MarkUp

SIXTY FIVE MILLION AND ONE BC

The **Latest** demo you've ever seen!

DETAILED STEP-BY-STEP TUTORIAL!

Artificial Intelligence

GAME DEVELOPMENT JOURNALS

God of Rock & Falcon Wars

GAME MAKER INSIGHT

Functions in Game Maker

AND MUCH MORE!

LEARN HOW TO PARTICIPATE IN

THE GAME MAKER RACE

ON PAGE 15

Interviews | Resources | Previews | Reviews | Tutorials

Editor's Desk

Open Source, and Open Source Game Development

We're back! There's another issue inside, with all the information you need to make your game shine. But before you get into that, I want to rant on a little about open source. Everyone who I've talked to for more than 10 minutes knows I love open source software, open source ideals, and open source content; it's part of the reason why MarkUp is open source.

But there is a lot more open source content out there that many people don't know about. For example, if you do an advanced Google search you can specify to only search for creative commons licensed content. What some photos to throw into your next project? Yotophoto can find them! Feel like doing some reading? Grab some classic books off gutenberg.org!

Maybe you want some open source music, to listen to or remix? http://www.owlmm.com/ has filters to only choose open source music. And, most everything on archive.org is under a liberal usage license too!

And don't forget sf.net and freesoftwaremagazine.com for open source software!

See you next month!

Robin Monks■

Contributors

Robin Monks	Sr. Editor
Eyas Sharaiha	Editor
Andris Belinskis	Editor
Philip Gamble	Writer
Bart Teunis	Writer
Thomas Hansen	Writer
John Leffingwell	Writer
Jonah Turnquist	Writer
Veeti Paananen	Writer
Michael Moore	Writer
Jono Alderson	Writer
Tarik Abbara	Writer
Mathew Malone	Writer
Robert Colton	Writer
Dan Meinzer	Graphic Designer

Table of Contents

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

65 Million And 1 BC

Overview

What an awesome, awesome game! From the moment I double-clicked that small icon until the moment I exited the game the experience was excellent. Right when the game opened, I knew I was in for a big treat--and boy was I right.

An early version of the game has already been released; however, I've had an amazing opportunity: an exclusive look at the new version of Sixty five Million and One BC, which is a huge improvement over the previous version. The incredibly long-titled game was created by **snailfox** of the GMC, who surely did an excellent job on the game, as you are about to discover.

I did not play the previous version of the game –apologies to **snailfox** – so I might be describing a feature with great excitement without realizing that it was available in an older version. I'd like to point out that even though other previews of the Game exist on competing magazines, this remains an **exclusive preview** of the full version of the game, not available elsewhere.

First Moments

When I opened the game, I was greeted with the most wonderful music and scenery. I've attached these four screenshots to give you an idea of how the introductory screen looks like right now. To be honest, screenshots don't do justice to the real beauty of the scene, as the sun is constantly moving, and smooth subtle animations are always occurring.

What improves the experience is the great music that is played during the introduction. It implies power and greatness, like the music in movies about ancient empires (such as 'Alexander the Great').

Introduction

Again, screenshots cannot do justice to this game, specifically not to the game's introduction. The introduction film is really cool; the interaction of the dinosaurs with each other is spectacular, and the view moves smoothly to cover different areas and even shakes the moment the meteor hits the ground. The effect is definitely commendable.

The background music becomes lighter and gentler to draw the readers concentration to the interaction between the dinosaurs and the text appearing on the screen.

The volume goes up and down according to the event and its significance, and the tempo changes based on the exact moments of the introduction. It is incredibly clear

65 Million And 1 BC Cont.

that the music was built carefully and tailored just for the introduction film. Oooh, awesome, awesome!

I've also appreciated the fact that the intro has some cool jokes--nothing too much, but the humor was appreciated. However, I was disappointed that the intro is way too long. If only the movie could be broken into smaller pieces separated by some cool but minor gameplay.

User Interface

Excellent! The screen is organized in a very logical way; the controls are always on the screen, yet they don't fill it. How so, you ask? well, at any one moment, only the relevant controls appear on the screen.

Other than the arrow keys, you only need two more buttons: the Space and Ctrl keys, though the Enter (Return) key is used occasionally. When you are walking, the Ctrl key changes from 'tailwhip' to 'run'; when you're jumping close to a vertical surface, Ctrl again changes to 'climb'; and when you're climbing, Ctrl becomes 'let go' while Space becomes 'jump'.

The health bar is jagged, which makes it look kinda "stoney" and I suppose this is done on purpose to give some character to the game.

The health bar is jagged, making it look kind of stoney, which seems to be done on purpose to give some character to the game. The UI screenshot above shows all these UI features. On the top are the two context-sensitive button instructions, on the bottom center is the health bar, and to the bottom right is another context-sensitive sign which only appears when it's relevant.

Unlike the Ctrl and Space indicators whom are always present (but with different labels according to the situation), the Talk indicator for the Enter key only appears when relevant. Otherwise, it disappears.

When I say "appear", "disappear", or "changes", I'm actually not describing the effect very well. Everything just fades in and out so smoothly; it's unbelievable!

Interaction

The way the player interacts with other objects is really nice. I've described this in the section above, but I'll emphasize it a bit more here. When getting near a vertical surface, the controls change. When getting close to a dinosaur, it might act differently, controls might appear, and a button which would normally do one thing would change its behaviour accordingly. Pretty cool.

Talking

When pressing Enter near a talking dinosaur, the 'talk mode' begins. When talking is taking place, there are two options for the user: continue or skip.

65 Million And 1 BC Cont.

If a user wants to continue, then Enter is the button to press. Otherwise, Escape is the button of choice.

Again, cool graphics are shown to reflect the changes that have occurred to the controls. On the bottom left is Skip and on the bottom right is Continue, corresponding to the Escape and Enter keys respectively.

I've also attached a screenshot that shows the 'talk mode'. As you can see, the game gets rid of the unnecessary UI and enters a quasi-widescreen mode. Again, this transition is done smoothly. It's not like the entire screen fades out; what actually happens is the UI components fade out, and the black stripes you see on the top and bottom of the screen gradually start filling the area.

Graphics

I've said quite a few times that screenshots don't do justice to the graphics. The reason for that is that the graphics themselves are good, but what makes them amazing is the animation.

Everything animates so smoothly. Walking is amazing; the legs and feet of the dinosaurs appear to move quite realistically relative to the speed of motion. Jumping has great animations, even when climbing and jumping off vertical surfaces.

The amazing thing is that it all happens very smoothly. It's almost as if there's a special transition set of sprites and frames for each sequence of moves.

Particle effects are used wisely for fire and smoke. Nothing is done over-the-top so that the result is light and delicate. Excellent job, I admit.

Another effect I noticed is the splashing of water – really cool.

Hitting, biting, and other moves are also pretty cool. The damage is a shock-like ring effect or else a cool, delicate blood effect that appears just briefly on the screen.

The Sound of Music!

The music sure does sound excellent all around the game. It has an ancient feel, and you can hear occasionally in the background the beat of those ancient instruments in a quick-

65 Million And 1 BC Cont.

tempoed "boom, tack-tack boom, tack". It makes the game more exciting and makes the graphics look better, also. When the volume of the music is decreased, birds can be heard in the background.

Though I'm not sure this is the case, it appears that the state of the music changes with the state of the game. When there's not much to do, or when you're idle, the music does seem to be less exciting and calmer than when you are in a battle situation.

Difficulty Curve

The difficulty curve is something I didn't like quite that much. The reason for that is the game becomes difficult way too early. This was actually a reason why I wasn't able to finish the game.

Right after you are asked to go into the forest and get the medicine, you'll have to fight some enemy dinosaurs, which doesn't seem to be hard. Right after that you are asked to climb past poisonous worms by jumping back and forth between two or more vertical platforms. That was really hard and cost me my life. At this point, I gave up.

Storyline

The storyline is quite simple and nice. It actually talks about, as the game's name implies, what supposedly happened Sixty Five Million years ago: the extinction of dinosaurs. A small meteor hits the ground, and the dinosaurs know that a bigger one will soon follow. The entire mission of the game is to stop that meteor from hitting and save the dinosaurs from extinction.

It's a sufficient storyline, and it works well to keep the player engaged. Snailfox did a good job there, too.

Overall

What a spectacular game. While it is still, even at this extremely mature stage, a work-in-progress, I'm amazed by how this game will turn out to be. It certainly isn't the typical Game Maker game we see anywhere in the GMC. It's special and unique.

It is really refreshing to see someone pay attention to all elements of a good game and to making sure the game is--in every standard--marvellous.

I've given only a couple of suggestions about the game, and these are to enhance the difficulty curve as well as make the intro shorter (or break it into smaller pieces). I hope my suggestions are heard, but even if they aren't--awesome job. See the GMC demo topic.

Up next: Exclusive Interview with snailfox.

Game Screenshots

65 Million And 1 BC Cont.

Interview

What do you find easier, building the engine then moving to eye candy or doing the whole thing in one go?

I'm sure that it's better to build the engine before you add eye candy, and it's probably easier in the long run, even if it's not that fun. The engine is the foundation of the game - the basic structures onto which you build the finer elements - and it definitely has to be well-established before you start tweaking and shaping the visible parts. Graphics and minor game features are usually small-scale elements that rely on the more basic working of the engine, so if you go adding those while it's still in development, they'll probably stop working as you change the engine code that they rely on.

But I wouldn't say that you should work exclusively on the game engine until it's complete - for most games that would be a long, dry process and you'd probably give up or go mad. As long as you aren't messing with the basic workings of your game, I think it's a lot better to work on other aspects in parallel with the engine development; elements like graphics and storylines and effects. I think you can be a lot more creative if you have a relatively flexible approach to building the game. What I've been doing is to focus mainly on building the engine, but take breaks from it now and then - to draw out a new sprite sheet or design a shaky camera effect or a moving sun - something minor that won't interfere with the game engine when I implement it. It's refreshing to have the change of focus, and it makes you think of ideas you might not otherwise have had.

The animations for 65 Million and One are amazing, any tips for newcomers on how to improve with their animation skills?

Thank you, and yes: use vector graphics if you don't already - they make spriting much easier because you don't have to redraw every single sub-image. While a lot of GM games purposefully strive for a classic look by having SNES Sonic-style graphics (the only example I can think of off-hand), they're pixel-based (raster graphics) and hard to manipulate. If you want to change part of a raster sprite, you have to redraw it, which is really irritating if you aren't great at drawing.

I'm not great at drawing, so I use vector graphics, which allow you to construct 2D objects out of individual shapes, then move, rotate, resize and deform them without any loss of quality. The shapes in MS Word and PowerPoint are vector-based, while the stuff you draw in MS Paint is raster.

As for the actual animating, that just takes trial, error and perseverance, I think. The GM sprite editor has the essential feature to help you animate - it shows you a preview animation of the sub images - so as long as you can paste your graphics into there, animating should just be a matter of reviewing the preview animation and modifying the sub images as necessary. Personally, I use Corel Draw to create the vector graphic sprites and Photo-Paint to clean up the individual frames, and then paste them

Super Sound System

Super Sound System is one of those classic Game Maker DLLs. It has been around ever since 2005, and is incredibly beneficial to games.

The DLL allows you to use the **OGG Sound Format**, which is an incredible open format. OGG has great impression, and high quality, and it can be used with Game Maker using the Super Sound System.

The DLL has some nice features, too. You can change the volume, panning, play multiple sounds, etc. Version 3.2, released in September 2007, solves multiple bugs previously found in the system.

Get it now!

gmc.yoyogames.com/?showtopic=120034

65 Million And 1 BC Cont.

from there into the GM sprite editor.

Have you ever tried working in a team? Which do you think is easier?

I really don't know which is easier, since I've never tried working in a team; I imagine there are both advantages and disadvantages of doing so.

The biggest advantage of working in a team, I should think, is that you have access to expertise different from your own - so you could potentially get better graphics and sound and innovation and coding than you would be able to produce yourself - and there are always people who can do things better than you can, so you might as well take advantage of that. I've been constantly scouring the GMC forums to find better ways of coding certain features, or to ask just how the heck you do vertical platforms; using the expertise of other people only makes for a better finished product. There would also be a smaller workload for you, and a more accurate, collective opinion on every aspect.

The drawback of working in a team, of course, is that other people are involved. While this can be good for all the reasons I mentioned above, it can also mean you end up relying on someone for something they can't or won't deliver, and creative disagreements can hamper the whole project. You'll have to share credit/delegate blame when the game turns out to be Immortal Defence/LOZ: Triforce of Trifle, or whatever; and finally, if you're like me, you'll get the (probably misplaced) suspicion that you could have done everyone else's work better than they did.

What do you generally use to achieve smooth and semi-transparent sprites?

Well, some of the title graphics and HUD elements have alpha channels, but the vast majority of the sprites don't. The problem with having no alpha channel is that Game Maker will only use one colour as the transparent colour of a sprite, which means you either have to draw the sprite onto a background of the same colour that will be used in the game room, or keep a jagged, aliased sprite edge that can be used against any colour - but that's pretty ugly.

I *very* ***slightly*** blended all my sprites with a medium-grey background, the average colour of all my different room backgrounds. This has worked out quite well, the grey around the edge of the sprites is pretty much undetectable against all of the medium-toned room colours in my game, but it takes the edge off the black sprite outlines and makes them look smoother. If I put them against a black or white background, the sprites would show up the grey edges and look ghastly.

Note from Eyas: then, may I recommend reading this issue's article about smooth edges?

How come the music so awesome?!

Thank you again! It's a matter of taste though - I'm not expecting everyone to like the music, which is why I added the function to switch it off. It's all my own (music keyboard, Adobe Audition and lots of audio cables) and I'm quite proud of it. The tunes are original, but mostly based on the soundtracks of various films and games, which are variously referenced throughout the game. There's some Zelda and Lord of the Rings, and of course, Jurassic Park - none of the music is ripped or directly copied though, even though I know the game would sound better for it. I think original music is a nice touch in a GM game, if possible.

I can't help but ask... where did snailfox come from?!

If I told you, I'm afraid I'd have to kill you.

Conclusion

Sixty Five Million and One BC is an excellent Game indeed. I was glad to review it, and also thrilled to interview **snailfox**, its author... whose name is (right now, for me) as mysterious as it ever could be!

I truly hope people would make use of the tips in the interview, and also try to create games as complete and homogeneous as this is. It is really educating (game development-wise) to hear from a creator of an excellent game.

Eyas Sharaiha■

Artificial Intelligence

Overview

Artificial Intelligence, AI. What is it? Some people say that it is the ability of a machine to act on its own. It is most often thought of as computers acting like humans. The reality is that AI is output based on circumstances, plain and simple.

It is called Artificial because the computer does not actually have any intelligence or the capability to comprehend what is being presented. Remember, computers and machines work with 1's and 0's, on or off. That's all they know, that's all they can understand. In the case of "Learning" AI, nothing is actually learned, all that happens is that the AI creates more circumstances and outputs based on what is needed to be "learned." For example if you wanted to make AI that "learned" to react based on what key the user presses most you could use something like the following:

```
if keyboard_check_pressed(vk_shift)==true
 {
 pressed+=1
 }
```

```
if pressed>10
 {
 //actions
 }
```

This will test if the user has pressed the shift key 11 or more times. If he or she has, it will perform the indicated actions. I often receive the rebuttal that this is not learning because it is pre-defined. That is because it can't actually learn. That is like asking it to write its own coding. A program can only do what you tell it to, nothing more.

Another common misconception of AI is that it must express human like quality and be very complex. This is not true. A single if, then statement could be considered AI. For example:

```
if x<10
 {
 x+=1
 }
```

This will add 1 to x as long as x is less than 10. The circumstance is x is less than 10 and the output is to add 1 to x.

In accordance to games, some simple AI could be a boss type monster that jumps to the left, shoots at the player, jumps to the right, shoots at the player, and jumps back to the left and repeats the process. The creator could improve it by making the attacks it does based on the amount of health or adding other such outputs based on circumstances.

When making human-like AI, such as

Artificial Intelligence Cont.

making a car drive on a road, I like to take it step-by-step and think of exactly what goes through my mind when I perform the action that the AI is supposed to do. The tricky part about that is that the brain does many things automatically. Think about the seemingly simple act of walking. When you get up to go get a snack you don't think about walking, you just do it. If you delve even deeper into the matter, when you walk you are actually fulfilling a long, complicated sequence of muscle extension and retraction. I won't go any deeper but the point is that what makes creating human AI so difficult is the vast complexity of every action that a human takes. The more of these minute details you include the more realistic it will be.

The first thing I did when I started making my Civilian Car AI example was think to myself, when I drive how do I keep on the road? To keep on the road you turn when the road turns. That raises another question, how do you know when the road turns? As humans, we can see what is in front of us and comprehend what we see and use that to base our actions on. The computer, however, cannot see the turn the road. In fact, all the road is a bunch of pixels arranged to look like a road. The method I used to test if the road is turning was to check points to the front left and front right of the car. If there is no road on the left side, the car should turn right, if there is no road on the right side, the car should turn left. Making the cars stop at traffic lights was not as complicated as you would think. To do that I had the cars check a point in front for the traffic light object. If it finds one it takes whatever state it is and acts accordingly. Here is how I checked for the light and found the sate of said light:

```
light=collision_line(x,y,x+32,y)
if light!=noone
 {
 state=light.state
 if state=3
  {
  speed=0
  }
 }
```

The above coding will check if there is a light in front of the car. If there is it takes the state of the light and stores it as a variable in the car object to test. If the state is red (3 representing red) he car stops.

The pattern explained above is the basics for developing AI. First take a desired action, interpret what steps you, as a person, take to accomplish that task, and find a way for the program to test for and act upon the same steps you take. As stated earlier, artificial intelligence does not have to be overly complicated or even express human characteristics. You must also keep in mind that the computer knows absolutely nothing except 1's and 0's. When creating your own AI convert the steps you take into code for the computer to test and run.

Michael Moore ■

Quick Reviews

Gamepad 2.2

Gamepad 2.2 is a powerful text editor script created by IsmAvatar. The scripts are available with an excellent GUI and example.

While ever since Gamepad 1, the editor featured multiple lines, pure GML, a text-cursor that uses the mouse and arrow keys to move, and other common editing features, some new exciting features were added Gamepad 2.

These include text selection, both with the mouse and keyboard, as well as replacement, cutting, copying, and pasting of selected text. An unlimited amount of lines, wheel scrolling, and line highlighting.

The editor comes with an excellent scrollbar interface which allows you to easily navigate through huge amounts of text.

One of the major features of this editor script in my opinion is the high speed (compared to other GML scripts) and the ability to easily process huge amounts of data.

Get it now!

gmc.yoyogames.com/?showtopic=328545

Faction Wars

Welcome back to the Faction Wars Development Diary.

In last week's article we looked at planning and structuring your game's organization, design process and infrastructure. In this, our second issue, we look at ways to better organize, design and manage your game's resources and memory usage. Faction Wars is a massive undertaking, and as such, memory management is something that we've had to plan for right from our first steps.

So, what is memory management, and how does it apply to my game?

Memory is used to store, process and manage all of your game's resources. We'll be looking specifically at textures and sounds, as they can have the biggest impact on the way that your game functions. Managing your memory resources is as big a design element as your level structure, storyline and gameplay.

Let's take a look at what happens when you don't manage your memory.

Imagine that you have a game with 20 levels.

Let's presume that an average player will tend to launch your game, play for an hour, and in that time complete two levels. Our theoretical game has about 10 hours of gameplay.

Now consider that each level is different. They have different tiles, room settings and sizes, enemies, textures, backgrounds and sound effects. Let's presume that each level has 20 different resources.

By default, Game Maker will load all of those 400 of our resources into memory when the game starts - even though the average player will only need 40 of those resources available for an average game session.

Obviously our theoretical game is wasting lots of memory. But how, why, and what does this mean? How much of a difference will this make?

Memory is used by every application that you might run on your system. As long as you've enough memory available to run your application, then there's no problem. But as soon as you start to run low, your system will look for ways to actively 'swap' memory around in order to make it available where it's needed, this is done by the Virtual Memory Manager, or VMM. The VMM swaps memory using the hard drive which is relatively slow and thus this can dramatically slow down your game. As such, it's good practice

Faction Wars Cont.

to make your games run as efficiently as possible.

So how do we manage memory in Game Maker?

As we've discussed, Game Maker loads all of your resources immediately by default. So, rather than storing all of your resources, sounds and textures within your executable (where Game Maker then has to decrypt and process all of them, even though only a fraction are likely to be used within that gaming session), you simply store the resources externally in the folder or sub-folder that comes with the game.

Now we only need load resources when they are required, and then free them from memory later when you they're not needed.

In the game maker manual there is a chapter named' Changing Resources' that focuses on the scripts and techniques used to load and unload resources. It's well worth a read.

Loading external resources

Let's presume that our theoretical game has 10 resources. Two of these are used in the first level, and two are used in the second level.

The script below is called at the start of the game, and defines the names and location of our resources (overleaf):

```
//ResourcePath[level,resource_number]
ResourcePath[0,0] = "/imga.gif"
ResourcePath[0,1] = "/imgb.gif"
ResourcePath[1,0] = "/imgc.gif"
ResourcePath[1,1] = "/imgd.gif"
ResourceAmount[0] = 2
ResourceAmount[1] = 2
```

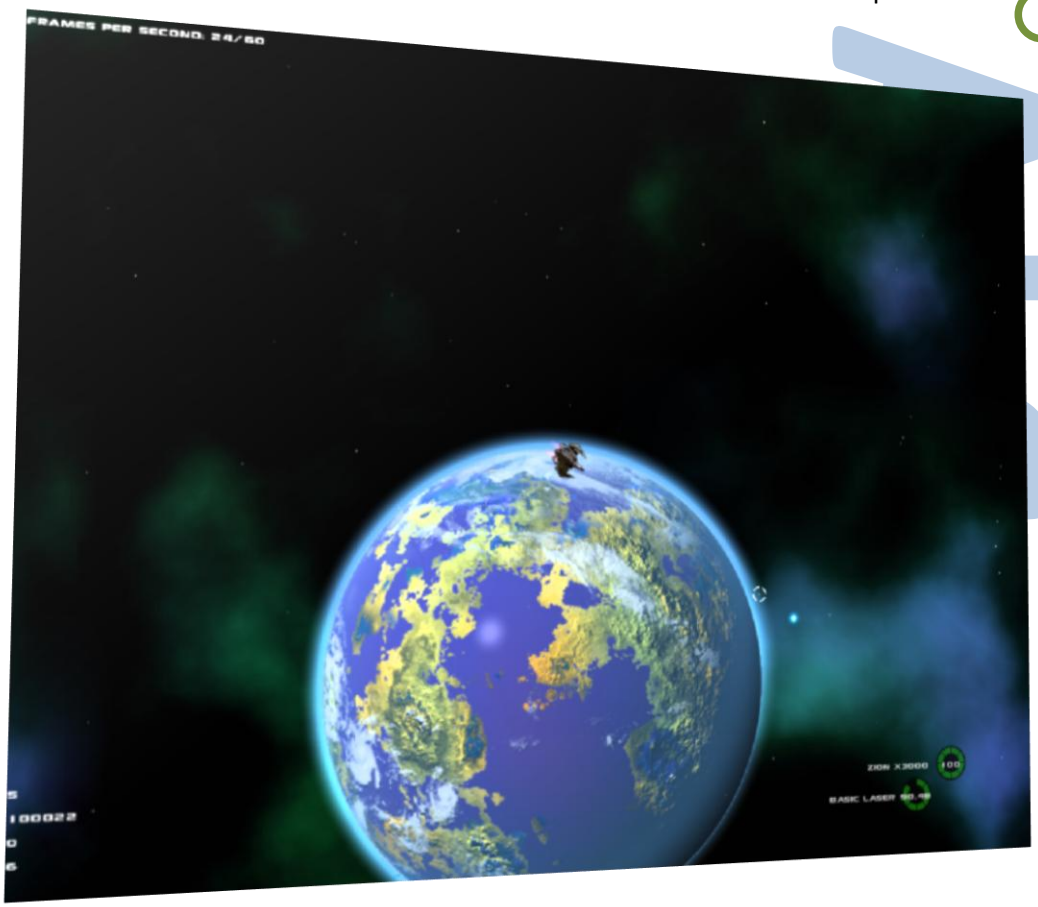
When specifying resource paths, the first index number states the level in which the resource is used, and the second states the resource number. We also create a variable to specify the number of resources in each room.

We now create a controller object for each room. Each room is assigned a number, which is stored in a variable named '`room_num`'. In our theoretical game we have two rooms, each with two objects.

Using the script below, the controller object will read the `ResourcePath` array and load only the resources assigned to its own room:

```
for(i = 0;
      i<ResourceAmount[room_num]-1;
      i+=1)
{
Resource[i] = sprite_add(
ResourcePath[room_num,i], 1, 1, 1, 1,
1, 0, 0 );
}
```

The resources within that room are now loaded and available to use. For example, in room one your 2 resources are '`resource[0]`' and '`resource[1]`', which can be removed from memory at the end of the level by using `sprite_delete()`, you can automate this with the same for loop I've used to



Faction Wars Cont.

load the sprites.

Of course these are only the very basic first steps of memory management, and there are drawbacks with a system this simple. You'll probably want to rewrite or modify the code for your own game to standardize names and variables – as well as considering using Constants to store names against your '`Resource[x]`' variables so as to avoid having to check which numbers correspond to which resources.

Note that it's a bad idea to add or delete resources whilst the player is actively engaged in play. Always do it between levels, at specified moments, so that you don't inadvertently slow down the game and thus render the whole process ineffective.

Tip: With a structured, scripted system to load resources, it's easy to create a custom loading bar or screen by tracking and displaying how many of your resources have loaded as a percentage.

Sprite and Texture sizes

When you save sprites in Game Maker, regardless of their size, they're stored in memory as the equivalent size of the 'next size up' in terms of 2^x.

e.g., 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, etc.

For example, an image sized as 500 x 200px is stored and processed as one of 512 x 256, and an image sized as 1100 x 1100px is stored as 2048 x 2048 - almost double the amount of memory is used!

When implementing your resources, it's very important to bear in mind that the size will always 'round up'. If your image is 66 x 66, you're using 128 x 128. Can you sacrifice a few pixels to bring it down to 64 x 64?

Tip: Always crop your images. Cropping is the act of removing transparent borders around the actual image, as these pixels are transparent they do not serve any function and thus only consume memory. Cropping can be done in many graphics editing software, including game maker's sprite editor.

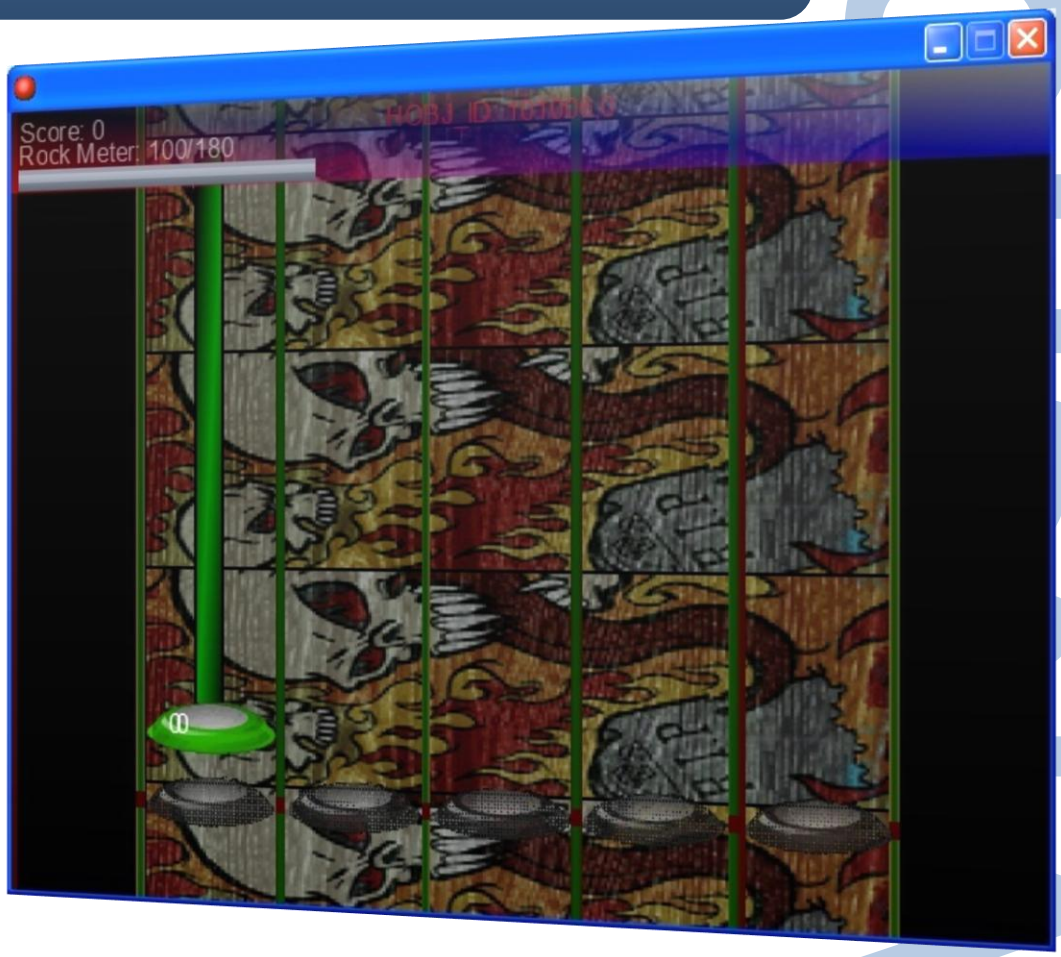
Parting Shot

It's been a busy month for all our team members, but we've still made significant progress. We're busy fighting away at pulling together final concepts and design ideas, and sourcing some top notch graphical and visual development resources. The framework is being laid for the future of Faction Wars, and we're eager to show you some of what the team is producing. As such, next month you can look forward to ongoing premier access to all of our tech demos, displays and events… We look forward to seeing you there!

Thanks for reading!

Jono Alderson & Tarik Abbara,
The Faction Wars Team■

God of Rock



Well, I am having more problems with the long notes since I am going to add chords. Now they cause errors and GM is logging errors every step so it slows the game down. So to remedy this I am going to use two-dimensional arrays.

Now that I have successfully got the note system working with the arrays, a new problem with timing arose. The problem was that the objects where setting their array information at around the same amount of time, and obviously that would cause problems. So I changed the note system to create all objects at the same time.

The only problem with this is that, this may put a lot of strain on the user's computer and potentially lead to freezing or lag because the room's height is 80,000. So I try to keep as fewer objects at a time in the room but, GM does a pretty good job of handling mass amounts of objects so my game is pretty much lag-free at least on my computer.

The next problem that occurred was when I added chords into the game. This issue involved notes snapping to the wrong x coordinate. So to fix this I had to completely rewrite the system this time, using Data Structures. The new Data Structure System had the same problem so I had to try it with writing a file in a temp directory. That would slow the game's FPS down massively. So I tried deleting the file every 15 steps. For some reason, that worked.

Then, only one issue remained and that was the issue of Star Power draining too slowly. To fix this I thought I'd subtract is by, about 0.02 or something like that, every step but in the end I decided just to subtract it every 10 steps, by 1. Now I just had to pre-set the update program that updates all my games to download God of Rock when its release date came.

Next thing you know the Update Program would continuously crash. I quickly discovered that I was actually writing to the EXE. I don't know how that code got in there. I must have been half-way asleep or something.

So now God of Rock is pretty much bug free so now all that's left is to actually release the new version. Now if I could just find a fast file host...

Mathew Malone■

Special

THE GAME MAKER RACE

Fire and Ice, fire and Ice!

The Race has started since the fifteenth of October, but you still have time to make a game and enter – the deadline of initial submissions is November 15, 2007, and you can submit updates to your game until a week after that date.

What *is* the Game Maker Race? It is the *ultimate* Game Maker challenge, created as a joint project between: our Magazine – **MarkUp Game Development Magazine**, and **Game Maker Technology Magazine (GM Tech)**.

The contest is ***creative***, ***fun***, and for you ***talented*** Game Developers, it'll give you a big chance of ***earning*** some awesome ***prizes***. Do not miss the ***opportunity!***

www.thegmrace.com

Prizes

All Top 20 Games!

For all the top 20 games submitted, the candidate will receive a free copy of AKUCHIZOKU, by cactus!

All Five Finalists

All five finalists will get one USB Flash Drive each, with a capacity of 1 GB.

The Winner

In addition to the 1 GB USB Flash drive, the first place winner will also receive a 2-year free domain, in addition to free cPanel hosting for that domain. Not only that, but the winner receives two PC games, out of this selection:

- FIFA '08
- Command & Conquer 3: Tiberium Wars
- Rollercoaster Tycoon 3 Deluxe
- Bioshock
- Company of Heroes
- The Sims 2: Bon Voyage
- Worldwide Soccer Manager 2008
- CSI: Crime Scene Investigation Hard Evidence
- Sega Rally
- Tiger Woods PGA Tour 08

How to Enter?

You can **enter the competition**, by navigating to the entrance page and following the instructions on-screen. The theme of the competition is the following:

- Make a game related to "Fire and Ice"
- The game must be 2-dimensional

How to Start?

Get Game Maker from the **YoYo Games** site, and start programming! Remember the criteria of the game above.

The Game Submission **must be** uploaded to the YoYo Games forum. If for some reason that is difficult or not possible, support is available through the **forums**.

Smooth Edges

Smooth Edges

"Smooth edges", in games, is almost certainly a graphic requirement for high quality games.

"Smooth Edges" as a concept is certainly different from the Game Maker feature with the same. While the latter could be used to achieve the former, other techniques also exist that make it easy to do so in Game Maker.

Figure 1: A circle with Smooth Edges

Figure 2: the same circle with no Smooth Edges

Games with no smooth edges (as seen in Figure 2, above) appear of much less quality than a game with full smooth edges (Figure 1).

What are Smooth Edges?

"Smooth Edges" as a concept, doesn't necessarily mean realistic graphics. You could always have bulky, cartoonish sprites and graphics – but 'smooth edges' is still needed. Enabling smooth edges in a game means any foreground sprite could well-blend with whatever is behind it (whether it is another sprite or a background). The lack of smooth edges might make certain (or all) objects feel out of place.

Smooth Edges in Game Maker

Game Maker 6 and 7

A cool feature exists since Game Maker 6, and available in all future versions of Game Maker: `sprite_set_alpha_from_sprite`, which is an excellent function in my opinion.

The function has two arguments, the first is the id of the sprite to be edited, and the second is the id of the sprite that will be used as an 'index' in determining the alpha of the first sprite.

The first sprite could be any wanted shape, with certain transparent and opaque areas (the transparent option should be disabled, most of the times).

The second sprite, which is the index of transparency, will have a grayscale color. **Black** means the pixel will be fully transparent, and **white** means the pixel will be fully opaque. All gray colors in the middle produce semi-transparent pixels, according to their distance from 'black' or 'white'.

Ex. 1: How the function works

In the first example, we want to turn the sprite in Figure 3 to a circular object.

Figure 3: The Original question mark sprite

In order to be able to make this shape a circle, we could easily use the function to crop out the four edges and make the center circular.

How do we do that? We get a black background, and draw a white solid circle in the middle.

Figure 4: The Alpha Index for the sprite

The result, when editing the sprite `s_q` (Figure 3) with the alpha mask `s_qm` (Figure 4) using this code:

```
sprite_set_alpha_from_sprite(s_q,s_qm);
```

Is what appears in Figure 5. The sprite, `s_q`, now looks like this. The action of the script cannot be reversed in runtime.

Figure 5: The new resulting sprite

Obviously, this example is mainly concerned with how to use the script, rather than smooth edges. So, there is no virtual benefit of using this exact set of actions on a sprite, as Game Maker's single color transparency would work just fine.

Ex. 2: An actual use of the function

If the same question mark in Figure 3

Smooth Edges Cont.

was to be edited into an actual smooth circular object, then a better mask sprite (Figure 6) has to be used.

Figure 6: The enhanced mask (s_qm2)

If we used the function to have `s_qm2` as an index, rather than `s_qm`, we would have the result shown in Figure 7.

Figure 7: The resulting sprite

Compare Figure 7 to Figure 5 on the previous page. Both are of the same base sprite, but their pixels have different alpha indices.

Game Maker 7: PNG

PNG stands for Portable Network Graphics, and is an excellent graphics file format.

The reason why it is really popular, especially in the web, is because it is an efficient form of **lossless data compression**, but also supports an alpha channel.

Lossless data compression means that the data (file contents) are compressed, usually resulting in a considerably drop in file size, but without suffering any loss of quality. JPG is an example of lossy compression, where the size drop occurs on the expense of quality occurs.

PNG can support an alpha channel, which means that transparency would be immediately integrated into the image, meaning there is no need for an additional mask: it's all in there.

Game Maker 7 introduces support for alpha channel PNG files. Unfortunately, access to alpha PNG files is only available via code, and adding a PNG image as a regular sprite would not make it transparent.

Two functions for alpha channel PNG files exist in Game Maker sprite handling: `sprite_add_alpha`, and `sprite_replace_alpha`. The former is similar to the `sprite_add` function, which returns the id of a newly created sprite from a file, and the latter is similar to the `sprite_replace` function, which the sprite with the given id is replaced with a completely different sprite, selected from a file.

The `sprite_add_alpha` function has the following arguments:

- File name: the name of the PNG file to be replaced, either relative to the game's `working_directory`, or a full file path leading to the file.
- Number of sub images: this indicates the amount of sub images to be used. 1 would result in a single still sprite, while multiple numbers mean that "numbered" sprites need to exist.
- Precise collision checking: this is the same as the feature in the sprite properties window. Precise collision requires more process but is necessary for non-rectangular objects that need collision checking.
- Preload texture: this option is also available in the sprite properties window, and defines whether or not should the texture be loaded when the function is called (or alternatively, upon first use of the new sprite).
- X-origin, Y-origin

Basically, an alpha-transparent PNG is either saved somewhere, or included in the executable (via the inclusion mechanism) and extracted to a temporary folder. Then, the function is used to add (or replace) a certain sprite. The sprite drawn would automatically be transparent, with smooth alpha channels.

PNG files could be saved using Photoshop's Saving-for-Web option, or many other editors.

Conclusion

Adding smooth edges to the game would indeed give a much better feel to the game overall.

In future issues, other techniques of smooth edges will be discussed. These include using surfaces, scaling along with GM's built in "Smooth Edges" feature, and others.

Eyas Sharaiha■

Online Highscore Tables

Introduction

Highscore tables or leader boards have long been a common feature in many genres of computer game.

Game Maker's built in highscore table function has been used in many Game Maker games enabling people to compete against themselves and family members, but unlike online games until now very few games have offered online worldwide highscore tables.

Online highscore tables enable gamers from around the globe to compete against each other whilst playing a game on their own computer. The concept by which they work is very simple – after the game is over the gamers score is submitted over the internet and stored in an online database of collected scores which can then be viewed by players.

Until recently there was no easy way to do this with Game Maker, but in the past couple of months not one but two free services have come along which claim to enable easy creation of online highscore tables.

Using a very basic game I have made I will compare the features offered by Blijbol OnScore and Vitarsi Vex.

Vitarsi Vex

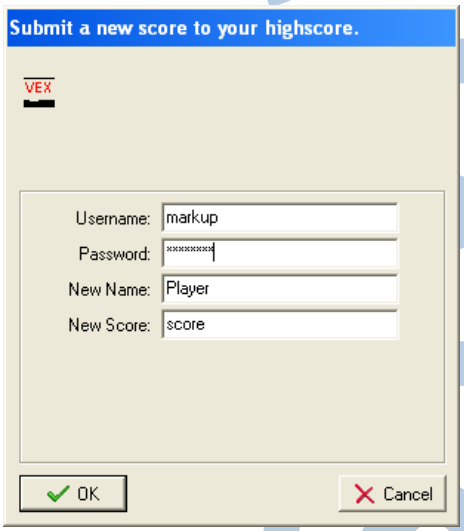
The Vitarsi Vex site looks like it has been put together in a slightly haphazard manner and some page elements could do with being realigned. Likewise the navigational links are not always sited in the most logical places; however this gives a false impression of the service.

After a quick registration form has been completed setup is extremely simple. The 'Game Settings' page asks for your game name and then enables you to choose from 4 pre-built themes for your highscore table.

Next you can preview you highscore table, but since no data has been entered yet this is a little pointless. Mysteriously the next option then enables you to embed your Vex highscore table onto your own webpage in an iframe – all this before you've actually seen how the service works. You then have to download a library file and copy it the /lib subdirectory located in the folder where Game Maker is installed. The download does include a very brief readme file to this effect, however if you have never used a none-default library you may not know what to do.

If you have Game Maker running restart it. Then create a new object which will be used as a button to submit the player's highscore online. Choose the left pressed event and then go to the newly added 'Vex' tab and drag the single action across. You then need to enter the username and password combination you used to register at the Vex website previously.

After entering your login credentials you then have to add details of the variables that store the player's name and current score in your game. The graphic below shows how the form should look if your player name is stored in a variable called 'Player' and you have used the inbuilt 'score' variable in Game Maker.



Clicking the button within the game then submits the player's data over the internet to your Vex account and places the score within your highscore table. The Vex highscore service allows you to create one high score table per account which can be used to display the top 10 scores.

Online Highscore Tables Cont.

Management options even enable you delete specific high scores from your table if you wish.

Links

GMC topic:
gmc.yoyogames.com/?showtopic=333478

Example highscore table:
vex.freehostia.com/Vex.php?user=markup

Example game:
www.gamemakerblog.com/gmtv/balloon-gmc.gmk

Highscores

Name	Score
Jason	72
bob	47
Jason	38
bob	22
bob	15
Philip	8
Philip	8
Phil	7
Phil	7
Phil	7

Powered by VEX

Blijbol OnScore

Blijbol.nl's OnScore download contains libraries for use with all versions of Game Maker from 5.3A and a HTML help file.

Unlike Vitarsi Vex, OnScore also includes encryption making your high score table safer from any potential cheaters. However another major difference between these two highscore services is that whereas Vitarsi Vex provides server space to store your high scores if you choose the Blijbol service you need to have access to your own web hosting with php and MySQL support.

The step by step instructions provided by the help file far exceed what is offered by Vitarsi Vex, however the service does appear to be targeted at more advanced users for the simple reason that web space is required. Getting the required files set up on your server is very simple thanks to the installation program provided which guides you through each action you need to take.

The admin panel looks a lot more compressive than the members area at Vex and supports an unlimited number of games and highscore tables.

I was a little disappointed to see that no demo tables had been set up, because of the quality of the documentation to date I was expecting to see this – a demo mini game linked up to a highscore table would be a good finish to the provided explanation.

After downloading a GML connection script from your server setup is very simple thanks to the included library which requires only one variable – your highscore table id.

When the user's score is submitted a webpage opens asking for the players details. All of the web pages included with the download are clean and lightweight and create the impression of a very professional service.

Links

GMC topic:
gmc.yoyogames.com/?showtopic=252864

Example highscore table:
gamemakerblog.com/Server/list.php?list=1

Example game:
www.gamemakerblog.com/gmtv/balloon-b.gmk

Conclusion

Both of these services were not without their problems, I did not complete the setup for either of them on my first attempt. Blijbol has the better documentation and admin area, however requires you to have your own web space. Vex on the other hand includes the hosting of your highscore table so is more suited for beginners.

Philip Gamble■

Functions

Introduction

Quite often in a game or program, the same piece of code is needed multiple times. A possibility would be to copy-paste all text, but that way, your code will soon become very hard to understand. The preferred solution here is to use functions. Functions allow you to group blocks of code and they offer an easy way to execute that code. Let's have a look at what a function is and what can be done with it.

Function: the general concept

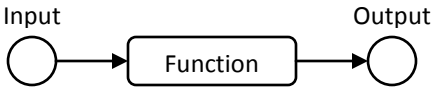
A function is a concept used in programming. All programming languages support functions in some way. A function consists of one or more statements that form the function code.

Simply said, a function is a piece of code that can be executed when needed. How can it be executed? Of course, you first need to write a function. Then, the function can be "called" by the program (or by another function). The syntax of a function call generally looks like this:

```
function_name(argument0, argument1, ..., argumentn);
```

- `function_name` is the name of the function
- the ()'s indicate that this is a function (it would be a variable, object id,... otherwise)
- `argument0` - `argumentn` are the arguments of the function. They are parameters that are passed to the function. These change the behavior and output of the function.

As you can see, a function can also receive arguments. These are parameters that are passed to the function and they usually influence the result of the function. Indeed, a function can also return some information. And this is where functions get interesting: functions can take a certain input, either a single variable or a reference to a data structure and they can return a value based on the input.



The advantage of functions becomes clear here: functions allow for modularity. A program can be divided into functions that are complete "modules" on their own. A function takes some arguments, a lot of code is executed within the function and the required results are returned. If you would associate the general concept of a function to an object in real life, then something real that resembles a function could be e.g. a coffee machine. The way you like your coffee (milk/no milk, sugar/no sugar, strong coffee, …) could be seen as the arguments. Pushing the start button on the coffee machine would equal the function call. And the returned value would be a cup of coffee. Note that the user of the machine doesn't need to know how the machine works internally. This makes it easy for the user to get him some coffee. In the same way, it is interesting for a programmer if he can use functions that perform certain tasks on their own. And it's even more interesting if those functions can be combined to make what one could call an "engine".

All very interesting, but we're not able to do anything with this. So far for the theoretical approach on functions. Time for practice with scripts.

Scripts: functions in Game Maker

Game Maker allows the creation and use of custom functions through "scripts". Scripts in GM are exactly the same as in other programming languages; it's just that they're called scripts instead. In this article, both terms will be used. However, the term "script" will only be used for the resource "script" in GM.

Now let's try to write a function that could be useful in more than one situation: a function to draw text somewhere in a room. There are a lot of ways to write a function. Some are very inefficient. We'll start with the

Functions Cont.

worst possible function and finish with the ideal, optimized function to perform this task.

There is already a function to draw text on screen, `draw_text()`, but we'd like to have a function that allows the user to customize a bit more, such as the text's colour and alpha. The code could look like this:

```
draw_set_alpha(0.5);
draw_set_color(c_black);
draw_text(5,5,"Hello!");
```

It should be clear that this function is worthless. Of course, you don't need to copy-paste this code anymore, but on the other hand: what's the use of a function that can't display the text you like.

The function needs to be able to accept parameters. Let's have a look at a possible way to do it:

```
draw_set_alpha(global.alpha);
draw_set_color(global.color);
draw_text(global.x, global.y,
global.text);
```

Okay, nice. Now the function can accept different parameters. Here's the corresponding piece of code to call the function (assume that the function/script is called `draw_text_custom`):

```
global.alpha=0.5;
global.color=c_black;
global.x=5;
global.y=5;
global.text="Hello!";
draw_text_custom();
```

That looks nice. And the function doesn't even need arguments!

But have a look at the 5 lines of code above the function call. This function is slightly better than the previous one, but there's one thing that should be done differently here: the parameters should be passed to the function as arguments, meaning that they should go between the brackets in the function call. And another note: never use global variables to pass parameters to a function (and don't use global variables if you don't need to).

Now let's replace the global variables with arguments. First a small explanation of "arguments" in GM functions. A function can take up to 16 arguments in GM. You cannot define the names of the arguments, they are named like this:

```
argumentn
```

where n is the integer index of the argument (0…15).

Or you can use the array notation:

```
argument[n]
```

Both expressions give the same result.

Something that is interesting to know is that each of the 16 possible arguments is initialized with 0. Now assume that you have a function that can take everything as an argument except 0. Then you could check the number of arguments passed to the function like this:

```
var no_args; no_args=0;
if is_string(argument[no_args])
argument[no_args]=ord(
      argument[no_args]);
while(argument[no_args]!=0)
{
      no_args+=1;
if is_string(argument[no_args])
 argument[no_args]=ord(
      argument[no_args]);
}
```

After the loop, the variable `no_args` contains the number of arguments passed to the function (assuming no argument is equal to zero).

Now that we know how arguments can be used in a script, it's time to rewrite the

Functions Cont.

function. It might also not be a bad idea to add some comments to make the use of the script more clear. I personally prefer a multiline comment at the start of the script that explains some things:

- What the function is used for.
- An explanation of each argument and the type of the argument with some extra information on the type, such as whether it is an integer, the range of the variable, a list id, …
- What the function returns (also including the data type).
- Possible remarks.

The script can now be rewritten:

```
/*this function draws some text on the screen with a color and alpha
arguments:
0. x position (real)
1. y position (real)
2. text (string)
3. color (real)
4. alpha (real, 0...1)
returns: nothing
    (remarks)
*/
draw_set_alpha(argument4);
draw_set_color(argument3);
draw_text(argument0, argument1, argument2);
```

This function is much more efficient than the previous ones. To draw the text string "Hello!" at (5;5) in a black color with alpha 0.5, you can use:

```
draw_text_custom(5, 5, "Hello!", c_black, 0.5);
```

Short and simple.

Now there's only one thing left to do. Clean up the mess, that is. By using the functions `draw_set_alpha` and `draw_set_color` the draw settings have been changed. Of course, that's not good. After the function has been called, nothing in the main program should be changed unless that was the aim of the function. In this case, it could definitely produce unwanted results. Other text might suddenly be drawn in a black color, while it should actually be drawn in white.

The solution is simple: before executing the code, we make a "backup" copy of the variables or settings that get changed and after the code, we restore the settings. If we add that to the script, we get the final code for the script:

```
/*this function draws some text on the screen with a color and alpha
    arguments:
0. x position (real)
1. y position (real)
2. text (string)
3. color (real)
4. alpha (real, 0...1)
    returns: nothing
    (remarks)
*/
var color, alpha;          //backup of variables
color=draw_get_color();
alpha=draw_get_alpha();

draw_set_alpha(argument4);
draw_set_color(argument3);
draw_text(argument0,argument1,argument2);

draw_set_color(color);
draw_set_alpha(alpha);
```

Et voilà, that's our final function. It won't get any more perfect than this. Note that the `var` keyword is used here to declare the temporary variables as local. You should always do this when variables are no longer needed after the execution of the function. Otherwise, the variables become local to the object that called the function and naming conflicts might occur, resulting in weird things. An example might make this clearer:

scr_function()

```
for(i=0;i<100;i+=1)
{
      //do something
}
```

Create event of an object

```
for(i=0;i<50;i+=1)  //i is declared and initialized as a local variable
{
      scr_function();     //i is reset to 0 and reaches 100
}                         //the loop ends here because i<50 returns false
```

The problem is this: in the first for loop (in the step event of the object) `i` is declared and initialized. In `scr_function`, that same variable `i` is reset to 0 and reaches 100 after the loop in the function. The result is that, after one execution of the function, `i<50` will return false because `i` is 100.

A solution would be to change the name of the counter variable `i` in each situation, but it is much easier and much more efficient to declare `i` local to the

Functions Cont.

script by using `var`:

scr_function()

```
var i;
      //declaration of i, which only exists inside scr_function
for(i=0;i<100;i+=1)
{
      //do something
}
//at the end of this function, i is destroyed
```

Create event of an object

```
for(i=0;i<50;i+=1)       //i is declared and initialized as a local variable
{
      scr_function();    //i is used and destroyed after the function call
}                        //the loop continues until i reaches 50
```

The variable `i` is now only known inside `scr_function`. That's why there will be no problem with `i` here. `i` and `i` are 2 different variables, but in the code, they have the same name.

Arguments that are passed to a function or script are also variables that are local to the function.

A last thing that could be of use is the functions `is_real` and `is_string`. With these functions, it is possible to determine the data type of a variable. These functions can also be used to get the data type of an argument passed to a function. And this can be used for some interesting things. Let's say we have a function to write text to a binary file. It would be interesting if we had a function that could do the following:

```
file_bin_write_string("C:\file.tmp", "Hello");
```

➔ write text to a binary file that is not yet open. This means that the file should be opened binary, then the string "Hello" is written to it.

```
file_bin_write_string(file, "Hello");
```

➔ write text to a binary file that is already opened and that has its id put in the variable file. The text now appears at the current position in the open file.

As you can see, the first argument, `argument0`, is of a different type in both cases (real and string). By using `is_real` and `is_string`, we can make sure that the function can work with both types of data.

The complete code of the function is shown in listing 1.

You should remember that, when you're using GM6 that the `file_bin_open` function doesn't create a new file when it doesn't exist yet. A simple way on how to solve this can be found in the article on binary files in MarkUp's 3rd issue.

The return statement makes it possible to immediately assign the file id to a variable that can then be used in a next function call as the first argument to the same function, like this (you can try this out in e.g. the create event):

```
test=file_bin_write_string("C:\test.tmp","Hello! ");
file_bin_write_string(test,"And some other text.");
file_bin_close(test);
```

```
/*this function writes a string to a binary file that is either open or not
    arguments: 0. file id of an open binary file (real, make sure the mode is write!)
               OR
                  string containing the link to the binary file that needs to be opened (or created)
               1. string to write to the file
    returns: the file id of the function
    (remarks)
*/
//declare temporary variable to store file id
var file_id;
if (is_string(argument0))
{
    file_id=file_bin_open(argument0,1);
}
else
{
    file_id=argument0;
}
//write the string to the file
var i;
for(i=1;i<string_length(argument1)+1;i+=1)
{
    file_bin_write_byte(file_id,ord(string_char_at(argument1,i)));
}
return file_id;
```

Listing 1

Functions Cont.

And that's it. What we've just done is an implementation of what is called function overloading. This allows functions with the same name to work with different types of arguments. Since GM only supports two types of variables (real and string), it is rather limited, but as you've seen in the example, it still allows for some nice possibilities.

External functions and extension packages

Since version 7, Game Maker has a new feature that makes it very easy to add new functionality based on scripts, action libraries and dll functions. This feature is called extensions. An extension package can contain new functions or action libraries.

This is how a script looks:

```
scr_function(arguments);
//script as a resource (purple)
```

But when the same script is now in the form of an extension package, it'll look like this:

```
scr_function(arguments);
//function from extension package
(blue)
```

Of course, the comparison between the colors is stupid. The real difference is that functions from extension packages are nicely integrated into the program. The different color also shows that these functions are treated as real built-in functions. This is the case for gml and dll functions. Also, when you use function and variable help, then this will be the result (see screenshot below)

All functions from extension packages will be shown with some information, like the built-in functions (assuming that the creator of the extension package has taken the time to add this information).

Also, a help file can be included with each extension package. This makes it easy to find help on the new functions.

Extension packages are definitely not a bad addition to Game Maker and they offer an easy way to add functions and action libraries to games and programs.

The End

Indeed, that's all for this tutorial. I hope it has helped you understand what functions are and how they can be of great use to any programmer.

Bart Teunis ■

Quick Reviews

GMZ: Game Maker zip Archiver

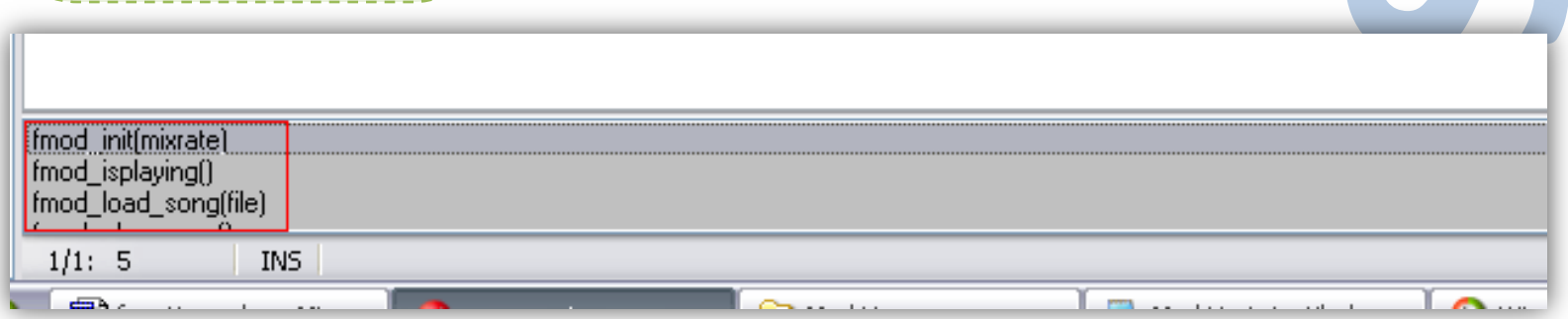
Though we have had our own fair share of .zip archiving DLLs for Game Maker, the GMC sure did have a decline recently!

This DLL is small, supports compression of ZIP files as well as the creation of password-protected zip files.

The Archiver DLL is non threaded, but for a reason according to Adventus, the DLL's creator; after calling the command for extraction, everything else could soon be executed afterwards immediately, thus narrowing the chances of a player/hacker tampering 'temporary' files.

Get it now!

gmc.yoyogames.com/?showtopic=326729



Script of the Month

This month's script is an excellent one called `sprite_replace_color`, a great script which is capable of replacing a certain color on a sprite with another. The sprite could be animated or even have alpha blending and it'll all work well! It also works pretty well if you want to replace a range of colors by repeating the use of the script multiple times. The script does not create a new sprite, but rather replaces the sprite given.

Sprite Replace Color

The script is awesome; it uses surfaces in a really cool way to create a new sprite. It creates a surface and two temporary sprites: one for the alpha blending and one for the actual colors, and 'mixes' them together to achieve a new sprite with alpha blending (like its 'mother' sprite). The script frees the temporary sprites and surfaces, so it shouldn't take much memory.

```
/*
**  Usage:
**      sprite_replace_color(sprite,oldcolor,newcolor)
**  Arguments:
**      sprite      sprite to change
**      oldcolor    color that will be replaced
**      newcolor    color used as replacement
**  Returns: (-1) on error
**  Notes:
**  This script replaces one color in a sprite with another.
**  No new sprites are created, the given sprite is changed.
**  GMLscripts.com
*/
{
    var sprite,tran,oldc,newc;
     sprite = argument0;
     oldc   = argument1;
     newc   = argument2;
    var w,h,n,i,p,t,s,l,xo,yo,surf,tempsprite,newsprite,alphasprite;
     w  = sprite_get_width(sprite);
     h  = sprite_get_height(sprite);
     n  = sprite_get_number(sprite);
     p  = sprite_get_precise(sprite);
     s  = sprite_get_smooth(sprite);
     l  = sprite_get_preload(sprite);
     xo = sprite_get_xoffset(sprite);
     yo = sprite_get_yoffset(sprite);
     surf = surface_create(w,h+1);
     surface_set_target(surf);
     for(i=0; i<n; i+=1) {
         draw_clear_alpha(c_black,1);
         draw_set_blend_mode_ext(bm_inv_dest_color,bm_one);
         draw_sprite(sprite,i,xo,yo);
         draw_set_blend_mode(bm_normal);
         draw_point_color(0,h,oldc);
     tempsprite = sprite_create_from_surface(surf,0,0,w,h+1,p,true,s,l,xo,yo);
          draw_clear_alpha(newc,1);
          draw_sprite(tempsprite,0,xo,yo);
          sprite_delete(tempsprite);
          if (i == 0) {
          newsprite = sprite_create_from_surface(surf,0,0,w,h,p,0,s,l,xo,yo);
             if (newsprite < 0) return -1;
         }else{
             sprite_add_from_surface(newsprite,surf,0,0,w,h);
         }
         draw_clear_alpha(c_white,1);
         draw_set_blend_mode_ext(bm_zero,bm_src_alpha);
         draw_sprite(sprite,i,xo,yo);
         if (i == 0) {
         alphasprite = sprite_create_from_surface(surf,0,0,w,h,p,0,s,l,xo,yo);
             if (alphasprite < 0) {
                 sprite_delete(newsprite);
                 return -1;
             }
         }else{
             sprite_add_from_surface(alphasprite,surf,0,0,w,h);
         }
         draw_set_blend_mode(bm_normal);
     }
     surface_reset_target();
     sprite_assign(sprite,newsprite);
     sprite_set_alpha_from_sprite(sprite,alphasprite);
     sprite_delete(newsprite); sprite_delete(alphasprite);
      surface_free(surf);
}
```

Contributors

While **xot** was the one to originally create the script, credit also goes to **EyeGuy** for adding the alpha channel support to the script. Well done to both of them, pretty good job.

Conclusion

As xot told me when I initially found out about the script:

> This is perfect for when you want to have sprites with different team colors. With carefully designed sprites and several calls to the script, you could replace a range of shades or colors fairly quickly.

There are many other uses the script, especially in conjunction with other GML code. You could make it duplicate a sprite several times and create a certain color variation, etc. based on events in the game. The possibilities are endless!

Eyas Sharaiha■

Extension of the Month

Sprite Generator: Tanks

For this week's Extension of the Month I have chosen a newly added addition to GMbase – A tank sprite generator, by Sandro. With this extension can actually generate tank sprites on the fly, during the game execution. The generator also allows defining options such as its color and flag. This allows for many features such as allowing the gamer to customize their tank, allowing tanks to have random tints and logos, and to save file space (as the sprites are generated *after* the game starts).

Main Functions

- **tnk_create()** Creates an new tank and returns an *id* to be used in other functions.
- **tnk_set_color(id,color)** Sets the color of the tank.
- **tnk_set_back(id,background)** Sets a background to be used as a texture instead of a color
- **tnk_set_tread(id,size)** Sets the size of the treads
- **tnk_set_banner(id,color,sprite)** Sets the color and symbol used for the banner located on the front of the tank. The symbol must be a sprite. The sprite MUST be 12x12, and have the origin set to (6,6).
- **tnk_set_turret(id,width,length)** Sets the length and width of the turret cannon. Width must be a value of 0-3, and length *should* be a value of 0-10.
- **tnk_set_sweapon(id,weapon)** Sets the secondary weapon. Use the following constants for *weapons:*
  - sw_mgun
  - sw_gl
  - sw_flame
  - sw_missiles
  - sw_homing
- **tnk_build(id)** Builds the sprites. When you are finished setting the tank settings, call this function.
- **tnk_tank(id)** Sets the tank body with *id* to be used for the calling instance
- **tnk_turret(id)** Sets the tank turret with *id* to be used for the calling instance.

Get it Now!

gmbase.cubedwater.com/?page=extension&id=157

Jonah Turnquist■

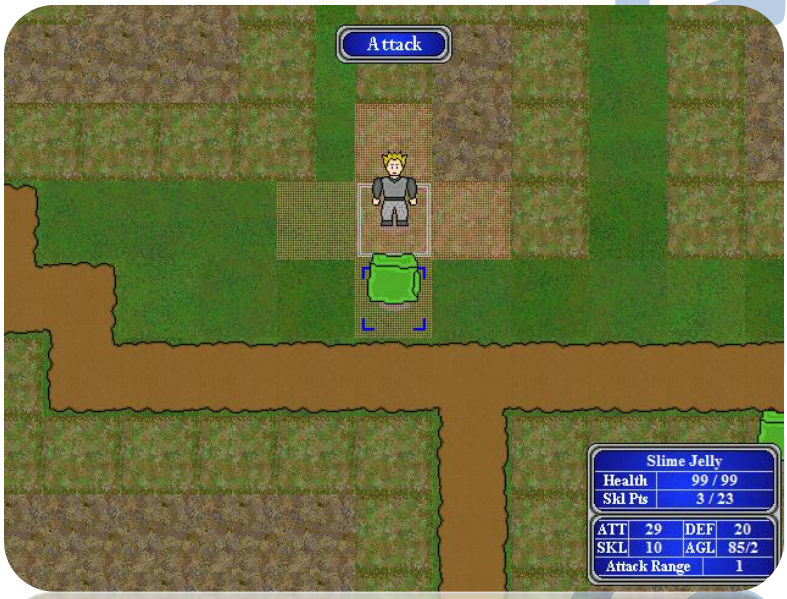
EXTENSIONS

Pick of the Month

Gythol Granditti

The wind will roar,
And the clouds will bleed,
And the dead will dance,
And the damned will head.

Okay, okay, okay. I admit that this game is pretty old. "Gythol Granditti" was created back in 2003. In 2005 it was submitted to the RPG category in the Game Maker Competition 2005 and ended up as the overall winner of the category. To make a RPG game in Game Maker is a pretty difficult and long task thus the number of GM RPG games is very low, and only a few of them are pretty good, and this one's one of them.



Gameplay

Overall, the game has a similar gameplay to most. You have to control one or more characters through the story and help them to victory in the very end. Besides that this game also contains some few minigames, which can be found around the world. The first one you will find is chicken race – bid at a chicken and hope that it is the fastest one, if it is you will win money, if it isn't you will lose money.

One of the strongest parts of this game is the battle engine. Many RPG games are using the well known Pokémon battle engine: you and your opponent are in turn selecting an attack to use… In this game it is different. Everyone in your party and all the monsters are fighting at the same time, but by turns. You are still able to move around and face different opponent, while you can attack the characters standing in the fields around you.

Pick of the Month Cont.

Controls

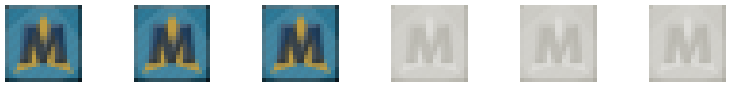
You control your party using the arrow keys. That's simply enough. But then it gets a bit tough Z, X and C are the action keys, and in my opinion it is a bit hard to remember what key is used for what, but here you go: Z is the primary key used for menus, talk, look and in general to accept/decline questions. X is simply enough as it is only used to turn on or off run mode. C seems to be used as joker.. It is also called the "start key" of some reason. Among other things, it is used to stop traders talking and make them sell something to you, for instance, a bed for Dolph.

Graphics and Design

The graphics are original. The characters are original as well, and not as the most RPG characters out there. It is a shame that they look like they have used tear gas instead of deodorant, so I'd like to see the characters improved.

Furthermore the game consists of some nice effects like lightings, mud and shadows. These effects are pretty good, if you keep in mind that this game was made before effects were introduced in Game Maker.

Sound and Music



It is impossible to judge the sound, as there are no sound effects at all. There are many different pieces of background music... The track depends of the area you are in. The music is great – or at least just for a while. It doesn't take long before

you turn your speakers off to get rid of the music. Especially the background music in town is very annoying.

Overall

The worst things about this game are the music and the funny looking characters. The things drawing up the score is the effects, the story and the engine in general. Also the idea about adding mini games is great and the very original battle engine.

Thomas Hansen■

Stats

Developer:	CZ Storm
Category:	Gythol Granditti
Version:	1.0
YoYo Rating:	3.6/6
Filesize:	9.9 MB

Download

Customizing GM7

Removing the YoYo Games Logo

When you install Game Maker 7 the bottom left hand corner of the screen is taken up with a graphical link to the YoYo Games website. Whilst this isn't a problem with the general running of Game Maker some people may find it intrusive. The good news is that it only takes 4 clicks to remove it.

File > Preferences [General] and then check the "Don't show the website image in the main window" [OK] (Figure 1, below).

Remove YoYo Games' branding (0:35)
http://uk.youtube.com/watch?v=5bBMU8f0Ctk

Change Code Coloring

If you are familiar with programming in a language other than GML or simply require more contrast to ensure certain types of coding stand out then this option is for you. Game Maker's default setting uses green for comments, purple for object names, blue for variables, brown for constants and black for pretty much everything else.

There are 16 different code elements which you can change the colouring of, or you can even disable colouring entirely if you are crazy.

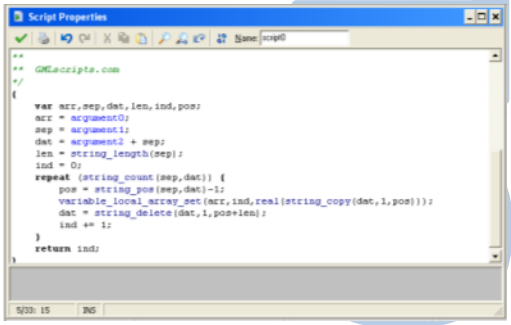
File > Preferences [Colors], then select a code element and press [Change] (Figure 2, to the right).

Change GML default colors (1:01)
http://youtube.com/watch?v=G_7MpKLD_tk

Conclusion

Obviously Game Maker works perfectly well without these tweaks, however if you are going to be spending a lot of time creating your game it makes sense to play around with these settings to see if it improves your game making experience.

Philip Gamble■



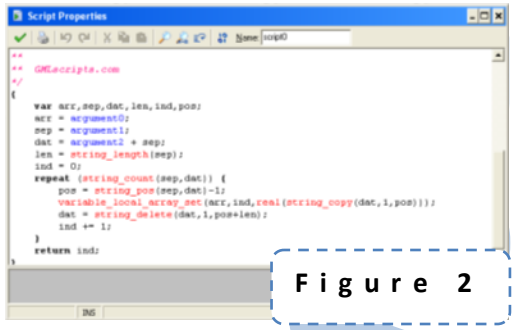


Figure 2

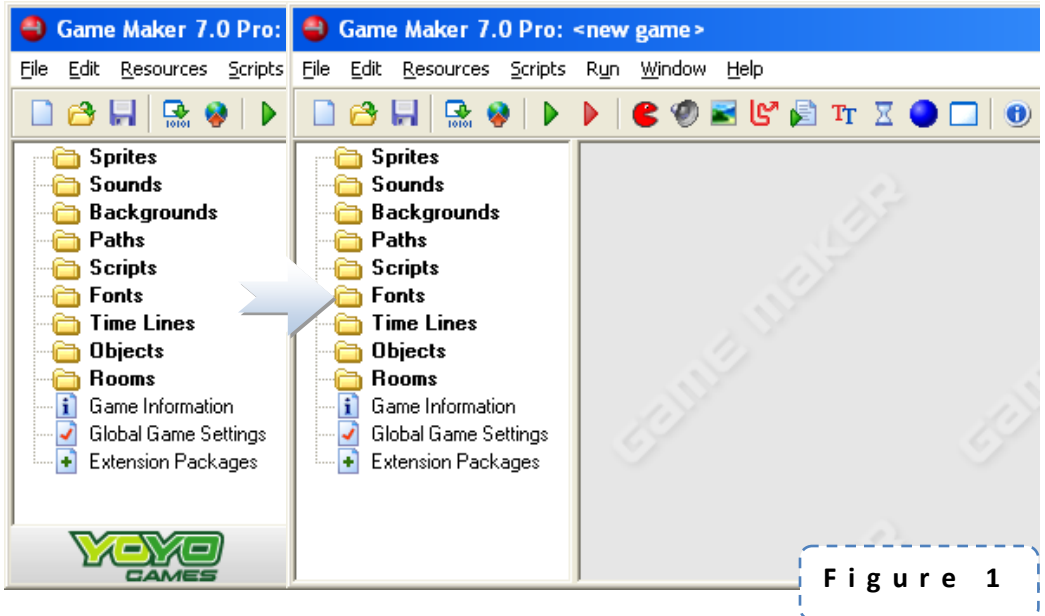
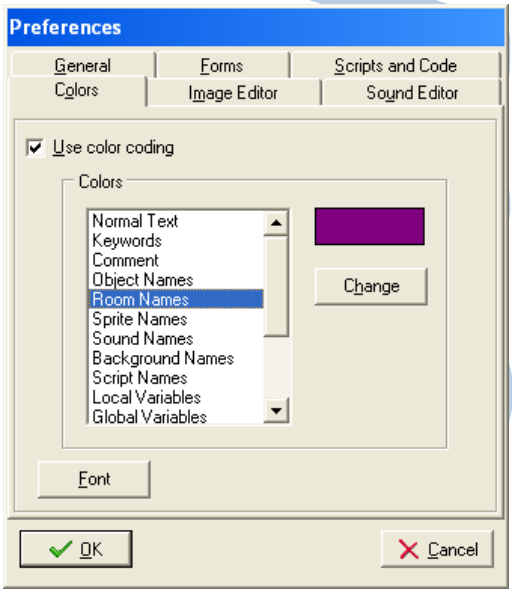


Figure 1



Reality Rampage

What they say

"cool!!! 6/6"

Description

The story involves a poke at FOX. It's *gotta be good.* Unfortunately (surprise) it's **not**. Let me explain:

I start the game and press new game. It's a top down shooter. Ok, I enter a tank. I can't move because other tanks that are not moving and won't move when pushed are in the way. I exit my tank when I suddenly teleport to a lower area in the screen stuck behind a building. Great.

From the tutorial I already got the feeling it has major flaws in it. Aiming is downright impossible with the bad control system (hold right mouse button to walk, move mouse to aim, press left button to shoot). It may sound like it works, but you have to aim pretty damn accurately to hit anything, especially with tanks. Not to mention grenades. You can't really throw anything with them, since you'll just end up destroying everything else than your target.

The second time I tested it I actually could pass the first level, but not without noticing many major problems.

The game gets very laggy in some points. The "command your team" system works poorly. And… tanks are very easy to cheat with. If you see an enemy tank, simply back off so that the enemy tank barely is visible in the screen. When you've done that it won't be shooting at you. You can simply fire at it.

The game contains no music either and needs a lot of improvement. Supposedly there is a level editor, but being unable to complete the game due to major bugs I were not able to test it.

Let me, however, add a warning. The game's method of distribution isn't the best: Click Team Installer, InstantPlay Incompatible, and on top of that: the game wasn't converted to be Vista compatible. It isn't quite easy to convert it locally, because the installer puts it in Program Files, which results in an "Access Denied" message in the converter. Oh well.

Pros and Cons

Reality Rampage is a top-down shooter that needs a lot of improvement.

Pros

- Good base
- Graphics are decent

Cons

- Poor sound effects
- No music
- Bugs
- Mediocre story
- Lag problems

Conclusion

Avoid until bugs are fixed. It could be a good game. One day.

Veeti Paananen■

Ratings

Graphics:	★★★☆☆
Sound:	★★☆☆☆
Gameplay:	★★☆☆☆
Storyline:	★★☆☆☆
Design:	★★★☆☆
Average:	★★★☆☆

Stats

Developer:	schmidtbag
Version:	1.0
Made with:	Game Maker 6
Filesize:	7.2 MB

Download

Age of Man

What they say

'Fun' is the usual word used to describe the game. The Game has potential but needs significant improvements in the graphics area to be a really good game.

Description

Well laid out and designed levels, strategic planning and idea accepting, sandbox freedom, and little things that are well fun to play with (like zapping a tiger so it don't eat your people).

Anyway who can think, this game is extremely fun to play for anyone of any age. You can simple quickly and easily begin playing requires not tutorials just simply start a new game although reading the Read Me file included is still recommended.

This game is fun to play has great design will keep you entertained for hours and so much more. The game itself and its' original ideas are just wonderful you indirectly control you people saving the user the hassle of moving armies from place to place and waging wars. You can quickly start a civilization in a small area and be fighting the enemy in no time. Simple things like sneaking in one of you men and killing enemies and then having your men attacked by a tiger well balance the game.

Pros and Cons

Pros

- Several well packed and strategized maps and skirmishes are included.
- You will literally play for hours you can't stop it is wonderful and should be put in arcades.
- Great design and game play go well with wonderful sounds.

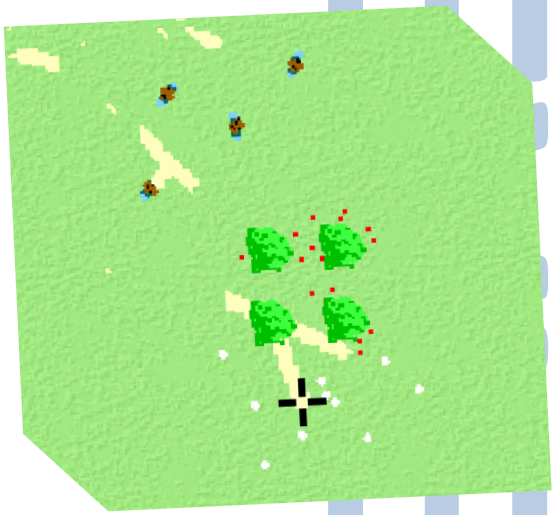
Cons

- Not very professional and menu is somewhat sloppy in other words needs polishing and refinement.
- Also you will find the missions fun yet there is not much replay value once those are completed.

Conclusion

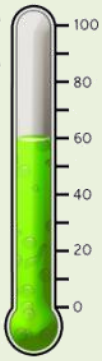
Though the game has a nice concept, it really needs more work to become a more interesting and fun game. In order for a player to really enjoy the game, a better layout, an interface, and a menu should at least be present.

Robert Colton■



Ratings

Graphics:	★☆☆☆☆
Sound:	★★★☆☆
Gameplay:	★★★☆☆
Storyline:	N/A
Design:	★★★★☆
Average:	★★★☆☆



Stats

Developer:	Hoytman
Version:	1.0

Download

Beginning .NET Programming with VB

About the Book

- 414 Pages
- Written By David Weller, Alexander Santos Labao, Ellen Hatton
- Paper Back
- Apress Publishing

What You Will Learn

You will be taken through the concepts and understandings of GDI (Graphics Device Interface) in Visual Basic. Object management, collision detection methods, project organization, transforming images, drawing primitives, 3D modeling and methods are all covered in this book.

Basic artificial intelligence programming is concepted and explained. You will actually create an AI in most of your games professionally and while reading this book.

What I Think

If you're in the market for buying a game programming book I am here to tell you not to buy this one. At first glance it looks like an all right book goes through several examples, steps into 3D and is written by Microsoft Developers, but this book is so confusing you'll kill yourself reading the first section it does not explain what kind of project you supposed to create or anything it says let's take a look at the outline for the block class (class = object) it does not say where the code goes if you were to make a console or a GDI program very confusing.

But I will say that if you are confused on how to do things such as rotate an image, then looking at this code will help yet to create the projects and get hands on experience; this book is not for the absolute beginners.

I also believe this book tried to cover too much, while they should have stuck with just 2D game programming because as short as the book was, a single chapter covered 5-10 concepts which in the long run is quite hard to grasp and not very fun to read.

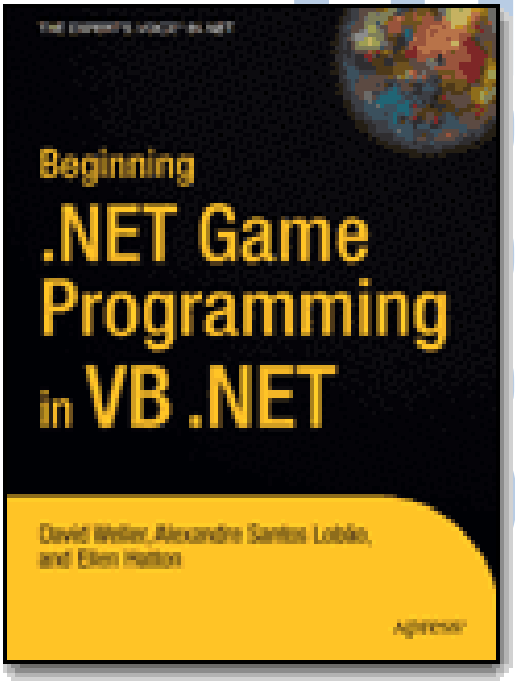
Although this book does manage and organize a well set of examples that will help a developer moving from one language to another who would just like to see the ways of doing things in VB. And I will say the recently added chapter to this revised book is the best because it actually explains how to start the project and where to add the code.

Overall, this book did work out for me but I still don't recommend it for the average reader.

Robert Colton■

Order the Book…

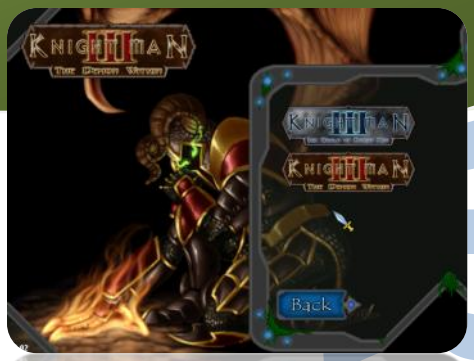
Here!



MarDar: Creations

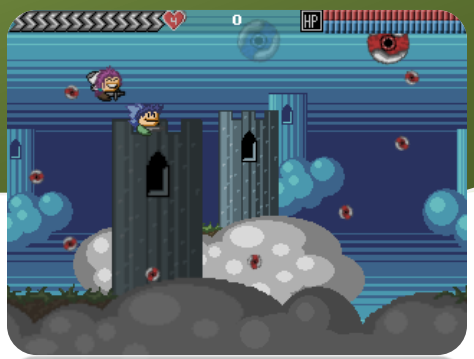
The Demon Within

The Demon Within is an 'expansion pack' for a Game Maker RPG series: Knight Man. The Role Playing game is a Diablo-style top-down 2D game that takes place in the ancient times, the times of magic, spells, but also honor. Fight 'bats', demons, and other enemies throughout multiple parts and levels. The game has really good music and suitable graphics, but the place where the game really shines – to me – is storyline. Smart and creative, well done. You must be warned though, the game has been created with Game Maker 5.3A, which has poor alpha handling techniques, etc. – so you'll almost definitely experience huge slowdowns when you're hurt, because of the effects. The game is recent, however, even though it is made with GM 5.3A.



Angels of Time

Angels of Time is an excellent game by jaj, developed for a competition in the period of two months. Angles of Time is basically a 2D shooter, made out of four levels. The game also comes with a highscore table, which usually is an excellent addition to games, as it gives an extra challenge to people: beating others' scores to advance. The game mostly revolves around fighting violence.. with violence. It's a cool, fast-paced game, nice graphics, a retro-ish style, nice music, levels, and just a good overall shine and polish.



MarDar: WIP

Almora Online – MMORPG

Almora Online is a Massively Multiplayer Online Role Playing Game, and it seems to me that 'massive' was never more appropriate to use! It is an incredibly polished game, the atmosphere is amazing, the music is great (credit is given to Rob Westwood for composing the music), and it really fits the game's overall atmosphere. Borek, the game's created, has done a magnificent job. Unfortunately however, the game doesn't seem to be in active development as it used to be, at least not on the Game Maker Community's side. However, what is available so far is good enough and deserves recognition for sure. Amazing gameplay experience.

Fatal Error RTS

Fatal Error is a Real Time Strategy game where you play as newly uploaded Anti-Virus program. This sums it up, basically: you are an anti-virus program, fighting viruses and worms… where? Inside a computer! An amazing game, action filled , and requires a good brain – I recommend it to fans of strategy games, it might be the soft-core solution you need! The game is actively developed, which is always a good sign, and is constantly being improved and new features are being continuously added.

Wrap up

Thank you for reading Issue 9 of MarkUp Magazine, for November 2007! In this issue we continued on our article format and review formats we introduced back in issue 8, including Development Journals and MarDar, as well as the game reviews with the new review format.

You might have noticed that 'The Making of…" is missing for this issue, but don't worry; it'll be back next issue better than ever, hopefully! We continued to publish b**ook Reviews**, in which our staff review books related to game development. The staff receive the books for free from all of the major game development book publishers in the world! You can take advantage from this offer by becoming a trusted staff member. Click here for more information.

Remember, MarkUp needs your help and contribution! Please contact us and contribute to MarkUp Magazine by either joining the MarkUp forum, or e-mailing the MarkUp staff.

The MarkUp Staff■■

Check out…

GMking.org is the parent network for MarkUp magazine. It has been recently redesigned to behave as 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. Come look at, and enjoy the redesign!

MarkUp has sister projects, also developed and maintained by GMking.org, all meant to help Game Developers. 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.

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!