A GMking.org Publication

MarkUp

Officially Endorsed by YoYo Games

Going Commercial: What to do?

Plus...

Level Editors Pt. 2
Cellular Automata
Speed up your Game
The Legend of Sudit

And Much More!

Interviews

Resources

Reviews

Tutorials

ISSN: 1913 - 6552

Editor's Desk

The Effects of summer on Development

We've all seen it, you're working on an exciting project, making excellent progress with great feedback. Then summer comes, you have a few weeks' vacation, and when you get back, that project seems old and you're looking for something new and exciting to do. It happens constantly and it's an effect I like to call "summer's Drought".

It actually happened with the August edition of MarkUp (if you were watching) vacations and lost interest abounded, but we succeeded! In fact, if you're reading this now, you're having the pleasure of reading the Game Maker magazine, past and present, with the most published issues. We just broke the record. There are some great ways to keep your projects from feeling the Drought too!

First and most importantly, have steady breaks from your projects; take at least a week off each month. If you have time away from it, you won't grow tired of it, and the Drought won't send in as fast. Secondly, don't work alone, find some other developers and friends to help with the project and make it more of a community atmosphere.

You also should plan for when you're taking off you summer vacation, and let the community know. Not only will they know when you're off and who's in change while you're away, but you also won't feel so pressed to keep "checking in". Finally, when you see someone suffering from Summer Drought, then send them a short note and let them know how much you appreciate their work. The Summer Drought isn't something you can predict or control, it strikes like an earthquake and once it's happened, you're just left to pick up the pieces and get everything back in order in as gung-hoe a fashion as possible. But, at least if you take some precautions, you won't have picked up quite so much. Before you know it, Summer Drought will be over!

See you next month!

Robin Monks■

Contributors

Eyas Sharaiha	Sr. Editor
Robin Monks	Editor
Andris Belinskis	Editor
Philip Gamble	Writer
Bart Teunis	Writer
José María Méndez	Writer
Michael Sharaiha	Writer
Thomas Hansen	Writer
John Leffingwell	Writer
Jonah Turnquist	Writer
Dan Meinzer	Graphic Designer

Table of Contents

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

Cellular Automata

Cellular Automata

A cellular automaton or CA (plural: cellular automata) is just a grid or matrix in which each cell has a state, and interacts with other cells. It can be used to create biological simulations, granular material simulations; cool graphic effects....For now just remember the grid/states part. We can represent the different states a cell can have by using numbers, like this:

0	0	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	0	0	0
0	0	0	1	0

This minesweeper-like grid is the base of a cellular automaton. As you can see, its cells can have 2 states: 0 and 1.

Ok. Now what can we do with this thing? Let's make an experiment. Each step, we will check all cells. If a cell has state 0, we ignore it. If a cell has state 1, we change it to state 0, and give state 1 to the cell immediately below it. This is like "moving" the 1 downwards.

What will happen if each step we draw a colored dot where the 1s are? The 1s (now represented as colored dots) will begin to fall, until they disappear from the grid. "Hey, this is kind of like falling particles, hehe". Well, not quite. Particles in a GM particle system can't interact between them. Cells in a CA can.

Let's give another state to the cells. Now our cells can have 3 states: 0, 1 and 2. 0 and 1 will behave like in the previous example, but 2s will stop 1s from falling down. So now if a cell has state 1, it will change to 0 and give a 1 to the cell below **if that cell isn't a 2**. Otherwise, we ignore the 1.

Now the 1s will fall until they find a 2:

0	0	0	0	0
0	0	0	0	0
1	1	1	0	0
2	2	2	0	1
0	0	0	1	0

Cells with state 2 behave like a floor, and cells with state 1 like gravity-affected objects. We could then make some more complex behaviors, like stacking 1s to simulate piles of objects, strange patterns in which if a 1 is surrounded by a 3 and a 5, it changes to a 2 and then the 3 goes 0....whatever.

Code

Let's apply this to simulate falling sand in GM. You could simulate a gas, or water, or even make a life sim, or little intelligent particles with pathfinding....the possibilities are endless. But for now simple falling sand is enough.

This automaton is quite similar to the second example. It has 3 states: 0 (there's nothing in that cell), 1 (there is a grain of sand) and 2 (there is a floor tile.). Let's code everything using scripts, which are more organized. We need a first script to initialize things: We need to know the position of the top-left corner of the grid in the screen (`grid_x`, `grid_y`), the size of the grid in cells (`grid_sizex`, `grid_sizey`), and the height and width of a cell for drawing purposes (`grid_width`, `grid_height`).

We will then initialize the grid (a 2D array) and (attention) a **special 2D Boolean array** to tell if we have already checked a cell in this step. We will run through the CA grid row after row, so if we don't use this boolean grid, when we "move" an 1 downwards, we will encounter it again in the next row and if we don't know if it has been simulated before, we will move it again, repeating the process each row until we throw it out of the grid. All of this in the same step, so we wouldn't see anything in the grid when executing the game.

The solution is to move it only the first

time we find it, and then mark the cell we moved it to as "simulated". This way, when we encounter it in the next row, we just skip it because we know it has already been simulated.

Initially both grids are empty, so the CA cells are os and the Boolean cells are "false" (not simulated yet).

```
global.grid_x = 0;
global.grid_y = 0;
global.grid_sizex = 40;
global.grid_sizey = 40;
global.grid_width = 8;
global.grid_height = 8;
global.grid[global.grid_sizex+1,
global.grid_sizey+1] = 0;
global.simulated[global.grid_sizex+
1,global.grid_sizey+1] = false;

for (i =
0;i<=global.grid_sizex;i+=1){
   for (j =
0;j<=global.grid_sizey;j+=1){
     global.grid[i,j] = 0;
     global.simulated[i,j] = false;
    }
}
```

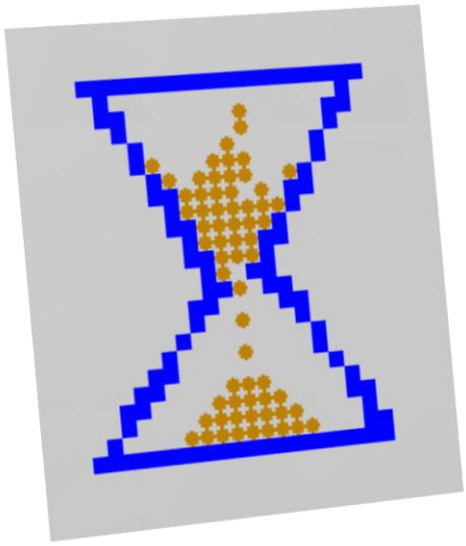
After that, we will make a script to draw a sprite where 1s are, and a different sprite where 2s are. The script is shown in Listing 1.

As you can see, we run through all cells in the grid, and make a switch statement for each one: if it is a 1, draw the sprite "grain", if it is a 2, draw "wall", if it is a 0 do nothing. Make sure to have small enough sprites, smaller than a single cell or they will overlap.

Listing 1

```
var col,i,j;

for (i = 1; i< global.grid_sizex;i+=1){
    for (j = 1; j< global.grid_sizey;j+=1){
    switch(global.grid[i,j]){

        case 1: {
            draw_sprite(grain,-1,global.grid_x+global.grid_width*i-
global.grid_width,global.grid_y+global.grid_height*j-global.grid_height);

  } break;

        case 2: {
            draw_sprite(wall,-1,global.grid_x+global.grid_width*i-
global.grid_width,global.grid_y+global.grid_height*j-global.grid_height);

  } break;
    }
  }
}
```

And finally, the cool part: the simulation script, which will run through the CA grid moving the cells around depending on their state.



The CA in Action

The sand simulation script is demonstrated in Listing 2 overleaf.

We run through the grid, checking the cells and moving them following some rules:

1. If a cell is 1 and below there is a 0, move the 1 to the 0 cell and mark it as simulated.
2. If a cell is 1, and has a 1 below:
   2.1. If the right diagonal cell is free (0), move there, unless the left diagonal is free, in which case we will move there, or if both cells are occupied, in which case we won't move. Then we mark the destination cell as simulated.

After simulating all the cells once, we

Listing 2

```
//Sand simulation
var i,j,r;
for (i = 0; i< global.grid_sizex;i+=1){
    for (j = 0; j< global.grid_sizey;j+=1){

        //If the cell hasn´t been simulated yet
        if (!(global.simulated[i,j])){

            //SIMULATION RULES
            if((global.grid[i,j] == 1) && (global.grid[i,j+1] == 0)){
                global.grid[i,j] = 0;
                global.grid[i,j+1] = 1;
                global.simulated[i,j+1] = true;
            }

            if((global.grid[i,j] == 1) && (global.grid[i,j+1] == 1))
            {
                r = 0;
                if (global.grid[i+1,j+1] == 0) r = 1;
                if (global.grid[i-1,j+1] == 0) r = -1;
                if (r != 0){
                 global.grid[i,j] = 0;
                 global.grid[i+r,j+1] = 1;
                 global.simulated[i+r,j+1] = true;
                }
            }
            //SIMULATION RULES END
}}}

//restart the simulated boolean grid
for (i = 0;i<=global.grid_sizex;i+=1){
    for (j = 0;j<=global.grid_sizey;j+=1){
        global.simulated[i,j] = false;
    }
}
```

restart the Boolean grid to "non-simulated", and repeat the process in the next step.

Ok. The CA part is over. Now we need to be able to put sand and walls in the grid (using the mouse, for example) and for that I will give you two scripts which involve just a little bit of math:

Walls

```
//put_wall(x,y);

posx = abs(argument0-
global.grid_x)div global.grid_width;

posy = abs(argument1-
global.grid_y)div
global.grid_height;

global.grid[posx+1,posy+1] = 2;
```

Sand

```
//put_sand(x,y);

posx = abs(argument0-
global.grid_x)div global.grid_width;

posy = abs(argument1-
global.grid_y)div
global.grid_height;

global.grid[posx+1,posy+1] = 1;
```

That´s it. If you apply these scripts to an object and run the game, you will see that the simulation behaves quite like real sand.

Conclusion

Cellular Automaton is a very interesting field of mathematics that can be used to create original eye candy or simulations. They are a bit limited in size because of GM´s speed, but big automata grids can be achieved through the use of dlls.

José María "ArKano" Méndez■

Level Editors pt. 2

This is part 2 of the Level Editor series, first started in the previous issue of MarkUp Magazine – issue 6. This time we discuss the interface of a level editor, and how to overcome certain problems that you might face when making it.

The reason why making a level editor interface is hard is not because people aren't creative enough to create a sufficient interface, but mostly it is to create an interface that is transparent and does not collide with the "game contents" – since the view of the level editor will be the actual game.

See, when viewing a level editor, what you should see is how the level looks like when played in the game (3D games could be an exception, where the level editor is only a map-view of what should appear in the game), and levels tend to be very big when it comes to area, i.e. does not fit in the window.

As a simple workaround to this problem, Game Maker's view feature is used, so that when in the level editor, you could navigate through different parts of the level easily. Now, since the entire room IS the level itself, the interface you need to construct must be transparent so that:

a. You could easily interact with it
b. Interaction with it might not result in unwanted interactions with other objects in the room
c. It does not take up considerable place from the in-game room

What we need

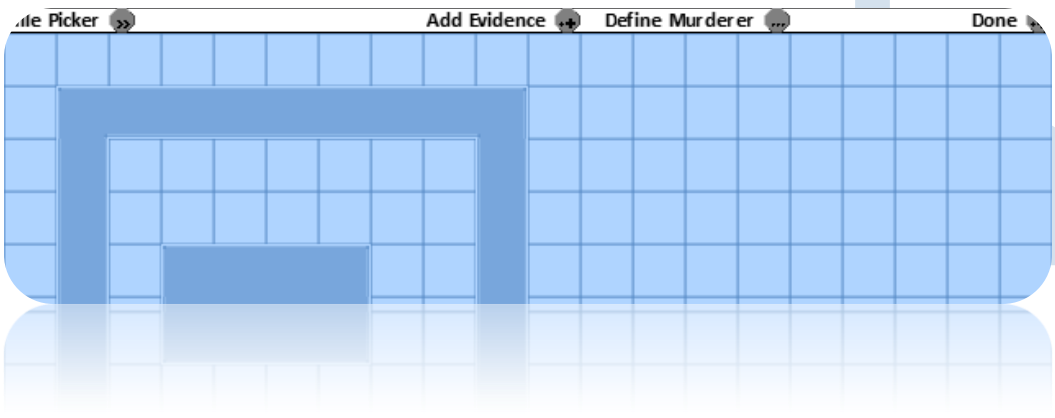
Ease of Interaction

To be able to easily interact with an interface, this means that all – or all frequently used – commands are readily accessible via a mouse. A keyboard shortcut is a smart way, but it really doesn't make it easy for the users to create their own levels. To make commands accessible in that way, the best thing to do is the creation of buttons – preferably with images as labels – that the user could click to choose a command.

However, if space problems arise, similar commands could be all grouped into one drop-down box, or dialogue boxes could also be added to the mix. This might make interaction a bit harder, but solves the problem of having everything accessible at the same time.

Interface and Editor Interaction is separate

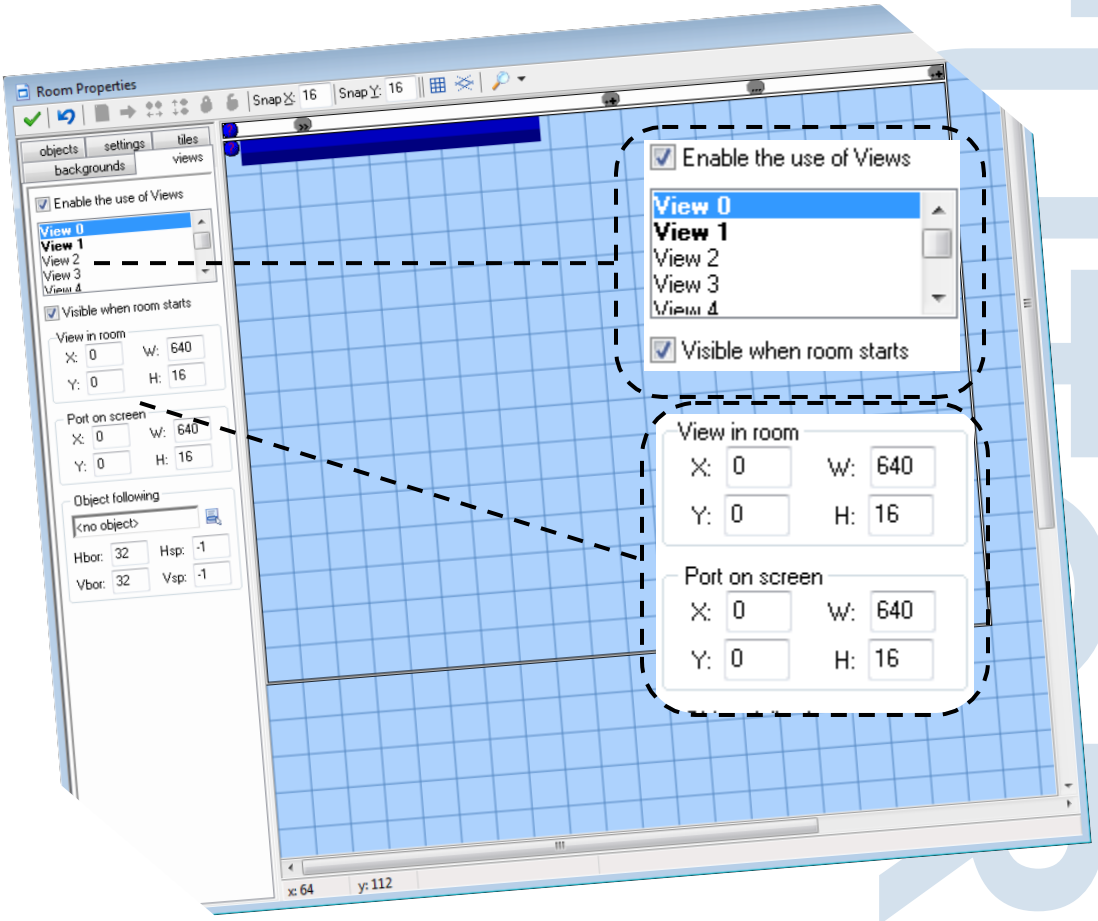
This means that when you interact with the interface itself, i.e. to add a new object or tile to the room, you will not have other object in the editor, i.e. a nearby tile or object that has already been created, respond to the interaction with the interface.



For example, if an object was created, and while moving the room throughout the view, the button that you now want to click was positioned above the object, clicking the button must not provoke the object to create its own mouse-click event.

This will be discussed later in more detail, but two ways of solving this are:

a. Adding more code in the mouse-click events to see if the mouse is in contact with the interface object
b. Separating the whole interface in a separate view located at a position that isn't accessible by the in-game view, and having them share the window.



Interface doesn't take away In-game space

This means that the interface should – at any time – hide (by being positioned over) the least amount of objects in-game. This also means that the interface should be so transparent that most – if not all – of the in-game area of the room is accessible to the user via the mouse or other simple tools.

There are also two ways of solving this:

a. Making the interface expandable and collapsible, and having all the interface objects hidden when collapsed, as well as a much smaller (in terms of area) sprite for the interface background.
b. Separating the whole interface in a separate view, as discussed in the previous section, would also mean that the entire room area is exposed and is completely separate from the interface objects.

Solution Attempts

Keyboard Shortcuts (No Visible Interface)

Having no visible interface immediately solves the two final points that were considered problems, because it can't physically interfere with the in-game objects. Everything is done via keyboard shortcuts, except

for final placement of objects which could be done by a mouse movement and mouse click.

As for how this could be done in terms of coding, it is pretty easy; a global controller object could be created that contains different key-press or key-release events (I for one like key-release as a method of identifying when a button has been selected). For each event, an object is created, or a dialogue box is shown, etc.

However, as I said before, this leaves the problem of having an interface that is easy to interact with.

Separating Views

An example image on separating views is shown in the figure in the previous page.

To do that, you easily create 2 views, one with the sufficient size you want the in-game view to be, and the other with the sufficient size you want the interface to be. The W and H both for the view in room and the port on screen should be the same, otherwise unwanted scaling would occur.

Now, for the "View in room" for the interface, simply locate the X and Y coordinates that the interface is physically located in the room. The port on screen decides where you want to draw it in the screen. In the example shown, the interface is both located upwards in reality and is also

viewed there, what you could do is have the interface be viewed below the in-game part.

An expand/collapsible interface

As it could be seen in the figure, featuring the interface of the game Command & Conquer: Generals (screenshot courtesy of GameSpot), the interface could expanded and collapsed by clicking the ▼ button on the right.

Even though this features a game and not a level-editor, the same concept could be used for level editors, where when the ▼ button is pressed; the entire interface would disappear, except for extremely important items and the button itself; to expand the interface again.

Instead of complex instance creation and destruction, it could be made very simple. All of the buttons could have a single parent: `obj_button`. When the menu is collapsed, the line of code is entered:

```
obj_button.visible=false;
```

This will make all children objects of `obj_button` invisible. Now, a simple condition before each click event could be added to evaluate if the menu is expanded or not. The code goes like this:

```
if (visible==true)

{

<do command>

}
```

As for expanding the menu again, the code could simply be:

```
obj_button.visible=true;
```

To summarize even more code, this could be changed into more of a toggle action, where this line of code is only used all the time:

```
obj_button.visible=!obj_button.visible;
```

Also as a part of the expanding and collapsing of the menu is changing something, that is its sprite, position, alpha, or yscale. That should also be done in the click event of the button.

Another issue that needs to be worked around is not having objects that are located under certain buttons respond when the button is chosen; this is done by doing something similar to the following:

```
if (place_meeting(mouse_x, mouse_y,
o_interface)==false)
```

If this code is placed before every mouse click event for all in-game objects, it would check if the mouse is actually also over the interface. If it is, then it wouldn't perform anything. This code also relies on the fact the collapse sprite is a different sprite, if it isn't (i.e. when collapsed it suddenly becomes invisible) then the code should be changed to the following:

```
if (place_meeting(mouse_x, mouse_y,
o_interface)==false or
o_iterface.visible==false)
```

Evaluating all the methods

The best way for me to evaluate all the methods that we have worked on is to compare them with the problems we faced in the first place – did they solve them? I also added another category which is eye candy, cause it is a plus.

I created the table below as a way of measuring the success of each one of the methods we worked on and created for this tutorial.

In reality both separating views and having an expand/collapse interface are good enough. It is easier to have separate views cause some might be bothered with programming the expand and collapse toggling part, or how objects are interacted with. Also, when it comes to eye candy, one can also make an interface in a separate view with good eye candy; it is just easier to have a photoshopped background for the interface with creative shapes and edges, etc. other than being limited by the rectangular shape of the separate view.

Conclusion

Level editors is an exciting feature of games, it allows for unbelievable expansion in the games – and takes a lot of the weight of game design off the chest of the lead programmer and divides that weight onto the whole community. Making it might be hard, but in the long term, it will be very beneficial to a game.

The good thing is, even with all the types of interfaces for level editors that have been introduced here; there are still many more, and not only that, but also room for innovation.

The trick of creating a new type of interface is of course to think about the problems the user is facing with the existing interfaces and try to solve these problems. Good luck with creating your own interfaces.

Eyas Sharaiha■

Method	Ease of Interaction	Interface and in-game objects are separated.	Doesn't take away space.	Eye Candy
Keyboard Shortcuts				
Separate Views				
Expand/Collapse Interface				

Speed Your Game

Including examples: GM6 and GMK

A very important aspect of a game or program is the speed at which it runs and the CPU time it uses. Many things need to be calculated and drawn. As a consequence, the CPU usage is high and the framerate might drop.

The way you write your code can have a serious effect on how fast your game or program will run. This article will explain some improvements you can do to your code to make your game or program run faster.

The idea behind these improvements is very simple: don't recalculate or redraw things when it's not necessary.

With this idea in mind, we'll try to find some techniques to gain speed.

Split expressions

Checking expressions might take some time, especially when many expressions need to be checked at the same time using the `and` keyword.

Let's start with an example: suppose we have a piece of code to check whether the mouse is inside a rectangular region:

```
if (mouse_x > 5 and mouse_x < 325
and mouse_y > 5 and mouse_y < 245)
{
        <<execute code>>
}
```

! Example 1: split expressions

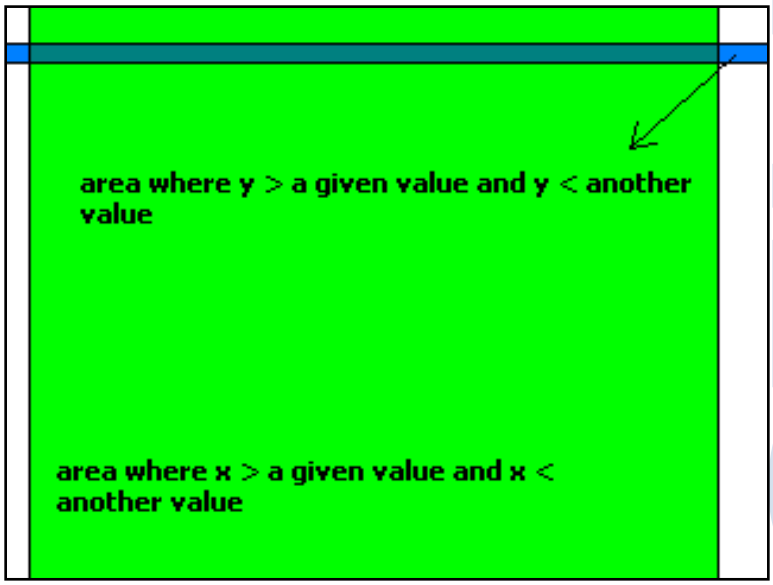
Nothing is wrong with the syntax of this code and if this condition needs to be checked only once in the step event of a single object, it won't slow down the game. It does take some time to check this expression and since it's a rather big expression (4 things to check) it'd be better to split this expression into two parts. The code then looks like this:

```
if (mouse_x > 5 and mouse_x < 325)
{
 if (mouse_y > 5 and mouse_y < 245)
 {
  <<execute code>>
 }
}
```

The advantage of using this piece of code is obvious: the second expression will only be evaluated when the result of the first expression is true. Only when `mouse_x` meets the condition, the expression for `mouse_y` will be checked. This improvement does not really speed up game or program execution, but it makes sure no expressions are evaluated when it's not necessary (here: when `mouse_x` doesn't meet the condition).

! Example 1: split expressions improved

The above example assumes a rectangular area starting at (5;5) with a width of 320 px and a height of 240 px. Now assume that the rectangle's width is much bigger than its height, like what is shown in the figure in the previous page.



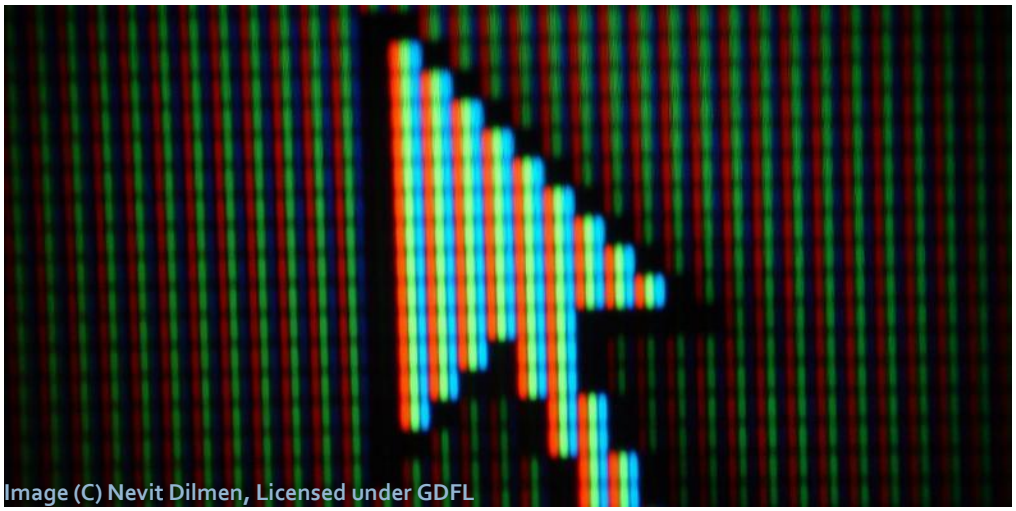


Image (C) Nevit Dilmen, Licensed under GDFL

Then it's very probable that the mouse will be in the green area more often than in the small blue strip. In that case it'd be better to change the order in which the expressions are checked:

```
if (mouse_y > 5 and mouse_y < 20)
{
if (mouse_x > 10 and mouse_x < 500)
{
 <<execute code>>
}
}
```

This is, of course, rather extreme. It'd be a bit stupid to check the width and height of each rectangle just to switch the order in which the expressions are evaluated. Yet, it can be useful. The idea can be expressed like this:

Always put a condition that is less likely to be true further outside the brackets than conditions that will be true more often.

This improvement will be used in examples in the next chapters.

! Example 2: split expressions (different order)

Don't execute code every step

Now we have seen that expressions shouldn't be too large. But they're very useful to determine whether some code should be executed or not. Expressions can be used to prevent many lines of code to be executed when it's not necessary. Suppose you'd like to know whether the mouse is over a certain item in a list. Then the following loop could be used to check this (assume that 10 items are displayed):

```
var i;
for(i = 0;i < 10;i+= 1)
{
 if (mouse_x > 5 and mouse_x < 325)
  {
    if (mouse_y > 5+i*15 and
mouse_y < 5+(1+i)*15)
       {
              selected = i;
       }
}
}
```

! Example 3: don't execute code every step

That's a very nice piece of code and the syntax is correct, but it's not well written to get a good speed. This piece of code can be used in the step event. Each step, the loop is executed 10 times. 10 times per step the expression for `mouse_x` is checked. That's something that's not necessary at all. Let's put that expression outside the loop:

```
if (mouse_x > 5 and mouse_x < 325)
{
var i;
for(i = 0;i < 10;i+= 1)
{
if (mouse_y > 5+i*15 and mouse_y <
5+(1+i)*15)
       {
              selected = i;
       }
}
}
```

This example gives us a nice rule: never put loops outside the brackets if you don't have to.

Note that, in the example, the expression for `mouse_y` must be in the "for loop" to work correctly.

The whole loop is only executed when the result of the expression for `mouse_x` is true. This piece of code will no longer slow down the game or program as long as `mouse_x` is not larger than 5 and smaller than 325.

Actually the program will still run as slow as before when `mouse_x` is inside the given area. That's something that can't be avoided. But we've done what we were supposed to do: don't recalculate or redraw things when it's not necessary.

Now let's have a look at an example that shows how to fully use this trick to get the best speed. For this, the user will interact with the mouse. We'll use one specific function:

`mouse_check_button_pressed(numb)`
Returns whether the mouse button was pressed since the last step.

This function is exactly what we need. This function only returns true when the user clicks a mouse button, and not while he is still pressing that button. So this function will return true for a very short time. The next step, this function will return false again. This function can be used in an expression as the condition that needs to be met for the code to be executed. We'll use this function in an expression that we will add to the previous piece of code:

```
if
(mouse_check_button_pressed(mb_left))
{
 if (mouse_x > 5 and mouse_x < 325)
  {
   var i;
   for(i = 0;i < 10;i+= 1)
    {
     if (mouse_y > 5+i*15 and
mouse_y < 5+(i+1)*15)
       {
       selected = i;
       }
    }
  }
}
```

! Example 3: don't execute code every step (fastest)

Note that `mouse_check_button_pressed( mb_left )` is put outside all other brackets, because that's the expression that is least likely to return true. There's only one improvement we can do to this code: replace the "for loop" by a repeat loop. It's quite logical that a repeat loop is faster than a "for loop" because a repeat loop doesn't need to check an expression with each iteration. Replace the code on the gray background with the piece of code below to get the same code with a repeat loop:

```
var i;
i = 0;
repeat (10)
{
       if (mouse_y > 5+i*15 and
mouse_y < 5+(i+1)*15)
       {
              selected = i;
       }
       i+= 1;
}
```

This is the best way to write this piece of code for optimal speed, highest fps and lowest CPU usage. Only if the left mouse button is clicked, the second condition is checked. If the result is true (`mouse_x` is in the required area), the loop will be executed. And only if the expression in the loop returns true, the code between those brackets will be executed. Now if we compare the above piece of code to this:

```
var i;
for(i = 0;i < 10;i+= 1)
{
if (mouse_x > 5 and mouse_x <325)
{
if ( mouse_check_button_pressed(
mb_left ) )
{
if (mouse_y > 5+i*15 and mouse_y <
5+(i+1)*15)
{
selected = i;
}
}
}
}
```

Quite an improvement, isn't it?

We have seen how to recalculate as few things as possible by using expressions and putting them in a logical order. Now it's time to advance to the next topic: improving the speed when drawing.

Surfaces and a redraw variable

Drawing sprites, text, backgrounds or anything else on the screen is something that takes a lot of CPU power. As an example, we'll attempt to draw a list. First we need to create a list and add some items to it:

Create Event:

```
list = ds_list_create();
var n;
n = 0;
repeat (500)
{
ds_list_add(list, "Item
"+string(n));
n+= 1;
}
```

This piece of code creates a list and adds 500 items to it. Now it's time to draw the list (this might be very slow on some older computers):

Draw Event:

```
var i;
for(i = 0;i <
ds_list_size(list);i+= 1)
draw_text(x,y+i*15,ds_list_find_val
ue(list,i));
```

! Example 4: surfaces and a redraw variable

If you look at the framerate, you'll see that it comes nowhere near the room speed. That's not so surprising, knowing that a text string is drawn 500 times per step. We'll have to find a solution to get around this. The solution is using surfaces (if you need to learn about surfaces, you can read a tutorial on it written by Eyas Sharaiha that can be found here: http://gmpedia.org/wiki/Surfaces).

First, we create a surface (add this code to the existing create event):

Create Event:

```
sf = surface_create(320,240);
```

Now we have created the surface. It's time to think about how we will draw the list items onto the surface. We can't draw 500 items on the surface at the same time.

We need to set a number of display items (items that will be displayed):

Create Event:

```
no_display_items = 10;
```

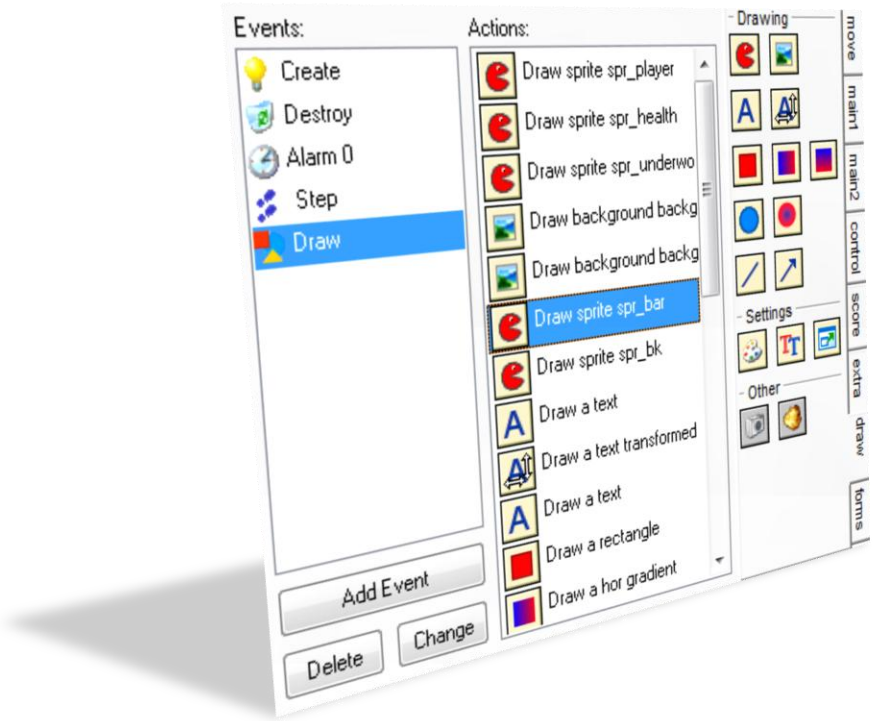
To be able to display the other list items later on, we'll add a scroll variable:

```
scroll = 0;
```

Now we can draw the text on the surface. The following piece of code will set the drawing target to the surface, draw a clear and the items and reset the drawing target:

Create Event:

```
surface_set_target(sf);
draw_clear(c_white);
var i;
for(i = 0;i<no_display_items;i+= 1)
{
draw_text(2, 2+i*15,
ds_list_find_value(list,i+scroll));
}
surface_reset_target();
```



Note that the index of the list value is not i, but i+scroll. This is to make scrolling possible.

Now we set a variable redraw:

Create Event:

```
redraw = 0;
```

It is set to 0 because the surface doesn't need to be redrawn right now (it's just been drawn in the create event).

This redraw variable indicates whether the surface should be redrawn "virtually", it doesn't mean that it should be redrawn on the screen (it's always automatically redrawn on the screen as long as automatic draw is on).

Now we can draw the surface on the screen in the draw event (you should first remove the other code in the draw event):

Draw Event:

```
draw_surface(sf,x,y);
```

Now the surface is drawn on the screen (and also the text), but there's a difference: the code to draw the text on the surface (`draw_text`) has already been executed and isn't executed anymore each step.

A comparison: a painter (the computer) needs time to make a painting (to draw the surface), but once the painting is finished (`surface_reset_target` is called), it can be displayed without the painter having to draw it all over again.

What when the surface does need to be redrawn? For that, we'll use the redraw variable and the step event:

Step Event:

```
if (redraw == 1)
{
surface_set_target(sf);
draw_clear(c_white);
var i;
for(i = 0;i<no_display_items;i+= 1)
{
draw_text(2,2+i*15,ds_list_find_val
ue(list,i+scroll));
}
surface_reset_target();
       redraw = 0;
}
```

This piece of code will make sure that the surface is redrawn when redraw is set to 1.

After that, it will be immediately set to 0 again to prevent that the surface would be redrawn unnecessarily.

To compare it to the painter: the painter makes sure the painting is white again (`draw_clear`), then redraws the painting.

Now we have an easy way to redraw the list.

Let's say we want to scroll down when holding the down arrow key. Then this piece of code can be used:

Keyboard Key Down Event:

```
if (scroll < ds_list_size(list)-
no_display_items+1)
{
        scroll+= 1;
        //increase scroll value
        redraw = 1;
        //redrawing is required
}
```

This will scroll down in the list until the end of the list is reached.

Now, we're able to control exactly when the surface needs to be redrawn.

Only when the surface is redrawn, the CPU usage will increase. After that, it returns back to normal.

This is a very effective way to increase the speed of your games and programs e.g. the room speed can be set to 60, then an alarm can be set that executes the following code each

two steps:

```
redraw = 1;
alarm[0] = 2;
```

This way, drawing on the surface happens at 30 frames per second and normal calculations are done at 60 fps.

! Example 4: surfaces and a redraw variable (improved)

Using surfaces this way is especially useful when creating drawing programs and for drawing huge lists (with thousands of items).

Particles & Destroyers

! Example 5: particles and destroyers

The particle system in Game Maker has some functions to switch off automatic drawing and updating of particles.

If your game or program runs at 60 fps and you'd like the particles to be drawn and updated at 30 fps, you can use a surface, draw the particles on the surface and use an alarm, as explained in the previous chapter.

Unfortunately, there's no special solution to make the drawing of particles go fast.

For each particle, calculations need to be made (note for each) and each particle needs to be drawn.

So the most important thing to do is to keep the particle count low. The best way to do this is by using particle destroyers.

Suppose you're making a side scrolling game where particles are drawn on the background. Right next to the source of the particles is a house that is in front of the particles (lower depth).

All particles that go behind the house are no longer visible so there's no need for them to exist any longer.

A destroyer can be put there to destroy the particles that come in that region. That way, the particle count will stay as low as possible.

Conclusion

Depending on the way you write your code, your game will run fast or slow. The most important thing is to only calculate or redraw things when necessary.

By using expressions, you can determine when calculations or redrawing are necessary, but evaluating expressions can also slow down your game or program when you evaluate too many at the same time. To speed up drawing, surfaces can be used.

If you keep all these things in mind, your games and programs will definitely run much faster.

Bart Teunis■

QUICK REVIEW

KX Tesla

This extension package allows you to easily add tesla effects (electricity effects), such as lightning bolts and electrical discharges, to your game. To use it, you can either use gml or use the action library included. You can also change some advanced parameters to modify the effect. If you're creating a game that requires this effect, you'll definitely find this extension package useful.

Get it now!

http://xrl.us/4day-kx

Script of the Month

Powered By: GMLscripts.com

Implode String

The implode string script and its variations is a fairly simple script that reverses the action of the explode_string script. The script basically joins all contents of an array to each other, and adds a separator between different array ids if wanted.

```
/*
**  Usage:
**      implode_string(sep,array,size)
**
**  Arguments:
**      sep         string used to between elements
**      array       name of an array given as a string
**      size        number of elements in the array
**
**  Returns:
**      a string comprised of each element of the
given local
**      array of strings seperated by the given
seperator
**
**  GMLscripts.com
*/
{
    var sep,arr,num,out,ind;
    sep = argument0;
    arr = argument1;
    num = argument2;
    out = "";
    ind = 0;
    repeat (num) {
        out += variable_local_array_get(arr,ind)+sep;
        ind += 1;
    }
    out = string_copy(out,1,string_length(out)-
string_length(sep));
    return out;
}
```

Variations

Implode Real

Adds an amount of real numbers in an array to a single variable string, separated with the separator defined.

Implode Real Global

Repeats the function mentioned above, but by importing the array data from a global array.

Implode String Global

Repeats the function of the original script written, but by importing the array data from a global array.

Creator

The script has been created by xot, who is also the owner of GMLscripts.com. All troubleshooting, reports, and related questions should be directed there. You can, however, contact MarkUp magazine for information about how to use the script, and other applications.

Eyas Sharaiha■

Extension of the Month

Powered By: GMbase The user created extension library

Max WinAPI

Max WinAPI is a great extension which allows you to use Windows API right inside your game window. Windows API, informally WinAPI, is the name of the core set of application programming interfaces available in the Microsoft Windows operating systems. Windows APIs are things such buttons, menus, message boxes and scroll bars that you often see in computer applications. With the Max WinAPI extension, you can put Windows API right into your Game Maker creations!

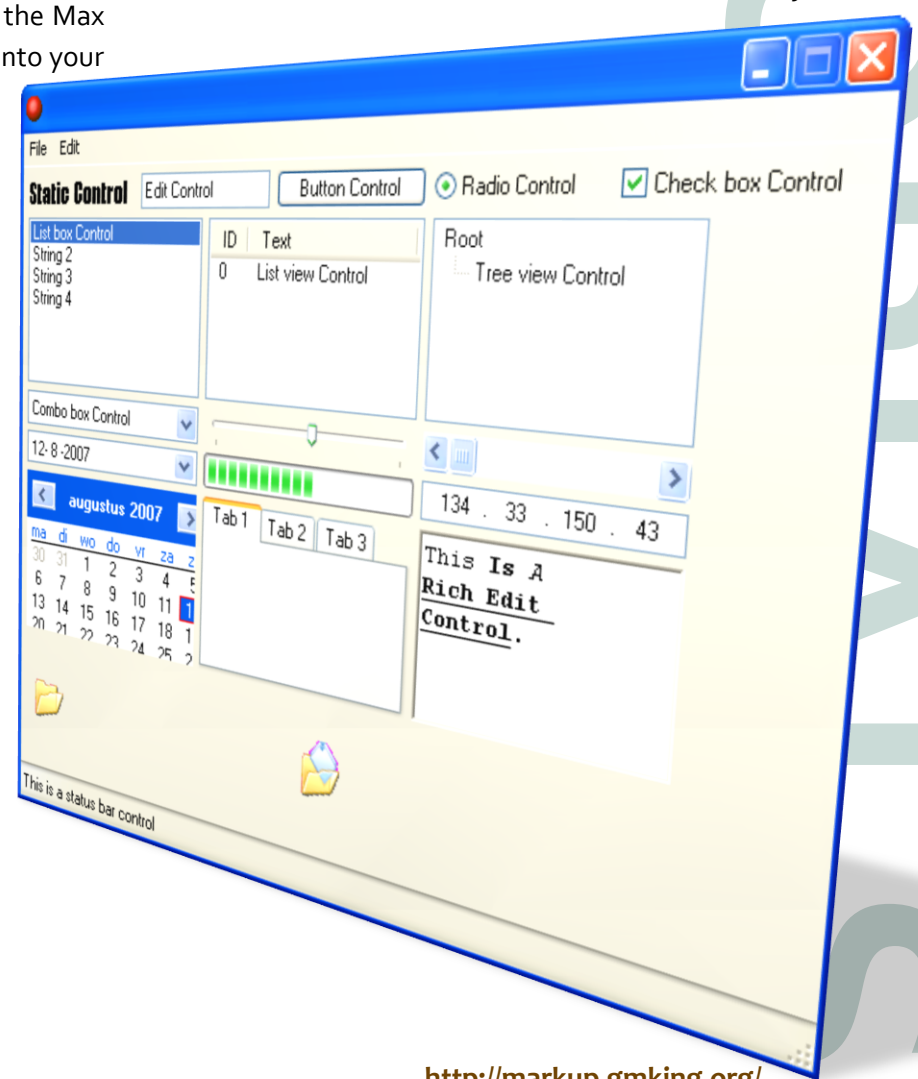
Max WinAPI supports the following controls:

- Static controls
- Edit controls
- Button controls
- Radio controls
- Check box controls
- List box controls
- List view controls
- Tree view controls
- Combo box controls
- Animate controls
- Progress controls
- Date-Time controls
- IP Address controls
- Calendar controls
- Rich edit controls
- Scrollbar controls
- Status controls
- Syslink controls
- Tab controls
- Up-Down controls
- Menu Functions

Max WinAPI also allows Game Maker to both check the status of each control, and to change them by using their special ids. This is a great extension for making applications such as Instant Messengers and internet browsers, and can also be used in games. The Max WinAPI dll was made by Hobble and wrapped into a extension by Callum.

Download:
http://gmbase.cubedwater.com/index.php?page=extension&id=119

Hobble,
Edited by Jonah■



Review of the Month

Pickup: Muffy's Great Escape

This month I am going to review the game "Muffy's Great Escape". So... Why this game? I am glad you ask ☺ As this is the first review, I decided to take a look at the first great game I found at YoYo Games when the first beta was released... Actually it's also the first game I marked as favorite and Staff's Pick when YYG became a reality. Enjoy.

Game Play

The gameplay reminds me of the good old TV-games. The concept is very simple: Get Muffy to the bell in the top of the room without letting him fall down. Along the way you have to collect fruits and coins, and of course avoid obstacles like fire, spiders and water drops. From the beginning you are provided with 3 lives, but you can earn more lives by collecting 100 coins (as usual in many arcade games). The game has no save feature, but instead you get a password in the beginning of each level... If you, of some reason, miss the password you can always find it in the settings.ini file.

The game has three different skill levels: Rookie, normal and hard. The only thing changed when you select another skill level, is the scroll speed, which respectively is 0.6, 0.8 and 1.0.

When you play Muffy's Great Escape the first time you might find it a bit difficult, but within long you will get used to make him running as well as jumping.

It contains 15 levels divided into 5 worlds. For instance jungle, winter and cave.

Controls

Well you only need some few keys to

MReg System

The MReg System is a powerful registration system that you could use with your game to enable users to register the game and get a full version.

The MReg system is a GEX powered by a DLL. The program allows you to create registration keys for users in your program or game. Each user can have an unlimited number of registration keys.

Get it now!

http://xrl.us/mreggex

QUICK REVIEW

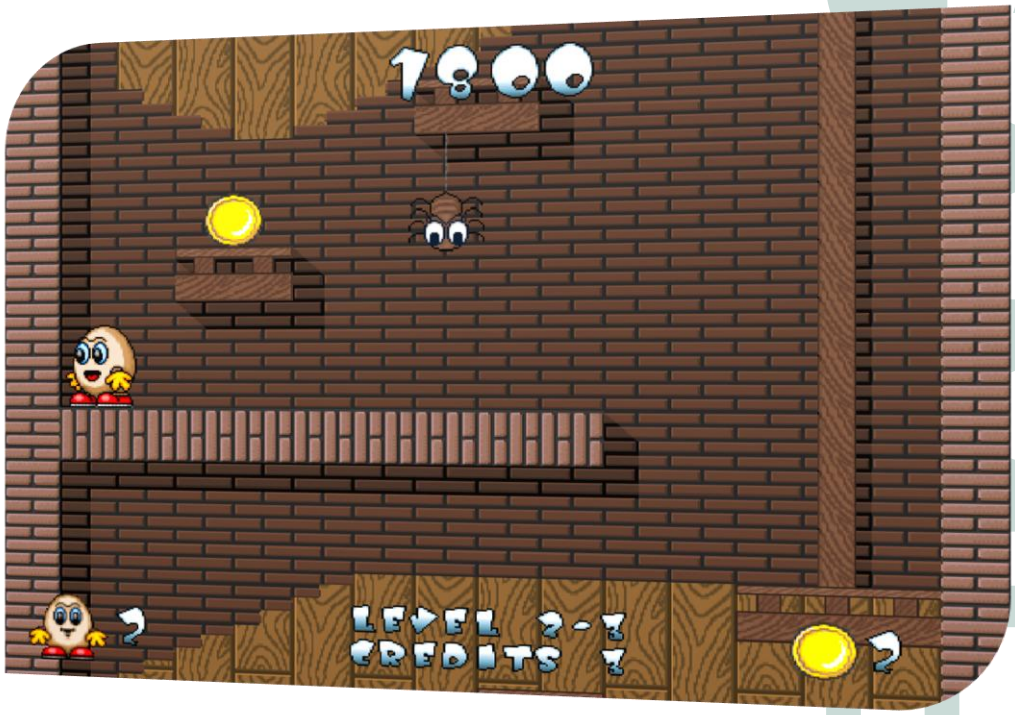
Pickup: Muffy's Great Escape

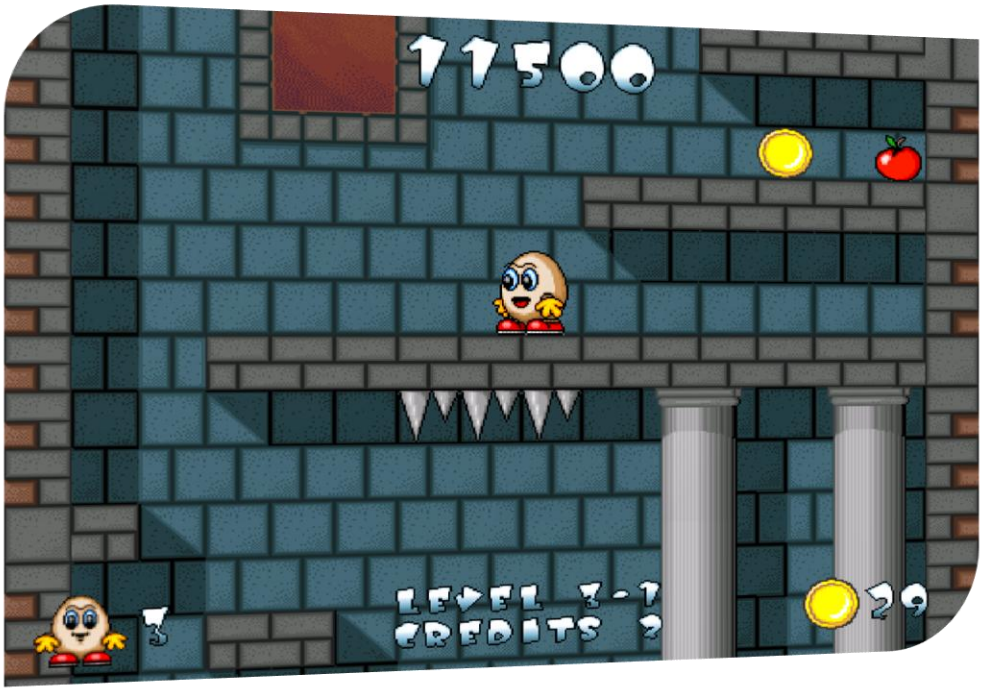
control the menus and the game. The menus are controlled by the arrow keys and Enter for selection. To turn of the music in the menu the ESC key is used.

In the game you control Muffy with the left and right arrow keys and jump with space. It doesn't take long to get used to this basic way of controlling a game. Once again these basic controls make the game look like the old TV arcade games.

Initially you get three credits which can be used when you run out of life. To use a credit you have to hit the Enter-key within 10 seconds. If you don't, the game will be over.

Furthermore the D-key can be used to open a debug window in the upper left corner of the screen, providing information about the FPS and the scroll speed and thereby the chosen difficulty. If the creator intended to make this debug key, or if (s)he forgot to remove the function, is unknown.





Graphics and Design

The graphics are original and well designed. The most amazing artwork in the game is, in my opinion, the title screen (see screenshot below).

Pickup: Muffy's Great Escape

The only annoying thing about the GFX is when you run the game on a widescreen monitor which makes Muffy a very fat egg, as well as making all of the images and the text on the screen blurred. This is mainly due to having it full screen.

As for the menu design, I thought it seemed pretty basic; only a simple list menu.

Sound & Music

The sound effects are good and "placed" so you understand what they mean. But the music consists of only two tracks; one for the menu, and one for the game itself. The in-game track is good... for a couple of levels, but afterwards it gets annoying. The creator really needs to add more variety to the background music of the game, for instance, one track for each world in the game. Luckily, sound and music can be turned off in the settings menu.

Overall

This is definitely a great game with only some few flaws. I already mentioned one of them, and that's the monotonous background music. Another thing, I'm missing is a confirmation from the checkpoint signs. I'd hope for a sound or a graphical change to the sign... Just to tell the player that (s)he has reached a checkpoint.

I give it 4 out of 6 stars. One or two starts could be added to the scores if the points I mentioned earlier would be sorted out.

Thomas "Snabela" Hansen ■

SUMMARY

Review Summary

Creator: Kama
Category: Platform
YYG Rating: 3.4/6 (29 members) (2007-08-25)

Summary: Help Muffy is escaping... From what, you ask? The bottom of your monitor of course ☺ The game was made in Game Maker 6 (I can say for sure as I had to convert it in order to make it run under Windows Vista).

Get it now!

Download here...

QUICK REVIEW

ColorScan DLL

The ColorScan DLL searches for a certain color on the screen and returns the number of pixels with that color in addition to the coordinates of each one of these pixels.

While the DLL is still not fast enough for large screen areas, it is certainly faster than using traditional GML methods to get the colors of pixels.

Get it now!

http://xrl.us/colorscan

FINAL RATING

Going Commercial

Introduction

In this deluxe editorial article, we study commercial games and give tips for those wanting to go commercial with their games. Next, I "review" the distribution methods of a number of commercial Game Maker games, interview their authors, and talk about their success.

A history

Game Maker, its community, and the games people are creating with it have certainly come a long way. When I first found the Game Maker Community in late 2002 there were very few commercial games that were made using Mark Overmars' software, in fact had someone suggested creating a commercial game back then they would probably have been laughed out of the forums.

Jump forward to 2007, when after a couple years spent away from Game Maker I returned to the community. I see a much larger number of games are now being released commercially - albeit with a varied degree of success. There has also been a huge increase in the number of Game Maker 'teams' who now have their own domain name, suggesting to me that the users of Game Maker have certainly grown up.

With many people believing that Game Maker itself has recently becoming more commercialized with the advent of YoYo Games it looks like the growth in the number of commercial Game Maker projects will only continue. In a world that is favoring more open-source and freely available software this may be surprising, but once people see that there have been successful commercial projects it is natural that they too want a piece of the action.

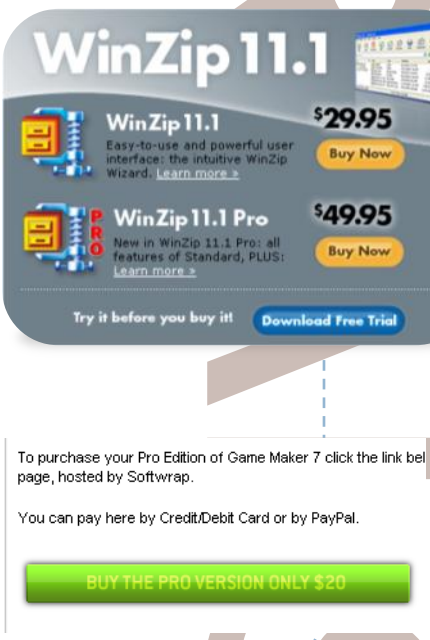
As well as what you might think are the traditional methods of making a commercial game, i.e. selling physical CDs or digital copies of the game there are other methods available to enable a game creator to receive monetary gain from their software.

Although people who have decided to go commercial may not like to share the facts about the success of their projects it is obvious that it has been varied. One anomaly to this rule is StudioEres who have been very open about their commercial project entitled "Immortal defense" posting updates of the game's progress on their forum. Revenue generation methods:

Shareware

Shareware is one of the oldest and most popular distribution methods out there. The user gets a chance to try a program for free before deciding whether or not to purchase the full version. The free version of software is offered normally with feature or level limitations or a gameplay time limit. The difference between shareware distribution and the use of a demo is that shareware can be converted into the full version by entering an activation code whereas use of a promotional demo would still require the full version of the game to

Listing 1 – Examples of shareware advertisements prompting the user to purchase the full version.



be installed on your computer.

Shareware games are often accompanied with 'nag screens' which encourage the user to purchase the full version of the software. WinZip is an example of a popular shareware program offering a 'free trial' version of their application. Game Maker itself is shareware as the pro version of the software has additional features which can be unlocked by paying a fee.

Adware

Software supported by in-program advertising is known as Adware. Once the game has been installed or is being run adverts will be displayed on the user's computer. Often the adverts are downloaded remotely from the Internet.

Advertising software is bundled with an application and earns the game promoter small revenue for each installation. If a piece of software is popular the advertising revenues can quickly add up.

A benefit of adware is that the programmer can be rewarded for their work without the user having to purchase the software, however this isn't always that case as some software producers ask people to pay a fee to upgrade and remove the advertisements.

Many people see adware as intrusive as the sponsored messages displayed can cause a nuisance if they disrupt their activities. Adware has had a bad press due to the misleading and deceptive tactics various companies using the method have used to distribute their software.

Other methods

It's may not be seen as commercial but some developers are rewarded by asking for donations to support their products future development. They may also choose to sell merchandise related to their product enabling consumers to choose whether or not and to what degree they would like to support the project. Likewise traditional internet advertisements can be placed on the creator's website. These methods cause minimal annoyance to users who are able to use the software for free, whilst enabling those who choose to support the product to do so.

The best way to see the different methods used to distribute commercial games is to look at specific examples. So I set about cataloguing a handful of Game Maker made creations which have tried (successfully or otherwise) to generate an income. I also caught up with the creators of some of these games and asked them how they choose which commercial path to take. If you are considering making one of your future creations commercial listen up to what the people who have already tried say.

Next Up

Next, as part of this editorial, I will examine four commercial Game Maker games and talk to the creators about the strategies they used and the successes, or otherwise, that they have had. It is important to note that the game itself is not being reviewed, but the distribution techniques involved.

Now Reviewing: Magi

In their words

"Magi" is a cross between RTS and RPG, yet it is very different from other games of those genres, actually it's a genre of its own.

In Magi the player creates a wizard (choosing from various available classes, attributes and styles) and immerses in series of magical duels in a never-ending pursuit for the power and immortality.

Magical spells and projectiles, summoning mythical creatures and calling spirits to curse the opponent are common ways to victory in Magi.

Limitations of the free version

- Only 5/10 characters available.
- You can only learn 4 spells
- No save option
- Limited to 30 in game years

Interview

Why did you choose shareware as your distribution format?

Frankly, I did because this is *the* way to sell Indie games. The downloadable games market is growing dynamically and there is some good money to be made there. Portals dealing in "casual" games (like BigFish or Reflexive) are making millions of dollars every year and many "Indie" developers are making serious business out of selling games from their websites. With the recent expansion of broadband Internet connections, downloadable games are getting more and more accessible. They also don't suffer from some issues that retail games do such as "shelf life" and relying on distributors.

When you started work on the game did you plan to use shareware?

MAGI wasn't started as a commercial project. It was just a game I always wanted to make. Still, I knew that if I'd sell the game in the end, I'd make it a downloadable shareware title.

After I got some really great feedback from the players and Indie gaming press, I decided to go commercial.

Do you believe your distribution method has

been successful?

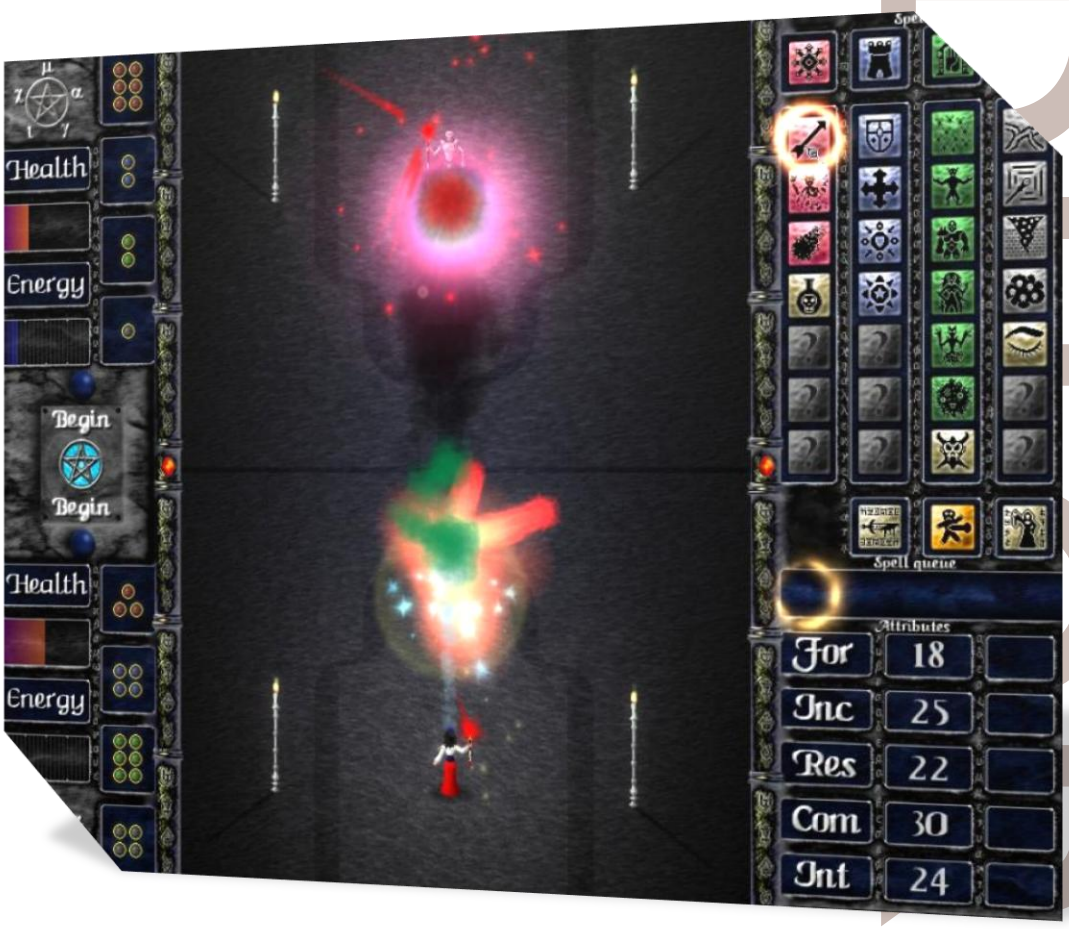
Yes. I wanted to break into the gaming industry with this game, while making some nice money out of it. The whole project taught me a lot about the business, got me a job in a big game-dev studio (CD Projekt - makers of www.thewitcher.com, I used MAGI as my portfolio) and earns me some nice cash at the end of each month. I've started as an amateur and ended as a professional game developer with some actual experience and a nice game under his belt.

What would you say to anyone who is considering making a commercial game?

Making Indie games can be much more than a hobby. If you are patient and prepared to learn a lot (mostly by mistakes), browse many forums and most importantly - develop a great game, go for it! The worst you can get is some good deal of serious experience. Just remember that the Game Maker Community is not the end of the world.

Any advice on promoting a commercial project?

Get your game reviewed by the Indie game sites and blogs out there. It's as simple and as hard as that. Remember that GMC (or GM sites in general) are not the end of the world. You want to advertise a game, not an amateur project.



My thoughts

Lots of useful advice here, and proof that all that time spent programming can land you a job. Of course promotion is critical if you want people to download your game, so finishing a decent game will be far from the end of your project.

Information

Genre: Magical strategy game
Developer: Thomas "TeeGee" Grochowiak
Distribution type: Shareware

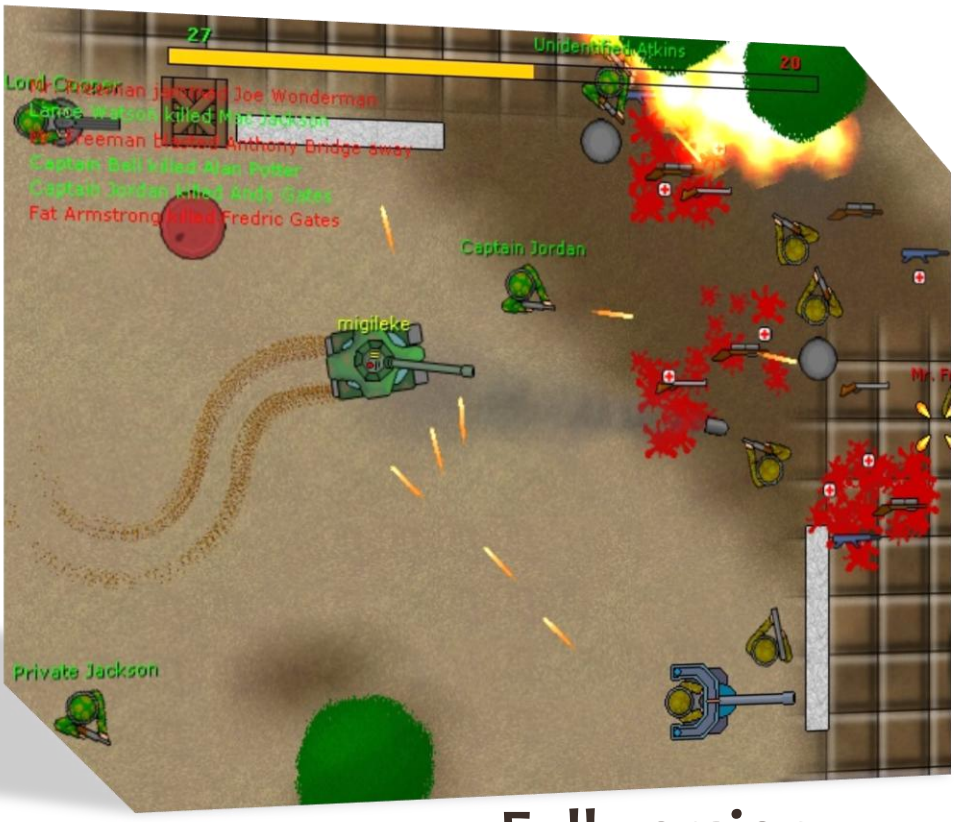
Free – full version $19.95 (USD)

http://www.getmagi.com/

http://gmc.yoyogames.com/index.php?showtopic=284869

SUMMARY

Now Reviewing: Headhunter



In their words

Headhunter is a top-down battlefield game. 2 armies clash and you are just one simple soldier. Not even the captain. But there are tanks, machineguns, helicopters, vehicles, anti-tanks, power-ups and lots more to aid you to victory. See you on the battlefield!

Full version extras

- Level editor
- Bonus in-game credits to unlock bonus features
- More challenges

Interview

Why did you choose shareware as your distribution format?

I am 17 years old, and there is a lot that I have to pay for myself. The €10 ($13.40 USD, £6.70) per month that I receive from my parents as pocket money just isn't enough to pay for everything. There were 2 options: a vacation job or trying to get enough money by selling my games. I decided to go for the second option.

When you started work on the game did you plan to use shareware?

Nope. Not at all. I started working on it 2 years ago. I had not enough money to buy Battlefield 1942 and decided to try to make it myself. I worked some weeks on it and had great fun testing/playing. Then I worked on other things, then on Headhunter again, then on other stuff again, and that cycle kept going for 2 years. When it became a really nice and fun game I thought: "Gee, if you'd give me the choice to play either Headhunter or Battlefield 1942, I'd pick Headhunter! And Battlefield 1942 is worth €30+!" So I thought that maybe it would be worth selling. Then I finished it, made a Free Edition and a Gold Edition, and right now the game is available for $4.99 on my website.

Do you believe your distribution method has been successful?

Nope. I don't know what the definition of success is, but $0.00

Now Reviewing: Headhunter

profit is not a success. But that has many reasons:

1. I give away like 90% of the game in the Free Edition. People don't want to buy the Gold Edition anymore; they are satisfied with the Free Edition.
2. My site uses a virtual currency: Migisoft Gold. If you want to buy a game, you should buy gold first with real money, and then you can buy the game with gold. At first it seems like a great idea, such a virtual currency, but apparently it's too far-fetched for a mere $5.
3. I haven't advertised a lot yet.

How would you define success for Headhunter?

If I had to define success, then I'd say: "success is reaching what you wanted to reach". In my case, I wanted to make at least $50 or $100, but $0 is way below that.

My thoughts

I quite agree with Migisoft's definition of success, I made this exact same point on GameMakerBlog a while back. I appreciate the honesty expressed in this article particularly as his chosen method does not appear to be working. The good thing is that Migisoft has taken note of reasons why no one has brought the full version of Headhunter. Migisoft Gold may have seemed like a good idea at the time, but if you want someone's money you really need to make the process as simple as possible.

SUMMARY

Information

Genre: RTS
Developer: Migisoft / migileke
Distribution type: Shareware
Free – gold version 50 Migisoft Gold ($4.99 equivalent)

http://migisoft.awardspace.com/

http://gmc.yoyogames.com/index.php?showtopic=313137&st=0

Now Reviewing: Immortal Defense

In their words

Immortal Defense is a story based space-themed tower defense game. It's a pretty big game, just to give you an idea, there are 100 levels, 11 types of towers, about 50 pages of story, 26 types of enemies, an original soundtrack, and about 6-10 hours of gameplay.

Free version limitation

- Only one third of the levels are playable

Immortal Defense was released with a very open approach to sales. Statistics of the number of downloads; site hits and purchases have been posted in a topic on the StudioEres forum to enable other independent game developers to learn from RinkuHero's experiences.

A month after its release the game had achieved 56 sales out of 4600 demo downloads - a 1.22% conversion rate. This gives is a very transparent insight into the amounts of money which have been involved so far. A retail price of $22.95 multiplied by 56 gives a total of $1,285.20. However this would only be true if every copy was sold at full price which hasn't been the case as there is a half price offer available for people who feature the game on their blog or website. The Immortal Defense demo is included on the cover DVD of the October 2007 issue of PC Advisor magazine in the UK.

Bear in mind that as Rinkuhero stated in an interview $1,285.20 in a month is below the minimum wage in many states, although of course you can choose how much time you spend promoting your game. In effect you are left with two jobs – first making your game, and second selling it.

The openness with which this game has been released may be useful to people who are interested to see the effect different promotion methods have on downloads and sales.

Information

Genre: Tower Defense/Shooter/RTS
Developer: Studio Eres/rinkuhero
Distribution type: Shareware
Free – full version $22.95 (USD)

http://immortaldefense.com

http://gmc.yoyogames.com/index.php?showtopic=302008

SUMMARY

Now Reviewing: Galactic Hacker

In their words

As a hacker in a futuristic world dominated by corporations, you must use your skills and technology to survive. The player must take jobs from corporations to earn money. Money can be used to buy equipment such as proxy by-passers, or other computer upgrades. No two games will ever be the same in this dynamic world. Galactic Hacker even features the ability to play with friends on a LAN, for some great multiplayer hacking action.

What they say about the Adware

Note that the installer includes adware… When you install Galactic Hacker it installs the adware. This is no secret! Some Antivirus software may say that the installer contains a virus. The antivirus software just detects the adware and reacts accordingly. If you install GH, you can easily uninstall the adware from your Control Panel > Add/Remove Programs. 90$ was spent to put Galactic Hacker on download.com, so I included the adware because I get paid per install.

Interview

Why did you choose Adware as your distribution method?

I didn't, my marketer did for a half-half split of what profits he could make from Galactic Hacker - and he chose adware.

Until the approach you had no plans to make money from the game?

That is correct.

Do you believe your distribution method has been successful?

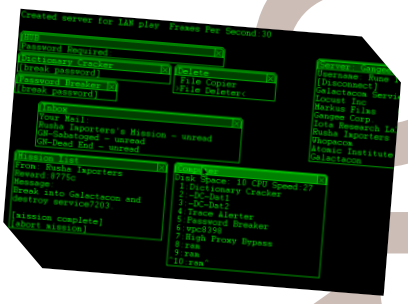
Relative to the time I spent making Galactic Hacker, yes I do. I do not believe it can be done again though. As stated, Download.com was used, and they no longer allow adware

My thoughts

An interesting introduction into using Adware here, Natso didn't have any plans to generate revenue from the game until an approach was received from a distant friend who wanted to distribute the game in this format.

As complaints inevitably came Natso released an adware free version of the software however with 24,000 downloads and counting of the pay-per-install adware supported version someone is making money somewhere. Thankfully a very open approach to the use of Adware was made, at least on the Game Maker Community.

Personally I think that Adware has past its peak now as more people have become aware of what it does, and computer security has improved.



EDITORIAL

Information

Genre: Hacking simulator
Developer: Natso
Distribution type: Adware supported
Free – adware bundled. Adware-free but otherwise identical version also available.

http://www.download.com/Galactic-Hacker/3000-7551_4-10628521.html?tag=lst-0-1

http://gmc.yoyogames.com/index.php?showtopic=37960

SUMMARY

Conclusion

In this feature we have seen a variety of completely different Game Maker created games which have attempted to go commercial. You have also been given an insight into just some of the methods used to generate revenue from games. If you are considering going commercial with one of your projects, don't. The most important thing to do is work on your game until you are happy with it and have received good feedback, then, and only then, should you consider taking your next step.

If you do go along the commercial route you should remember that Game Maker is far from the end of the world, and you should promote your game in as many different avenues as is possible.

Thank you to everyone who has contributed material to this feature, especially the game creators who have shared their first hand experiences of different monetization methods with me.

Philip Gamble,
GameMakerBlog■

R E V I E W

xWUNG

By Cactus with music by John Marwin
http://gmc.yoyogames.com/index.php?showtopic=321963

It seems that Cactus just can't stop pulling out great games. After "Clean Asia!" and "Fractal Fighter" comes another fast, high action game with equally retro graphics.

xWUNG is controlled with the mouse, swinging a ball behind it on a rope. When the ball collides with an enemy they explode. In essence, this is extremely simple. But the additions of possible combos and the fact that the explosions and debris will kill you if you're too close add to the complexity and make the game more challenging.

As I said, the graphics are up to cactus's typical retro style. They're formed of few colors and just simple straight lines. This makes the surroundings less distracting, allowing you to focus on playing – and winning – the game.

With online high score tables, there is always a special drive to get to the top of the list – and this game is no exception. You can find yourself playing for ages, determined to at least get on the list, if not to get to the top of it. And, judging by the highest scores, you'd have to keep at it for quite a while!

Grego Tyler■

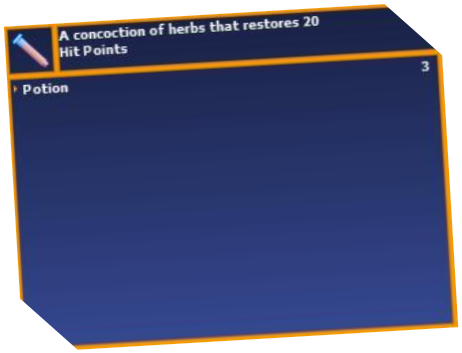
The Legend of Sudit

Introduction

Despite big name similarities between this game and the Zelda games, brod – its lead programmer – makes it clear that The Legend of Sudit is not a Zelda clone. In fact, he says, it's closer to Final Fantasy games than Zelda.

The game is a Work in Progress, and probably in terms of stages of game and level design this game is at the earliest stage I've ever reviewed.

How's it looking so far?



So, how is the game doing? What parts of it are complete and what parts of it aren't? Well, engine-wise it appears to be close to completion. The battle engine is functional, the inventory is also functional, items work pretty well, and the movement system as well.

What isn't complete, however, is one of the most important parts of the game's aspects – level design. You appear to be "trapped" in this extremely small area, and there's very little place to walk. I've seen only one type of monsters out there, and I've also seen only a single attack for the player.

Since the game is still in development, game and level design will not be judged or considered a criterion; the author did not start developing them yet.

The Ratings

Graphics

The game has this general old-school look and feel. The atmosphere the game has makes it normal for you to see graphics without smooth edges and with such low color depth. The graphics are not amazing, but they do fit the stereotypical old-school-game look and feel.

The thing is, old-school should not be used as an excuse for the graphics. I know good old school Game Maker games that have went around that issues and created very polished graphics with smooth edges and a

QUICK REVIEW

E-Browse

E-Browse is not a typical browser component DLL; it is a big DLL and an IE-browser-component DLL is only a small part of it.

It contains many other exciting features, including an HTML viewer with syntax highlighting, FTP support, environmental variable viewer, complete logging, system statistics, and, it uses the VDI console.

Get it now!

http://xrl.us/ebrowse

cartoony look and feel while still having the same (actually, enhanced) atmosphere old school games tend to have.

The game gets 7 out of 10 on graphics.

Sounds & Music

It is fine... it's seriously fine. So far, I feel like the game has a single song (when in the overworld) that keeps repeating on and on, but still – it does this smoothly and in a nice way. The t one of the music is also pretty good.

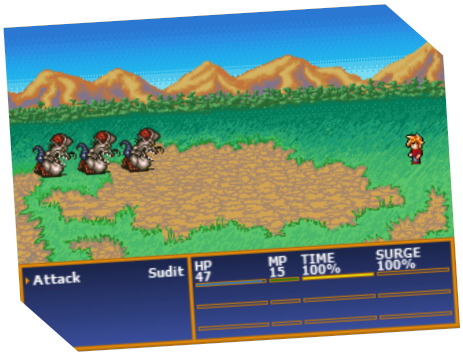
Also, I've noticed the game has different music for the battle, and the way the transition occurs between both sounds isn't rough at all.

That being said, I was disappointed by the game's lack of sound effects. In such a style of games I do admit I don't expect that many – footsteps and such are not required, to the contrary adding them might make the game annoying – but still, I expected to see sounds of buttons and attacks throughout the game.

In general, the game's sound and music gets 7 out of 10.

Bugs

I'm impressed; the engine is really smooth! I didn't find anything that I would call a bug, in general the game ran very smooth and I was not interrupted with any error messages. I also didn't get any outcome I didn't expect.



9 Out of 10, what more can I say?

Smoothness

The game had a bunch of things that I call "stupidities", these things really annoy me. For instance, during the battle you could select non-existing attacks, etc. In general, however, it was fine. Movement was smooth; you never got stuck, etc. The game gets a 6 in smoothness.

Overall

All in all, the game gets a 7.25 score, which is pretty good.

Conclusion

Old School games aren't always bad, and this game is a proof for that. There's not much to play and enjoy right now, but this seems like it could become a promising product in the future.

Eyas Sharaiha■

QUICK REVIEW

GMFTP

GMFTP is an FTP connection DLL for Game Maker. It allows you to connect to an FTP server on the internet, and is then capable of viewing files and directories in the FTP server via a friendly interface.

The DLL could be used either directly to allow someone to make their game download certain server files from an FTP server and upload them again, or simply to create an FTP client.

Get it now!

http://xrl.us/gmftp

Sonic & Knuckles Flicky Panic

I remember my Sega consol from when I was a kid, simple, not too many buttons and so easy a 5 year old could use it. The reason I am bringing this up is because of this game, which I played the other day and made me finally find a sonic game worth playing since the Sega days.

Don't get me wrong, sonic games are ok but they lost their original touch; the platform game feeling has been lost when the game became 3d, and most of the concepts of the game changed. Changing to 3D is not necessarily a bad thing, but the change made did make it less Sonic-like.

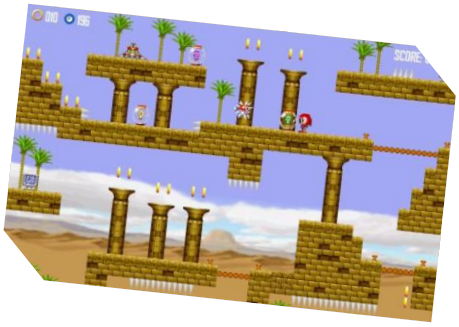
When I downloaded the game and played it feelings came back, I could clearly remember playing against Robotnic in Sonic 1 and 2, I remembered the excitement and finally I remembered the love I had for that hedgehog.

The Ratings

Graphics

I really doubt the game had hard work done on the graphics; as most, if not all of the graphics, were actually ripped. However, I did find that the sprites in this game were relatively better than the general quality level of the 2D sonic sprites available out there. Ripped graphics is obviously a bad thing, but what also made it worse was the fact that no extra work has been done on the graphics, to e.g. make the sprites smoother or more realistic.

For graphics I give it a 6/10



QUICK REVIEW

Force Feedback

The Force Feedback DLL is a great dll; why, you ask? This dll can enable the use of gamepad rumble and joystick forces.

So for any of you who are targeting make a game that requires a Joystick/Gamepad then here's a good way to keep the player attached: make the gamepad rumble and shake just like Dual Shock and Dual Shock 2 gamepads shake for the PlayStation console.

Get it now!

http://xrl.us/forcefeed

Game play

As I have been saying this game is a return of sonic as we all liked and grew up to love. This game is the same yet different; it still has the same basic concepts that the old game had, but it builds upon these concepts; you need to rescue the "Flickies" in order to go to the next stage. I have no doubt that the level designer has put a lot of effort to create such a new form of sonic game play. It was the same old concept but taken one step further.

For game play I give it a 9/10 an outstanding game indeed.

Sound effects

Sounds made at jumps taking the rings or hitting an enemy are a lot like the old sonic if not the same, which is an advantage especially to those of us who have enjoyed sonic the way he was. Even though the sounds themselves are good, I would've personally preferred more unique sounds (made for the game).

For the sound effects I give it 5/10.

Story

The story is the typical sonic freeing a creature story. To be honest it could have been more elaborate but that would have made it lose its "Sonic-*ness*" this is an old school type game giving it a complicated elaborated story would defeat its purpose.

So for the story I give it 8/10.

Music

The music is not that good for a game such as this. A sonic game's music should be stimulating this game has failed in this aspect, the music represents the theme of each stage but it does not stimulate the excitement appropriate for the game.

So for music I give it 6/10.

Overall

This game receives in total 6.8/10.

Conclusion

In conclusion this game has lost it overall rating because of music and graphics, but I do salute the designer for this game and if he is thinking of a sequel sonic game please work on the music making it more exciting.

It all starts here,
Michael Sharaiha■

Setbacks and Going Forward

Here at MarkUp, we realize and recognize the concerns and criticisms that have been directed towards reviews included in the magazine, and, starting from the next issue, we promise you'll start noticing a review overhaul. The selection of games reviewed and the types of games reviewed will be different, so that only games which are interesting and beneficial for developers to know about will be reviewed.

Not only that, but the existing format of reviewing games will be changed radically and improved so that the whole game is described and reviewed, and not just the few specific sections of the game. Numerical ratings shall remain, but will be based on group votes instead of opinions of a single reviewer. All game reviews will adopt a unified style and format.

Not only that, but you'll also notice new types of articles that were never present in MarkUp before, starting next issue, we'll have the new and fabulous **MarDar** – the MarkUp Game Radar that reviews, in a few sentences, new games worthy of your attention. Also an all new section is **The Making of My Game** where we'll have a developer talk about the methods, strategies, ideas, and programming methods that they used to make their high quality game. You can help improve MarkUp Magazine and shape up the next generation of Game Maker game development by a number of ways:

You could apply for staff for a reviewer position at MarkUp here, and start working on the all new reviews format and contents for the upcoming issues of MarkUp.

You could simply join the GMking.org Network forums, and participate at the MarkUp forums in there by giving suggestions about future issues and issues in development.

You could comment in the MarkUp site about all newly released issues and give us suggestions from there if you don't want to register at any forums.

And finally, you could jump in the GMC discussion topic about MarkUp to give feedback, discuss and debate issues with other members at the GMC, and therefore help make MarkUp a better magazine by letting us know your opinions on the matters being discussed.

Eyas Sharaiha■

Good Bye!

And that's it, our seventh issue of MarkUp Magazine – the September 2007 issue – ends here. The number seven celebrates more than one thing; other than being the lucky number of many people, the seventh issue means that MarkUp Magazine is now the Game Maker Magazine with the most amounts of released issues, past and present! So congratulations to everyone who contributed to MarkUp for breaking this record – others might follow, but remember, we got here first!

This issue is not as content-intensive as Issue 6 was, but yet it achieves a good amount of articles and content in general. The reason for this month's 'decline' is as mentioned in the editorial: summer! Issue 7 marks the beginning of the school year for some of you – so for those of you whose school year started, good luck on your studies! I won't say anything too direct, but I'll just say the next issue of MarkUp is going to be special in many ways, so I hope you stay tuned to our news via the MarkUp site, forums, and GMC thread. It's also exciting to note that we have crossed our half-year anniversary last issue, and we are now getting closer to Issue 12, our full 1-year anniversary issue!

From now till that time, we plan on growing considerably; getting more writers, supporters, and readership to achieve an amazing twelfth issue and hopefully many more to come.

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...

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.

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!