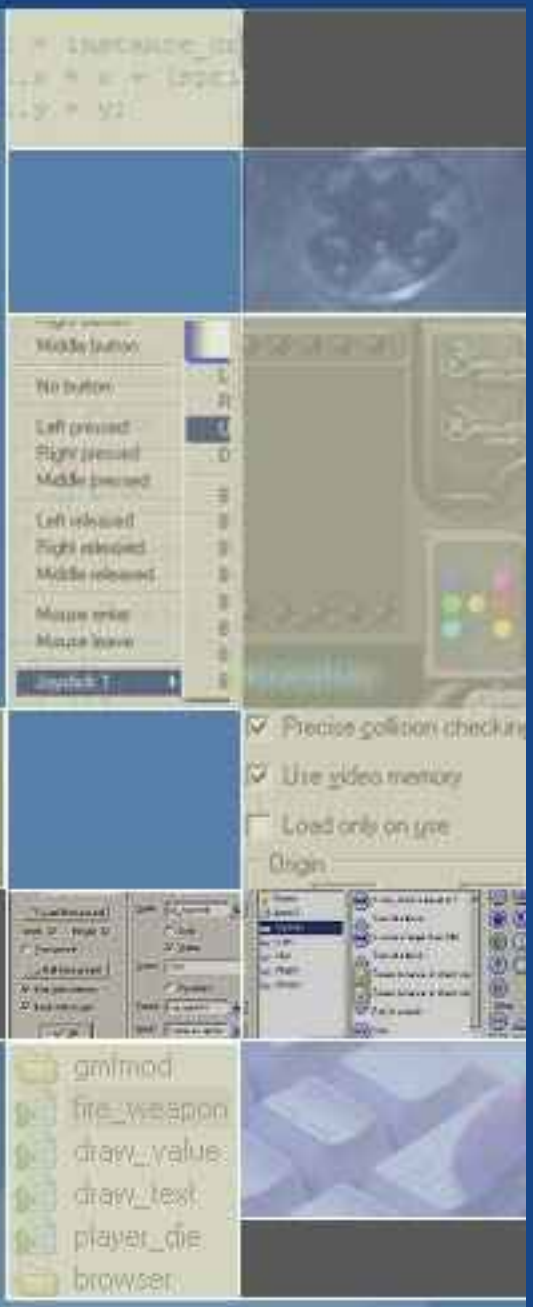
game reviews  game development  interviews  code  tutorials

# GM data mag

morphosis enter-active



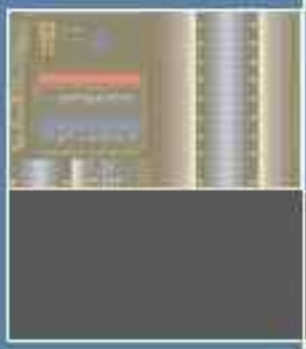
# it's alive

## Game Maker 5

1st Look at the New Game Maker 5
You got to see and read it!

## 3-D game design

Check out the top 30 programs





## 10 tips to follow

10 Steps to Great Game Design

## good game?

What makes a game good? Read this by Mark Overmars

## reviews

Aliens Attack On Colony - Jetz Rampage - Ore no Ryom

GMDM Issue #2

# Contents



## Adobe Acrobat Tips

By: Morphosis

Ctrl L- "Full Screen"
Ctrl +- "Zoom In" - Ctrl - - "Zoom Out"
Home - "First Page" - End - "Last Page"
Arrow Keys - "Page Down or Up"
V - "Text Select Tool"
H- "Hand Tool"

## Credits


**Morphosis Enter-Active**
**Morphosis Games**
**© John Hempstead**
**http://mea.gmcommunity.com/mea**
**morphosisgames@aol.com**

Edited, Designed, Written and Produced by:
John Hempstead-Morphosis
Morphosis Enter-Active (MEA)



Other contributions to GMDM © 2003 include and get a big THANKS!
Mark Overmars
Chris Spicer
Allen Cheung
DT
C.E. Forman
Curtis LeMay
Simon Danaher



Look! More Room here for you so help out! Email me for information. Thanx

# From the editor

Morphosis: John Hempstead

Welcome to the 2nd issue of Game Makers Data Magazine. I am very excited about this magazine and it's success. Reports from Mr. Overmars state that in a few weeks the magazine was downloaded over 2000 times. That's great but since it's getting a large audience, who many of them noticed my errors writing this magazine, forced me into being more aware of my mistakes. I did rush to put the 1st issue together and rush to complete this 2nd issue. Because of this, I have decided that the magazine will not be released every month as intended, but every two months. That way I could plan better, check mistakes, give people who can't make the deadlines more time, and make better decisions of what is in the magazine. This may change if I do get people who are dedicated to their work. I don't think a two month wait is that bad, but it's not what I want.

I would like to thank Mark for his decision to place this on his site, it's an honor. I would also like to thank all that have read the first issue and had made comments on it. I hope you continue to enjoy this magazine and remain looking forward into the next and future issues.

Morphosis
John

# News Flash

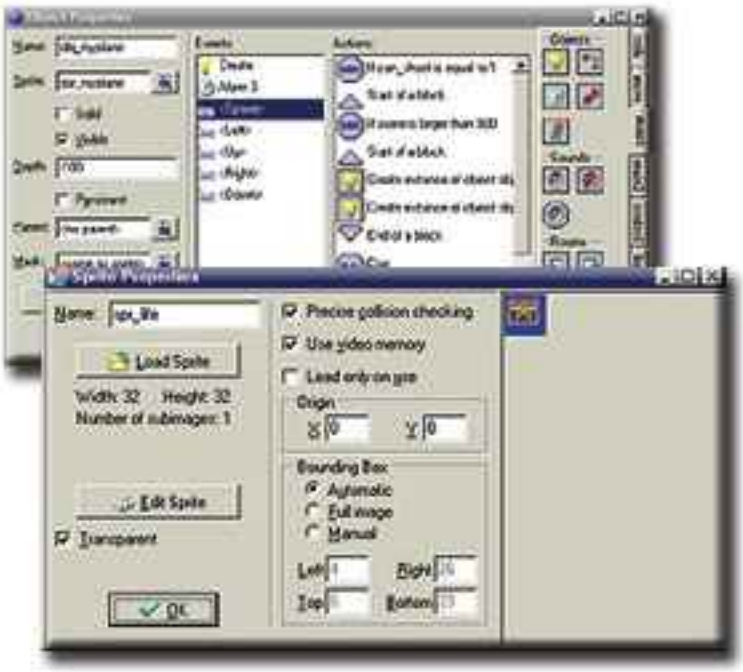
Morphosis: John Hempstead

## Game Maker 5.0 Beta Released

Mark Overmars has done it again, it seems like he will never stop because improvement is important to him. He shows this with the most recent development of Game Maker 5.0.

This is exciting to all game makers, as other releases were. The big change that may have shocked GM users was the release of Game Maker 4.0 due to it's almost completely new appearance and power. Game Maker 5.0 was thought by some to only be small minor improvements and bugs that were in GM 4+ to be fixed. But it is not just a few changes, it's many changes and many new additions. Make sure you read my review of this greatness.

For more information about Game Maker: http://www.gamemaker.nl/

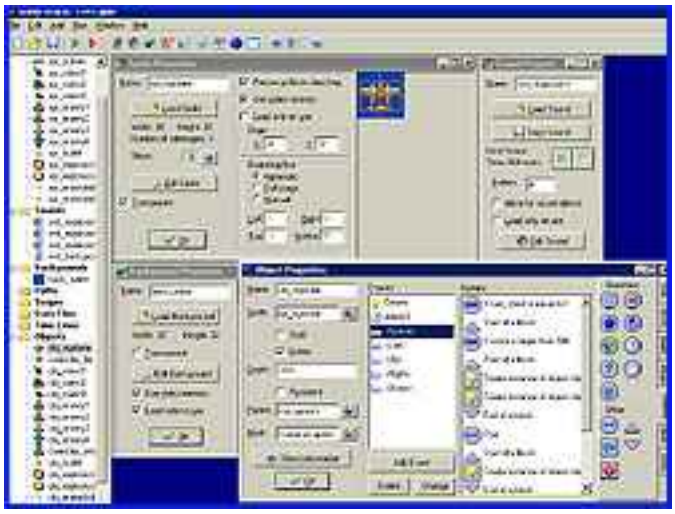



## Game Makers Data Magazine Help

If you are interested in writing for the GMDM please feel free to drop me an email. From there I can send you some guidlines to follow. I am open to almost anything that deals with the developement of games, web, programming, multimedia, and art. Even some humor can be thrown in that may make us smile. So don't wait on your ideas, send questions to Morphosisgames@aol.com.

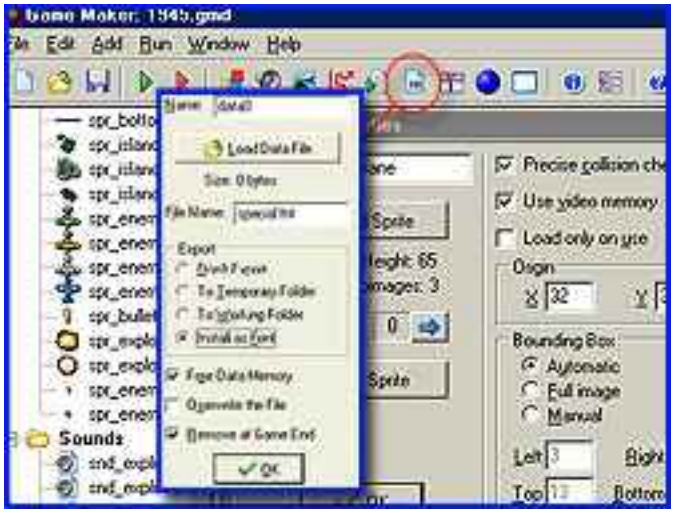
Morphosis: John Hempstead

# It's Alive - Game Maker 5



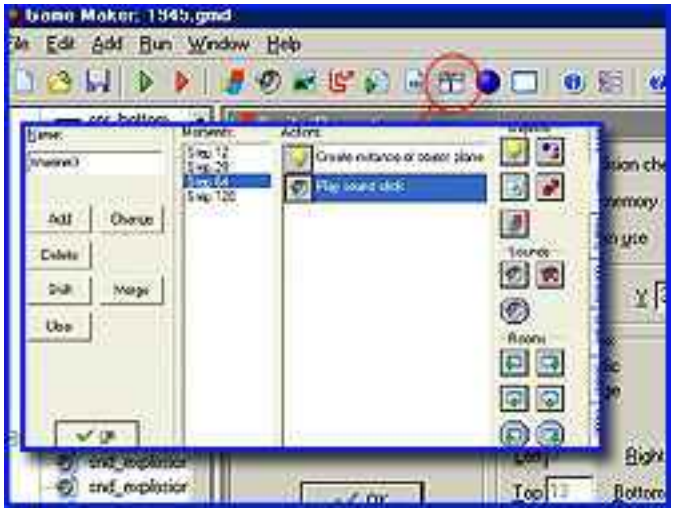
One of the first things when you run Game Maker 5 is it's XP styled graphics. This is a nice change that was done really well. The icons are clear which makes it easy to identify it's meaning. Even though the icons changed, some still look like the old GM illustrations, with a little xp touch up, and some icons are brand new  icons like the comment, rooms, time line, score, lives, and health. Most of these icons are brand new features in GM 5.

New graphics for GM 5 are not really needed to create a great game, but it was a really nice addition. The main GM icon is still the red ball and it's preloader remains to be the one that I had created. But don't forget, these can be changed in your game.




Two new and interesting features in GM 5 is the "Data File" and "Time Line"

**Data File:**
In more advanced games you often need to use additional files, for example files that describe certain properties, backgrounds and sprites that you want to load during the game, movies, DLL files or your own fonts. You can distribute these files with your game but it is nicer to embed them in the game itself. For this you can use data file resources. A data file resource simply stores the contents of a file. When the game starts this file is written to the disk and can then be used in the game Data files can take a lot of memory but loading and exporting them is fast.




**Time Lines:**
In many games certain things must happen at certain moments in time. You can try to achieve this by using alarm events but when things get too complicated this won't work any more. The time line resource is meant for this. In a time line you specify which actions must happen at certain moments in time. You can use all the actions that are also available for the different events. Once you created a time line you can assign it to an instance of an object. This instance will then execute the actions at the indicated moments of time. Let me explain this with an example. Assume you want to make a guard. This guard should move 20 time steps to the left, then 10 upwards, 20 to the right, 10 downwards and then stop. To achieve this you make a time line where you start with setting a motion to the left. At moment 20 you set a motion upward, at moment 30 a motion to the right, at moment 50 a motion downwards and at moment 60 you stop the motion. Now you can assign this time line to the guard and the guard will do exactly what you planned. You can also use a time line to control your game more globally. Create an invisible controller objects, create a time line that at certain moments creates enemies, and assign it to the controller object. If you start to work with it you will find out it is a very powerful concept.


**Advanced and simple mode:**

Another interesting new feature is for the user to switch from an advanced mode to simple mode. I could remember some people saying that GM 4 was hard to use, not this is solved. Even if a game was made in Advanced mode, it still can be used in simple mode. That way samples and tutorials should not be effected. As you can see in the right image, simple mode compared to advanced mode.

You can select the mode in which to use the program. In simple mode many resource types and options are made invisible, making it easier to start using the program. In advanced mode it is easier to access the advanced options.


Other changes inclued buttons that let you draw the value of the variable, draw the value or image of lives, draw health value, set the room caption, mouse enter/leave options, 2 joystick control options, and I am sure a lot more!

GM 5 is a top notch program to create games and even presentations, movies, and other multimedia tools. It is free but you should register it to help the development of future versions of game maker.

“Game Maker can be used free of charge. There are no restrictions on the games you create with it. The games will show no nag screens and you can even sell them if you like. See the enclosed license agreement for more details.

Developing Game Maker though does cost time and money. If you wish to support the development of Game Maker or wish to disable this nag screen, please register Game Maker. In the future there might be other benefits for registered users.”

And last, GM 5.0 release is the beta version and should be used for bug testing for the final release in April.

A small test on the 1945 game between Game Maker 4.3 and 5. Tested on Windows ME. 256 megs of RAM. 933 P3.

| GM TEST | GM 4 | GM 5 |
|---|---|---|
| Opening game in GM | 00:87 sec | 01:15 sec |
| Game Testing Time | 04:00 sec | 04:09 sec |
| Creation of .exe | 01:08 sec | 00:82 sec |
| Loading Time of .exe | 02:43 sec | 00:82 sec |
| .gmd file size | 215 k | 97.1 k |
| .exe file size | 1.35 meg | 1.29 meg |

# Tutorials

By Allen Cheung

## Guide to Good Programming and Game Making Practices with Game Maker. Part 2

Continued from issue 1

**Examples**

Since scripts are so important, I want to present a few examples as to good script programming. Perhaps the best way to practice this is by writing scripts by themselves as I am doing here; you will be forced to remove yourself from the context of a program and thus generalize your scripts. Feel free to copy and paste them into your code, using them as per the comments.

```
// Returns the square of arg
{
  return argument0 * argument0;
}

// Concatenates two strings, returns it
{
  return argument0 + argument1;
}

// Moves object arg1 spaces on a diagonal
// Arg0 is the object, Arg1 specifies distance
// Arg2 is direction – 1 = NW, 2 = NE, 3 = SW, 4 = SE
{
  localx = argument0.x; locally = argument0.y;
  if (argument2 == 1 || argument2 == 3)
    localx -= argument1;
  else localx += argument1;
  if (argument2 == 1 || argument2 == 2)
    localy += argument1;
  else localy -= argument1;
  argument0.x = localx; argument0.y = localy;
}

// Find the angle between two objects, in degrees
{
  disX = argument0.x – argument1.x;
  disY = argument0.y – argument1.y;
  tanRadians = tan(disY / disX);
  return tanRadians * 180 / (2 * pi);
}
```

I may create more scripts later if demand is great, but hopefully these few examples will serve to show the clarity that a good script can provide.

## 6 Variables

It is fair to say that a lot of programming and video games revolves around variables and their usage. They are what make programs interactive – the user can input data, have his input stored as variables, and the program outputs meaningful data. As such, it is worthwhile to explore and explain the value and good use of variables.

**Data Structures**

What are data structures? They are merely types of variables; for example, while **S** may be a variable, the type string can be considered a data structure. As explained in Section 2, General Concepts, GM provides three easy basic data structures: strings, real numbers (reals), and booleans.

**Arrays**

One important data structure that GM provides is an array, up to two dimensions. An array is a collection of similar data structures that can be easily accessed; it is an orderly progression of data. They are declared as follows:

Arrayname[number]

So that arrayname is name of the array, and number is the counting number (starting from zero) that contains a variable, also known as an index. In a way, an array can be thought of as a row of variables:

[0] [1] [2] [3] …
15 178 3 84

In the example above, arr[0] would return 15, and arr[3] would give back 84.

The reason arrays are useful is that they can be accessed via simple counting, which is perfect for loops. By going from 0-9, for instance, one can quickly go through 10 similar variables and assign values to them. Consider the code:

```
{
  s = ‘’; // ‘’ is the null string, kind of like the number 0
  for (n = 0; n <= 9; n += 1) {
    s += chr(n + ord(“A”));
    arr[n] = s;
  }
  for (m = 0; m <= 9; m += 1)
    draw_text(10, m*10+10, arr[m]);
}
```

Looks intimidating, but it really isn’t so bad. All the code does is first set the string variable s to the null string, then enter a loop that repeats ten times. Each time, s adds another letter to the end (remember that s += n is really shorthand for s = s + n), with that letter being the next letter in the alphabet. (a quick explanation: ord() gives the numerical ASCII code of a character, so adding to that code will produce further letters down the line, and chr() will convert that number back to a character) It then stores that string into an array arr, which will have 10 elements (0-9) by the end of this loop. The last loop will then print the contents of that arr, which should be:

A
AB
ABC
…
ABCDEFGHIJ

With correct spacing from draw_text(), of course. The point here was that because of the nature of arrays, you can easily run through the entire thing without inventing ten variables to store ten variables that differ only by their value.

GML also allows the use of two-dimensional arrays, formatting as follows:

TwoDimensionalArray[num1, num2]

Where num1 and num2 are the two indices of the array. Just as a 1D array can be thought of as a row of data types, a 2D array is merely a table (rows and columns) of data. Since GM is in the business of creating 2D games, such arrays may come handy in, for example, representing the contents of a nxn grid (in a sliding puzzle game). Unfortunately, this is as high-dimensional as GM goes, but should be enough for all purposes.

**Arrays of Objects**

An excellent use of arrays would have to be in storing similar objects and instances. GML itself already provides one such grouping for you – instance_id[n]is an array of all the instances of an object. You may also manually store instances as a part of an array as well – for example, in an RPG with a large party, one can use something like:

```
…
wholeParty[0] = objHero;
wholeParty[1] = objEvilHero;
wholeParty[2] = objNinja;
…
```

Then, when the time comes to do something to the entire party (like taking damage from a mass-destruction spell), we need not stress ourselves over the code:

```
for (n = 0; n < numPartyMembers; n += 1)
  with (wholeParty[n]) {
    damage = random(defense) + enemyMagic;
    HP -= damage;
  }
```

And voila, our entire party has just taken damage from the spell, where each individual has calculated his/her own damage levels according to their own stats. Mark also notes that with instances of the same object, they are already stored in an array (the instance_id[n]above is that array), and that to cycle through all the instances, the command with is provided. More on this command can, of course, be found in the official manual.

Constants

We now move onto the discussion of constants in a program. Constants are global variables that are declared in the beginning of a program; they are there usually as general bounds to certain parameters. Taking our RPG example above, we can set the maximum damage to no more than 9999, and max health the same number in the beginning of game:

```
global.MAXDAM = 9999;
global.MAXHEALTH = 9999;
```

Then in your actual code:

```
if (HP > global.MAXHEALTH)
  HP = global.MAXHEALTH;
if (damage > global.MAXDAM)
  damage = global.MAXDAM;
```

Programming naming convention usually gives constants all capitals. The point here is that by declaring these variables in the beginning, you can easily look up what the numerical values are. Furthermore, if your code uses a certain constant multiple times, when time comes to change the number (from play-testing, perhaps, when you find that 9999 isn't enough health), all that is required is a quick change in the beginning of the code, and everything else will fall in line.

**Variable Usage**

As you begin to make more advanced games or simply just more games in general, you will begin to realize that not a lot of numbers need to be explicitly given (this is not true for strings, however) – explicitly using a "magical number" is known as "hard-coding". You will also begin to notice that your program will behave just as well with one value as well as another, given that your data structures and overall organization is reasonably solid.

You can take advantage of this by providing the gamer with options. From experience, most games created in GM tend to be started linear – one loads the game, reads the introduction/instruction screen, then one begins the game itself. Unlike most professional games, an options screen is not provided, when having one does characterize quality in that game. They do not even need to be tedious; a platform game, for example, can have the player choose between easy, medium, or hard, and assign lives 5, 4, and 3 respectively to the gamer. Or he can up the speed of his monsters with increasing difficulty. Perhaps the main character will walk slower/faster, and jump lower/higher. The numerical nature of games and programming almost begs for manipulation, so why not add a professional polish to your game?

**Global Variables**
GM allows for global variables in the form of:

```
global.var = x;
```

Where the keyword global signifies that the variable can be used, called, and changed anywhere in the program. While this certainly sounds great, the old adage "too much of a good thing will ultimately reduce you to the shadow of a man you once was and make you beg for mercy" (paraphrased) is certainly applicable. In programming, overuse of global variables is shunned, and GML is no different in this regard.

While GML avoids the issue of namespace (it's just a fancy term for the available names that you may give your variables…makes a difference in complicated and large programs) by requiring the use of global, the practice is still bad because of the potential to make your program less object-oriented (this is explained in further detail in the next chapter). By all needs, you should not even

need global variables – **controller** objects (also explained in the next chapter) should be enough to keep track of all variables that are needed in your game. In other words, they are provided as a convenience, and there is often the temptation to use them boldly to avoid otherwise clean code (for example, to avoid passing arguments to scripts by setting a few global variables and changing those). They are not completely horrible, per se, but extensive usage simply shows that you have organized your program poorly.

**Custom Data Structures**

As mentioned above, it does not appear to be the case that one can make one's own data structures in the context of GML. However, there are ways around this, though they may not be so obvious to the casual GM user. The general idea is to create a custom **object** that houses all the data that you need.

You may be at a loss to figure out why anyone would need such a structure, so let's give a quick example. Suppose that you want to write a script that returns two distinct strings. Normally, this is impossible, as a script can only have one return value. Hence, you would need some sort of container or data structure to house both strings. The answer? Create a dummy object that has two local variables of strings, then in the script, stuff those two strings in there and return that.

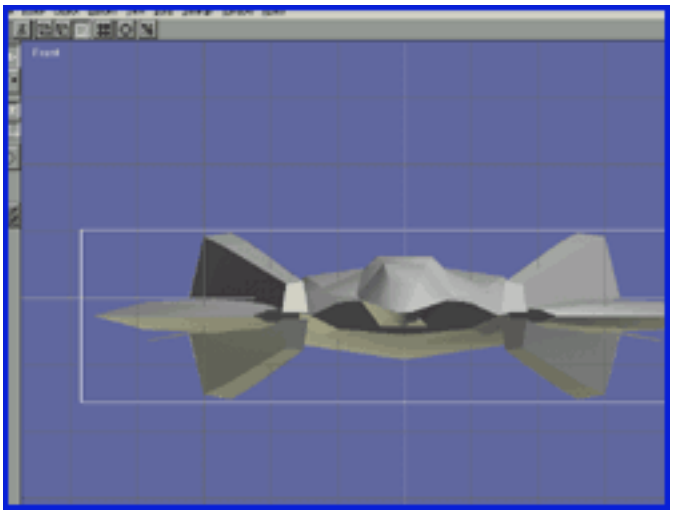
For the most part, however, you probably will not need these custom data structures. As Mark has so carefully informed me, GM is not based on dynamic scope, but rather seems to be more lexically scoped (for those that don't have any idea what I'm talking about, ignore it), so variables tend to stick around, it seems. In any case, this little trick is here for those that would use it.

**Continued in next issue.**

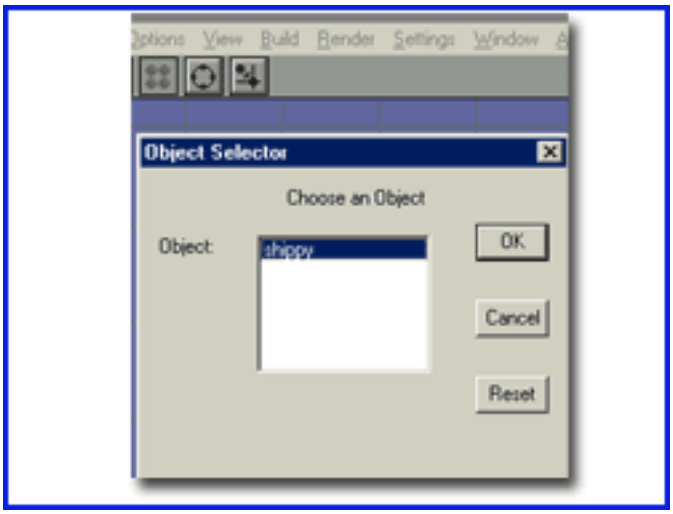
# Tutorials

## 3D Sprites with Anim8or and Game Maker

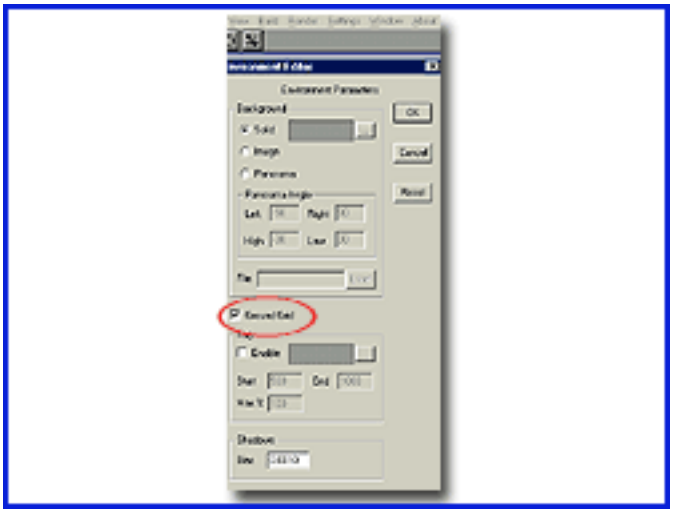
It first may take some time to create a 3D animated image, but I am sure in time your images/sprite will be really cool. You can use for this tutorial a simple box rather than making a ship, as in mine. But if you would like to make a ship, go and do it. When you are ready, have your object in anim8or and ready to go.



With your object in Anim8or, we now must place it in the “Scene”. So, go to “Mode” and select “Scene”

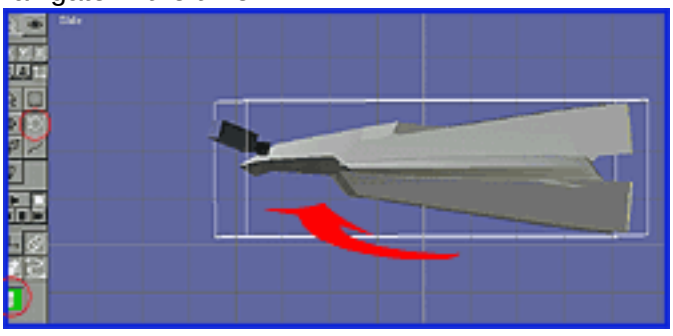



Now when in scene, select “Build” and ad your object in the scene.



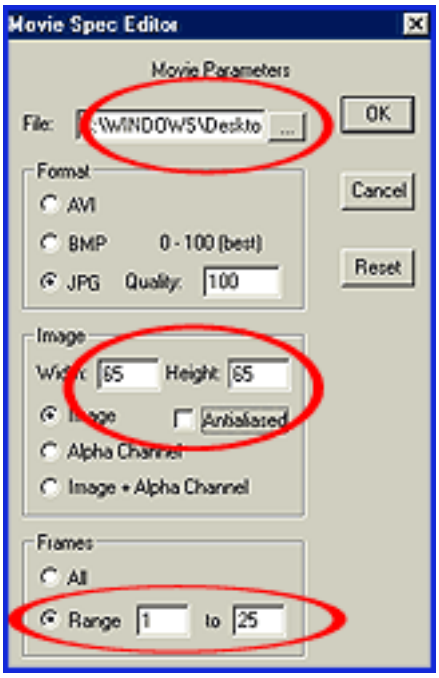
Now, to get rid of the floor, since we don’t want that as part of the sprite, select “Settings” then “Environment” and uncheck “Ground Grid”


This is the keyframe area, by clicking on your object you set a key frame. By clicking in the time line you navigate in the time.



Now look at what needs to be checked when you begin to animate the object. It may be a lot easier if the rotation is done from the side of the object. And the goal is to get it to rotate 359 degrees. That way a 360 can loop and look smooth.

Just set your 1st  with the object like this and go to the 25th  and rotate the object almost 360 around.




## About Rendering

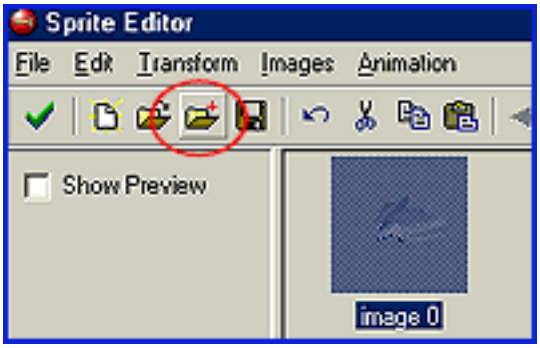
**You can render your images as image files and import them into Game Maker one at a time. Or you may have some type of software such as Ulead Gif Animator 5 which lets you add all or as many images that you want and then lets you save it as a gif. You also can render out an avi file and import it into a program (Ulead GA5) and save it as a gif.**

**As for the Alias and Antialias, when this is turned on and brought into Game Maker, the edges of the images may not be transparent. Antialias blends the edges of the images to make it smooth. It's hard to make these smooth areas transparent.**

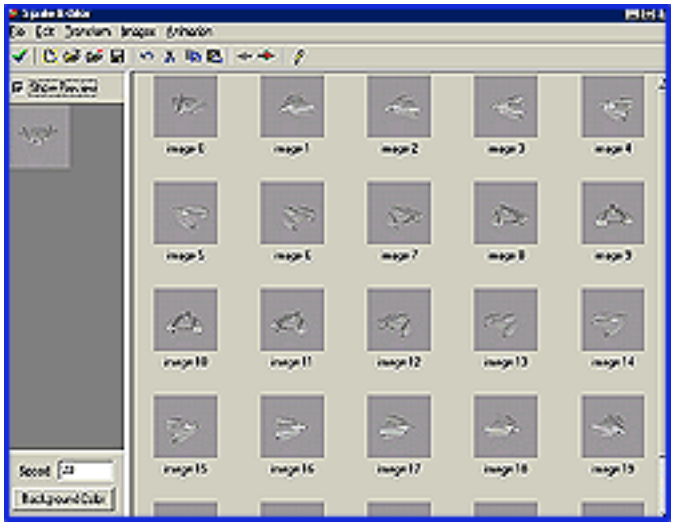
**But then, you could use the Alpha Channel and have the Antialiased turned on only if you have some program that knows there is an alpha channel (which is the object itself). This means, anything around the object will be "Keyed" or transparent.**

**I have done all my sprites with antialiased turned off, so that may be your best bet.**

Now it's time to render away. Before this select the view you want to render. It may be perspective, side, front... Now go to "Render" and select "Render to file" Look at the options in the image above. Select where you want to save your file, selection of the image size is up to you, I would make it small for now. And also make sure "Antialias" is unchecked.




Add from file in Game Maker.



Final sprite.

In Game Maker:
In the objects create event:facing_direction=90

In the step event: image_single = facing_direction/10

Left: facing_direction = 90

Right: facing_direction = -90

```
{
        if facing_direction < 0
        facing_direction = 360
}
```

end

# Development

By C.E. Forman

## 10 Steps To Great Game Design

You've solved every Infocom game ever released. You've FTP'd countless text adventure games from Internet sites in a desperate attempt to quench your insatiable thirst for interactive fiction, but still it's not enough. So you decide to take the final step, to write your own parser adventure. But--how do you know for sure that people will like it? How can you avoid making the same mistakes you've seen in many of the quests you've been playing for years? What exactly constitutes a "good" text adventure game?

That's what I'm here to help you with. I've taken it upon myself to analyze my favorite works of interactive fiction, determine why they're my faves, and compile a list of their common characteristics that first-time adventure writers can use for reference.
Keep in mind that this is not an article on programming a game. These ten tips deal exclusively with game design and the authoring of the game's storyline. My intent here is to point out the most common mistakes beginners make, and identify methods of avoiding falling into these traps.

### 1 Develop a good parser.

This is the single most important element of any work of interactive fiction. Unfortunately, it's also the one most frequently neglected by beginners. Even the most cleverly designed adventure isn't going to hold players' interest for very long if they have trouble communicating what they want to do. The earliest adventure games, such as the original "Adventure in Colossal Cave" and the Scott Adams series, used crude, verb-and-noun parsers that accepted only two words in each command. Due to the limitations of computers in those days, a standard parser's vocabulary was often very limited, leaving gamers dissatisfied.
The Zork Implementation Parser introduced by Infocom in the late 1970s is really the accepted standard for parsers today. If you're using an IF design tool such as Inform or TADS, developing a good parser isn't as much of a problem. If, on the other hand, you've decided to write your own parser, pick yourself up a good thesaurus and use several common synonyms for each noun and verb. Make your puzzles the challenge of your adventure; don't force players to "guess the verb."
In addition, the more options you can supply your parser with, the better. An "undo" command, built-in hints, the ability to allow players to configure the function keys as typing shortcuts, and automatic mapping will all contribute to the reduction of frustration on the part of the player.

### 2 Good puzzle structuring.

Don't just force players to wander aimlessly from one puzzle to the next, halting their progress completely until they solve the only available puzzle. Branch out your puzzle structure and make it as nonlinear as possible. Interweave your puzzles with one another and allow players multiple paths through the adventure. That is, don't make players solve the puzzles in the same order every time; give them some flexibility. The only point in the game where there should be only one path for the player to follow is at the conclusion, where all the branches of puzzles come together to form a final challenge.
Puzzle connectivity is also important. Make sure each puzzle "fits in" with all the others. If you have an extremely challenging puzzle, but you can't make it fit logically into your adventure, don't just throw it in for the sake of using it. Save it and use it in another game, where it is appropriate. One of the biggest abuses of this that I've seen comes in the form of mazes. Often adventure writers will simply throw in a maze to make the game more difficult, when in reality it is totally inappropriate and has nothing to do with the game at all. Making maps of games is tedious, and mazes are generally frowned upon in adventure games today, unless they have a truly unique twist (such as the catacombs in "Leather Goddesses of Phobos"), or can be solved without mapping (such as the wet tunnels in "The Lurking Horror"). In summary, designers should ask themselves this: "Is this puzzle connected to the game in some way, or is it in the game merely for the sake of its own existence?" If it's the latter, you should probably consider scrapping it.

## 3 Diffulculty of puzzles.

What's the fun of getting all the way to the end of an adventure only to discover that the final challenge is the easiest puzzle in the history of the universe? In most cases, the best adventure games are the ones that curve the difficulty of their puzzles. Keep 'em fairly simple at first, to allow the player to get into the game, then gradually raise the challenge as players go deeper into it. Don't get me wrong. It's perfectly okay to throw in a difficult or obscure puzzle or two in the early stages of the game, but the key is not to overwhelm the player at the outset.

Of course, the primary deciding factor as to the difficulty of the puzzles should be how difficult you've chosen to make the game as a whole. If you're writing for expert players, design your puzzles accordingly. If beginners are your target audience, include a lot of simple, one- or two- step puzzles. In all cases, after a particularly arduous puzzle, reward the player with a few simpler ones. You'd be surprised at how many players lose interest when a game's puzzles aren't balanced.

## 4 Good puzzle structuring.

This one is basically just common sense, but it's still sometimes overlooked. If you're writing a sci-fi adventure, pay attention to the laws of physics. Don't let players enter the vacuum of space and survive without spacesuits. Realism is less of a problem in fantasy games, as much can be justified by the use of magic. The point, though, is to make sure everything makes sense in some way. This is especially important in the area of puzzles. Avoid making players do things that have no logic or purpose behind them. The more realistic your adventure, the more it will draw players in.

Resident Evil

## 5 Be Discriptive

You've created a whole other world, so why not let the player enjoy the beauty of it? How many times have you played a game with such lame location descriptions as "You are in a forest," "You are at the bottom of a tall cliff," "You are outside a cave," etc.? The term "interactive fiction" is not an arbitrary one--players are essentially exploring a form of writing, much like a good novel, and adding their own input to it. So let the player see the world you've created, much like your favorite fiction authors let you see theirs. Take care, though, not to overwhelm your players with prose. If you give them little opportunity to interact, they just might decide that they may as well be reading a book. Don't get too bogged down in descriptions. Usually half the screen is the absolute maximum for a room or object description, and this limit should only be reached on rare occasions. However, if a particular puzzle requires a lot of text in order for the player to see it, one or two full screens are acceptable. (A good example of this case is the mirror box in "Zork III".) Don't make players read the text over and over unless they want to, though. Make sure your parser has the option of changing the length of room descriptions. Using phrases such as "You are in the forest" the second time a player goes there is perfectly acceptable. (Just make sure that players can still get a better description if they want it.) While we're at it, I'd like to mention one variation on this subject. Most players, when writing good room descriptions, like to include several objects or features in each location (for example, a tavern might have a fireplace, a bar, and several tables and stools). Nothing is more aggravating than typing "EXAMINE THE STOOLS" only to be told, "I don't know the word 'stools.'" This is guaranteed to instantaneously shatter the fantasy and destroy any hope of players ever really getting into the game. Do this enough, and you'll alienate them forever. If you're going to put an object in the location's description, you'd better let the player interact with it, even if it's only in a limited way. Just a message saying, "There's nothing special about the stools." will suffice. Incidentally, I feel that this is one of the biggest problems with the Zork-based MUDs I've played. Players see that term, "Zork-based," and they telnet in expecting the same level of realism that Infocom gave us, and unfortunately, they rarely, if ever, get it. I myself have on occasion experienced difficulty in simply trying to interact with what the game claims is in the scene with me, and I'm afraid this is the rule rather than the exception.

## 6 Be Fair to the player

I know, I know. Life isn't fair. Never has been, never will be. But adventure games aren't real life; they're a form of entertainment. And the only way players will be entertained is if they're treated fairly. Here are some general guidelines you should follow to ensure that this is the case:

Don't let the game get into an unsolvable state too much, without giving the player some indication of it. There's no definite line here, so you'll have to use your own judgment. As an example, consider the KULCAD scroll in Infocom's "Enchanter". (Warning! Spoiler to follow...) You are supposed to use this spell to dispel the illusion of the infinite winding staircase, as this is the only way to overcome this particular obstacle. However, you can also use KULCAD to get rid of the guarded doorway and the Gordian Knot around the jeweled box. If you do one of these, though, the scroll is gone and you can't win the game. Despite the frustration that could be caused by this, I still don't consider it unfair, because if you use the spell on anything other than the stairs, your master Belboz will appear before you and warn you that the evil Krill has been alerted to your presence. You receive a definite hint that maybe there was a better way. On the other hand, I've heard numerous complaints about the game "Curses" because you can inadvertently do something out of sequence and blow any chance of being able to win, and no indication whatsoever is given. So save yourself and your players a lot of trouble, and give some kind of message if they unintentionally do something to get themselves stuck. (On the other hand, throwing all your possessions off a cliff is not a very smart move to begin with, and players should be able to figure this out without a hint. Only tell them they've screwed up big-time if there's a chance they can't determine that for themselves.) Incidentally, a good way to allow players to keep going after they've lost an item is to allow them to re-obtain the item in the place they originally got it. For instance, allow them to pick another apple off the tree if they eat or lose the one they originally got. However, when you're dealing with a unique item, such as the KULCAD scroll, this isn't a feasible option. Again, you'll have to use your own judgment here.

Don't force the player to have too much foresight. Inventory management is a crucial part of an adventure game in which the number of things a player can carry is limited. Often players will wander into a new location and only be able to take so much along with him. Obviously, if they're going into a dungeon they'll need a light source, a weapon, and probably some food and water, but if they're going to need something less obvious, you'd be wise to provide a hint beforehand. Of course, players can always restore, but going through a lot of moves to get back to where they was before can be frustrating, and too many save files can become difficult to keep track of. It's best to give players a general idea of which items they won't need and thus can leave behind. Don't make them pick and choose too much. A few very good games I've seen are designed so that the player's inventory is pretty much managed as the game progresses. That is, the player uses two items to solve a puzzle, thus removing them from his inventory. Then she finds another object and adds it, and later gets rid of it in another puzzle, and so on. If you have items that don't have multiple uses, this is a good technique to use.


Resident Evil Inventory

Don't overwhelm players at the start. If you have a large area they can explore in the beginning, you'd be wise to point them in the right direction to start with. Legend Entertainment's "TimeQuest" is a perfect example of this. With 80 different time-places to visit, any of which can be reached from the beginning, it's vast enough to make the player feel burdened. But this game directs you to Rome in 44 B.C. from the start, which gives them a sense of direction and helps you establish a path through the game. In addition, the most crucial time-places are all listed in the game's documentation, so the player knows where the important places are. This makes it easier to get started.

Don't put off the entire reward until the end. Congratulate players when they solve a difficult puzzle, possibly by giving them a special item or power. According to Joseph Campbell's monomyth, the challenges a hero faces become more and more difficult as his quest continues, but the rewards become greater. This should apply to adventure games as well.

Don't create puzzles that absolutely have to be solved within a specific time frame, unless you give the player a reasonable hint.

Include good error messages in your program to tell players if they're doing something wrong, but don't insult players in the process. Be clever, but not verbally abusive.

Random events are good for spicing up adventure games, but never, EVER base the decision of whether a player lives or dies upon the outcome of a random-number generator. I mention this because I did it once. The player had to cross a pit by placing a wooden plank over it and then walking across. But you fell in 1f the time anyway. Talk about frustrating! The only instance where this is at all acceptable is when there is an alternative solution that is not random. For example, in Sorcerer, players have a 10% chance of successfully jumping a gorge, but if they use a flying spell, they'll get across every time.

You might be saying, "Well, this is my game, so I'll do whatever the hell I want, and I don't care whether the player thinks it's fair or not!" Keep sending this attitude, and pretty soon you won't have any players who care about finishing your adventure. Book authors who don't show respect for their readers don't stay book authors for very long. The same holds true for interactive fiction writers. Players are your lifeblood; they keep your game alive. If you want to write games solely for your own pleasure, that's fine, but you won't gain any recognition from doing it. Treat your players as you would treat a paying customer, because after all, that's essentially what they are.

## 7 Killing the player off.

At the same time, don't hold the player's hand all the way through the game. Let them experience a game death if their actions aren't clever enough. Dying is a natural part of adventure games. Can you picture what the Zork Trilogy would be like without the constant threat of being eaten by a grue in a dark place? And just try to imagine "The Lurking Horror" without death! Which would you rather see after doing something intentionally stupid in an adventure--a detailed, possibly amusing, account of your alter-ego's untimely demise? Or a lame message saying, "That would kill you, so I'm not even going to let you try it"? Believe it or not, it's FUN to try to find new and inventive ways to kill off your character (especially after you've already finished the game). Pampering players with feelings of invincibility is only going to make them severely disappointed. And besides, this is one of the few ways you can get away with murder nowadays. If you've put an UNDO command into your game, don't be afraid to let players use it.

## 8 Challenges and Story.

Puzzles are fun, but the story itself should be the main point of an adventure game. Rather than having the player wander aimlessly around solving puzzles, develop the story as the player moves along. An unexpected plot twist or the introduction of a new NPC can really liven things up, especially when it occurs in the midst of a good puzzle. Puzzles alone can only carry a game so far. Another thing to keep in mind is that a game should have a good introduction and ending. Actually, an introduction is optional. Some writers may prefer to simply have the game begin as soon as it loads, much like "Zork I", while others may choose to follow the route of "Beyond Zork" and have introductory text spanning several screens. A few games, such as "Zork Zero" and "Demon's Tomb", even have short prologues--opening sequences which play much like the game itself, but which exist for only a limited number of terms. Any of these methods will suffice. A good ending, though, is indispensable. Is it worth struggling through a game just to be rewarded with the words, "Congratulations, you win"? A good game ending should tie up any and all loose ends the story may still have, pave the way for the sequel if you're writing a series of games, and leave players feeling as though they have truly accomplished something. Good endings will be read again and again by players, but I can guarantee that a lame ending will only be seen once.

## 9 Information in your game.

Most of the time, when writing the game's introduction, you'll want to tell the players only so much about your world. Let them learn the various intricacies and details of it themselves. If your world is vast and complex, build several sources of information into your game to help the player accomplish this. You could implement an encyclopedia (Infocom uses these in several games), newspapers, a computer database, or some other form of information storage (the tape spools in Stationfall come to mind). In addition, you might want to make one or more characters act as primary information sources. The player could then ask those characters about various people, places, or things in the game. The more you tell the player about your world, the more complex and realistic it will appear.

## 10 Keep the longevity intact.

A lot of good adventure games become dust collectors after players have solved them. Often this can be prevented, or at least delayed, by a little extra effort on the part of the author. Give some of your puzzles multiple solutions. Think up imaginative ways of dying and humorous tricks for the player to try. Some adventures even have multiple endings depending on various things the player has done (or not done) during the course of the game. All of these things can keep players interested for quite some time after they've been through the entire game. The "Zork" and "Enchanter" series are particularly good examples of this tip. I'm still finding things buried in them that I never knew existed. A little extra programming can go a long way.

-End-

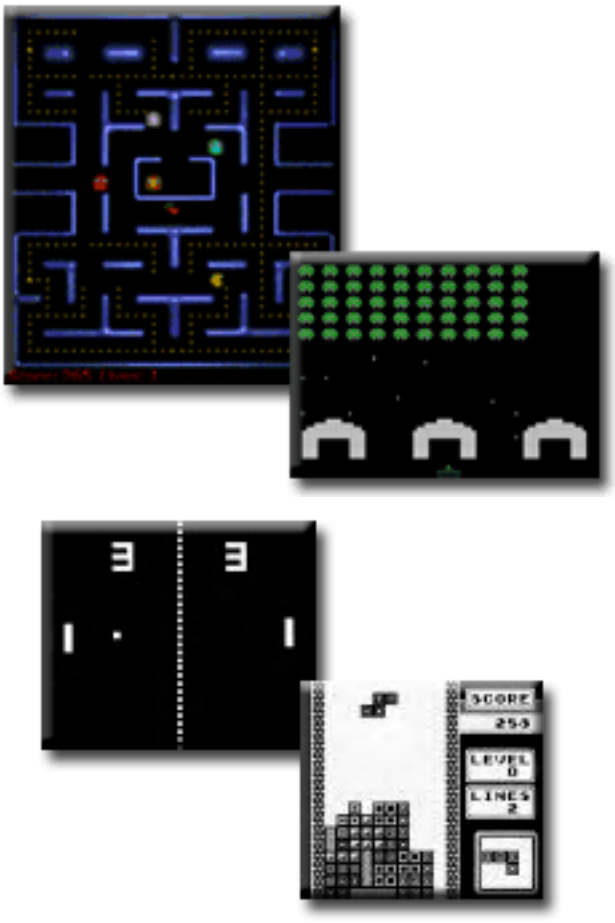
# What is a Good Game?

**By Mark Overmars**

## Games, Design, What is it?

When Atari produced its first game console in the seventies is was not very popular. This changed drastically when the game Space Invader was created and bundled with the console. Within a short period of time Atari sold a huge number of consoles. The same thing happened when Pacman was produced. And for the Nintendo Game Boy Tetris was the absolute winner. Why are these games so special that they mean the difference between success and failure of the devices they were created for?

The same applies in PC games. Some games become extremely popular making their creators instant millionaires, while other games, that look almost the same, become miserable failures. And then there is also a large collection of games that you never see because they were cancelled halfway the production and their creators went bankrupt. What makes a game a winner and what leads to failure? This is a very difficult question to answer. It involves many different aspects. In this tutorial we will delve into some of these aspects in the hope it will help you to create better games.




PacMan and other games are still very popular.

### What is a game?

Before talking about good games we should decide what a game is in the first place. There is a surprising amount of discussion about this issue and there are many different definitions. It is easier to say what is not a game.

**A movie is not a game**
This is rather obvious, but why? What elements of games are missing in movies? The main difference is that there is no active participation of the viewer in a movie. The viewer does not control the movie and cannot make decisions that influence the outcome of the movie. The same is true for stories and plays in theater. Also the final outcome of the movie is fixed (even though the viewer does not know it). This is a crucial aspect of movies and plays. People in general don't like plays in which the outcome is not predetermined. In games the opposite is true. People do not like it when the outcome of a game it fixed.

**A toy is not a game**
You play with a toy while you play a game. With a toy there are no predefined goals although during play you tend to set such goals yourself. A number of computer games actually are close to being toys. For example, in SimCity or The Sims there are no clearly defined goals. You can build your own city or family and most likely set your own goals (like creating the biggest city) but there is not really a notion of winning the game. One could add this (e.g. you could add that the game is won when your city has reached a particular population) but this can be frustrating because it is not a natural ending. This being said, there is nothing wrong with creating a nice interactive computer toy.

**A drawing program is not a game**
A drawing program is fun to play with and encourages creativity, but again it has no clear set goals. The user defines the goals and it is the user who decides whether the goals are reached.

**A puzzle is not a game**
This is a more difficult one. Clearly many games contain puzzle elements. But a puzzle is static, while a game is dynamic and changes in the course of playing it. A satisfying game can be played over and over again and there are different strategies that lead to success.

So what is a (computer) game then? Here is my definition:

**A computer game is a software program in which one or more players make decisions through the control of game objects and resources, in pursuit of a goal.**

Note that the definition does not talk about graphics, or sound effect, or in-game movies. Such aspects obviously do play a role in making nice, appealing games, but they are not the essential aspects of games. Let us look at the different ingredients of the definition in some more detail.

Is SimCity a game?

Is this Sim City a game?

**A computer game is a software program**

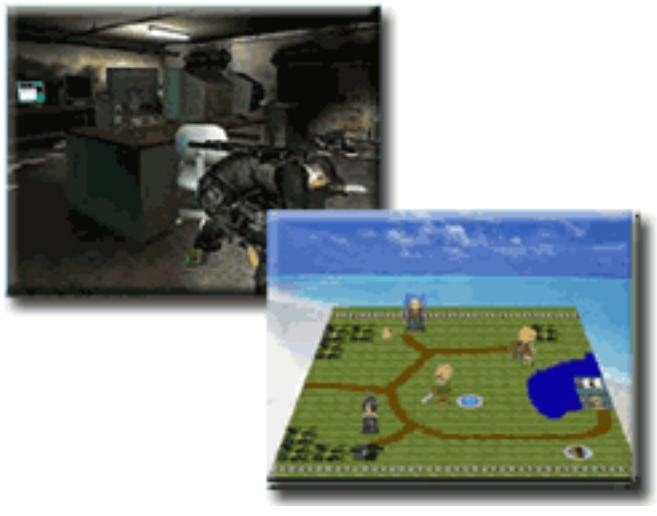
This makes it rather different from for example board games or sport games. It takes away some of the fun of games. There are no pieces to move around and there is no physical satisfaction. Also the social aspects are less prominent. But we get quite a bit in return. A software program can much better react to and adapt to the players. Most computer games have a real-time element that is not present in board games. The game continues even when the players do nothing. This can lead to enhanced excitement and a better feeling of presence in the game. Also computer games can adapt to the players making it satisfying for largely different players, both beginners and advanced players. The possibility of having computer-controlled opponents adds many new challenges. Computer games can also be more complex because the game itself can help the players understand the different aspects and teach the player how to play. Finally, computer games can create a more emissive environment by adding wonderful graphics, music and cut-scenes.


**A computer game involves players**

This is rather obvious. A game is not something to watch. You should be involved in a game. Still I want to stress the importance of the player. Beginning game designers often forget that you make the game not for yourself but for the people that are going to play it. So you always have to think about who they are. A game for children should be rather different than a game for adults. And a game for hard-core gamers should be rather different from a game for less experienced players. You need to pick the correct audience. Bad games are often written for the wrong audience. For example, a very experience flight simulator freak wants to be able to control every aspect of the plane and want things to be as realistic as possible. For a player that just wants a bit of quick flying fun this is frustrating and boring and such a player will most likely never get the plane to take off, let alone to get it to land.

**Playing a game is about making decisions**

The player makes decisions that influence the rest of the game. In fast paced action games such decision typically involve in which direction to move and which weapon to choose for shooting. In complicated strategy games the decisions involve were to build your settlements, which units to train, when and where to attack, etcetera. Of course decisions should have an effect. Surprisingly, in many games the effect of decisions is only marginal. For example, often it does not really matter which weapon to use. This often leads to frustration. Carefully balancing decisions and their effects is crucial for satisfying game play.



**Playing a game is about control**

The player should feel in control of the game. Not the other way round. Uninterruptible sequences in which the control is taken out of the hands of the player still occur in many games and often lead to frustration. The more freedom there is for the player, the better. There is though a catch here. A game is also about surprises and dramatic effects. Such effects can be created much better if the player is not in control. For example, in a movie, when the main character approaches a door you can let the music rise. The viewer knows that something is going to happen. Together with zooming in on the door, this can create a great dramatic effect. But if the same happen in a game and at the last instance the player decides not to open the door, most of the effect is gone and even becomes absurd. Careful balance of freedom of control and dramatic effect is difficult. (There is another less valid reason for not allowing too much control. More freedom and control for the player makes it more work to create the game.) Whenever you need toconstrain the user, try to do this in a natural way. For example, in Riven the player moves between different parts of the game world. By letting the user use some kind of train system it is natural that this motion goes automatic and cannot be controlled by the player.

**Game objects and resources**

In a game you normally control certain game objects, like the main character, units, a car, etc. In some games you can control just one object while in other games, for example strategy games, you can control many different objects. Besides the game objects that the player controls, there are normally many other objects that are controlled by the computer. The game objects the player controls play a certain role in the game. This is an important property. In other programs you also control certain objects, like buttons, but these do not play a role in the program. They are only meant to give certain commands to the program. Besides controlling game objects you must often also control certain resources. This is most evident in strategy games and simulation games in which you must control the amount of food, wood, stone, gold, etc. But also in many other games there are resources to control, like ammunition for your weapons, a shield that can be used a limited amount of time, etc. Careful planning of resources and their use can add many nice aspects to the game play. The game designer must balance the availability of resource with their need to achieve interesting game play.

**A game needs a goal**

This is a crucial ingredient in a game. People want to win a game and, hence, there must be a goal to reach. For long games there should also be sub-goals, like finishing a particular level, defeating a certain monster, or acquiring a new spell. Reaching a goal or sub-goal should result in a reward. Such a reward can consist of a score or some nice movie, but it is

better if the reward is actually part of the game play itself, for example a new weapon, some additional useful information, etc. We will talk more about goals and rewards in a moment.

So now we know what a computer game is. But is does not say much about when a game is good. Think about the following computer game:

**You have to rescue the princess who is held in a fortress. On the screen you are shown two roads, one leading to a fortress and the other leading to a cave. You have to decide which road to take. You choose the road to the fortress? Congratulations. You rescued the princess and won the game. You choose the other road? Bad luck. You are eaten by the cave monster and died.**

If you verify it, this game has all the ingredients described above. There is a player, there is a decision to make, the player controls what is happening, there are game objects (the prince, the cave monster, etc.) and there is a clear goal. But it is obviously a rather boring game. There is no challenge. The game is too easy. So clearly we have to do a better job to make an interesting game.

**Reaching goals**

The most important part of a game is that there is a goal and the game challenges the player to try and achieve this goal. Actually, there are often many different sub-goals. Goals come in all sorts and shapes. A goal can be to try and shoot an enemy plane, or to finish a level by collecting all diamonds, or to reach the highest score or to finish the game. Clearly some of these goals are short-term goals while others are long-term goals that can only be reached by playing the game for weeks. A good game is filled with these goals and the player should be rewarded when he reaches one of the goals (what else is the fun to try and reach it).

Goals should not be too easy to achieve. There must be a challenge. And when the game progresses the goals should become harder to reach and the player has to become better at the game to achieve them. This learning curve is very important. In the beginning the player needs to understand the controls and the mechanisms in the game. This is best done by achieving some simple goals. Later on, the player understands the game better and will be ready for a bigger challenge.

Obviously, when goals are hard to achieve, there is a big chance of failure. You have to be careful with failure though. It can easily put the player off, making him stop playing. And that is

definitely not what you want to happen. To avoid this it is crucial that, in the case of failure, the player always has the feeling he made a mistake that he could have avoided. It should not be the game's fault that the player lost, but his own. It is one of the aspects that distinguish games like PacMan and Tetris from other games. You always have the feeling you did something stupid. You can be pretty angry with yourself when it goes wrong and you are determined to avoid this mistake the next time. This feeling keeps you playing the game. On the other hand, consider a maze game in which from time to time at a random spot a flash of lighting occurs, killing you if you happen to be in the neighborhood. In this game you, as a player, did nothing wrong. You just had bad luck to be at the wrong spot. This is very frustrating. You are not angry with yourself but with the game. And you probably soon stop playing it. Don't think that commercial games are perfect in this matter. Many games produce random enemies and if you have bad luck they appear at the wrong moment and slaughter you.

I hope you learned from this that you have to be careful with "luck" in your games. Whether the player can achieve a goal should not depend on bad luck. But it should neither depend on good luck, even though that is less frustrating. Imagine that you can be lucky and find a super bomb just before facing the main enemy. Having the super bomb make the fight very simple while not having it makes it a major challenge. Having got the super bomb does normally not give the player much satisfaction in finishing the monster. It would have been much better if the super bomb was always there but the player had to make a difficult move to get it, for example, jumping over a dangerous pit. Now the player has an interesting decision: performing the dangerous jump to make the fight easy, or not risking the fall and taking on the monster with lesser weapons.

**Decisions**

As we saw in the last example, creating an interesting decision enhances the game play considerably. In general, decisions are a crucial ingredient of games. The more interesting the decisions, the more interesting the game is. There can be very simple low-level decisions or very high-level strategic decisions.

Let us look at the well-known PacMan game. It is packed with decisions. The most important decision that you constantly have to take is which direction to move in. Are you trying to stay as far as possible away from the monsters or are you going after the dots, even if the monsters stay close-by. And will you go to a corner, where you might be caught or will you stay in the center where you can move in more directions but can also be attacked from multiple sides. The second type of decisions lies with the pills you can eat to chase the monsters. When are you going to use them? Do you leave them to the end and only use them to get the final dots or do you use them early on to clear most of the maze. And if you eat them, are you going to hunt for the monsters to get extra points or are you going to use the safe time to eat more dots and try to finish the level? And finally there is the bonus item that appears from time to time. You can try to get it for extra points, but you will run the risk of being eaten by a monster.

When there are many decisions to make, like in PacMan, the player will make mistakes. In PacMan these mistakes are not immediately fatal, but it will require you to work harder to finish the level or to get the highest score. This is important because everybody makes mistakes and you should not be punished too much for such mistakes. Like a reward should be related to the achievement you made, a punishment should be related to the seriousness of your mistake. If the player looses, this should be the result of a grave mistake or a series of smaller ones. In such a case the player will definitely feel that he himself is to blame for the loss, and will continue playing to try to do better.

**Balance**

In a good game different game aspects are balanced. For example, the player should have the weapons with which he can fight the enemies. The weapons should not be too strong. That would make the game too easy. And they should not be too weak because then the player can only survive if he has a lot of luck, and remember what we said about luck before. Balance is difficult to achieve. And players are very clever in finding out where the game is unbalanced and exploit this unbalance, thereby often ruining the fun of the game.

A lot can be said about game balance. In a future tutorial we will study game balance in detail so here I will just make some general comments. There are actually three aspects of balance that are rather different: balance between players, balance between the player and the game play, and balance between different features in the game.

**Balance between players**

If you create a two-player game, you better make sure that the best player normally wins, and not the most lucky one. Imagine a strategy game in which two players compete with each other. As in most strategy games they have to build up a city and for this you need wood. Now imagine there is just one forest in the world and one player starts very close to this forest and the other is far away from it. This gives the first player an advantage that will most likely win him the game. So the game is highly unbalanced.


A game of chess on the other hand is highly balanced. Each player has the same pieces and can make the same move. The only problem is that one player can start and this is actually a rather big advantage in chess. But this is balanced out because in a match each player can start the same number of times.

Chess is a symmetric game. Symmetric games are well balanced. But symmetry is also a bit boring. Imagine that in the strategy game I mentioned the world looks completely symmetrical and each player plays the same race with the same units. That would make the game less appealing. Still it is used rather often. For example, the multiplayer maps in Red Alert II are very symmetrical. The real art lies in making a non-symmetrical game that is still rather balanced.

One way of achieving this is to use fake asymmetry. Let me demonstrate this with an example. In our strategy game we let the first player start behind a mountain range while the second player has his city behind a river. The first player we give the ability to create boats while the second player can create helicopters. This looks very asymmetric but the helicopters can pass the mountain range and in a similar way the boats can pass the river. So balance is restored again. Many strategy games use this type of fake asymmetry. Races might look rather different but in the end the possibilities are very similar.

**Balance between the player and the game play**

The game play is there to help the player, not to fight the player. As I said before, the player should loose because he made a mistake, not because he forgot the key combination to fire the canon. Careful design of the interaction (the use of the keyboard, mouse, joystick, etc.) is important to avoid this type of problems.

Also you need to strike a good balance between what the player must do and what the game does for him. For example, in most games the player does not need to keep on pushing buttons to make a game character walk. The game does this automatically for him. But the player must press a button to make the character shoot. In many strategy games, soldiers automatically start attacking enemies that come in close range rather than letting the player constantly check on all the units. But the player must decide when to start an invasion into foreign territory. But also well-known games make the wrong decisions here. For example, they force the player to constantly bring food to the troops or they force you to manually withdraw wounded soldiers from the battle. For example, one of the things many people complained about in Black and White was that when your people were praying you had to bring them food all the time.


Let me give another example. In the early adventure games one of the major problems was to find out where you should click on the picture to get certain things done. For example, to open a door you had to find the secret button to press. Only after pressing on all the 100 stones in the wall you found the one that opens the door. This adds no fun to the game. In modern adventure games the mouse cursor changes whenever you move it over a place where you can click and often a message appears indicating what there is to click on. Good visual cues are also given, for example by giving one of the stone a slightly different color. This will improve the game play a lot. The player still has to come up with the idea that there might be a secret button but once he has that idea it is easy to find.

The bottom line is that the player should spend his time and energy on the important aspects, and the game program should do the rest. The game should try to understand what the player wants and take action accordingly, rather than the other way round.

**The balance between game features**

A game contains many different features: different weapons, different enemies, different units, different roads, all sorts of resources that can be use, and so on. These features result in decisions for the player: which weapon to use for what enemy, which road to take, how to use the resources, and so on. This makes the game interesting. But you better make sure there are some real decisions here. For example, when your game features four types of weapons, but one is superior to the others, the player will never use the other three weapons once he got the best one. So there is no decision left anymore. To keep the decisions interesting you should balance the good aspects of the features with the bad ones. For example, the powerful weapon can fire only one shot per second, or the ammunition is more expensive, or it cannot be used in a cave, or one opponent is more sensitive to a particular weapon than another.

Also you have to balance the powers of the player with the power of the opponents. When new opponents appear during the game, you should give the player new powers to fight them. But be careful that you don't fall in a well-known trap in which you simply increase the firepower of the player while the opponents get equally stronger. This does not lead to more interesting game play. There is not must difference in driving with a slow car against slow opponents or with a fast car against fast opponents (unless, of course, steering the fast car is more difficult). A key issue here is that the player should improve during the game, not the character he plays (or car he drives).

Don't forget that a player must learn to play the game. That is, the game should start easy with easy decisions for the player to make. When the game progresses and the player becomes better, he should get more and more complicated decisions to take. This can be achieved by introducing new features gradually during the game. The features should match the players' abilities. Make sure that there are still new features appearing far into the game. Too many games show all the features in the first few levels after which the game becomes just more of the same. Good games come up with surprises, all the way till the end.

**Rewards**

You need to reward a player when he achieves a goal. A reward can take the form of a particular score, some nice graphical or musical feature, or items that can be used in the game, like better weapons, power-ups, spells, or knowledge about the game world. The last type of reward is definitely the most rewarding to the player and whenever possible you should try to create this type of rewards. The effect can be permanent or temporary. Temporary rewards are typically given when a player achieves minor goals. It makes the playing easier for a while. Examples of this type of reward are some extra ammunition, or temporary invisibility to opponents. Permanent rewards are given when bigger goals are achieved. For example, you get a new weapon or spell or car. This will change the game play from that moment on, hopefully extending the range of decisions the player can make.

Giving the player the right type of rewards is actually an issue that is harder than you might think. People are picky about their rewards. If the rewards are too small they will not work hard to achieve them. If they are too large they get greedy and want even bigger rewards. It is a well-known psychological phenomenon that players start expecting rewards and if you somewhere during the game decide that a particular reward is no longer available they get angry. Let me give an example of this. If in the first level of the game you give the player a bit of extra health for each opponent he kills, the player starts expecting this. If you decide in the second level that the player should now be more experienced and you stop giving this reward the player tends to get upset and might stop playing the game.

You also need to decide whether rewards are predictable or more random. For example, in your game you might give a bonus item for each 50 kills. Alternatively, bonus items might appear more randomly. The effect of these two choices on the player is completely different. In the first situation, in the beginning the player is not very interested in killing opponents. It will take way too long before it will result in a bonus. This will make the game play less intense so there should be other aspects that keep the player interested, like exploring the environment. But when the number of kills approaches the 50 the game plays starts becoming very intense and the player will work very hard in killing opponents. So there is a high variation in intensity, which is appealing to certain types of players. When the award is more randomly there is always an interest in trying to kill opponents because it might lead to a reward. So the average intensity of the game will be higher. But there will be no peaks in intensity, which can lead to a more dull game.

Make sure the player notices the rewards he gets and starts understanding why he gets them. If the player does not know the relation between his actions and the rewards he gets this will be frustrating to the player and will lead to less focused game play. So clearly indicate when points are scored or power-ups are obtained.

**Presence and immersion**

You might have wondered why we did not talk about graphics yet, or about sound and music. Many people consider them crucial ingredients of a game. New commercial games try to achieve great new graphical effects and hire famous musicians to create the music. So isn't this important? Well, yes and no. If you look at the games available on the Nintendo Game Boy, they have very poor graphics and the sound is horrible. Still they are great fun to play and many people are addicted to them. On the other hand, some great three-dimensional games create a special spooky atmosphere using the right type of music and stunning graphics effects like dripping water, smoke, and flickering torch lights.

The key issue here is immersion. Game play is largely enhanced if the player feels immersed in the game. If he feels that he is present in the game world and that his decisions and actions really matter. If he becomes emotionally attached to the main characters in the game and really wants to help them. Important ingredients to achieve this immersion are the story behind the game, the surroundings in which the game takes place, the way the main characters in the game look and behave, the music, and the special effects.

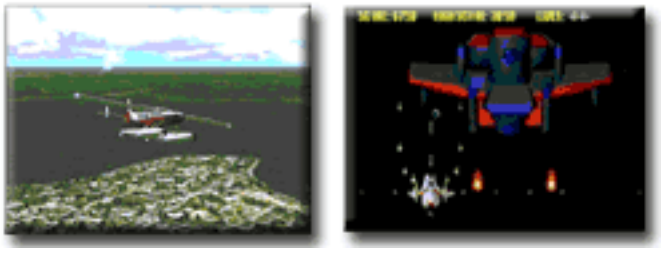
**The story**

There is a lot of discussion about whether a game needs a story. Popular games, like PacMan or Tetris do not have a real story (although the designers still give it some sort of story). And in many first person shooting games, the story is almost always the same: rescue the world from some kind of evil. Most people never read the story and it does not influence the way you experience the game. (You are not trying to save the world; you are simply killing the monsters that attack you.) On the other hand, for adventure games the story is crucial.

It forms the basis for the puzzles you need to solve, and the story actually helps you solve the puzzles; they often only make sense when being part of the story. Also other games can benefit from a good story, again because they give a meaning to the actions you are performing and deepen the satisfaction when reaching your goals. This can be achieved by making sure that different tasks or levels in the game form a logical sequence and by putting cut-scenes or movies in between them to enhance this storyline. Designing a good storyline with movies, etc. is probably beyond the skills of most beginning game designers, but it is good practice to at least put some logic in the game you are creating and such logic normally comes from a story.

**The game world**

A game takes place in some world. This world can be presented in exact three-dimensional realistic detail but also in a more abstract or cartoon-like two-dimensional way. Some games just use text and some static images to represent their game world. Designing an interesting game world is an important part of game design. And picking the right type of representation is important too. For a first-person shooter a well-detailed three-dimensional game world with lights, shadows, and special features like mist and water is crucial to give the player the feeling of presence. He has to see what a real fighter would see, otherwise the game becomes artificial. For a flight simulator the world should also look as realistic as possible. For an adventure game a realistic three-dimensional world is not so important. Here it is the story that creates the feeling of presence and this is often accompanied by two-dimensional images. In puzzle games and many arcade games the game world is rather abstract and often two-dimensional. For example, in a scrolling shooter planes don't fly in natural ways nor do the bullets behave natural. And power-ups might float in the air. This is all perfectly acceptable for the player when the game world is rather abstract but would be out of place when the game world would look realistic. So it is really important to adapt the game world to the type of game you are creating.



A flight simulator should be realistic,
while a scrolling shooter can be more abstract.

A realistic three-dimensional world can also hamper game play. For example, many strategy games use an overhead view (called an isometric view) of the game world. This makes it easy to track your units and to quickly see what is happening. You can easily scroll over the world to steer you units in doing the right things. Trying to do the same in a full three-dimensional world is a lot harder. You quickly loose your orientation, and have difficulty in keeping track of what is happening in the world. Moving around is more difficult. Again you must adapt the representation of the game world to the game play that is required.

**The main characters**

Many games have one or more main characters that the player controls or meets. Like in a movie it is important that the player becomes emotionally attached to these characters. He can hate them and try to kill them or like them and try to help them. So characters and their behavior need to be designed carefully. Again, this depends on the type of game. For example, in a first-person shooter the player himself is the character. He should fully identify himself with the character. In such a case it is advisable not to give the character a strong personality. This makes it more difficult to identify yourself with him. Or at least give the player the possibility to choose between different characters to pick one that suits him. For third-person games and adventures a strong personality is often important. If done right, the character can get some kind of hero status, like Lara Croft from Tomb Raider.

**Music**

Music and background sounds can play a very important role in immerging the player in the game. Even very soft background sounds can have a dramatic effect in games. For example, dripping water in a cave gives a creepy sound. Rolling thunder can raise the players fear, etc. Background sounds can also provide clues to the player about what is going on. For example you can hear footsteps in the distance or a door that is slammed shut. Modern games use positional sound such that the player also knows where things are happening. Picking the right kind of music for your games is as important as picking the right kind of graphics. A cartoon style game should have cartoon style music. Creepy games should have creepy music, and funny games should have funny music. Better have no music than the wrong kind of music. Modern games nowadays use adaptive music that changes with the action that is happening. This can further increase the dramatic effect but is definitely beyond the possibilities for beginning game designers.

**Special effects**

Like in movies, special effects can have an important effect on the player. Some great explosions or sound effects can temporarily highly enhance the game experience. But be careful. The effect soon wears off. After 10 of such explosions you won't even notice them anymore. And they might even become annoying if they hamper the game play, e.g. by slowing down the refresh rate, or distracting the player. For example, some puzzle games have beautiful color changing or animated background. Soon these become very annoying and you really want to switch them off. So don't spend too much time and effort on special effects. Better concentrate on good game play.


Gunnerman, Good graphics and Good Game Play?

**Game genres**

Games come in many different types. Over the years a number of different genres have been created. If you are very creative you can try to make a game that is completely new, but if you want to be on the safe side you better pick a particular genre and make a game that fits in this genre. The following are some of the most important game genres:

**Arcade games**, where reaction speed is the most important aspect of the game. Typical examples are scrolling shooters, some maze games like Pacman, breakout type of games, etc. These games are relatively easy to make and normally 2-dimensional graphics is good enough for them. These are definitely the type of games you should first start creating. A particular type of arcade games is the pinball game. These are a bit harder to create because you need natural ball movement.

**Puzzle games**, where clever thinking is the most important aspect. Many maze games are actually more based on puzzle solving rather than on reaction speed. Other examples include board games and sliding puzzles. These games are also normally 2-dimensional and are relatively easy to create, unless the game has to be played against a computer opponent in which case it might be difficult to program the way the computer plays the game. (Think about trying to program the computer to play chess.)

**Role playing games** (RPG), where you steer a character through a difficult world. Typical examples are Diablo and Baldur's Gate. The most important part of such a game is the development of the character you control learning new skills, becoming more powerful, and finding additional and better weapons. At the same moment the opponents become more powerful as well. Such games are often isometric, that is, they have a fixed viewpoint on the world, but this is not crucial. You can also create 2-dimensional RPG games or 3-dimensional ones. RPG games are harder to make because you must create the mechanism of character development. Also the games normally need to be large because otherwise they are soon finished. Good level design is crucial.

**Strategy games**, either real-time (RTS) or turn-based. Here the player normally only indirectly controls the character in the game but he does set out the strategies that the characters need to follow. Examples include Age of Empires, Caesar, Theme Park, and other city or empire building games. Strategy games most of the time use an isometric view. They take a lot of time to create because they require many different game objects, like characters and buildings, that all need their own animated images and specific behavior. Many GOD games can be considered as strategy games as well.

**Adventure games**, where the story line is rather crucial. Most adventure games are largely 2-dimensional and use the well-known point-and-click interface. The difficulty in creating an adventure game does not lie in the actions but in creating an interesting, funny, and surprising story line and in creating the corresponding artwork. You really need to be an artist for this.

**First-person shooters**, which can be seen as the 3-dimensional version of the old arcade games. Here the emphasis is on fast-paced action and reaction speed, not on cleverness and puzzle solving. Famous examples are obviously Doom and Quake but huge numbers have been created. First person shooters need a 3-dimensional world to create the feeling of being there.

**Third-person shooters**, where the player directly controls a game character through a hostile world. A clear example is Tomb Raider. The main difference with role playing games is that there is not much emphasis on character development. It is more a matter of fast action and discovering the game world. Many third-person shooters also have a storyline and borrow aspects from adventure games. Third-person shooters do not need to be 3-dimensional (think for example of GTA) and can be create with relative easy.

**Sport games**, in which an existing sport, like soccer or baseball is simulated. Many such games exist but they are often rather boring. Creating a convincing and fun-to-play sport game is a big challenge. **Racing games** are in some sense a special type of sport game. Because there are so many of them they deserve a category of their own. Some racing games, like for example many Formula-1 games, try to model the driving of a car as perfect as possible. Other games are more arcade style and make racing very easy. Racing games can be both 2-dimensional and 3-dimensional. One of the major challenges when making a racing game is to create convincing racing of the computer controlled cars. **Simulators,** like flight simulators. Such games try to realistically simulate some mechanism, like a plane. They are popular because people like to understand how such systems work and like to be able to control them. Creating simulators is rather difficult because they must be convincing.

Clearly we did not cover all types of games in this list but it at least gives you some indication of the various genres.

You can of course produce a game that has aspects of different genres, but you should be careful with this. The player picks a game from a particular genre because he likes that genre. For example, assume that you, as a designer, decided to create an adventure game with some added action. Somewhere in the game the main character has to move to a different city and for this he has to steel a car. Chased by the police the player has to race to the next city, avoiding being caught. This may sound like fun, but be careful. A player that chooses an adventure game likes the story aspect, the fact that he has to solve complicated puzzles, and the fact that he can take his time and is not hurried. The racing part suddenly requires him to play a completely different type of game in which reaction speed counts much more than clever thinking. Probably this is not his type of game and he might be unable to finish the race and will stop playing the game. Similar problems occur for example when combining strategy games with first person shooting action. So best pick your genre and stick to it for the whole game.

**Learn from other people**

This tutorial should have given you a rough idea of the things that matter when trying to create a good computer game. But in the end the best way to learn is to do it yourself and to critically look at your results.

Another piece of advice that I would like to give you is to learn from other people's mistakes. Whenever you plan to make a particular type of game, look at similar games. Play them and see what they did right and what they did wrong. It is amazing to see how often people repeat mistakes made by others before them.

There is a lot of information on game design available on the web and is this tutorial I borrowed a lot of information from these sources. You are strongly encourages to read some of the articles experienced game designers have written. For some links to get you started, see the links page of the Game Maker web site:

**Game Maker Web Site**

**Morphosis Says**

**“Check These Out”**

The Art of Computer **Game Design**

Quotes on **Game Design**

**Game Design:** Theory & Practice

History of Computer **Game Design**:

# In The Spot Light

By Morphosis

## Interview with “Freegadgets”

Many people know of him and his work with Game Maker and the creation of a 3D like game. You may have seen some of his games and asked how in the heck was that done. You may have traveled to his website and downloaded the tutorials for creating a 3D like game. But what you may have not done was to get to know a little background on the character. Well I asked him to interview him and with excitement he replied quickly with an interview and screenshots of his new projects.


Nice photo of him I made into a 3D look. It’s him don’t you think?

When was the first time you knew you wanted to use a computer?

I pretty much grew up with computers. We got the first one when I was about 6, it was an Odyssey, then came the VIC-20, the Commodore 64 and so on. I started making video games on the Commodore 64 when I was 11, mostly platform and space games.

When did you first find out about Game Maker and how did you hear about it?

I was off work with a shattered wrist. Nine months in recovery and I was bored out of my skull. Remembering how much fun it was to make games on the old C-64, I did a search. The exact phrase was “Game Maker” and Mark’s site was top of the list.

What was the first game you made with Game Maker?

Finished my first game Mental Block, about a month after downloading Game Maker. It’s actually a remake of a game I made for the C-64 many years ago. It did well in the competition and is still on the GM games page. I put it on download.com where it received over 8,000 downloads and growing.

What was the first 3D encounter you came in contact with?

I was taking drafting is high school, designing floor plans for houses. I wanted to draw my plans in 3D so I made a simple 3D engine on the C-64. I would type in the x, y, z of a point and it would in return tell me where to plot that point on a sheet of paper. I could then connect the points using a ruler, it was a slow process. I used light blue and red pencil crayons to draw pictures that would pop off the page with 3D glasses. I actually went from store to store wearing 3D glasses to find the right colors, I was a strange kid.

I started the first 3D engine a little over a month after downloading Game Maker. It took less than a week to make and was done using all drag and drop icons. It took a little longer to make the demo for it which is also on the GM games page. It was made using nothing but 2D sprites drawn to scale, no walls, no panoramic backgrounds. My engine has come a long way.

Why did you want to try to emulate a 3D look in Game Maker?

I wasn't trying to make a game or anything, it was an experiment, just to see if I could do it. When I announced I was building a 3D engine with GM, I got a lot of criticism. My favorite remark was "big foot, tooth fiery, 3D in Game Maker" or something to that effect. If anything, the criticism made me work harder, Thanks Guys!

Do you think 3D is needed in a game?

Certainly not in every game, but there are some types of games that just can't be done in 2D. Many classic 2D games could benefit from that 3rd dimension.

What are some benefits of 3D over 2d?

In a top-down game the person playing can see everything, it's a God-like perspective. If there are walls, how does Pac-Man know exactly where all the ghosts are? Not knowing what might be lurking around the corner can give a more intense realistic feel. It's also easier to show objects at different heights, this is very hard to do in a top-down game. In a 2D game your basically stuck with objects that only move 2 dimensionally.

In creation of a 3D like game, what is the most difficult thing to do? Is it the programming, Graphics, or something else?

Obviously building my 3D engine was fairly hard, using it to make games takes some getting used to but it's not unlike building a top down game. I think the hardest part is finding the graphics. To make your own original graphics you would likely need to use a 3D modeler which can be tricky. It is for that reason I'll be adding a lot of graphics to my site, some ripped from games, and later some original stuff made by me and other members. www.lexdark.com/freegadgets/

Do you make any other 3D games with other software designed for the creation of making a 3D game?

No, I have tried many of the 3D game makers out there like 3D Rad and Blender, but they are not user friendly. Most require you to know a lot of code or cost a lot of money. I consider myself a fairly intelligent guy, so if I can't get anywhere with these programs, I think few people will. There is a real need for an easy to use 3D game maker. There are so many websites out there devoted to re-skinning Doom for example. I believe these people would prefer to design their own weapons and A.I. if they had the chance. Making games should be fun, that's why I do it. I'd rather use a program that's only capable of basic old school 3D, than one with all the features but is so aggravating it takes all the fun out of it.

And last, will you continue to experiment with 3D and Game Maker?

This month I'll be releasing a new and improved version of my 3D engine, I'm calling Gadget 3D. It will take 3D rendered in Game Maker about as far as it can currently go. From then on I will focus on writing the tutorials, making graphics, and of course games! My game Doomed which many of you have already had a chance to sample, should be finished this month as well. If Mark decides to add new features to Game Maker such as textured polygons, I'll of course add them to my engine and write up the tutorials. Rather than pester Mark about it, I suggest people show an interest in 3D. I'm willing to help people with their projects, you can also help me by contributing graphics, demos, and tutorials for my website.

# Freegadgets Vs Xception

By Curtis LeMay

## The 3D War

Since Gm's beginning, 3D engines have been slow to develop and rarely proved to work well. Top-down games have dominated the GM world, simply because there were no easy-to-use 3D engines. That is, until now. Enter Freegadget's and Xception's engines. A new leaf has turned in GM; a new realm for developers to explore.

**Comparing the engines**

At first look, the engines seem very much alike. Not so. The deeper you dig into them, the more you realize how different they really are. Time for a quick engine comparison…..

Xception's DLL at work

### Graphics

Because Freegadget's DLL uses images already in the .gmd, the 3D effect is attained by scaling. Though this scaling effect is used well to make it seem that you are actually moving through the levels, in actuality, the walls are moving to you. The closer you get, the more pixilated the walls become. However, in Xception's DLL, images are loaded externally, meaning that they stay the same no matter where you are, but from a distance away they seem somewhat blurry. Overall, Xception takes the ball on faster computers.

### Level Design

Yes, the way an engine is built does affect what you can make with it. Xception's does allow you to change wall size, textures, and coordinates very quickly and efficiently. But Freegadget's has many of the same features. Freegadgets also can make cooler-looking buildings. Freegadgets actually takes the level design category

Freegadget's Doom Spinoff

### Loading Speed

Speed may be an issue with some people. Freegadget's runs everything internally-a modern wonder (that isn't counting the DLL). Though this may look appealing in a folder, it doesn't help the game speed. Xception's runs images externally, but when a room is loading up, it has to load in the images. The images ARE crystal clear, but there is a definite speed advantage between rooms in the two engines. In our speed tests with 5000*5000 rooms, Freegadget's loaded almost 5 seconds faster than Xception's every single time. Advantage Freegadgets.

### Ease of Use

Freegadget's uses many variables and scripts to run levels. These scripts and variables aren't the easiest things to specify either. Xception's is way easier to use. A few lines of code using Xception's may produce a wall. To do the same in Freegadget's, you may need to use 10 variables and a few scripts. Plus, Xception's has md2 and 3ds compatibility, meaning you can use old Quake 2 skins and scenery files. Xception scores again.

## Customizability

Because images load internally in Freegadget's, people cannot customize the levels in a standalone version of a game. With Xception's, and walls, floors, and roofs can be changed as easily as opening an image in paint and changing it. 3ds shapes can be changed from hills to houses, and enemies from monsters into Pokemon. With all these possibilities, Xception takes the win.

## Conclusion

With all its power, the limiting factor with Xception's engine may be the steeper system requirements. However, granted you have a newer PC, it is the winner. Not to say in any way that Freegadgets has made a bad engine, but it does not yield the great graphics and customizability that are almost a given in today's times. Both of these engines have unlimited possibilities, though, and I hope that they continue to improve. With the arrival of GM 5, maybe these engines can take it a step further as far as engine uses and graphical works. They can also get faster. Freegadget's is most compared to the first Doom, given the demo's that have come off of it so far. Xception's is pretty much a better-looking Doom. Expect to see the 3D invasion by June of this year. A few additions to both engines will produce many good 3D games.

One thing is certain at this point. The days of only seeing 2D games are numbered. The pioneers of 3D games in GM have blazed a clear trail, and in a little while the common people will follow. The realm has been entered. It is time for change in GM.

**Curtis LeMay**

**:Email:**

**cjlemay@cox.net**

**:Web site:**

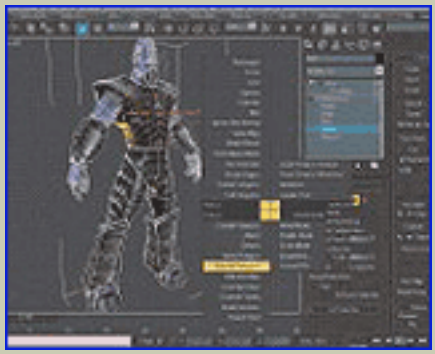
**www.planetearth.tk**

# Tools of the trade

By Morphosis and various sources.

## GMDM's Top "3" 3D Software

### 1 3D Studio Max



3D Studio Max 4 is a substantial new upgrade to the famed 3D production tool. Max has always been the games designer's favorite 3D application, though as time goes by Discreet doubtless feels its grip on the games market gradually becoming more tenuous.

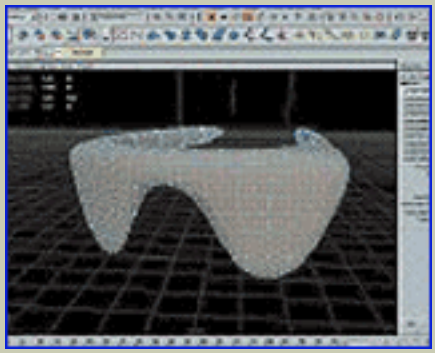
Max has always relied on complex procedural tools and a rather convoluted work flow. Modeling operations, effects, and utilities are layered in the Modifier Stack which, while offering a sort of construction history, makes working with Max a very disjointed affair. Version 4 has gone some way to improving things thanks to a new pop-up menu system, called the 'Quad menu' – similar to the Hot Box and marking menu system in Maya. There's also a new polygon-modeling system, which makes you wonder how Max was so successful in games before. The new Polygon Object gives you access to a true polygon modeling toolset. You can access the various modeling functions from the Quad menu, though the toolset is not as comprehensive as some of the other packages we have on test here. The new IK system is impressive and much improved over previous versions. Bones can be displayed with fins which help define the volume of the object. This is useful, and as they act as real objects they can be rendered for fast previews. Any object can become a bone, so you can use Max's modeling tools to build a custom skeleton structure.

The limb solver is a two-bone IK solver specifically designed for games production, since it can be modified for use in a games engine. FK and IK can be mixed and blended on the same IK chain using key frames. Skinning – the binding of a character model to a jointed skeleton – has also been improved. The Morph Angle Deformer lets you create bulges and skin deformations based on bone angles. You have to make a duplicate of the geometry, though, so it's not as elegant as is could be. The best workflow enhancements are the Custom Attributes and Parameter Wiring features. Custom Attributes are custom UI widgets, sliders and the like that control animated parameters and are displayed right in the view ports. Rather than having to hunt for the parameter in the modifier stack, all the animated parameters can be in one place.

Wiring lets you link animated parameters together – and combined with expressions, add a lot more power to Max's animation toolset. Max 4 is undoubtedly a great system for games development. However, despite the advances and the large user base, Max remains a powerful yet inelegantly designed system.

Price: $995.00 US

## 2 Maya

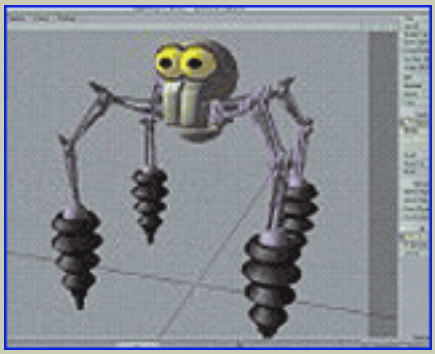


There are three versions of Maya: Complete, Unlimited, and Builder. Complete is the base version featuring advanced character animation tools, NURBS and polygon modeling and sophisticated rendering. Unlimited adds Cloth, Fur and Subdivision Surfaces to this package. Builder, of interest here, is a stripped down version geared towards game development. While it lacks rendering and NURBS modeling, there's a single two-bone IK solver that's ideal for game characters (the source code of which is included and can be integrated into a games engine). As you'd expect, Builder is also much cheaper than Maya Complete and, at around $2,000, represents good value for money. It has the full complement of polygon modeling and texturing tools found in Complete, which includes UV editing and 3D painting. Multiple textures and lighting (including shadows) can be 'burned' into an object's UV texture map allowing for rich effects and environments to be created with no extra overhead.

Where Maya 3.0 scores highly for games is with MEL scripting (Maya Embedded Language). MEL lets programmers create their own UI elements to help manage the creation of multiple levels and provide access to the full Maya API (Application Program Interface). Artistically Maya scores highly too. The interface is fully customizable, and it's generally easy to use. Maya Complete features more sophisticated IK solvers – including an IK Spline handle that's ideal for animating multi-jointed or tentacle-like creatures. Attaching skin to a skeleton is easy, and editing joint weights can be done visually by painting on the object's surface. The painting interface, called Artisan, extends much further than this. You can paint-select objects and components, sculpt surfaces like clay (also available in Builder), and use Paint FX. The polygon tools are good, and thanks to Maya's Construction History, you can reach back to any previous step to make changes. This offers plenty of opportunity for creating versions of a single model to create different characters for a game. Like XSI, Maya features non-linear animation – extremely useful for game development, since you can build up a library of animation clips that can be reused and accessed by all animators in a workgroup. Maya is an exceptional package that offers several levels of sophistication and price points for game producers. Builder can be used to create levels and environments, build characters and for simple animation, while Complete and even Unlimited can be used for more complex work in the game – and of course for an FMV. It may not be cheap, but for most of us Maya is all the animation software we'll ever need.

Price: $2,180 US

## 3 Lightwave



LightWave is a solid 3D program with a considerable history of achievements. Its credits span film and broadcast – and, of course, game production. LightWave has an excellent modeler that's mainly polygon based, so it's ideal for creating models for games. The polygon toolset is vast and extremely versatile, with excellent control over the finest details of poly modeling. There are other available options, of course, such as procedural primitives (generated as polygon meshes), metaballs and MetaNURBS, which are LightWave's version of subdivision surfaces.

MetaNURBS can convert models that contain three- and four-point polygons and subdivide them to form smooth surfaces. Unlike XSI, you're limited to just these three- or four-points – any more or less and MetaNURBS will fail to convert. They are, however, very flexible and feature weight painting that lets you vary the influence of points in the control cage over the subdivided surface. LightWave 6.5 has good UV editing, and there are some nice extras like per-polygon UV mapping too.

Animation is good, although there's no non-linear animation as yet. Soft body dynamics are courtesy of the Motion Designer 2000 plug-in, and there's good expressions and a function-curve editor. The IK system is also good and skeleton setup has been made easier thanks to Skeletons, special polygons assembled in Modeler that can be converted to bones in Layout. LightWave has an excellent raytrace render capable of stunning output and very sophisticated effects. Hypervoxels is LightWave's technology for creating realistic volumetric effects, and there's also radiosity and caustics should you need them. For all aspects of games production – including in-game video – LightWave is a worthy option. It offers a more hands-on approach and in many situations, such as for modeling, this is no bad thing.

**Price: $795.00 US**

First there was the graphics-impaired Adventure by Crowther and Woods, then there was Atari's masterpiece of minimalist gaming, Pong. At the time, of course, it was mesmerizing, and soon we had Asteroidsand Space Invaders,which were even more beguiling. The games industry and the games it produced have certainly come a long way since then, and so have the tools used to create them. Intense graphics and realistic motion coupled with impressive effects and audio are the going commodity in today's games.

For years the games industry has been dominated by 3D Studio Max, and there are still many staunch Max supporters, there has been a trend away from it and towards the seemingly unstoppable force of Maya. Although almost any polygon-based 3D program can be used for modeling for games, some are better suited than others. As computers and consoles become more powerful, the restrictions on polygon counts will be less severe. Games will become more interactive and yet look as realistic as a movie, so the ideal games system for the coming future will combine the efficient heritage of games production with the no-compromise toolset of film effects and character animation. Graphics cards such as the new GeForce 3 – which can render high-resolution characters and produce near broadcast-quality effects and imagery in real-time – demonstrate that things are changing

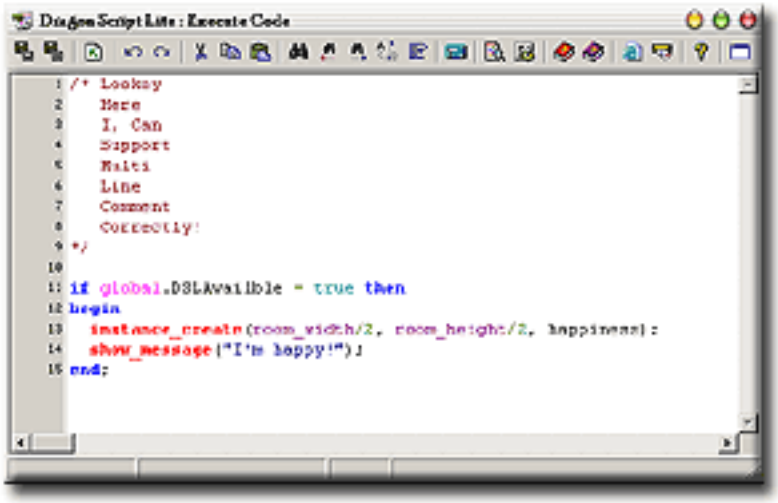
# Dragon Script Lite

By DT

## GML Editor with the Dragon's Touch


Dynamic Terminology was an unofficial (yet very powerful) GML editor developed by one of GMC administrators, DT, which made its first appearance to the public on January 25th, 2003 as a beta release. It has been downloaded hundred of times since its first debut. However, statistics still weren't very ideal then because of the lack of close integration between the program and GM. People want to avoid the hassle of using an external editor, because it requires the unnecessary effort to paste it into GM. But now, all hesitation aside, for the creator of GM himself has added the option for external script editor support, which is how project DSL (Dragon Script Lite) was born.

A great deal of care has been put into the development of DSL. Its user-friendly interface closely resembles GM's, eliminating productive time being spent on manual reading. Even with a simple interface, its power will not be compromised; its functionality will easily surpass all competitions (including the official built in editor of GM). Despite the fact that the program's name consisted of the word "lite", in this case it means "light", as in small and compact. Recycling the same robust syntax parsing engine as its predecessor, DSL will feature all the core functions of DT's plus more extras. For more information on what the changes are, please see the




### Changes - Quick Glance

#### Syntax Highlighter

Code coloring has been completely rewritten from scratch to fully support the GML syntax to the core. The current grammar file comprised of over 3000 lines of raw coding, the most comprehensive version yet. It can handle GML elements even better than GM's itself can!

#### Graphical Interface

The program's GUI has been painstakingly designed over and over again, dynamically as well as on papers to find the simplest yet elegant layout. Modeled to closely resemble GM's, anyone should be able to just pickup and use it right off the package, without the need for tedious manual reading.

#### Execution Speed & Size

Optimized to consume the least memory as possible and because it's "lite", its execution speed is vastly improved.

#### Code Proposal & Hinting

One of the many highlights of Dynamic Terminology was the syntax proposal and hinting, as a result this department was carefully adjusted to better suit with the global design. The hint-lookup is no longer restricted to GML functions, but has expanded to other GML related elements as well.

#### Many Smaller Changes

Such as less clustering in the preferences, actions being more responsive, auto completion and completion are under fine-tuning, more depth to syntax coloring configuration, etc...

**Release Plan:** Atmost,no later than a week after GM5's stable release.

**http://dsl.gmcommunity.com/**

# Reviews

By Morphosis

* I Need people to review games, contact me for info and guidelines.

## Cool games to be or not to be?

### Aliens Attack On Colony


By far on of the largest GM games that I have ever downloaded. With the wait of this 30meg game, I soon realized it was well worth it.

This Gory and eerie alien sci-fi action game is filled with excitement, great sounds, good graphics, and good game play. The story line is that from an alien sci-fi movie packed with 9 types of weapons and explosives. If you are a fan of a good gory overhead game give it a shot. I was rather impressed with the game and you may be too. If you like SNES Blackthorn, Fallout, X-Com, and PSX Project Overkill, that's a sign you'll like this. The downfall, some menu items do not function, but this doesn't hinder the game.

**Grzegorz Borkowsk** Click Here For Web Site

### Jetz Rampage


An interesting game mixed with humor, action, and a touch of gore, well a little more than a touch. This side scroller is one of the more interesting side scrollers I have played. The character movement alone makes it fun, not to mention how the character is strapped with a gun and grenades with only one goal, destruction.

The graphics are done really well and the sounds from the TV show south park make you laugh. This bizarre game and theme may seem violent, but it's fun, at least to me. I do recommend for you to check this out.

**Shawn 64's Creations** Click Here For Web Site

### Ore no Ryom


"The Customer is always right" that is what you have to say when playing this game. It's a great restaurant simulation game where you have to serve the customers food and drinks, and if you give them what they ordered you'll make money. But if you mess up an order, you loose money. From pizza to a frosty beverage and even washing the dishes, this game will make you laugh and keeps you on your toes. The game has three game modes to play, a nice tutorial mode, franchise mode, and arcade mode.

Good sounds and music but the graphics are a bit weak and could be improved, but the game play and fun theme make it a winner. Get this!

**Vertigo Games** Click Here For Web Site

# Development Misc.

By Chris Spicer

## Teh Noob, part 2 of my development diary


Hello everyone! I am Teh Noob, and welcome to part 2 of my development diary!

Apparently, this diary is being published in a GM Magazine! See? I've only been using GM for five minutes and already I'm famous! To celebrate, I've drawn a cool new logo for my column! I made it entirely in GM's own graphics editor - isn't it great?!? It took me five minutes - I've never spent that long on my graphics before!

I must be famous - someone sent me an email about the last issue!!! After I said I was a l33t H@x0r, they wanted to know what hacking
I've done! Well, one time I asked my friend what his computer password was at school, and I went onto the network and changed his desktop to a picture I'd drawn saying "I am a loser!" See? What did I tell you? Beware my digital fury!!! Anyway, remember how in the last column I said I'd have my first game finished for this issue? Well, I did it! Unfortunately, my RPG, Soul Strike: Legend of the Vengeance Demon (final title still pending), has been put on hold due to Technical Issues with the 3D engine (because at the moment it still doesn't have one), but I still completed my first game! I couldn't be bothered to learn all that complicated drag-and-drop stuff (I thought GM was supposed to be easy!) so instead I based my game around one of the example games that came with GM called Click The Ball. Click The Ball isn't very good, but I did loads of work on my version and now it's almost unrecognisable! My game is called Kill The Ball. See? Completely different!!!

In the original game there were these balls bouncing around and you clicked on them to make them disappear - totally lame! My game is much better. In my game there are these balls floating around and you click on them to KILL them!!! It's great, when you click on them there's blood everywhere and everything!!!! I drew all the graphics myself, except for the ones I took from the original game.

Anyway, my game is so cool that I submitted it to a review website and tomorrow I'm going to email it to all the games magazines. Depending on how good they think it is, I might start charging for it as a shareware game! (Don't worry, I'll make sure I give out a secret link so all my loyal readers can download it for free!!!)

Those l@amers at the GM Community are still all l@merz, though! I told them all about my game, and they were all d1ss1ng me! This one guy was like "All you did was edit the lame Click The Ball example that came with GM!" I mean, DUH! Of COURSE that's what I did! He says it like it's a bad thing!

Just enough time to tell you about my next game. I've decided to scrap all those games I planned last issue - I decided there were too many 3D games around already, it's not that I couldn't program them or anything - and concentrate on something new. I want to make something original and experimental, something that will make people think "Wow, I've never seen anything like that before, and certainly not made with GM!" So I've decided to make a platform game.

I know what you're thinking, and you're right to be excited!! I haven't decided on a plot for the game yet, and I don't have any ideas for levels or characters at the minute, but I do know one thing my game is going to have: L33TNESS!!! Believe me, this is going to be teh best GM platform game ever! Without a doubt! Nothing can prevent it!

Until next time, remember: I'm GREAT!!!1!

Noob Out!

Teh Noob's development diary was transcribed by Chris Spicer