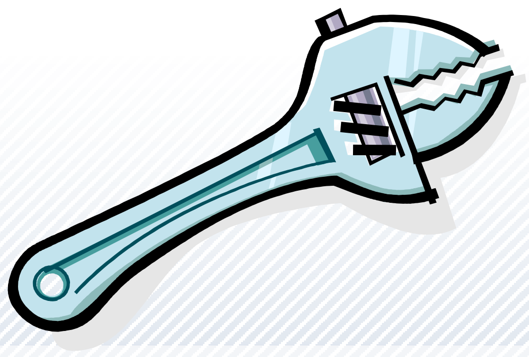
id	destination ds_grid to copy values to
source	source ds_grid to copy values from
x1,y1	upper-left corner of the region to be copied
x2,y2	lower-right corner of the region to be copied
xpos,ypos	position in the destination to copy values to

Returns

Nothing.

Notes

This function replaces the buggy Game Maker 7 version of the same name. It also adds the same functionality to Game Maker 6.



Eyas Sharaiha

An exclusive from GMLscripts.com

```
{
   var copy,xoff,yoff,i,j;
   if (argument0 != argument1) copy = -1;
   else {
      copy = ds_grid_create(argument4+1,argument5+1);
      for (i=argument2; i<=argument4; i+=1) {
         for (j=argument3; j<=argument5; j+=1) {
            ds_grid_set(copy,i,j,ds_grid_get(argument1,i,j));
         }
      }
      argument1 = copy;
   }
   xoff = argument6-argument2;
   yoff = argument7-argument3;
   for (i=argument2; i<=argument4; i+=1) {
      for (j=argument3; j<=argument5; j+=1) {

ds_grid_set(argument0,i+xoff,j+yoff,ds_grid_get(argume
nt1,i,j));
      }
   }
   if (copy != -1) ds_grid_destroy(copy);
}
```

Features

Conclusion

This script can be very useful in many conditions; grids in games sort all kinds of two-dimensional data, including player and enemy positions, items, inventories, etc. When, for example, a new enemy is created, or the same situation or 'pattern' of data repeats itself, this pattern could be easily copied from one place to another – something that should have been available from Game Maker 'out of the box', but was unfortunately flawed. Now, we have the chance of using that functionality again.

File Extension 1.0

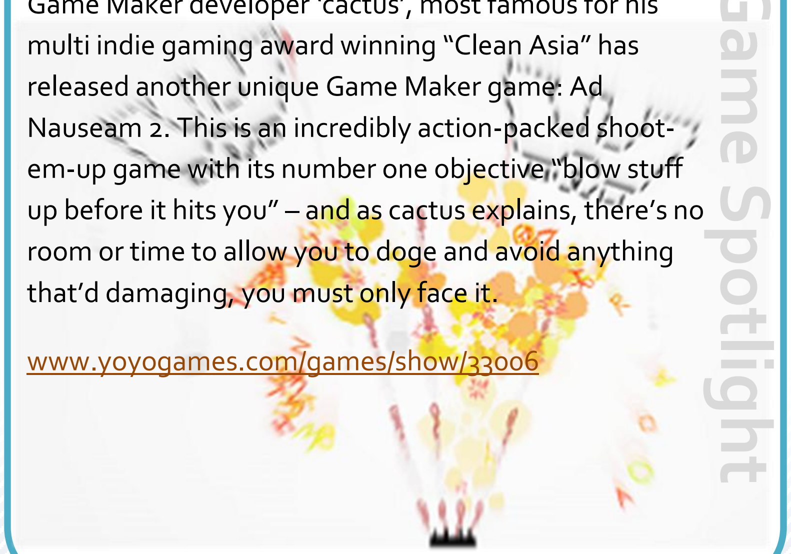
Quick Reviews

The 'File Extension' by SyncViews is an excellent Game Maker extension for reading and handling various types of files. The extension can write various data and read it from files, such as strings and signed/unsigned short data to binary files, points to path files, and, my all time favorite: all sorts of data to CSV files. If you remember reading the 'CSV: Extreme Data Storage' article from issue 4 you'll know how much CSV files could be helpful for data storage, and this script provides all the functions described in the article and more, using a single line of code!

gmc.yoyogames.com/index.php?showtopic=364768

Ad Nauseam 2

Game Spotlight



Game Maker developer 'cactus', most famous for his multi indie gaming award winning "Clean Asia" has released another unique Game Maker game: Ad Nauseam 2. This is an incredibly action-packed shoot-em-up game with its number one objective "blow stuff up before it hits you" – and as cactus explains, there's no room or time to allow you to doge and avoid anything that'd damaging, you must only face it.

www.yoyogames.com/games/show/33006

God of Rock

Matthew Malone

I have finally begun to code some sort of a BPM system into God of Rock. So far it's not going too good due to GM's horrible precision problems. But, I am going to continue working on it in GM7 because I have heard the issues where corrected in that version. And having a bigger EXE but, better math precision is worth it.

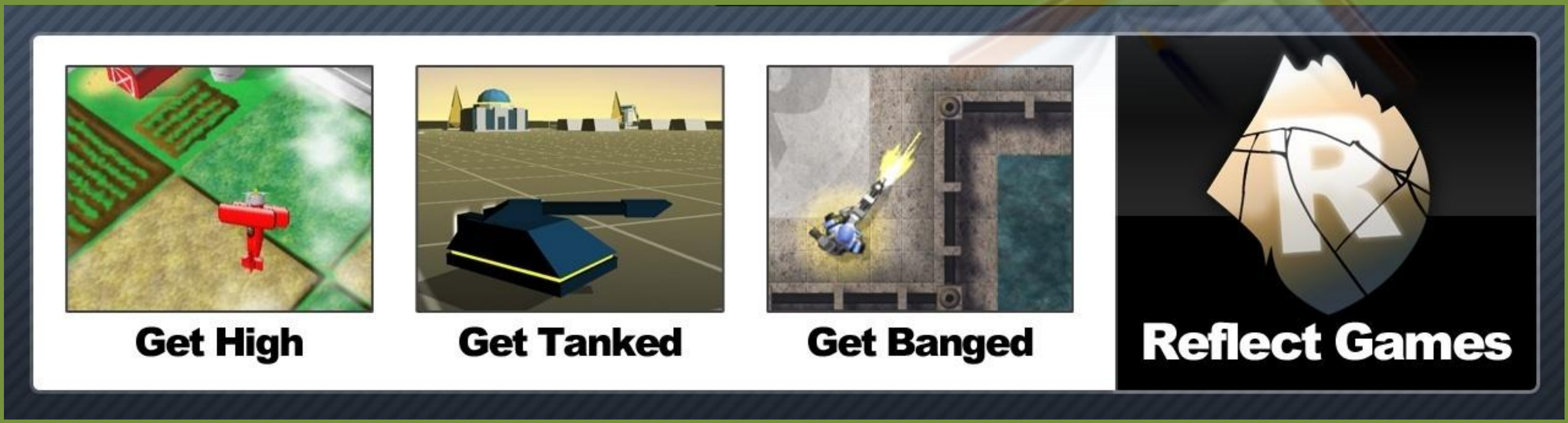
After all of that, I will begin coding the "Advanced Note Editor Program" which may eventually be distributed to the players. But, unfortunately this will delay the release of the new demo even more, but will result in very professional and well-made note charts.

Once I get that set up, I will be adding the option of 3D. One of the exclusive things that will be in the 3D version is lighting. Luckily, this will be easy, since GM has easy-to-use lighting functions. Another thing that will be exclusive in the 3D version is a couple of stage models or, just a stage background.

Some of you guys were wondering about the online and multiplayer also, these modes are almost done, a matter of fact, the current GOR demos have enough data in them to enter the lobby but it's all cut off by one line of code.

Some of the features the online version will have are: Pro Face Offs, Battles, and Co-Op. Pro Face Off being two players playing and the one to hit more notes wins, as for Battles it's the same as Guitar Hero III except, I plan to add more power ups.

Co-Op would be where one player plays Lead and the other plays Bass/Rhythm. Well, those are all the features that will be in God of Rock. And this concludes the April God of Rock Development Journal.



MarkUp Advertising

http://www.reflectgames.com/

Music Theory and Composition

David Perritte (MischiephX/Ds)

Music can be the difference between a good game, and a great game. This article will help you understand basic music theory and composition.

Let me begin by stating that I am writing these articles expecting you know the basics of reading music. (Knowing the musical alphabet)

Article 1: Major Scales, the Rudiment of Music Composition

A major scale can be classified as a rudiment of basic and complex music composition. It is a building block for several different kinds of scales, and can easily be skewed to create different sounds. First lets take the look at the format of a major scale:

Key:

H=half step

W=whole step

```
1 v 2 v 3 v 4 v 5 v 6 v 7 v 8
 W  W  H  W  W  W  H
```

So the eight notes of the scale are split in this way. You can start on any note and create the major scale by increasing each note by the indicated whole step or half step. So for example let's take C and create the major scale:

```
C v D v E v F v G v A v B v C
  W   W   H   W  W    W   H
```

It's really simple to build! The main importance of the major scale is the ability to change a few notes to create a different sound or scale. Learning your major scales can increase the quality of your composition. It adds a lot of stuff to your musical vocabulary when you learn these changes. Most seasoned musicians and composers will agree that this is an important first step to building music successfully.

Example

C-Major

```
C v D v E v F v G v A v B v C
  W   W   H   W  W    W   H
```

To convert this to a minor scale, simply add 3flats to the key signature. In this case we add Bb, Eb, and Ab.

C-Minor

```
C v D v Eb v F v G v Ab v Bb v C
  W   H    W   W   H    W   W
```

You can see that the whole step and half step format changes, this set up is characteristic of the minor scale and you can create a minor scale if you start on any note and adjust the note based on the indicated whole step or half step intervals.

As you can see its very easy to adjust the major scale to create a new and different scale/sound. This is why they are considered a rudiment in composition and theory.

I would like to answer questions you might have regarding anything about music theory and composition. Feel free to e-mail me at mischiephX@yahoo.com with any questions you might have and I will possibly address them in the next article. ■

Persistent vs. Global

Joshua Smith (jsmithlmsl)

Here it is; the famous question: "If I transfer from one room to another, how do I get my object to show up in the correct position of the next room, or the current room if I transfer back?"

Room Placement and Memory Preservation

In most cases, this question is referring to a "door" object and the programmer is trying to get that object to appear right in front of a door in this new room he is transferring too; possibly a house or shop in the game, whichever. The most common reply to this question will be "check the persistent box of the rooms you transfer between." The ol' quick and easy... Yes, persistent is an option and serves its purpose, but it is a costly tool to use.

For a small game, it really won't make a difference, but in a larger game with tons of rooms and objects, like an RPG, you will eat a lot of memory, your game will run slower, and you will lose some fan base because of that. In my opinion, people need a hell of a catchy theme or consistent change and game play before boredom sets in and they're off to another game, so saving every frame per second and loading second you can is very crucial towards the end of a large project. That's where global variables come into play. Let's talk about persistent.

By checking persistent within an object, that object will remember it's place (x and y) in whatever room you place it in; not good for room changing where doors are anywhere on the screen: top, bottom, middle, etc... For changing rooms and character placement, most would check the persistent box option of the room, so the player's exact position is remembered where it was last at before the transfer of rooms takes place.

By using that method, you are telling Game Maker to remember that exact rooms position at the time of room transfer on, regardless whether or not you will be going back into that room later on in your game. Why should you need to? Why make GM remember a room that is discarded? This also results in the objects sprite_index showing the incorrect direction for entering the room. Little things like a "Hero Character" entering a castle and switching to a room where he is now nose first in the door he just came through looks bad; it's lazy work.

Now I'm not saying that persistent is bad; sometimes like in puzzle games, it is needed dearly, but I simply recommend to all: if you can get around it, do so.

Room Changing Object

The most common method around persistent room placement is the "room changing object." You draw a small sprite, you create an object that holds this sprite; this object is not visible, not solid, and placed directly in front of a "door" object or tile in which the player can press a key and initiate a "room change." This method works, but will require a lot of different objects for all of the rooms you need to transfer around too; one to go to, and one to go back.

The common answer to what I just stated is "well you could just check the room you're in and then move to the next room based off of that answer." That does work, but at the same time, what if you had a room with 3 doors in it? What if you were in an outside area like The Legend of Zelda and there were 10 or 15 different rooms you could transfer to? You could no longer check the current room only, as there is however many possible rooms to transfer to. The answer is checking where your character object is (x and y), in what room, and basing your transfer off of the answer you receive. (EXAMPLE) If when my object initiates a room change, and I know he's currently in room_world, and his current y location is larger than 300 but smaller than 350, and his current x location is larger than 450 but smaller than 500, than I know where he is in the room and I can use this pinpoint to send the player directly to the next room they should go to. By checking the current room and narrowing down the players location, you can use 1 single object for every single room transition in a game, regardless of the games size, and you also do all of this without ever checking a persistent box in a room!

Room transfers covered, that leaves us with properly spawning the player in the correct position with the correct sprite of newly entered rooms. If I were to actually get into this with a full explanation here, I would eat up a lot more space than I'm sure MarkUp wants to give me, so I'll quickly cover the basics.

I use global variables to represent every room; one per room actually. Every time we change a room, I set the global variable for the room we're transferring to true in order to have a way of checking the current room, as well as the room we just came from. From this result, I set the player objects x and y coordinates to directly where they should be (generally in front of another door object) and assign it the correct sprite, then I switch some variable values again in preparation for when the player exits this room, then I exit the code. Here is a snip of my example showing a transfer from room2 to room 1:

Transferring to the new room:

```
if (obj_player.x<100) and room=room2
 {
 global.r2=false;
 global.r1=true;
 room_goto(room1);
 exit;
 }
```

Placing the player in the correct place with the correct sprite of the new room:

```
if global.r1=true and global.r2=false
 {
 obj_player.x=272;
 obj_player.y=160;
 obj_player.sprite_index=spr_left;
 global.r2=true;
 exit;
 }
```

Like said at the top of this column, the purpose here is to get around that persistent option and save yourself some memory, but even this example aside, you should always be looking and questioning everything you do and asking yourself all the while, "How can I make this better, faster, stronger, smaller, more efficient, more user friendly, better suited to my needs..." The list goes on and on.

As a programmer, that is your and my responsibility. The better and faster the route with less space used we find, the better for all of us it is in the long run. We're just lucky to have a simplified (yet still brain shaking) program such as Game Maker to do so with. On that note, I'll wrap this up and I hope you leave from reading this with a new idea and better confidence in what you do with your game.

I have written a fully commented example on how to use this method of room changing and persistent dodging. Within that example, there are 4 rooms, 1 of which has 3 doors. There is one single transfer object for the entire example, and I also demonstrate how to correctly position your player in the next room with the correct sprite facing the correct direction.

You can visit my topic in the GMC, post your comments, and download my full RPG Character Placement Example at this link:

http://gmc.yoyogames.com/index.php?showtopic=334149

Thank you MarkUp for the opportunity to try and help others and I hope you all have enjoyed this and hopefully learned something new in the process.

Beginning Theoretic Artificial Intelligence

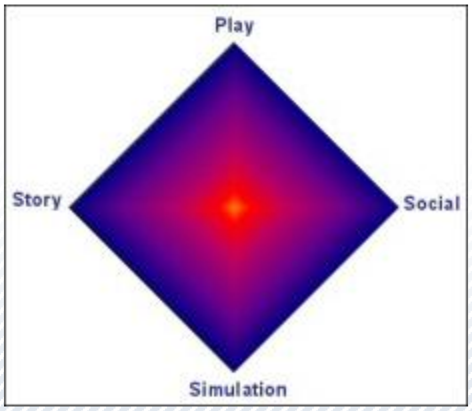
Darragh Tobin (Shadow Master)

One may say that Artificial Intelligence is the study and practice of intelligent behaviour, in animals, and the attempt to engineer this intelligence into machines, robots and programs. This said, beginners are allowed the presumption that Artificial Intelligence is reserved for Computer Scientists or Programming Gurus, but this is not always the case.

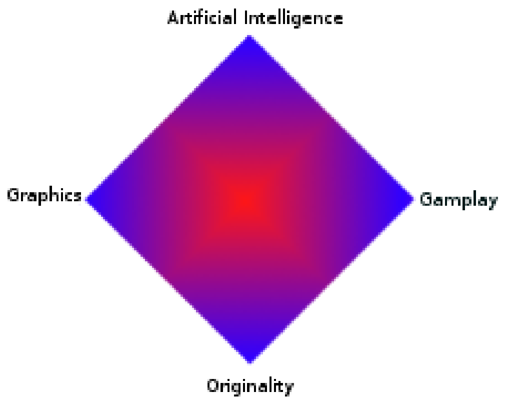
I'm writing this article, not to teach you the code behind quality AI, but rather the theory behind it, where or how one could apply it, the practical extent of its applications, and hence how to learn and even teach yourself what good gaming AI is.

In video games and most programs, AI has the principle goal of using known information to make a decision, similar to that a human would make. Depending on the situation, AI may need to be excellent, such as in sports games, in other games, this is not necessarily the case. However, when the AI needs to be at its best we may need to look towards some basic psychology to capture the essence of a character and what truly creates them (this really helps to immerse a player in a game.) Before beginning a new project, one should always plan out the extent of the AI needed for each object (for example, a bird in the background of a platform game would not need to do/know much, in contrast, a leading villain would.) Nevertheless when AI reaches a certain point, it can just make a game plain boring, make it just like normal life. After all, many people play games as a form of escapism, making them too realistic would take from the experience.

This brings me onto my next point about what Mark Overmars has aptly named "The Game Focus Diamond":



This measures what a game is going to be like (for example, Tetris is mainly in the 'Play' section), but I propose a new version, on the focuses of how the game would actually be made, not saying a game couldn't have all of these aspects, but it helps to know what the main concentrations are on, especially when time and/or capital are limited:



Each one is as important as the last; each one helps to create a great game, but very few games can capture them all. Getting back on track, this figure should help you to create balanced and 'well-polished' games, once you try to stay out of the blue/purple zones (the edges) in where your game is, but not in the very centre either. (For example, some games are driven by their originality, but not completely they also owe their success to enticing gameplay.) Play around with the chart and positions on it, until you find a good balance for you and your game (keeping your skills and capabilities in mind.)

It is clear that AI is not the only important part in any game, but it is, even if only in small quantities needed. As such, it needs to be done right, for the type of game you're making. In small, say puzzle based games, AI isn't big, but when it does come in, get it right. Think of the following:

- What stage is the player at, as in what level of experience they're at, where are they on the learning curve?
- What is the difficulty of said 'AI needing object' going to be? How much thinking do they need to do?
- How realistic does the AI need to be?

That last one is interesting, sure enough, most of the time AI in basic games will be kept to a minimum, and consequently doesn't need to be too realistic when it is in use, but if you're going for top range human AI, in a war game, as I have said already keep psychology in mind. Psychology is a very broad topic; you could be qualified in it, but still have so much to learn, in general, it is the scientific study of mental process and behaviour. Only the basics would be used in AI, (very few games would dare to try out the unexplored realms of intelligence).

We all know these basics, it's just a matter of how we would put them into context, and capture them in our games, we'll look at them socially, through Intrapersonal (existing or occurring within the individual self or mind, although not always consciously) and Interpersonal phenomena to try and capture what is needed for believable (therefore higher-quality) characters. These phenomena include:

- Attitudes: They must be created through and even developed throughout the story. Depending on the type of person, their attitude would differ (for example, a witty character (the games comic relief) would usually have a good attitude) depending on their situation, ones attitude to something or someone could change drastically. In serious games, this is important.
- Social Influence: This is the way that people affect others around them, this would be difficult to implement with code, but instead of that, implement it in the story or script.
- Interpersonal Perception: Simply put, this examines the beliefs that interacting people have about each other, such as their feelings, agreements, similarities, et cetera, et cetera.
- Social Cognition: This is growing more popular, and although it is an interesting study, it can be complex. It covers how people perceive, think about, and remember information about others.

There are more that we could add, but in my opinion they are the most important. They cover the social side of things, but in many games, that isn't too important. So now we're going to take a look at environmental psychology, to cover the, well, environmental side of things. This is an important topic, but it can get to be a very tricky. It boils down to how we react to our surroundings. But of course we all react differently, and that is where the problem arises. To find out how somebody would react, we need to look at his or her social psychological situation. What is their attitude, social cognition, et cetera? Because there are so many different types, it is best to map this one out for yourself. You would need to map out what the objects current environment is, and how to do that? I don't want to delve into code, but there are ways (`point_distance(x1,y1,x2,y2)`, `distance_to_object(obj)`, `distance_to_point(x,y)` `position_empty(x,y)`) so the situation is mapped out, but how to decide what to do or how to react? The answer is to remember the characters personality and attitude. That means that you won't be copying and pasting much code for main characters (assuming they're different) but that is a good sign. It means that you have reached a state, where your game is not all the same, which gives the player more variety and more to play for. Look at a soap opera, for example, wouldn't it be boring and unrealistic if just followed the events of one person?

So this brings my article to a close, the end for my teaching you, (for now anyway) but far from the end for you. If you want to learn more, start trying out new games, make them simple, but try to do them all in code. Then you can grow more advanced with each new project. For further reading, or if you're stuck, go to the GM help document, or look through the built-in functions/variables, and make sure to check out Wikipedia for more psychology information. I hope I've encouraged you to make something great now, so good luck, and happy coding!

Efficient Source Code

Bart Teunis

Although a finished game or program doesn't need the source code anymore, that source code is quite important. Suppose you want to make changes to the source code later on. Depending on the way you wrote that source code, it can be easy to make changes to it, but you can also have a lot of work. This article will explain a few reasons why it's best to keep your source code neat and some ways to actually achieve this.

Maintainability

This is definitely an important reason to keep your source code neat. Suppose you need to draw 3 rectangles with a size of `320x160 px`. You can do this by placing this piece of code in the draw event of some object:

```
draw_rectangle(x,y,x+320,y+160,0);
draw_rectangle(x,y+164,x+320,y+324,0);
draw_rectangle(x,y+328,x+320,y+488,0);
```

Suddenly you think that a window size of `160x120 px` would be better. You now need to change all numbers in the piece of code. In this example, it wouldn't be too hard to do that, but for larger pieces of code, it takes a lot of time to change all the numbers. This also means that you'll have to do all the calculations again. And looking at the apparently random numbers, you wouldn't expect this code to draw 3 rectangles of the same size. This way of coding is called hard coding. The good way to do this is to set a variable (or a named constant) for each property, then replace the constant values in the piece of code with the necessary variables (or a combination of them). In this case, we need three variables: width, height and distance. distance is the distance between two rectangles. The piece of code now looks like this:

```
draw_rectangle(x,y,x+width,y+height,0);
draw_rectangle(x,y+height+distance,x+width,
y+2*height+distance,0);
draw_rectangle(x,y+2*height+2*distance,x+width,
y+3*height+2*distance,0);
```

In the create event we define the required variables:

```
width = 320;
height = 160;
distance = 4;
```

Now the only place where you need to change the width, height and distance between rectangles is the create event. Also, you can now clearly see in the draw event where exactly the rectangles are drawn. The first one is drawn at the object's x position and has a width determined by the variable `width`. The same for y. The next rectangle's y position is `distance` farther than `y+height`, which is the y position of the end of the first rectangle. It gets quite complicated for the third rectangle, so it'd be better to put the code in a loop:

```
var i;
for(i=0;i<no_rectangles;i+=1)
draw_rectangle(x,y+i*distance+i*height,x+width,
y+i*distance+(i+1)*height,0);
```

To further demonstrate the use of variables, the number of rectangles to be drawn is determined by a variable no_rectangles.

This example should make clear that it's best to always use variables or named constants where possible. Then it'll be much easier to change the code later on. Also, all calculations are now done when the game or program is running; you can simply see them in the source code. That way, you have a nice overview.

Readability

The following example will clearly demonstrate why it's important to keep your code readable (note: the dir_get_content script gets all files in a directory and puts the filenames in a list):

```
var tmpdir;
tmpdir=get_directory_alt("Open  a  directory:
","")
if tmpdir!=""
files=dir_get_content(tmpdir);
sf_files=surface_create(316,320);
surface_set_target(sf_files);
draw_clear_alpha(c_white,1);
var n;
for(n=0;n<ds_list_size(files);n+=1)
draw_text(2,2+n*15,ds_list_find_value(files,
n))
surface_reset_target();
```

This piece of code is anything but neat. Time to make some improvements.

There are two ways to make this piece of code look a bit neater. First of all, you can indent some lines of code. This is very useful for large blocks of code starting with { and ending with }. In the example above, the lines after the if statement and the for loop can be indented to make the code neater. GM has a nice option to make code indention easier: smart tabs. This option can be enabled in File ➔ Preferences ➔ Scripts and Code. When this option is turned on, text on a new line will start at the same position as the text on the previous line (or on a logical position, again based on the text on the previous line).

Another thing you can do is add comments. You can add comments anywhere in a piece of code. The most logical places to put them are above a block of code or to the right of a line of code. You could also add comments in the middle of a line of code:

```
draw_text(5,5,/*draw some text, ...*/"text");
```

But it doesn't really improve readability, does it?

You can also separate blocks of code by an empty line. It also improves readability a lot.

And last of all, it'd be best to keep to the correct syntax. In GM, you can do a lot without having syntax errors e.g. leave out semicolons, not place brackets around **if** conditions, use = instead of == for comparison, ... But it's still better to keep to the correct syntax. An example: which one is the easiest to read? :

```
if tmpdir!=""

if (tmpdir!="")
```

The brackets improve the readability, don't they? This is also useful with operator precedence. Although a calculation might be correct without brackets, it might not be a bad idea to add brackets. In this case, brackets are not obligatory, but they can improve readability a lot.

Now that we know all that, let's have a look at the improved piece of code:

```
//prompt for directory and get filenames
var tmpdir;
tmpdir=get_directory_alt("Open a directory: ","");
if (tmpdir!="")
  files=dir_get_content(tmpdir);

//create a surface to draw on
sf_files=surface_create(316,320);

//draw on the surface
surface_set_target(sf_files);

  draw_clear_alpha(c_white,1);

  var n;
  for(n=0;n<ds_list_size(files);n+=1)
    draw_text(2,2+n*15,ds_list_find_value(files,n));

surface_reset_target();
```

As you can see, some comments have been added before each block of code, the code is syntactically correct and the text after the if statement and for loop has been indented. Also, some empty lines have been inserted to group lines of code dealing with the same action (which is described in the comment above each block).

Naming conflicts

Common causes of errors are naming conflicts. Suppose you have a character object and a sprite for the character. Since a sprite and an object are two different resources, you might think it'd be possible to give them the same name. So you add a sprite called character and an object, also called character. There is one case where this wouldn't cause a problem, which is when the id of the sprite and object happen to be equal. But most likely that's not going to be the case so character, which is actually a constant, can be either one of the two values. That means that if you're going to draw the sprite, it might not be the sprite you were expecting. If you're referring to the object, you might be referring to another object. Game Maker has a feature to check for conflicting resource names: Scripts ➔ Check Resource Names. It's an easy way to find resource naming conflicts.

Not only resources, but also variables can be the cause of naming conflicts. An example: a script is run several times in a for loop. The loop and the script look like this:

Loop

```
for(i=0;i<25;i+=1)
{
  scr_iterate();
}
```

Script

```

//this script repeats a certain action 27 times

for(i=0;i<27;i+=1)
{
  test = 0;
}
```

How many times do you think the loop (not the loop in the script) will be executed? If you think 25 times, you're wrong. The loop is executed only once. The reason is that the variable i is the same variable in both cases. Both i's do not only have the same name, but it's just one single variable (in memory). That means that both loops change the same variable. After the script has been executed, i is 27. That same i is now no longer smaller than 25, so the loop ends. It's an annoying problem and it can be hard to find. That's why you should always use **var** to declare these variables local to the piece of code. In the example, it's sufficient to add "**var** i;" before the loop. The two variables i are now two different variables (in memory), while they have the same name.

The best thing to do is to use local variables wherever you can. That's the easiest way to avoid naming conflicts with variables. And you should use as few global variables as possible.

A proper definition of everything

It is very important to name everything correctly if you want to keep an overview over your code. Something important you need to do here is the following: Never, and I repeat, never, turn on "Treat uninitialized variables as value 0" in the global game settings. Why not? Well, you're not defining everything properly then. And it might cause errors, too.

An important thing are resource prefixes e.g. instead of naming an object character, you'd better call it obj_character or objCharacter or even ObjCharacter. This has a few advantages:

- When looking at a piece of code, you immediately see whether a variable is just a normal variable or a resource id.
- The problem of resource naming conflicts is solved.

Efficient Source Code Cont.

A resource prefix usually consists of 3 letters; in the example above the prefix is obj. Some other object prefixes that can be used: spr, bck, fnt, rom, lst, map, sys, sf …

I also added a few prefixes for some other types of resources (the ones that aren't shown in the resource tree). It's a good thing to also add a prefix to those resource names.

Another thing to keep in mind: when naming a resource, make sure the name explains the function of the resource, rather than it's content. You could e.g. have a font called fnt_arial. This font is used as the font for the text in a custom messagebox. But its name implies something else. It is rather probable that the font Arial is required for other text, e.g. a text header. But the font for this header will probably be larger. You could call this font fnt_arial_large. Imagine you need to do this for several fonts. There is a better solution. A good name for the messagebox font would be fnt_msgbox. Now you know immediately that this font is used for the text in a messagebox. The name for the header font would be fnt_header.

If you want to define everything properly, it might not be a bad idea to add a short header to scripts. This header contains some information on the script: explanation, arguments, return value, remarks…

Modularity is another interesting thing. It means that certain parts of a game can work independently. Examples would be a physics engine and a resource management engine. Both have nothing to do with each other and thus can work separately. Later on, if you need to make changes to one of the engines, you can edit it without having to be afraid that anything will go wrong in the rest of the game.

Be consistent when programming

There is not always a single correct solution to a coding problem and everyone writes code differently. The most important is that you should be consistent when programming e.g. always use the **var** keyword, always use the same naming convention (capital letters, underscore…), always use the same header for scripts, always use `&&` (or **`and`**)…

That way, it will be easy to interpret your code later on.

Conclusion

This article has explained how to write and maintain efficient source code. There are a few reasons to write efficient source code: maintainability, readability, avoids naming conflicts. Also, you have to define everything properly and you should be consequent when programming. It'll definitely make things easier for you as a programmer.

AI Command Stack

icuurd12b42

I developed the AI Command Stack when I started to hits a few limits in what I could do in a step event. I was getting lost in a code block 1000 lines long which would make my AI patrol or attack or find a base for repair or hunt for health packs. It all sounds simple but when the code gets to be too big, it becomes unmanageable and slow.

Also, the ability to make my AI change its mind and resume what it was doing... That added a level of complexity that became extremely difficult to manage. And I was limited to only one level of "resume prior task". I won't bore you with the details of the implementation of the original troublesome AI code.

I needed a system where I could tell my AI to patrol an area; if, while patrolling, it would see an enemy, it would attack it; if, during the attack, it would need to repair, hunt down repair packs while defending itself; then resume the attack; then resume the patrol...

Add on top of this, AIs that actually defend and escort another AI (or player) and your code becomes too complicated... Or the number of object types increases to the point you are confused as to what type of object you should use.

In my case, in The Tank Game (yoyogames.com/games/show/6777), we ended up having so many tank types which none would do the job well enough for multiple roles.

I needed a system where I could change the behaviour of the AI on the fly with no consequences to what it was doing before. Turn a patrolling tank into an escorting tank... Even upload a mission in the tank, like stop patrolling, go over an area and destroy the enemy base there, then escort the POW back to your base...

I needed to make the amount of code executed smaller to bypass GML's slow interpretation. The huge amount of code in the tanks would limit the number of AIs to a maximum of about 12-20 tanks. The problem caused not by the amount of code that made the tanks run, but by the amount of code that was skipped over using the "if" statement. That alone caused the biggest strain on the CPU.

It hit me that I could code the patrol behaviour in a script and the escort behaviour in another script and that I could switch between the two scripts in my step event. Even write my player control code in a script so I could actually switch to that script to control any tank.

Then it hit me that I could write little scripts instead of a huge control script/state machine and that I could string the scripts into a series of commands using ds_list and ds_stack. Resulting in a very powerful system.

How it works

Let's take a look at how it works. Starting from a simple movement script, removing the stack system from the example to ease the understanding of the concept, you can make a simple patrol system this way:

Scripts

Script MoveToXY:

```
//MoveToXY(xto,yto,movespeed)
//returns 1 when task is complete, 0 when
not
var xto; xto = argument0;
var yto; yto = argument1;
var movespeed; movespeed= argument2;
//Move To XY at speed, or at distance left
so not to overshoot
move_towards_point( xto, yto, min(movespeed,
point_distance( x, y, xto, yto)));
if(x = xto and y = yto)
    return 1; //done
return 0; //not done
```

Objects

In your AI create event you have:

```
m_curpoint = 0;
```

In your AI step code you have:

```
if(m_curpoint == 0)
m_curpoint+=MoveToXY(10,10,5);
else if(m_curpoint == 1)
m_curpoint+=MoveToXY(10,100,5);
else if(m_curpoint == 2)
m_curpoint+=MoveToXY(100,100,5);
else if(m_curpoint == 3)
m_curpoint+=MoveToXY(100,10,5);
else if(m_curpoint == 4) m_curpoint=0;
```

This makes the AI do a square patrol. m_curpoint increases each time the move is done. But you see, all these "if" will eventually task GM's interpreter the more you add to the system. Especially when they are a lot of them (10-50).

Now, let's add in an imaginary list system that allows storing the script and it's parameters in it.

Assuming

- list_create creates a command list
- list_add_command adds a command script with its parameters to it
- list_execute_command executes the script at position specified, returning the value the script returns (1 when done, 0 when not)
- list_num_commands returns the number of commands in the list

Objects

In your AI create event you have:

```
m_curcommand = 0;
m_list = list_create();
list_add_command(m_list, MoveToXY,10,10,5);
list_add_command(m_list, MoveToXY,10,100,5);
list_add_command(m_list,
MoveToXY,100,100,5);
list_add_command(m_list, MoveToXY,100,10,5);
```

In your AI step code you have:

```
m_curcommand += list_execute_command(m_list,
m_curcommand);
if(m_curcommand                              >=
list_num_commands(m_list)) m_curcommand=0;
```

This makes the AI do a square patrol. m_curcommand increases each time the command is done, using our imaginary system.

OK, I see, but where does it attack?

The second part of the **Command Stack** is the stack system (Where the name originally comes from). The stack is like the list but commands in the stack are executed in a last in - first out manner. And commands in the stack are executed first, before commands in the list... Commands in the list are not executed if there are commands in the stack.

The Command Stack system actually removes the current list command from the list and pushes it in the stack which is more efficient but for the explanation here, I will keep it simple to avoid confusion.

Let's add another script:

Scripts

Script PatrolToXY:

```
//Script PatrolToXY
//PatrolToXY(xto,yto,movespeed,enemyobject,
// range)
//returns 1 when task is complete, 0 when
//not pushes AttackEnemy to the stack if
//enemy found
//var xto; xto = argument0;
//var yto; yto = argument1;
//var movespeed; movespeed= argument2;
var enemyobject; enemyobject= argument3;
var range; range= argument4;
//get closest enemy
var                enemyid;                enemyid                =
instance_nearest(x,y,enemyobject);
if(enemyid)
{
```

AI Command Stack Cont.

```
  //is it in range
  if(point_distance(x,y,enemyid.x,enemyid.y)<range)
  {
    //push the attack on the stack
    stack_push_command(m_stack,
AttackEnemy(enemyid,range,argument2));
    //not done;
    return 0;
  }
}
//move towards patrol point;
return MoveToXY(argument0,argument1,argument2);
```

And another script:

Script AttackEnemy:

```
//Script AttackEnemy
//AttackEnemy(enemyid,range,speed)
//returns 1 when task is complete, 0 when
not
var enemyid; enemyid= argument0;
var range; range= argument1;
var movespeed; movespeed= argument2;
//is enemy still alive
if(instance_exists(enemyid))
{
  //is it in range
  if(point_distance(x,y,enemyid.x,enemyid.y)<range)
  {
    //move to it

move_towards_point(enemyid.x,enemyid.y,movespeed);
    //kill it if we touch it... simple
    if(place_meeting(x,y,enemyid))
    {
      with(enemyid) {instance_destroy();}
      //done;
      return 1;
    }
    //not done;
    return 0;
  }
}
//done;
return 1;
```

Assuming

- stack_create creates a command stack
- stack_push_command pushes a command script in the stack (top of stack) with its arguments
- stack_execute_command executes the script at the top of the stack, returning the value the script returns (1 when done, 0 when not)
- stack_is_empty returns if the stack is empty
- statck_pop removes the top script from the stack

Features: A full level system complete with exp, leveling, health, defense, attack, and skill levels - Completely compiled using GML, no drag and drop at all - All expressions, actions, and statements are thoroughly explained along side a definition for every function used - a simple built in attack system - level based sprites for the player - drawing the hud and all level information onscreen using only GML and much, much more...

"This is a great, simple, engine which still does the job. 10/10"

JsmithLMSL's

Level System Tutorial v1.2

1500+ downloads and counting...

"This is certainly the best exp system I've ever seen"

MarkUp Advertising http://gmc.yoyogames.com/index.php?showtopic=346888

AI Command Stack Cont.

icuurd12b42

Objects

In your AI create event you have:

```
m_curcommand = 0;
m_list = list_create();
list_add_command(m_list, PatrolToXY,10,10,5,
EnemyObj, 200);
list_add_command(m_list,
PatrolToXY,10,100,5, EnemyObj, 200);
list_add_command(m_list,
PatrolToXY,100,100,5, EnemyObj, 200);
list_add_command(m_list,
PatrolToXY,100,10,5, EnemyObj, 200);

m_stack = stack_create();
```

In your AI step code you have:

```
if(stack_is_empty(m_stack))
{
    m_curcommand +=
list_execute_command(m_list, m_curcommand);
    if(m_curcommand >=
list_num_commands(m_list)) m_curcommand=0;
}
else
{
    if(stack_execute_command(m_stack))
stack_pop(m_stack);
}
```

This makes the AI do a square patrol. m_curcommand increases each time the command is done, using our imaginary system. If the patrol script detects an enemy of type or parent type EnemyObj within range, the attack script is pushed in the stack (that code is in the PatrolToYX if you missed it). The attack ends if the enemy is destroyed or moves out of range and the patrol resumes.

Conclusion

That is how the Command Stack fundamentally works. You have your basic behavior defined in your create and step events and the behavior changes according to the circumstances by the way you implement your scripts. And, at anytime, you can destroy the basic behaviour set and replace it with another by simply emptying the command list and stack and adding another set of commands.

The actual system can be found here.

http://gmc.yoyogames.com/index.php?showtopic=335600

PatrolToXY
10,100
Pushes: AttackEnemy

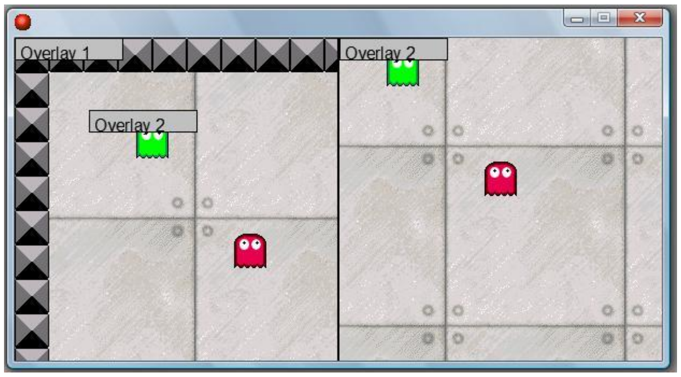
MarkUp Advertising

http://markup.gmking.org/

Split Screen

Multiplayer games are huge these days. Many games you find use a split-screen system to allow two or more players to play on the same computer. Many beginner level GM users have some problems when making split screen games.

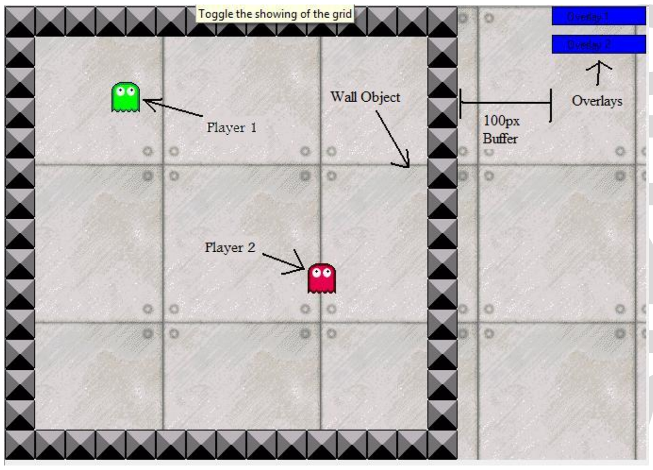
The main issue is adding an overlay, to display a score, timer, etc. The problem that a lot of people seem to be having is that when you add an overlay, and one player enters the other's view, the overlay is shown in the wrong view, and it looks absolutely horrible (Picture at right). I have two simple solutions to remedy this problem.



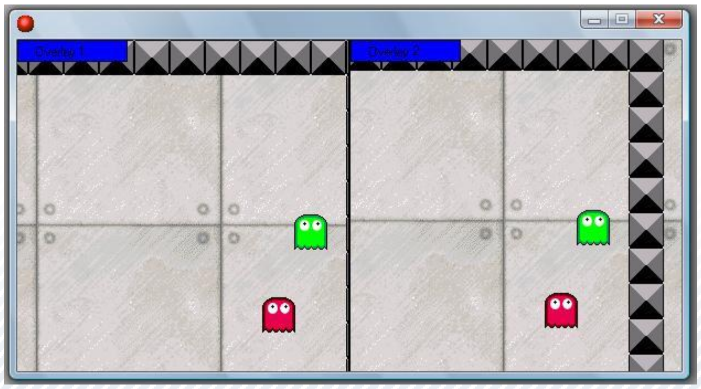
If you are a beginner in GML and have little to no experience, here's what you do. First, you want to make a wall object that the players can't pass through. Line the right border of the room with this object. You might already have a wall object around your entire room, which is great. If your game lets the players warp across the room, this wall object can be invisible and on collision warp the player. Either way, the purpose of the wall object is to cordon off the right side of your room.

Now that we have our wall, we want to extend the width of our room. However big your overlay is, you want to add 100px buffer to the width of your overlay, then add that amount to the width of your room.

This creates a 100px buffer zone between the edge of the playing field and the edge of where your overlay will be. Now you want your overlay object to draw the overlay at the right edge of your room. If you have multiple overlays, put one on top of the other (picture below of what the room should look like).



Ok, now we need to add some extra views, one for each overlay. We position those views over the overlay objects, and set the port on screen to where ever the overlays would show up in the game. So if our overlay would be placed at the top left corner of the view, which is where we would place the view of the overlays. I completed these improvements, and placed a picture below of what the game would look like. One last thing, if when your player moves all the way to the right of the screen and you can see the overlay a bit, increase the 100px buffer until you can't see the overlay anymore.



Split Screen Cont.

Another solution to this problem requires a bit of GML coding under your belt. This will accomplish the same task with a little less work, but it's a little more complicated. I recently found, in the help file, a snippet of code that allows you to turn off the overlay if it is in the wrong view. The statement `view_current` allows you to check what the current view is. So, if the current view is the wrong one, we simply don't display the overlay. Here's a little what the code would look like:

```
{
//player 1's view is view[0]
  if (view_current == 0) {
  //this would be the draw functions for
the overlay
  draw_rectangle(view_xview[0],
view_yview[0], view_xview[0]+100,
view_yview[0]+20, false)
  }
}
```

It's that simple. The code for player 2's view would be the same, just change the 0 in the if statement to whatever player 2's view is.

Conclusion

There you have it, two simple ways to add in overlay in your split-screen game. I hope this helps a bit to solve some of the problems people have had with split-screen games.

GM Color Extension

One of the basic features of Game Maker 7's Extension Mechanism was to allow the addition of constant variables directly into a game's source file in a way that makes them seem native to the IDE. Indeed, one of the first hypothesized uses of this capability was the simple addition of new colors to Game Maker, and indeed again, one of the first useful extensions made for Game Maker 7 *is* the Game Maker Color Extension. This adds more than 260 colors to Game Maker, many of which are interesting colors. A very simple extension yet could be needed if you're not too familiar with RGB and HSV colors.

gmc.yoyogames.com/index.php?showtopic=343006

Seven Minutes

In the 'Creative Game Ideas' department, we have "Seven Minutes" by 'Virtanen', a puzzle platform game that lasts for exactly seven minutes: your last seven minutes of life. The game starts after touching what appears to be an important object that reveals to you the end of the world, and is then given seven minutes of life to find out. Extremely challenging, extremely intense, and – at times – very philosophical, try it out!

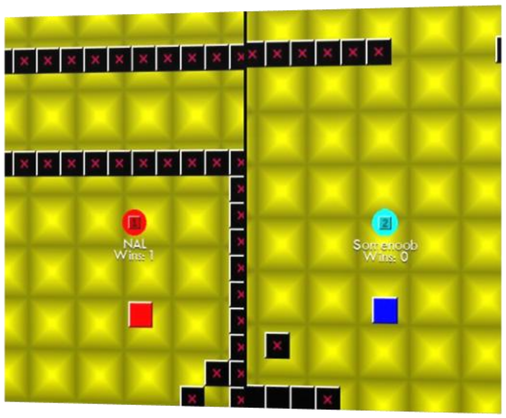
www.yoyogames.com/games/show/26339

Elemence Aux

"Take control of one of six elements in this strategic platformer. Use each one's unique power to help otherwise-uncrossable gaps and avoid danger. Simple graphics, five difficulties and a three-level multiplayer race."

What they say

"It's a good game I like it. Worth download **[sic]**.*"*



Description

Elemence Aux is a puzzle platformer, where you control an element. The idea is to simply go past them, while facing dangerous jumps and other sections, where death is very close. The player can collect points, which can be used to create blocks underneath the player, to go through otherwise impossible sections of the map. The level design is a bit repetitive, however.

Elemence Aux is an addicting minigame. The sound & graphics aspects of the game are, however, quite poorly done. The music consists of MIDI songs, and the game features little sound effects. The graphics are very simple (although the game features small particle effects).

The game has a small backstory; *"Four scientists were on a quest to turn gold into the most expensive element discovered so far – Californium. Unfortunately, the electrons misfired, warping the gold into six mind-of-their-own gold derivatives."*

The game is a fun play, and has been designed well. The technical execution suffers of some minor problems (for example, the player got stuck at one point of playing). It also features many difficulty levels, and a multiplayer mode. In overall, it's a game worth testing and playing.

Reviews

Pros and Cons

Elemence Aux is worth a try, and a quick timekiller.

Pros

- Good idea
- Addicting gameplay
- Technically implemented well

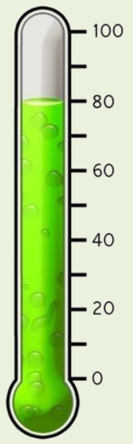
Cons

- Poor sound effects and music
- Poor graphics
- Repetitive level design

Conclusion

While Elemence Aux suffers of some problems, it's still a good game.

Ratings



Graphics: 4/6
Sound: 4/6
Gameplay: 5/6
Storyline: 4/6
Design: 5/6

Developer: NAL Games
Game Maker: Version 6

PLAY NOW

Ratings

Instant Play

Phillip Gamble

Since YoYo Games added the "Instant Play" feature to their website in September 2007 gamers have been able to play Game Maker created games without the need to download and extract game files. Until the end of January YoYo Games was the only site offering this feature. Users of a compatible browser that have installed the YoYo Player plugin can take advantage of the ability to automatically download game files to a location on their computer which are then executed.

It makes good commercial sense for YoYo Games to offer this feature. Firstly, I am not aware of any other free gaming sites that offer anything similar to Instant Play. The announcement generated some publicity, being picked up on many Indie Gaming sites.

The main reason for the addition of *Instant Play* is of course that it brings people back to the YoYo Games website, much more lucrative to YoYo than someone who plays games from a folder on their desktop.

Thirdly, and this isn't a point I fully agree with, *Instant Play* makes it easier for gamers to play Game Maker games. Certainly the need to choose a location to save the game, and then extract the folder is removed – but I don't think this is a serious problem. Finally bandwidth is saved, for both YoYo and gamers, as you only need to download the GameMaker runner the first time you *Instant Play* a game made with a particular version of GameMaker.

Instant Play is currently compatible with IE 6+ and FF 2.0+ and requires Windows 2000 or a later version. If you haven't yet used instant play you can install the ActiveX control or FireFox plugin when you next go to play a game hosted on YoYoGames.com

Making your game *Instant Play* compatible:

- Either upload the executable of your game directly or use a zip file containing the executable and other files your game need. The other files can be in a subdirectory if you want but make sure the game executable is in the main directory. Don't use another compressor than zip.
- Don't use an installer for your game.
- If you want to upload an editable version of your game, also include the executable in the zip file.
- Make sure there is only one executable in the main directory of the zip file. If you need additional executables (such as a level editor), make sure they are in a subfolder.
- Use *Game Maker* 7 or 6 for your game.

Source: http://mark.glog.yoyogames.com/?p=13

Instant Play FAQ

What is downloaded when I Instant Play a game?

The original .exe file is separated from its GameMaker runner and is given an extension which relates to the version of GameMaker in which it was created. For example .g70 games were made with GameMaker 7 and .g61 games were compiled in GameMaker 6.1.

What about saved files, DLLs and external sounds, will those work?

External resources such as DLLs and data files are also downloaded and work as they would with the normal .exe file. Save files are stored on your hard disk.

Do uncompiled GameMaker files (.gmd, .gm6, .gmk) work with Instant Play?

No. Only GameMaker 6, 6.1 or 7 executables work with Instant Play.

Instant Play

Where are the Instant Play files stored on my computer?
Instant Play files are stored under a subfolder of the Game ID in a folder called "YoYoGames" in My Documents.

What about the Instant Play runner?
Each version of GameMaker requires a different runner. The runners are automatically downloaded from YoYo Games when you try to instant play a game created with a particular version of GameMaker. They are installed under *DEFAULT\Documents and Settings\All Users\Application Data\YoYoGames*.

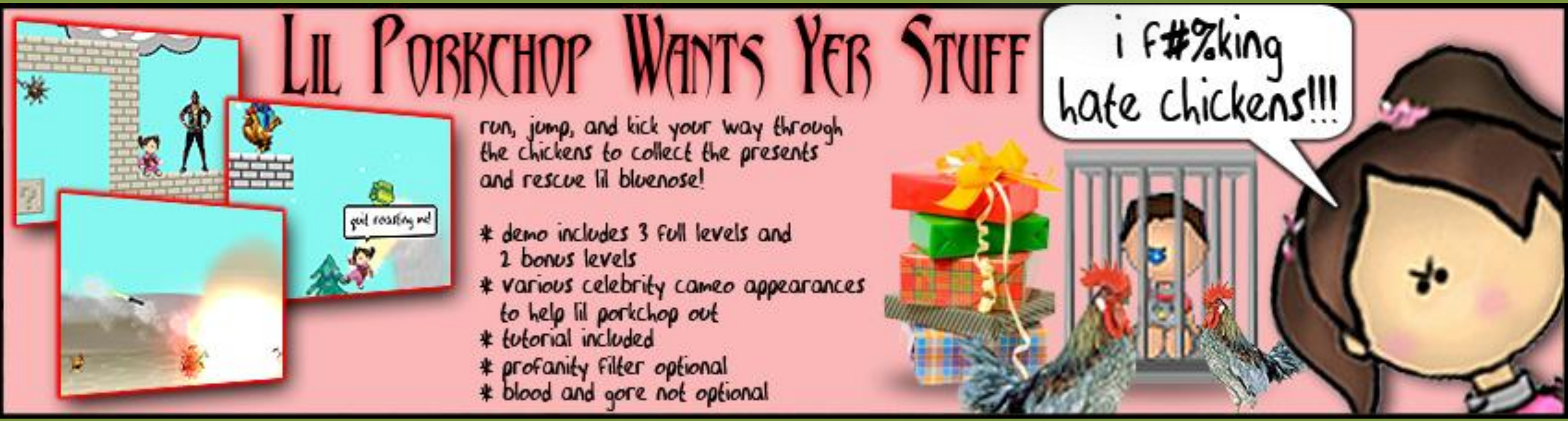
Do the files delete themselves after I have finished playing?
No, after you have finished playing the files remain on your computer.

Can I use Instant Play when I am not connected to the Internet?
No, you need to be online to run *Instant Play* programs.

Why isn't my game working with Instant Play?
First check that your game is compatible (see previous page). It takes a while for the files to be prepared for *Instant Play*.

Why is there a delay between my game being uploaded and my game working with Instant Play?
Uploaded files are scanned for viruses before being made *Instant Play* compatible. Games also need to be split from the GameMaker runner which is included in all GameMaker created executable files.

Compiled from the GMC and information provided on www.YoYoGames.com


LIL PORKCHOP WANTS YER STUFF
i F#%king hate chickens!!!
run, jump, and kick your way through the chickens to collect the presents and rescue lil bluenose!
* demo includes 3 full levels and 2 bonus levels
* various celebrity cameo appearances to help lil porkchop out
* tutorial included
* profanity filter optional
* blood and gore not optional
quit roasting me!

MarkUp Advertising
http://www.yoyogames.com/games/launch/15130

Goodbye!

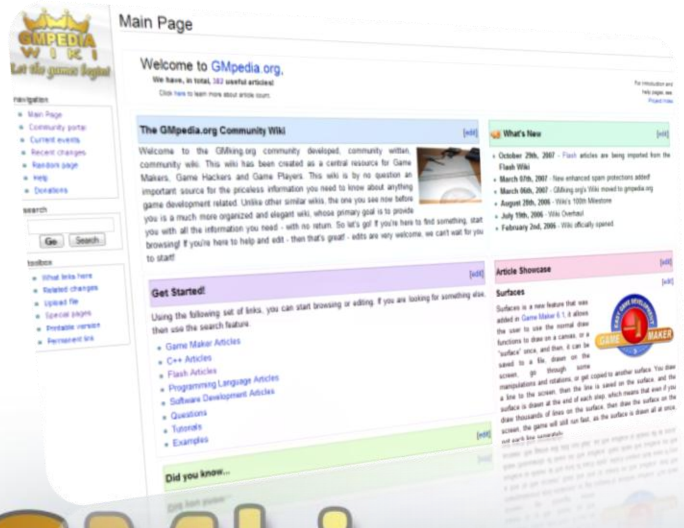
And that was Issue 13 of MarkUp Magazine! We truly hope you've enjoyed reading this little issue.

MarkUp Magazine is supported by contributors! If you are a reader of MarkUp Magazine, then you can help us become better by sharing your experience with us and the readers! With feedback, you can help by either joining the MarkUp forum, or e-mailing the MarkUp staff. You can also help us by sending us your own content, to do so, please check out the contribution page on the MarkUp Magazine website! The contribution page also includes a list of advantages to becoming a contributor, including information on advertising space and free game development books. Remember, you don't *need* to apply to become a staff member, you can just write for us!

The MarkUp Staff ■■

Be Sure to Check Out…

GMking.org is the parent network for **MarkUp Magazine**. It is constructed as to behave like a centralized portal that links to the four main aspects of GMking.org's projects: The GMking.org Site [which is now a sub-site of the main gmking.org page], The GMking.org forums, GMpedia.org, and MarkUp magazines. Visit the site for MarkUp's entire set of sister projects!



One of MarkUp's sister-projects, also developed and maintained by GMking.org, is GMpedia.org. To learn more information about your Game Platform of choice, you could check out GMPedia.org. GMPedia is a game development wiki with a growing community-base and content. GMPedia is not limited to Game Maker, but expands to include all forms of game development, including Flash, etc.

GMPedia.org has also been tied deeply with MarkUp Magazine. For more information and detail on certain topics discussed in MarkUp Magazine, visit GMpedia.org!

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!