game reviews game development interviews code tutorials

Music Superstar

First look at the new Morphosis Enter-Active sim game of fame!

Ore no Ryomi 2

First look at the hit sequel!

Interviews

- Ari Feldman

Game Maker Legends

- Carl Gustafsson
- Mr. Chubigans

Color and game arcade graphics

By Ari Feldman Part 1

Reviews

Bicman: Gravitized - Undead Fred - Rounders (a) - JWS Racer - and more!

GMDM Issue #3

ISSUE 3

Contents

Credits

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

Edited, Designed, Written and Produced by:
John Hempstead-Morphosis
Morphosis Enter-Active (MEA)
© 2004 MEA

Other contributions to GMDM include and get a big THANKS!
Mark Overmars
Ari Feldman
Chris Crawford

SIGN UP for the Morphosis Enter-Active Newsletter
email me with the subject line "MEA Newsletter"
You will get emails about updates, gmdm release dates, an more!
CLICK HERE
morphosisgames@aol.com

Adobe Acrobat Tips

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”

From the editor

Morphosis: John Hempstead

Welcome to the 3rd DOUBLE issue of Game Makers Data Magazine. I am so sorry for the wait, for those hundreds of emaisl asking “When is it coming out?” I hope you will be pleased.

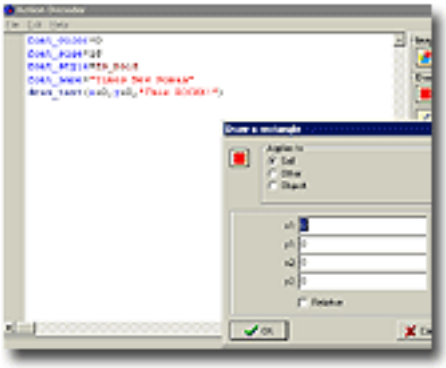
I am very excited about this magazine and it’s success. Reports from Mr. Overmars state that in a few weeks the 2nd issue of the magazine was downloaded over 2000 times. I hope that this issue and future issues will be just as great!

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 Enter-Active
http://mea.invisiblefury.com
John

GM Tools

Tools to use with game maker



Action Decoder

Translates Game Maker 5.0 Actions into working code. Great script editor with Syntax-Highlighting, Code completion proposal, auto block commenting

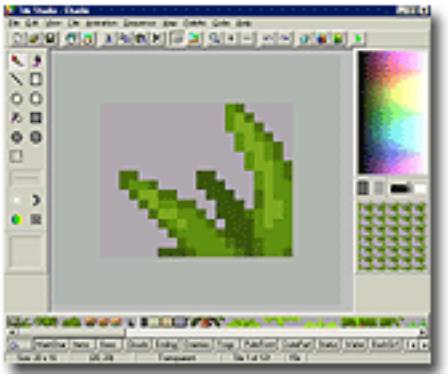
http://home.tiscalinet.de/xception/

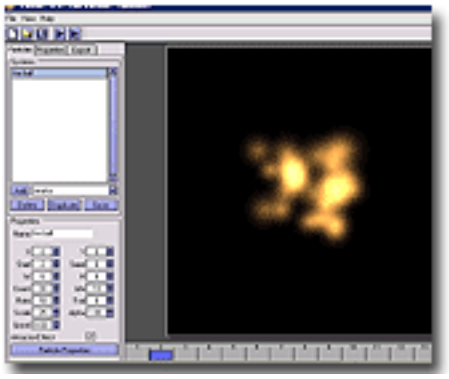
Tile Studio

Tile Studio is a combination of a tile editor and a map editor and was made to design games.

With Tile Studio, you can only design graphics and levels, not complete games.

http://www.cs.kun.nl/is/ts/





Ex Gen

ExGen was created because of a need to create particle effects for a video game. Equivalent programs were searched out but the closest to what was wanted were found lacking in certain areas.

http://www.binarymoon.co.uk/

GM Top Files

Top tutorials for game maker

Beginner's Multiplayer Library LIB
http://www.formingaband.com/Ominous/MGLibTutorial.htm
A great multiplayer library with .gmd example.

Data LIB
http://free-du.htnet.hr/hrvoje_ban/datalib.zip
This library covers most used functions of writing/reading data in Game Maker. It is made for novice user who wants to use Game Maker's ability to write/read data from files and registry without need to learn GML.

360 Styled game LIB
http://insane.tauniverse.com/TASC/360.zip
Acceleration
Braking
Turning Left
Tunring Right
Reverse

Shooting
Shooting in middle
Shooting on wingtips (thanks to grasshopper for the example) and more!

Replacement LIB
http://www.geocities.com/fabianswebsite/replacing.zip
Replacement library lets the user change a background or sound from the executable. It also enables you to change a background, sound, or sound volume at any time.

Advanced Library LIB
http://www.freewebs.com/danshow777/downloads.htm

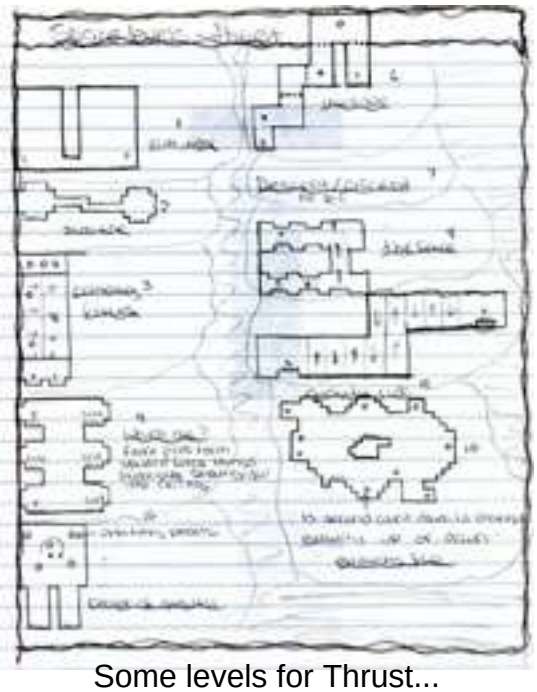
ISO Library LIB
http://de.geocities.com/ben0032003/libraries.html
This library containes many actions dealing with motion on a isometric grill. You don't need to know much about GM to use it but advanced users should also find this useful. A script library is also included so that you can use the power of this library also through code.

Mini golf example
http://www.oddworldz.com/claret/minigolf.zip
“I tried several minigolf examples, but I didn`t like them and made my own work that I used in my game Now I made an example out of it, showing the way of playing shots. It`s only 9 kb.”

Tatcics Engine
http://images.trafficmp.com/tmpad/content/Monster/monster.html

Concept art

From paper to computer

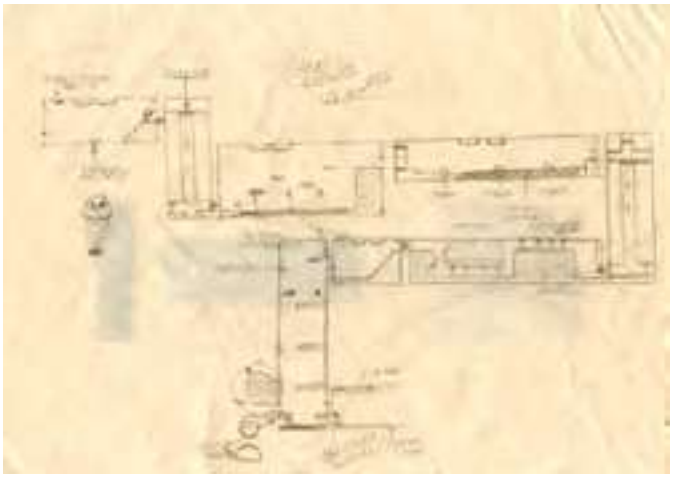


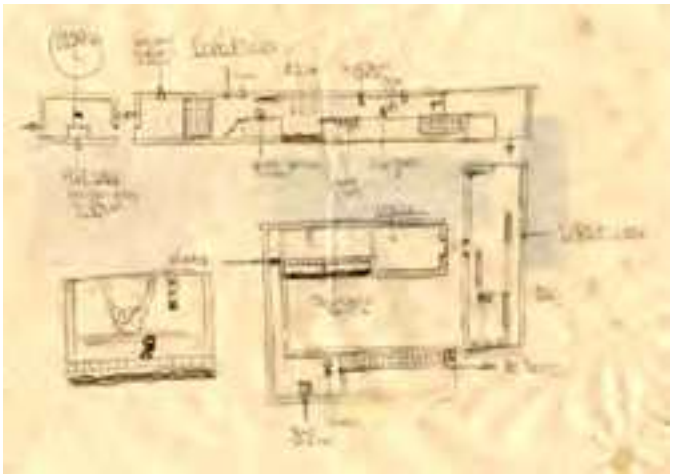
Some levels for Thrust...

Shawn64.
http://www.shawn64.com/

These drawings were done by me mostly at school during some boring class. Alot of times i draw stuff for my games and never even look at them again but remember ideas i came up with. yeeaah....

From the editor:
Shawn64's concept art is very well done. I think it's important before beginning a game to first create concept art of levels, characters, game look, and game functionality.





Do you have concept art?Send me a link on the web and you can be in gmdm!

Game Documentation

Part 1

As a game designer, you will be primarily concerned with what is com monly called the design document. How ever, there are many other pieces of documentation used in the creation of modern computer games.

Even though you may not work with all of these documents, it is important none the - less to under stand what each of them is supposed to contain and how the different documents are interrelated. So before delving into the nature of design documents, a survey of the different types of documents is appropriate. Different people at different com pa nies or in dif ferent situations will invari ably call the documents listed below by a vari ety of different names, so you should be aware that the naming convention I employ here is not universal, but the types of documents used are quite common through out the game devel opment industry.

Concept Document or Pitch Document or Proposal

These are usually the first for documents created for a given game. Often they are written in order to sell the idea of a game to a publisher (if the author works at a developer that does not pub lish its own work) or to upper man agement (at a com - pany which publishes internally devel oped projects). In short, this document is shown to the green-light com mittee, the money, the suits, the decision makers, or whatever one may call them, in order to con vince them to spend a lot of money on the idea, thereby funding its development. Con cept documents are usually short in length, customarily no longer than ten pages, and usually include plenty of concept art. Concept documents are commonly written by committee, typically involving the game's producer, lead designer, lead programmer, what ever market ing people may be on hand, and the lead artists who contribute a variety of sketches, conceptual pieces, and screen mock-ups. Concept doc uments dis cuss all aspects of the game idea in question, including how it might be positioned in the marketplace,
budgets and devel opment timelines, what tech nology will be used, what the art style of the game will be, mini-bios of the team who hope to work on the game, and some broad descrip tion of the gameplay. These documents are not much use in the game's actual development, though they can be a spring board for creation of other documents, such as the design document or the art bible. Since concept documents do not apply very much to the game's actual development, I will not go into fur ther detail about them.

Design Document

In other parts of the software development industry, the equivalent of the design document is often called the functional specification. Indeed, some game developers refer to the design document as the functional specification. I prefer "design document" because it is the more widely used term and because it better represents the contents of the document. The design doc ument's goal is to fully describe and detail the gameplay of the game. For large team projects, the design document serves as avital ref erence work for how the different aspects of the game need to func tion, with, ideally, team mem bers referring to it through out the game's development. Producers will often use the design document as a spring board from which to schedule the project. A well-written and complete document can also be of vital importance when a game is sub sequently converted to another platform by a different develop ment team. The document can serve as an ideal reference tool for this new team to understand how the game is supposed to func tion as they start port ing it to a new system.

Whereas a functional specification for, say, a spread sheet application can be extremely detailed and complete, a design document for a game is necessarily less complete because of the more organic, dynamic, and iterative nature of game development, as As a designer work ing on a large team project, the design document will be the primary specification with which you will need to be concerned. The guts of a design document are the detail ing of game mechanics: what the player is able to do in the game-world, how they do it, and how that leads to a compelling gameplay experience. Design documents typically also include the main components of what ever story the game may tell and a detail ing of the different levels or worlds the player will encounter in the game. Also included will be lists of the different characters, items, and objects the player will interact with in the game-world. One can think of the important aspects of the design document as not dissimilar from what a journalist looks for in a news story: what the player does (which actions the player can perform), where he does it (the game's setting), when he does it (at what time and in what order the player must perform different actions), why he does it (the player's motivations), and how he does it (what com mands are used to con trol the game).

The design document can also be defined by what it does not include. Most of the content contained in the other documents listed in this should not be found in the design document, including the bulk of the information found in the script, the technical design document, and the art bible. In particular, a design document should not spend any time describing the game's development from a technical stand point. Plat form, sys tem require ments, code structure, artificial intelligence algorithms, and the like are all topics that should be covered in the technical design document and there fore avoided in the design doc u ment. The design document should describe how the game will function, not how that functionality will be imple mented.

Similarly, discussions about the marketing of the game, explorations of how it will be positioned compared to other games in the market place, and sales projections are all inappropriate in the design document. In addition, schedules, budgets, and other project management informa tion should be left out. This infor mation should cer tainly be recorded in some documents, such as the pitch document or project schedule, but it should be strictly excluded from the design document. I would think that such an exclusion would be obvious to any one under taking a design document, but I have seen many design docu ments that spent half their pages con sidering how the game will be sold. The design document needs to describe how the game functions so that some one working on the develop ment team can see exactly what she needs to create. Including materials which are more about the business side of the game's development will only get in the way of more appropriate information.

Storyboards

Storyboards are an established film and television device for sketching or mockingup shots before they are actually filmed. Storyboards may be included as part of the art bible or can stand alone as their own separate doc ument. Storyboards are most handy for map ping out non-interactive cut-scenes, which are quite cinematic in nature and are thereby well suited to storyboarding. This allows members of the development team to provide feedback and correc tions on those cut-scenes before someone goes to the trouble of filming or rendering them. Storyboards can also be used as concept sketches or mock-ups for how the game-world will appear to the player if the game's engine is not yet ready to be used. Such storyboards can be use - ful both for mak ing the entire team under stand at an early stage where the game is heading, as well as convincing financiers to fund the project's development.

end

Steps to success

Shareware Part 1

Research the Potential Market

A great shareware program is going to enjoy little success if there are dozens of other fabulous programs of its kind already on the market. Make sure there is a need for your program before you spend months writing something somebody else has already done and flooded the market with. (Of course, don't let a little competition deter you if you think you can do better -- or the competition hasn't found its way deep into the market or you can add a new feature that would likely be in high demand.) Search for the key words on the various places that distribute shareware.

As you try to identify a void for your product type, also keep in mind that your program must be useful to a lot of other people for it to work in the shareware market. That is, even if you can't find another program of the kind you wish to write, maybe there's a reason. For example, you may have a great program that helps someone manage a worm farm, but not too many people will find that useful. If you can, however, make it general enough to be a management program for any kind of farm, kennel, etc. you have effectively altered it from being a product in a vertical market (where only a certain group of people would use it) to one that is in several vertical markets (where it would be useful to many more potential users).

There are times when marketing to a vertical market may be advantageous and/or feasible. Naturally if you have already written a program for yourself and you think it would find a niche in a vertical market, the costs and time for you to do so won't be as great -- and you don't have as much to lose since much of the program has already been written anyway. Another exception is if the vertical market represents a lot of potential customers despite being useful to only a certain group of people. If this is the case and you can specifically target the people who would be willing to use (and eventually pay for) your program, you may enjoy a lot of success with your program. The key will be to advertise it in trade journals and selective places online frequented by the people who would find your program of some use to their hobby, occupation, etc.

Program for the Shareware Market

Shareware users are getting more and more accustomed to great software at affordable prices. Obviously, no one is going to pay for your program if they do not find it useful and friendly. However, with that said, even a useful, friendly program being used by a happy, somewhat honest user is often not enough to elicit a payment.

Once your program is being used by a user, it is crucial that they know how to pay for your product. This can effectively, and tastefully, be done via readme files and reminder screens. I can not over emphasize the importance of reminder screens. I'm not talking about obnoxious screens that pop up every minute "nagging" the user to pay up or else. However, you should put information about how to pay for it in strategic places in your program. Because readme files get lost or deleted, you may consider adding the information directly into the program itself.

One example of tastefully placing payment information in your program is to make it a menu option. For example, if you have various options your user needs to select while running the program, make one of them something like "How to Purchase..." or "If You Like this Program..."

You may consider using the following method:

Don't immediately start "reminding" the user how to pay for it via pop-up screens, etc. -- they need to try your program before they can decide to pay for it. However, if after a period of time they are still using your program, it is probably time to start "reminding" them of their obligation to pay for it proportional to the time they have evaluated the program and/or number of times they have used it.

For me personally, at first I don't display any reminders about payment beyond allowing them to choose reading the information from a menu. However, I feel if the user has used the program for a certain period of time (say, 30 days), then they have been "evaluating" the program at least enough to warrant a screen explaining how to pay for it after performing certain functions. As the length of time increases, so do the type of functions that invoke the message. Eventually, common functions will display the message and I feel this is certainly fair, and adheres to the shareware concept of trying before buying.

Some would argue that some users delay paying for a long time, and therefore you're loosing a sale. If you feel this way, then this method is not for you. However, I think I'm pulling in more impulse purchases than I'm loosing from the ones that "delay" paying. These procrastinators (if they intend to pay at all) are delaying, I might add, beyond the 30-day trial period I (and many others) typically set for evaluating the program.

One thing you do have to be careful of is if your program is such that someone might not use it often, then you will want to track actual usage rather than the time it has been used. If someone installs it and then does not use it for a couple weeks (or it only gets used a few times a week), then you will probably be better off tracking the number of times it has been used rather than the days it has been used.

Pulling in the Payments

Payments should be a quick, easy process -- both for you and the user. Don't give them an excuse not to purchase the program! One way to do this is to accept credit cards, especially through an 800 number, dedicated fax, and securely via the Internet. NorthStar Solutions is a company that will do this for you for a very modest fee.

Incentives are often what pull in payments. Even some honest users need that extra "reason" (beyond the satisfaction of a good feeling) to get them to pay for it. If they are in need of something, they are more willing to give something: such as paying for the program. Coupled with a convenient way to purchase (see above paragraph), this is an extremely effective way to pull in more payments.

Distribute to Key Places

Another crucial element in writing shareware is distributing it. I'm not going to spend too much time on this subject, because it only makes sense that a user has to be using your program before they can consider purchasing it.

Distribution can be expensive if you elect to mail diskettes to vendors, etc. However, many of the organizations that distribute shareware are online. Todays shareware author can get a lot more distribution for his or her money now that online services and the Internet have become a major distribution channel. One of the best places you can distribute your product is via the large information services such as CompuServe, America Online, Prodigy, the Internet, etc.

Run Your Shareware Business Like a Business

Treat your shareware business as just that -- a business. It's not enough to be a great programmer, you must be a good businessperson. You have to know how to market, be creative, research your market to make sure there is a demand for your product, constantly adjust to the changing market, and run your business efficiently.

We have included several resources here in the SARG that should help you. Most importantly, if you keep in mind that your shareware business is a business and treat it as such, you will benefit greatly. In fact, you can even get several tax benefits for doing so.

Why do people play games?

Part 1

by Chris Crawford

Game-playing requires two components: a game and a player. The game designer works to produce a game, and so her immediate preoccupation is with the game itself. Yet, her final goal is to educate, entertain, or edify the game-player; hence, the human player is the proper primary concern of the game designer. Why do people play games? What motivates them? What makes games fun? The answers to these questions are crucial to good game design.

One way to address the question of the purpose of games is to inquire into their history. Games now are too varied, too intricate, too involved, to indicate a single clear function. Perhaps their fundamental nature would be more evident in their earliest incarnations. How far back must we go? To MONOPOLY, created during the Depression? No, card games were played long before that. Indeed, the discoverers of King Tutankhamen's tomb found among the wealth there a wooden surface with regular divisions that appears to be some sort of boardgame. But even archaeology does not take us far enough back. If we wish to get back to the beginnings of games, we must go beyond the realm of the archaeologist and into the realm of the paleontologist. We must reach not thousands but millions of years into the past to find the earliest games, for games predate not just history but all of mankind. They are not a human invention.

Fortunately, direct recourse to paleontology is unnecessary. A trip to the zoo will suffice. There we find two lion cubs wrestling near their mother. They growl and claw at each other. They bite and kick. One cub wanders off and notices a butterfly. It crouches in the grass, creeps ever so slowly toward its insect prey, then raises its haunches, wiggles them, and pounces. We laugh at the comedy; we say that the cubs are playing a game, that they are having fun, and that they are such fun-loving, carefree creatures.

We are right on the first count: these cubs do indeed appear to be playing a kind of game. We can certainly see in their behavior all four of the fundamental game attributes described in Chapter 1: representation, interaction, conflict, and safety. We may be right on the second count; who knows if lions can have fun? But we are dead wrong on the last count. These cubs are not carefree. They do not indulge in games to while away the years of their cubhood. These games are deadly serious business. They are studying the skills of hunting, the skills of survival. They are learning how to approach their prey without being seen, how to pounce, and how to grapple with and dispatch prey without being injured. They are learning by doing, but in a safe way. Better to make mistakes with butterfly and sibling than with the horns of the wildebeest.

Games are thus the most ancient and time-honored vehicle for education. They are the original educational technology, the natural one, having received the seal of approval of natural selection. We don't see mother lions lecturing cubs at the chalkboard; we don't see senior lions writing their memoirs for posterity. In light of this, the question, "Can games have educational value?" becomes absurd. It is not games but schools that are the newfangled notion, the untested fad, the violator of tradition. Game-playing is a vital educational function for any creature capable of learning.

The incidence of game-playing in animals is itself instructive. Game-playing has been observed only in mammals and birds. The phylogenetically earlier orders (fish, insects, amphibians, and reptiles) have not been shown to engage in game-playing. (See Animal Play Behavior, by Robert Fagen, Oxford University Press.) Game play seems to be associated with that quality which we have clumsily attempted to measure with brain size, intelligence, and ability to learn. This correspondence cannot be attributed to accident; clearly game play is an important component in the development of many creatures.

We commonly associate the playing of games with children. Indeed, "play" as an activity is considered to be the almost exclusive preserve of children, and the term is applied to adults either disparagingly or jocularly. Children are expected to play games because we recognize (perhaps unconsciously) the fundamental utility of games as an educational tool. As children grow up, cultural pressures change and they are encouraged to devote less time to the playing of games so that they can devote themselves to more serious activities.

I claim that the fundamental motivation for all game-playing is to learn. This is the original motivation for game-playing, and surely retains much of its importance. This claim does not conflict with my other primary assertion that computer games constitute a new art form. Consider, for example, humans and food. The fundamental motivation to eat food is the base desire for nourishment, yet this has not prevented us from embellishing this fundamental activity with all manner of elaborate and non-nourishing customs, rituals, seasonings, and garnishes. I do not mean to imply that food is an art form; only that we humans can take an activity far beyond its prime cause without denying that prime cause.

I must qualify my claim that the fundamental motivation for all game-play is to learn. First, the educational motivation may not be conscious. Indeed, it may well take the form of a vague predilection to play games. The fact that this motivation may be unconscious does not lessen its import; indeed, the fact would lend credence to the assertion that learning is a truly fundamental motivation.

Second, there are many other motivations to play games that have little to do with learning, and in some cases these secondary motivations may assume greater local importance than the ancestral motivation to learn. These other motivations include: fantasy/exploration, nose-thumbing, proving oneself, social lubrication, exercise, and need for acknowledgment. I shall examine each in turn.

Fantasy/Exploration

A very important motivation to play games is fantasy fulfillment. Like a movie, a book, or music, a game can transport the player away from the tawdry world that oppresses him and create a fantasy world in which he can forget his problems. Games are potentially superior to the traditional means of escape (movies, books, music) because they are participatory. Instead of merely watching a movie, reading a book, or listening to music, the player is actively involved in the game. Indeed, the player drives the game, controls it in a way that is quite impossible with the passive fantasies. This need to escape, to fantasize is certainly an important motivation.

Fantasy fulfillment frequently takes the form of symbolic exploration. There’s a big world out there, full of exciting things, people, and places, yet most of us are confined to a world ,of asphalt, plastic, and paper. Many art forms attempt to transport the audience into a different world, to present experiences or feelings not often known in the everyday world.

Consider, for example, the success of Disneyland. This place is undoubtedly the most successful of its genre. Such parks are often called "amusement parks" or "theme parks." These terms are misleading, for the success of Disneyland cannot be attributed solely to its amusements and diversions. These elements are technically excellent, but other amusement parks sport technically excellent rides. The success of Disneyland can be summed up in one word.

Nose-Thumbing

A common function of games is to provide a means of overcoming social restrictions, at least in fantasy. Many games place the player in a role that would not be socially acceptable in real life, such as a pirate or a thief. An excellent (albeit extreme) example of this is the game CRUSH, CRUMBLE, AND CHOMP by Automated Simulations. In this game the player is cast as a 1950's-vintage monster going on a rampage through his favorite city. He stomps on police cars, crushes buildings, swats helicopters, and creates general mayhem. The box art shows a monster about to attack an IRS building as terrified citizens flee. This represents an extreme case of anti-social behavior made acceptable by the safety of the game.

Sometimes the player's role is itself socially acceptable, but the actions taken are discouraged in real life. MONOPOLY encourages players to engage in what the Federal Trade Commission delicately calls "predatory trade practices." Wargames encourage players to start and win wars. Some games address sexual matters, allowing players to indulge in make-believe behavior that they could never exhibit in the real world.

The most telling example of this nose-thumbing phenomenon lies in the arcade games. These games emphasize violence, and lots of it. The theme is almost universal in arcades: destroy somebody. The coup de grace is not delivered discreetly or elegantly. On the contrary, the victim is dispatched with the most colorful animated explosion possible. Like a Sam Peckinpah movie, the violence is the whole point and purpose of the enterprise. Yet, even as we pander to these distasteful emotions, we delicately mask them in less offensive garb. We never, never obliterate human beings; instead, we vaporize ugly space monsters. The monsters have perpetrated some odious interstellar crime, so the player is cast as the defender, the protector, or the avenger. The case is often presented that the game represents a time of extreme crisis ("THE FATE OF HUMANITY IS AT STAKE!!!"). This heightens the player's sense of urgency; it also conveniently justifies the use of extreme violence, thereby allowing the player to have violence without guilt. The player can thumb his nose at social strictures and engage in violence and mass murder without risking censure. The game provides a safe way to thumb one's nose.

Proving Oneself

Another function of games is as a means of demonstrating prowess. All games support this motivation to a greater or lesser degree. Many game-playing communities sponsor tournaments or player ratings. Arcade games support this function by recording and displaying the initials of the top-scoring players. There are also players who carry this to extremes. Their prime goal is not merely to win, but to beat somebody, preferably somebody worth beating. Chess has an unusually high concentration of such sharks; so do wargames. A common question asked during a wargame is "Are you playing for blood or for fun?" Such players normally prefer games that allow their skill to be properly brought to bear, so they tend towards games in which chance plays a minimal role.

Despite this concentration of such players in deductive logic games, almost all games have sharks preying on the playful players. When a shark plays for serious rewards (e.g., social dominance) and -takes serious risks of failure, the crucial element of safety is eliminated from the game, and the game ceases to be a game; it becomes a conflict. Inasmuch as all games have the potential for being played in an overly competitive way, some people who are especially sensitive to the social risks of game-as-conflict refuse to play games, for they do not perceive the games to be safe. If they do play, they prefer to play games of pure chance, not so much to disable or discourage the shark as to create a situation in which winning is patently unrelated to prowess. If winning is arbitrary, social risk is eliminated and safety is restored.

It is impossible to design a game that is unalterably safe (i.e., invulnerable to sharks) without resorting to pure chance as the sole determinant of victory. If the game in any way allows individual prowess to affect the outcome, then the outcome is perceivable as a reflection of individual prowess. In most games, safety from social risk is conferred onto the game by the attitudes of the players, the willingness to say, "It's only a game."

Social Lubrication

Games are frequently used (especially by adults) as social lubricants. The game itself is of minor importance to the players; its real significance is its function as a focus around which an evening of socializing will be built. Card games and some light board games serve this function. An excellent example of such a social lubricant game is a game utilizing a large plastic gameboard about four feet square that is marked with colored spots. On each player's turn, a random process is used to determine which of four appendages (arms or legs) is to be placed on which spot on the board. As the players contort to fulfill the game requirements, they inevitably make physical contact with each other in innocent and foolishly humorous ways. Social interaction is thereby fostered.

Exercise

Exercise is another common motivation to play games. The exercise can be mental or physical or some combination of both; in either event, the game is an entertaining way to stay in shape. Some players like to exercise their cognitive skills, while others prefer the use of intuition. Some players prefer to exercise their athletic skills. Furthermore, players need to exercise their skills at an appropriate level. A chess player will get very little exercise out of a game of tic-tac-toe. Similarly, a person who finds tic-tac-toe challenging will get little useful exercise out of chess. These preferences sort players out and route them to the different games available.

Need for Acknowledgment

We all need to be acknowledged, to be recognized by other people. The acknowledgment we crave is not merely an acknowledgment of our existence, but of our personalities. For example, when we meet a casual acquaintance, we usually get a perfunctory acknowledgment ("Hello there, Jones.") We are more gratified when the greeting in some way acknowledges us as individuals with special personalities and problems ("Hello there, Jones; is that knee still bothering you?") The popularity of pets provide another example of the need for acknowledgment. Why on earth do we keep in our homes animals that require food, veterinary attention, and sanitary maintenance? Because they acknowledge us. We can interact with pets; we talk to them, play with them, and emote with them. A dog is an especially responsive creature; it can read our facial expressions and interpret our tone of voice. A smile will trigger tall-wagging; a kind word will precipitate jumping, licking, barking, or some other expression of affection. Goldfish, by contrast, neither appreciate nor express emotion. Thus, even though goldfish are much easier to care for, most people prefer dogs as pets. People value acknowledgment enough to expend the effort to obtain it.

This is one reason why interaction is so important to a game; it allows the two players to acknowledge each other. A truly excellent game allows us to imprint a greater portion of our personalities into our game-playing. Such a game allows me to play in a way that only I could have played it. My opponent must look beyond the playing pieces and acknowledge my cleverness, my rashness, my deviousness, my entire personality. When such a game ends, my opponent and I know each other better than we did before we sat down to play.

Summary

Many factors play a role in motivating a person to play a game. The original (and almost instinctive) motivation is to learn, but other motivations come to bear as well.

MOTIVATION VERSUS SELECTION

We must be careful to distinguish between factors that motivate people to play games in the first place and factors that allow people to choose between games. In other words, the answer to the question, "Why do people play games?" can be quite different from the answer to the question, "What makes one game more fun than another?" Some factors motivate a person to play games; other factors help that person select a particular game. For example, sensory gratification is such a selection factor. A player who has decided to play a particular type of game will prefer a game

with excellent graphics over games with poor graphics; yet the graphics alone will not motivate many people to play games. Motivating factors get people to approach games in general; enjoyment factors help them make their choice of particular games.

Distinguishing motivation from enjoyment is not tantamount to denying correlation's between motivating factors and enjoyment factors. Clearly, any game that does not deliver the experiences implied by the motivating factor will not be enjoyed. Thus, some (but not all) motivating factors will also be used as enjoyment factors. If a player is motivated to play a game for mental exercise, that player will probably prefer those games that offer better mental exercise than do other games. A game cannot be fun if its factors do not satisfy the motivations of the player. Two enjoyment factors that are not in themselves motivational are game play and sensory gratification.

Game Play

Game play is a crucial element in any skill-and-action game. This term has been used for some years, but no clear consensus has arisen as to its meaning. Everyone agrees that good game play is essential to the success of a game, and that game play has something to do with the quality of the player's interaction with the game. Beyond that, nuances of meaning are as numerous as users of the phrase. The term is losing descriptive value because of its ambiguity. I therefore present here a more precise, more limited, and (I hope) more useful meaning for the term "game play". I suggest that this elusive trait is derived from the combination of pace and cognitive effort required by the game. Games like TEMPEST have a demonic pace, while games like BATTLEZ0NE have a far more deliberate pace. Despite this difference, both games have good game play, for the pace is appropriate to the cognitive demands of the game. TEMPEST requires far less planning and conceptualization than BATTLEZONE; the demands on the player are simple and direct, albeit at a fast pace. BATTLEZONE requires considerably greater cognitive effort from the player, but at a slower pace. Thus, both games have roughly equivalent game play even though they have very different paces. Pace and cognitive effort combine to yield game play.

Sensory Gratification

Sensory gratification is another important enjoyment factor. Good graphics, color, animation, and sound are all valued by game players. They support the fantasy of the game by providing sensory "proof" of the game's reality. We see a related phenomenon in the movies: special effects. Some of the newer movies have excited great interest because of the excellent special effects they utilize. These movies have placed us in the thick of space battles, let us meet strange and wonderful creatures, and taken us to faraway places. The things we see look so real that we believe the fantasy; we know (subjectively) that the fantasy is real. Similar processes can be applied to games. Special effects, graphics, sound, animation-these factors all help distinguish a good game from a bad game. We must not confuse their role, however; sensory gratification is a crucial support function, not a central feature. Sensory texture enhances the impact of the fantasy created by the game or movie, but wonderful graphics or sound do not by themselves make the product. A movie without a believable or enjoyable fantasy is just a collection of pretty pictures; a game without an entertaining fantasy is just a collection of interactive pretty pictures.

INDIVIDUAL TASTES

So far I have discussed motivational and enjoyment factors as if they were absolute quantities whose significance is independent of the individual player. Such is not the case; the response to a given game depends heavily on the personality of the prospective player. How are we to deal with the personality differences that dominate the individual's response to games?

One academic solution to this problem is to postulate the existence of a very large number of personality traits that determine the individual response to a game. We next postulate a like number of game traits that, taken together, completely define the psychological profile of the game. Next, we measure and catalog all of the personality traits of any given individual, presumably with an omniscient "personalitometer". Then we measure all the game traits of the game in question with an equally powerful "gamometer". We then perform a matrix multiplication of personality traits against game traits. Sometime before the sun enters its red giant phase, our monster computer returns a number telling us how much that person will enjoy that game.

This approach will for the moment remain a gedanken-experiment. We must devise simpler, admittedly less reliable means of coping with individual differences. One alternative route is to observe and catalog groups of game-players, and identify the game traits valued by these groups. This method is made difficult by the youth of the computer game industry. We can at this time identify only a few broad, vague, and overlapping groups of players: skill-and-action enthusiasts, D&D enthusiasts, and strategy gamers. There remain several other game types, but they have not attracted so large a following as to present us with a definable group of players. The passage of time and further research will certainly give us more information with which to work.
Individual tastes in games are not static; as a person changes so do the tastes. The following analogy with music illustrates this point.

As children, we are all exposed to music in a variety of forms, but it has little impact on us because our tastes are poorly developed. We sing and dance to simple songs, but a full appreciation of the emotional range of music eludes us. The power of music arises from our ability to associate musical expressions with emotions. It takes years to develop these associations, and they are made in the context of our experiences. For many in my generation, the first deep contact with music came with rock 'n roll in the 60’s. The pounding beat, simple themes, and short durations were easily grasped by our adolescent and unsophisticated minds. We could understand this music. Moreover, the act of listening to and enjoying this music was itself an educational experience. As the range of our musical experience expanded, we learned more complex components of the musical lexicon and developed a wider range of associations. Soon we were able to understand and appreciate other musical compositions previously inaccessible to our untrained ears. Rock music changed to reflect this maturation; some of us stayed with rock. Others moved to jazz, country, or folk. Like some others, I moved from rock to classical in a series of stages. As I moved along this evolutionary path, the lessons of one stage enabled me to understand the material of the next stage. Other people followed their own paths, exploring and learning the areas of musical expression that most appealed to them. The common experience was that our musical tastes evolved, no matter what direction we chose. Rock music was the broad base we all shared, the entry point or ,junk out of which sprang many branches.

Just as rock 'n roll was the entry point into the world of music for an entire generation, so will skill-and-action games be the entry point into the world of games for the whole population. Like early rock 'n roll, skill-and-action games have broad appeal, and are easy to understand. As people become more sophisticated with games, their tastes will evolve down different branches. Like rock 'n roll, skill-and-action games will not go away; they will change to reflect the evolving taste of the public. We can see this happening already. The early arcade games are tame pussycats compared to the rip-snorting, fire-breathing games of 1982. Had TEMPEST been released in 1977, it would have intimidated and repelled players. Times change; people change. Skill-and-action is here to stay and will always provide an entry point for new players, but the public will not stand still. Many people will move on to explore other areas of game-playing.
People play games for many reasons. In this chapter, I have touched on a variety of these motivations. I readily admit that my treatment of the subject matter is thin, speculative, and uncompelling. People are complex creatures; we will never fully understand human motivations to play games. Yet me must appreciate the importance of these motivations and at least try to understand them if we are to master the art of computer game design.

end

The Education of a Computer Game Designer

by Chris Crawford

So, my young friend, you want to be a game designer, and you have turned to me for advice. I will offer you my best advice, but I suspect that you'll reject it and take the advice of those who tell you what you want to hear. But that's fine with me -- all I can do is tell the truth and hope that it will get through to a few people.

First, you must make a major career decision: training or education? Training gives you specific skills that you can use to get a job straight out of school. Education gives you broader skills that won't have immediate application, but will in the long run serve you better. It's basically a choice between a quickie approach and a strategic approach. If you're in too much of a hurry to plan strategically, then go ahead and attend a school where they'll teach you the details of handling the latest, greatest computer technology. Energy, not patience, is the strength of youth, so I can understand if you just can't stomach the thought of not plunging straight into your avocation. When I was your age, I too was impatient with all the irrelevant courses that the University forced upon me; now I blush at my impertinence and thank those teachers who pushed me so hard.

The quickie route will indeed yield faster results. If you attend a school that is dedicated to game design, or major in computer games at a decent college or university, you'll likely learn many of the details of present-day game design. You have a good chance of landing a job right out of school at an actual games company, working on games before you're 23.

But hold on here, hotshot. There's a difference between working on games and designing games. That first job you land will surely be the gruntiest of grunt jobs. You'll be assigned to some tiny task, like animating a minor character in the game who does nothing but walk across the background, or writing the code that asks, "Are you sure?" when the user decides to quit the game. If you do a good job with that, after a few years you might get promoted to handling more complex animation, or writing a more important piece of code. And after a few more years, you might even get promoted to a position where you're handling some pretty serious work.

But don't count on it. The basic problem is that there are hundreds of thousands, perhaps even millions of students just like you who are bursting with eagerness to become part of the computer games industry. Think in terms of supply, demand, and price. When the supply of workers is ten or a hundred times greater than the demand for workers, the price goes way down. You can expect to be paid starvation wages, and you probably won't be treated with any respect. You can complain, but the answer they'll give you is simple and honest: if you don't like it, feel free to quit. There are a hundred more kids just like you who are dying to have your job.

In fact, that is exactly what happens. Sometime you ought to wander around the halls of the Game Developers' Conference; it's held in San Jose every March or April. You don't need to actually pay the money to enter any of the events; just wander around the San Jose Convention Center and take note of the people in attendance. You'll find two surprising rules: first, everybody is dressed in black, and second, the average age of the attendees is between 25 and 30.

I don't know why everybody dresses in black; it seems to be a standard that everybody conforms to. I can, however, tell you why they're all so young: everybody leaves the industry after a few years. The games industry is like a big building with one entrance and a lot of exits. There are thousands of eager young kids crowded at the front entrance, pushing and shoving to get inside; only a few make it in. But for every person who gets in, another person leaves -- that's what keeps the industry in balance. And the fact that so many of the people in the business are so young demonstrates who quickly people bail out of the industry. Not many survive until their thirties.

If you think about it, it really does make sense. If there are thousands of kids eager to work for peanuts to build games, then you can hire them at a dime a dozen, work them like slaves until they drop, and then hire replacements. You need only a skeleton crew of managers to keep the kids working. The system works perfectly.

The only question is, do you want to be part of this system? I hope not. However, if you're too fired up with enthusiasm about making your big break into the games biz, then go ahead -- no amount of talk from an old fool like me will deter you. You just have to learn these things for yourself.

But there is an alternative I can offer you. Here's how it works. First, get yourself a real education, not some one-night-stand training. Go to a real school and major in anything except games. Almost anything will do: biology, physics (that's where I got my start), art, literature, history, psychology, linguistics. Just make sure that you get what used to be called a "liberal education". Take lots of courses outside your major. And yes, you should probably minor in computer science.

On the side, you should be experimenting with building games. Don't go for the snazzy graphics just yet -- that can always be slapped onto the design. You want to concentrate on the guts of the game, the architecture and game mechanics. How do the little gears and levers inside the game operate? Don't try to build games that are just as good as the commercial games -- for crying out loud, those games have dozens of people working on them; anything that little ole you can do will look pretty pathetic next to those extravaganzas. Think of your process as rather like building a car. Don't worry about the chrome and the paint job just now; you want to concentrate on learning how to put pistons together, how the valves operate, what the carburetor does. You want to build little go-karts, not shiny Rolls-Royces. They're all experimental; you should never think that your designs have any commercial potential. Build them and throw them away. Creativity requires you to murder your children. If you are so enthralled with your designs that you can't let them go, then you'll never have the hard-bitten creativity of a truly good designer.

Meanwhile, keep building the intellectual foundations for your creativity. There's no way you can compete with the formidable creativity of a seasoned game designer, so for now, concentrate on building your strength. Hey, even Neo couldn't take on Agent Smith until he had spent enough time building the foundations of his skills. Learn everything you can. Do not graduate without having examined every bookshelf in your library; you'd be surprised what interesting things you will stumble on in those dusty aisles.

Once you get out of college, don't rush into the games biz. Get a real job at a real company and earn some money, but keep expanding your education. You'll learn a lot about organizational behavior and how to handle yourself in a corporate environment. You'll learn how and when to stand up to your boss -- which is rarely, by the way. And you'll prepare yourself to swim with the sharks when you do enter the games biz.

But continue to work on games in your spare time. Build lots of different games go-karts, trying out each one for its handling, its speed, and its other characteristics. Once you've gotten six or ten games built, you might want to think about putting together a substantial project, but still on your own. Recruit a few like-minded folk to help you out, and build something really impressive. Show it off to the world. Then you can use that game as your resume when you do apply for a position in the games industry. If your game is good enough, you'll get a job as an actual game designer, not some dime-a-dozen minion. You'll still be a junior assistant to the assistant game designer, but you'll be in the right place, and if you work hard and do your job well, you might actually have a future in the games biz.

I realize that this is not what you wanted to hear. What you want to hear is a quick fix. Take such-and-such courses and you'll be guaranteed a high-paid job with a big office, all the best computers, and complete creative control. Sure, everybody wants that -- but nobody gets it. Anybody who tells you that kind of story is a shyster trying to get your money. The sad fact is that the pioneering days of game design are over and it's now a big industry; nobody gets "discovered" and turned into a superstar overnight. It's a long, long slog for beginners.

You've got the passion, the energy, and the drive to make it happen -- do you have the strategic insight to plan for the long slog, or are you going to rush in before you're truly ready?

Good luck, kid. I'm rooting for you.

end

Life as a game developer

Part 1 - Getting a job

By Ryan Mc Mahon

Is the video game industry right for me?

Just about everyone who spends a lot of time playing video games sooner or later wonders about the possibility of making them for a living. Yes, there really is a whole industry devoted to the creation of these electronic amusements, with its own trade magazines, yearly conventions, tools, economics, culture, language, and attitude. However, the first thing anyone should know about the profession of game development is that it's **NOT** an easy one. Making it in this industry takes hard work, attention to detail, a smart understanding of the average game player's psyche, good business sense, organization, teamwork, and the iron will to stick to a project for months after the initial excitement factor has worn off. The cool ideas are truly just the tip of the iceberg -- for every minute spent dreaming up features, game developers must put in hours and hours of painstaking busywork to bring them to life. A lot of the time, working on games seems to be a never-ending list of code to tweak and pixels to push.

But on the other hand, every job has its tedious, exhausting parts -- only in this one can you say that you're making something flashy and cool for others to play and enjoy. I always knew I wanted a job like this, and I never consider the possibility of going back to writing business software for faceless corporations (except to scare myself into working harder). I think that most people who become game developers have a similar sense of belonging here. Like born musicians, artists, writers, or engineers, they have certain innate talents and see the game industry as a natural place to apply them. Are you a natural? Well, the only way to find out is to start programming, building levels, creating character models, or designing gameplay. You may find that the work is less fun than you expected it to be. Or you may find that, heck yeah, you're so good, you deserve to get paid.

Is this game for you? What isn't so great about this crazy industry:

Let me begin with a dose of reality. The video game industry is very, very competive. There are hundreds of studios pumping out games, most battling for limited shelf space in retail stores. Any game company that wants to get its product under lots of Christmas trees has to work relentlessly to keep up with the latest technology, stay within financial budget, stick to a tight schedule, and still make something that lots of consumers will enjoy playing. For many companies, especially newly formed ones, the pressure results in long hours, relentless deadlines, and a generally high stress level. If you don't deal well with a demanding, chaotic work life, this industry is not for you!
In recent years, the fierce competition has driven the production values of retail games to a high benchmark. Consumers now expect a lot of bang for their buck, with features like CGI cutscenes, multiplayer modes, soundtracks by popular artists, and lots of other bells and whistles. As a result, production and marketing costs are high, and profit margins notoriously slim. Around half of all projects never get finished, fewer than 5% make much money. It's not uncommon for game companies to run out of capital and fold.

Another problem with the game industry is its often adolescent personality. There is an endless flood of inexperienced young people who are eager to get in, and willing to work long hours for low pay (probably like you). The result is plenty of immaturity, a lot of naivete, more than few inflated "rock star" egos , and an overall lack of professional knowledge and discipline. Some companies fall apart simply because their members can't come together and make a game the sensible way. By the time someone has been in the industry long enough to know how the system works, he or she may well be burned out.

The very last thing to keep in mind about joining this industry is that you'll be working on someone else's game, not necessarily the kind you want to make. The business ultimately revolves around what sells to a wide audience, so salaried game developers, even the best ones, inevitably must put up with the demands of management types (the ones with the money). For that reason, and the ones mentioned above, some people choose to be "indie" developers, earning little or no money, but making games on their own terms. Others just accept the cog-in-a-wheel nature of the business.

Without a doubt, if you want to get into the business of making video games, be prepared to pick up a few bruises during your career. There are no guarantees that you will always have fun, or even like your job.

What's great about the game industry:

columnist Earnest Adams, it's the only game industry we have. And it's one that attracts a lot of creative, brilliant, intense, and unconventional people. Look at the progression of video games over the past twenty years, not to mention computer technology in general. The outside world can call us freaks and geeks if they like -- the fact is, this is a pretty happening place to be! Just pick up a copy of Wired or The Wall Street Journal, and you'll see that all this energy has not gone untapped by the business world. Large corporations like Sony and Microsoft are pumping billions into owning a share of the market, and have top engineering teams hard at work creating the next generation of game machines. There's no slowdown in sight!

Then, of course, there's the fact that, hey, you're making video games for a living. There's nothing quite like watching a project that you've contributed to grow and take on a life of its own. When that piece of AI code finally works, or when you get that detailed character model walking around, or when you've tweaked that level to be just right, it's a wonderful feeling. As the game comes together, creating new waves of inspiration among you and your teammates, the sense of accomplishment is a true rush. And when your game finally makes its splash, it's enormously gratifying to see a fan club grow around it. Imagine walking into a store, seeing the fruit of your labors on the shelf, and thinking, "I made that." Despite their youthful faults, game developers are a close-knit bunch, and there's a lot of camaraderie and energy in the field. There are some very talented people who spend their days making games simply because they love doing it, even though their abilities could earn them more money for less stress somewhere else, and life in the game industry is kept interesting by the idiosyncratic, spirited, and off-the-wall natures of these individuals. If you are fortunate enough to join a good team, you will find that you actually look forward to coming to work each day. It's not an average job.

Many great books to look into!

What kinds of jobs are there in the game industry?

There are several different career tracks within the game industry. I cover each one in its own section on subsequent pages, but here's a quick overview:

1. Programming: Programmers write the software that underlies a game. Without them there wouldn't be a game industry. In the old days, the people who made up the industry were mainly recreational programmers with ideas for games and some ability at drawing pixel art. Today, programmers have a more serious and exclusive focus on technology.
2. Art: Artists create the visual elements and aesthetic of a game. Their work includes 2D art, 3D modeling, and animation.
3. Design: Game designers are the people who come up with the ideas and content of a game. Their work includes coming up with gameplay and story concepts, writing design documents, and building levels.
4. Production: Producers are the management and logistics people in the game industry. They handle all the miscellaneous tasks related to getting the game done on time, such as managing QA, keeping track of game assets, coordinating with the publisher and outside contractors, and managing the game's schedule.
5. Audio: Sound and music composers in the industry make a living working for big companies, or on contract.
6. Quality Assurance: The job of QA revolves around play-testing a game in order to find bugs in it. This is not particularly glamorous or lucrative work, but it's the usual training ground for future designers and producers.

No career track is necessarily more important or fun than the others. It takes all of the above to make a great game. Each job has its thrills, tedious chores, risks, and rewards. Which one is for you depends on what kind of challenges you like the most.

(You should note that programming and art jobs are a lot easier to get without industry experience than design or production jobs are. This doesn't mean that art and programming are easier - it just means that you can pick up experience and training in these areas outside the industry.)

What is the business model for the game industry like?

Developing a video game requires funding. A team must pay for its equipment, its software (usually!), its licensing fees, the manufacturing, marketing, and distribution costs of its game, and, of course, the salaries of its members. Since few game developers are independently wealthy, a typical game company must form a relationship with a publisher (e.g., Activision, Electronic Arts, Sony, Nintendo). Typically, this involves negotiating a contract, designed to insure that the publisher won't be throwing its money away on a questionable product or team. In many deals, the publisher will usually gain influence over the design of the game, control of intellectual property, non-competition agreements from the company's key members, and, of course, a cut of the profits. In return, the publisher pays for agreed-upon expenses, and marketing and distribution costs. Some developers are closely tied to (or part of) their publisher, which brings them advantages in extra manpower, special projects, and hot licenses, in exchange for independence.
The system certainly has its critics within the development community, but the way of the western world is that "the money" makes the rules, even if the rules are less than ideal from the point of view of "the talent". For better or worse, game developers who want to pay the rent must learn to live with capitalism, and to leverage it to their advantage, even though it may mean compromising one's vision and "selling out" a little.

(A more complete description of the game development business model can be found in a few books that exist on the subject.)

How hard is it to get into the video game industry?

It depends on exactly what you want to do, but in general, the video game industry is not easy to break into. It takes a lot more than simply having ideas for games to get a job. This is a highly competitive business, and game companies provide little in the way of formal job training. With the enormous number of people who want to get in, they don't have to. Plenty of applicants are willing to learn the necessary skills on their own.

To land a job, you first need to consider which career track in the industry you are best suited for, be it programmer, artist, design, or something else. There are no "sit around and think up game ideas" jobs, especially not for entry-level candidates. There are few story-writing jobs. There are few openings for people who have basic skills in programming, art, and design, but aren't particularly adept in any one area. You need to pick one.

Even when you have a career track lined up, getting a job is still tough. You'll have to have demonstrable talent, ability, and passion for the kind of work you are applying to do, above and beyond the average wannabe. You'll also have to be persistent in getting your resume out there, since entry-level job openings tend to be snapped up fairly quickly.

What qualities do game developers look for in new hires?

Skill: You have to have proven talent in the specialty you're applying for a job in. You don't have to have contributed to a full-blown game before, but it makes a big difference if you've worked on small games or projects that involve related skills. Your background shouldn't say, "I wanna be..." It should say, "I AM".

Self-motivation: You have to be the kind of person who attacks challenges and can work with little supervision. You'll need to be able to quickly adapt to a company's way of doing things, to keep up with changing technology, and to push yourself to take on new responsibilities.

Teamwork skills: Game development is a highly cooperative undertaking between people with different skill sets, which makes solid communication skills, willingness to compromise, and a generally positive personality a must. You'll need to be good at gathering information from your coworkers and at explaining things to them.

Knowledge of the industry: Even entry-level hires need to be informed about the state of the art of making video games. Read articles in Game Developer and other places to keep up with the latest tools, techniques, and technology.

Passion for games: This one seems obvious, but is easy to understate. If you want to make it in the stressful world of game development, you really have to love video games. The ideal job candidate is constantly thinking about how they work, what makes them fun to play, and how they can be improved.

Although a lot of "help wanted" ads are aimed at people with "industry experience", there is sufficient work for those who aren't industry veterans. Entry-level hires are useful to game companies because they provide fresh energy, can be given work that more senior members aren't interested in doing, and can grow to fit the company's style. Sometimes, industry outsiders are actually more productive than certain jaded, sloppy, and whiny so-called "professionals".

SOON From Vertigo Games

Should I get a degree?
Although it wasn't always so, it's become difficult to get into the industry without some sort of degree. Nowadays, most game companies expect applicants for entry-level jobs to have one. Why? Having a college education shows that a person is disciplined, goal-oriented, and capable of handling assignments on his or her own. Furthermore, if someone has a degree in computer science or art, then you can depend on his or her having a certain knowledge base.
A college education will teach you the theories and concepts underlying a field. There's a lot more to programming than just knowing C++. There's advanced mathematics, data structures, algorithm design, computer architecture, and software engineering. All these fields of knowledge connect to game programming, and there's been a **TREMENDOUS** amount of study devoted to each. Why not learn from the expects? Some "self taught" game programmers will scoff that a degree is useless, but the fact is, I had been programming for nearly eight years prior to college, and I found my computer science courses highly enlightening. Video games are becoming more and more like any piece of software, and it pays to get formal training in what really involves a large amount of theory.
As for art, even though artistic ability is a largely innate thing, there is a lot an art student can learn from being exposed to the theories and techiques of perspective, composition, form, color, expression, etc.; or to the styles of famous artists and movements through history.

Similarly, an aspiring designer can only gain from courses on game theory, psychology, literature, politics, physics, art appreciation, written communication, history, computer programming, biology, and mythology. In fact, I think that a good designer has to be a veritable Rennaisance (Wo)Man, interested in a wide variety of subjects, and the underlying connections between them. If you want to be a game designer, for God's sake, don't just play videogames, read Tom Clancy novels, and watch action movies. No, be interesting and deep. Learn to play Go, read Gravity's Rainbow, go to art museums. Get out there and go for the **edge**. If videogames are a true art form, this industry **needs** it.

Another thing to keep in mind is that you may not **ALWAYS** want to work in the game industry. Although working on videogames for a living may sound like neverending bliss to you, trust me, the long hours and chaotic atmosphere can take their toll over a few years. It's wise to keep your options open so that you can find work or study elsewhere if you get sick of video games, or just want to take a break from them for a while. Many professional jobs require at least a bachelor's degree.

Finally, going to college is a great life experience. You live away from home, meet people from different backgrounds, try new activities, and study topics you may not have thought of studying before. Then there are road trips, late night "deep" conversations, parties, opportunities to meet members of the opposite sex free of all that baggage we picked up in high school, all those quintessential moments unique to college. Over those four years, you grow in intellectual awareness, confidence, and maturity. If I had to list the most valuable lessons I learned in college, many of them would have nothing to do with video games or computers. So, if you have any brains or talent, don't even **CONSIDER** not getting an education of some kind. Your future will thank you.

How good are schools like Digipen and Full Sail?
This is a section of my website that generates a lot of questions. Digipen and Full Sail are schools that exist with the goal of training people to be videogame developers. , according to its website, offers a 2 year degree in "3D computer animation" and 2 and 4 year degrees in "real time interactive simulation". The latter is a subset of a traditional computer science degree, focused on game development. (whose website is the cheesier by far) is a school with a broader multimedia focus, offering studies in computer animation, digital media, film, show production, recording arts, and game design and development. The 'game design' program takes 15 months and blends programming, game design, media, and business courses. From a little Net research, I'd have to say that Digipen, with its exclusive focus on games and deeper course load, seems to be the better of the two schools. Full Sail seems to be, at best, more like a supplement to a regular education than a replacement for it.

In its early years, Digipen wasn't highly regarded within the industry, but it's since built up its program, becoming a viable alternative to a regular college degree for aspiring game developers. I remain neutral about whether it's "better" to go to such a school than to a university; both choices have their tradeoffs, and game companies, overall, don't seem to have a lot of preference for one route or the other. Obviously, the pool of regular college grads is much larger than Digipen's, and game companies are already well accustomed to hiring from it.

Here are the main facts I've heard about Digipen:

- The main selling point of Digipen is that students get **A LOT** of hands-on experience working on games. The workload is intense and the students very dedicated. Some attendees quickly find out that game development isn't for them, whether due to lack of lasting interest or lack of talent. Unlike with a regular college, you should be sure about your career choice before going.

- Digipen does not provide a broad, general education, but a narrow, focused one aimed at getting students jobs at game companies. Personally, I think that an education is about more than just getting a job: it's about expanding your mind, teaching you to think in new directions, and making you a more interesting, adaptable human being. Since game development is such a rapidly expanding field, it's worth knowing about topics beyond its borders, since they may well soon be a part of it. A traditional computer science degree will give you a much broader understanding of software development than Digipen's will. Plus, you may not want to spend your whole working life as a game developer, and a regular college will allow you to explore other possibilities while in school, even if they are not immediate career choices for you.

- Digipen isn't currently an accredited university, so while its programs are comparable to an art or computer science degree, they are not true BFA or BS degrees. I doubt that this distinction would make a difference to very many game companies, but it is an important detail to keep in mind if you anticipate seeking work or education beyond the game industry, where paper credentials are given more importance.

- Digipen is fairly expensive, about $50,000 for the four year course of study, $25,000 for the two year one.

I don't know much about the quality of Digipen's courses and staff, but students there say that they are excited and are well-challenged. Game companies seem to find Digipen graduates to be competent, still having some rough edges that need to be smoothed out with on-the-job experience and guidance, but capable of cutting it (which is my assessment of the one Digipen grad I've worked with). As with a regular university grad, a Digipen grad's long-term success in this field relies heavily on his or her own talent, drive, and intelligence. Some fizzle out, some make waves. No path is guaranteed.

Digipen and Full Sail aren't the only schools to capitalize on the videogame market. In recent years, there have been a number of game development related programs and courses of study that have popped up at schools around the country, some more technical in nature, some more artistic. Some are course concentrations within general curricula, some are special programs unto themselves. One example is the program at Carnegie Mellon University. At this time, I don't have much specific knowledge about the various offerings out there, but I suggest you look around.

Sites to look into!

DIGIPEN INSTITUTE OF TECHNOLOGY

FULL SAIL Real World Education

Game Maker

How to learn on your own

A common complaint on game industry newsgroups is: "Game companies all want to hire people with experience! How do I get experience if no one will hire me?"
The usual response is: "Make some games on your own!"
It's not that hard to do. All you need is a computer, an Internet connection, a little free time, a little money, and a lot of willpower. You don't need to make a technically amazing game, or even one that's fun to play. Just make something that you learn from, that brings you closer to understanding the issues that the pros deal with every day. It can be crude, it can be buggy. Just get it done.
When that happens, pat yourself on the back, take measure of the lessons you learned, then start a new project. Try to make it a little more challenging, a little more polished, a little more professional, a little cooler than the last one. Repeat the cycle enough times, and you'll eventually have a game that you'll be proud to include with your resume.
When it comes to finding information and tools, the Net is your friend. A lot of information there may be disorganized or outdated, but there are thousands of resources, and a large hobbyist community. For programmers, there are a ton of game programming books, and a lot of web sites. For artists, there are good freeware packages and lots of models and textures to play around with. For designers, there are packages that enable you to modify existing games, to create your own levels, characters, and gameplay, or even turn them into whole new games. The ease with which data can be swapped on the Internet makes it a perfect resource for aspiring developers.
Don't, and I mean **DON'T**, attempt highly ambitious projects while you're still learning the ways of game development. Though it's fun to dream up grand designs, you'll never compete with the production values of commercial games, and you'll eventually get stuck in the mud. Stick to projects you know you can finish. You'll learn a lot more that way, and will gain valuable confidence when you have a game that you can call "done", even if it impresses no one other than you.
Also, if you plan to work with friends, keep in mind that this presents its own challenges. You'll need to develop and maintain a clear plan for what you're going to do and how you're going to do it, otherwise everyone will just wander off down their own separate paths. Generally, the fewer people who are involved, the less of a headache team management is.
Overall, I would say that learning to make games requires being both patient and persistent. Break the challenge down into a set of simple and realistic goals. Give information a chance to sink in. There will be times when you'll feel so frustrated, you'll want to throw your keyboard at the wall. There will be times when things suddenly click in your head and you'll find yourself dancing around the room. Both experiences are part of being a game developer.

Salaries of Game Developers

According to the salary survey in the July, 2001 issue of Game Developer (a lot more data can be found there), the average salary for a person with one to two years industry experience in his or her field is around $55,000. It's about the same in each career track, but varying somewhat in different parts of the USA. At six years in the industry and above, a programmer makes around $85,000 a year on average, an artist around $70,000, a designer between $65,000 and $80,000, and a producer around $80,000.

My experience is that entry-level salaries are a bit less than this. A junior programmer or artist is more likely to be offered $35,000-$50,000. The exact number is affected by location, the experience of the applicant, and the funding level of the company (cash-strapped shops are more likely to hire an inexperienced candidate... cheaply). Most established companies provide standard benefits like health care and a 401K package. Many also sweeten the deal with royalties, a share in the profits of the games you work on.

If you sign on to work for someone else's company as an artist, programmer, or designer, don't plan on getting rich. Salaries for in-the-trenches developers who have lasted 15 years or so in the industry cap out at around $150,000-$200,000. Royalties and stock options only kick in significant cash when your game or company does exceptionally well, in which case you might become comfortably wealthy, but don't count on it. If you want to make lots of filthy lucre, start your own company and make it successful enough to be bought out, or get an MBA and start working your way towards upper management.

How are game companies run?

There are a variety of different schemes and strategies for running a video game company, some more successful than others. Some studios have a college dorm feel, where empty pizza containers, heaped reference manuals, and sci-fi action figures make up the office decor, where the game design changes and mutates according to the daily whims of the team, and where employees can be found in the office all hours of the day and night. Other studios have more corporate structure, where cubicles partition the office, where employees have rigidly defined jobs, and where the feeling isn't all that unlike an assembly line. Of course, neither of these two extremes is ideal (although there are plenty of companies that exemplify one or the other) -- it takes a balance between individual freedom and collective order to make a game company work in the long run.

Core project teams in the industry can consist of anywhere from one person to several dozen people (not including the various support functions such as publishing, marketing, or contracting). Ten to twenty-five, however, is most typical for retail games. Among the different job groups, my own company breaks down into 9 programmers, 11 artists, 7 designers, 5 producers (including the president), and 1 receptionist. It's hardly a static figure, or one that represents all companies, but it should give you an idea. Testing and audio involve outside help for us.

The life of a game project usually goes something like this: A team comes up with an idea for a game; develops a conceptual design, a technical analysis, and a prototype (a very simple first pass at the game); and pitches the whole thing to a publisher. Once the publisher is satisfied and the contract signed, the design team will work out a more detailed design document, the programming team will develop a technical design, the art team will do some "look and feel" storyboards and sketches, the production team will put together a schedule, and management will focus on securing whatever assets are needed and hiring people to staff certain positions.

Then, the "construction" phase begins. During this time, all team members are working full tilt at creating code, assets, and content for the game. Work is merged as frequently as possible, with the intent of keeping the game at least roughly playable throughout the project. The team leads and the publisher will set milestones for the project, indicating what work is to be complete by when. Often, some design rebalancing will need to be done along the way, and possibly a few feature cuts. Eventually, the game will reach its "alpha" stage, at which point it is playable in "rough draft" form. After that comes "beta", where the game is pretty much complete, barring a few bugs. Finally, the game is run through a quality control check, is manufactured, and released to the public.
Many companies will often try to overlap projects, so that while one is winding down, an advance team is laying the foundation for the next. A single project typically takes anywhere from eight months to two years (longer than that, and you will probably find yourself being lapped by changing technology).

What makes a game company successful? I think that the key ingredients to the magic formula are an attitude of professionalism, an intense desire to make cool games, a healthy respect for the gaming public's interests, a good relationship between management and the development teams, and a clear set of goals.

Location of Game Developers

Many game companies are located around several big cities in the US (and UK). A lot of well-known companies are clustered around San Francisco, LA, San Diego, Austin, Seattle/Vancouver, and London. Smaller enclaves can also be found in or near Boston, New York, Washington DC, North Carolina, Chicago, and Tampa. (Then there's Japan, of course, but I'm not sure of the cities)

The fact is, game companies are popping up (and shutting down) all over the industrialized world. You can find game companies from Malaysia to Croatia. There might even be one quietly working away in a small, non-descript office right down the street from you. The best way to locate game companies is simply to get online. Use a search engine, ask around on message boards, and check the company directories at Gamasutra.com. Some companies don't advertise their location, so it takes a bit of detective work to find them.

What's the best place to work? In searching for a job, you should take into account the number of companies in the area you are thinking of moving to. Given the instability of the industry (as mentioned above), it's good to have other job prospects nearby in case one doesn't work out. You should also consider the cost of living in the area when evaluating a job offer. For example, $60,000 a year might seem like a lot of money, but in San Francisco, it doesn't go very far. (Personally, I'm happy with the LA area. It's warm and sunny, and there's plenty to do if you can put up with a little bit of traffic, smog, and superficiality)

Are the hours really that bad?

Horror stories abound of sixteen hour workdays lasting for weeks at a stretch, and a few people have asked me about this. Unfortunately, at some companies, bad management turns what ought to be an enjoyable job into a brain-sapping death march. Few games developed under such conditions turn out to be very good, and company morale is usually destroyed in the process.

Most companies aren't that bad, but crazy hours do come with the business, particularly around the infamous crunch time. Expect to put in at least 50 hour weeks under normal circumstances, maybe 60 or more as deadlines loom. Also, be prepared for a working environment somewhat less organized than what you might find at a "regular" company. If you're the kind of person who dislikes chaos or stresses easily, you may not fit in well.

Also, be warned that years of the kind of long hours and stress that the game industry sometimes produces can lead to physical or emotional health problems, especially if you put your job ahead of having a normal life. I've heard stories of overworked people being driven to substance abuse, nervous breakdowns, or, in one case, suicide. Be careful and pace yourself. Don't work for any company that demands 110% from its employees, all the time. While it's not necessarily that bad all the time or everywhere, there is a high risk of burnout in this field.

How is the current economy affecting the industry?

Unfortunately, the economy has slowed down over the past year, thanks to recession, recent terrorist attacks, and the possibility of war. This has had an effect on the game industry, making publishers more conservative about what projects they invest in, and game companies more reluctant to hire new staff. However, even (or especially) in troubled times, the public still wants inexpensive diversions, and video games help fill that role. So, unless all hell breaks loose, the economy is destined to pick up again, and video games to rebound. So, sit tight, and don't get too discouraged if you are having trouble finding a job in the industry. While you wait for things to get moving again, keep learning. As crazy as things are right now, history has shown that they can be a lot worse.

Is it harder for older people to get into the game industry?

"Older", for purposes of this site, means 35 or above. Realistically, yes, the game industry is less friendly towards folks who want to break into it, but are no longer in their twenties. If you've comfortable settled into the wages and hours of a "normal" job, the low salaries and hectic schedules of an entry-level game development job may not be very attractive to you, especially if you're burdened with family responsibilities. Also, you can expect a bit of prejudice from people who have pursued a ambitions to be game developers since their teenage years or younger, and might regard you as a Johnny-come-lately. And, sadly, too many management types in the industry equate youth with "cheap, eager labor". Still, not all companies share the same attitude. If you are particularly talented and experienced in a field that has application in video game development (for example, you're an animator from the film industry or have a Phd. in Mathematics), there is opportunity. However, it's up to you to get past the age barrier and really sell yourself to game companies.

Is it hard for foreigners to get into the US game industry?

Securing an H-1 visa for an overseas hire involves a certain amount of effort and expense for a game company, but some jobs in the industry are difficult to fill with US applicants alone. If you are looking to get a job as a programmer, you're in luck, because good American programmers are tough to find (mainly because of the large job market - I'll ignore comments about our education system, thank you very much). About half of the programming team at my company is foreign (English), and similar ratios seem to exist at many other companies. For other career tracks, being foreign is somewhat more of an obstacle. One feature to being in the US on a visa is that it's much harder to pack up and quit a job than it would be for a native. Thus, some companies see foreign hires as being more "reliable"! (take that for good or bad as you will)

Be warned: being in the US on a work visa means that if you get laid off, you only have two weeks to find another job in your field before being forced to leave the country. In a bad economy, two weeks is rarely enough time. So, be prepared for this possibility, and explore green card options with any company that offers you a job.

What's it like to work on console games vs. PC games?

Both worlds have their pros and cons. The PC side regards itself as being the home for "serious" gamers and its games as being deeper and more sophisticated than the "kids' stuff" on consoles. Console developers see themselves as being more disciplined and businesslike, while PC developers are "spoiled" by the plentiful amounts of memory and more friendly API's on their Windows machines. There's a bit of truth and a coating of ego to both views.
From a business standpoint, console games definitely make a lot more money. The gaming systems are cheaper to buy, and therefore more widely distributed. While the PC world has a few smash hits, like Everquest or Quake, the PC market in general is in a bit of a slump. My prediction is that the PC will continue to be the main forum for a few game styles (the ones you just can't play without a mouse and keyboard), while more and more developers move into the expanding console market (witness the sheer hype-storm over the Playstation 2).

From the development standpoint, PC's are definitely the more "open" system. While console developers have to start from scratch at learning the inner workings of each new generation of gaming machines, the technology on PC's follows a more continuous curve. Also, PC developers usually have a lot more memory and overall processor speed to work with. The main advantage to the "closed" systems of consoles is that they are fixed targets. While PC developers have to support a variety of graphics cards, memory allotments, and system configurations, console developers can depend on ONE system being in every home. Although consoles are trickier to work with and impose more constraints upon the developer, their architectures are better streamlined for games. By the time a console's had its run in the market, developers will have learned many of its tricks, and the last generation of games will be much better than the first.
Now, with hard drives, online capability, and other PC-like features on the way, the console world is starting to encroach on its big brother's territory. We shall see.

Bad game designer no twinkie

By Ernest Adams

Lately I have been playing a number of old games, and I've noticed something interesting in comparison with today's games. The technology has changed enormously, of course. But some of the design mistakes we made in the past are still being made in modern games. The same irritating misfeatures and poorly-designed puzzles that appeared in games as early as fifteen or twenty years ago are still around.

Herewith a list of game misfeatures that I'm tired of seeing. This is a highly personal perspective and your opinion may differ, but to me, these are a sign of sloppy, or lazy, game design.

Boring and Stupid Mazes

The original text adventure, Colossal Cave, had two mazes. One was a series of rooms each of which was described thus: "You're in a maze of twisty little passages, all alike." The other was a series of rooms described as, "You're in a twisting little maze of passages, all different" (or "You're in a little twisty maze of passages, all different," or "You're in a maze of little twisting passages, all different," etc.). These were the prototypical boring and stupid mazes. Colossal Cave was the first adventure game ever, though, so I cut it a little slack. But that was over twenty years ago; there's no longer any excuse for doing that now. Somebody gave me a copy of The Legend of Kyrandia a few years back, and I played it with some pleasure – right up until I got to the maze.

Mazes don't have to be boring and stupid. It's possible to design entertaining mazes by ordering the rooms according to a pattern that the player can figure out. A maze should be attractive, clever, and above all, fun to solve. If a maze isn't interesting or a pleasure to be in, then it's a bad feature.

Games Without Maps

I have a notoriously poor sense of direction inside buildings, so maybe it's just me. Still, in the video game world where all the walls and floors use the same textures, places look too much alike. In the real world, even the most rigid cubicle-hell office building has something to distinguish one area from another – a stain on the carpet, a cartoon posted outside someone's cube. I played Doom and had a great time. I fired up the Quake demo, found out there was no map, and dumped it. I want a map. There's no reason for withholding a map from me unless it's just to slow me down, and that's a poor substitute for providing real gameplay. Bad game designer! No Twinkie!

Incongruous or Fantasy-Killing Elements

Sometimes an adventure game will present you with a puzzle, or other obstacle, that is completely outside the fantasy you're supposed to be having. In my opinion, that's a case of the designer running out of ideas, and it's disappointing to the player. If you've taken me away to a magical world where I'm a heroic knight on a glorious quest to rescue the fearsome princess, don't make me sit and play Mastermind with the dragon. If I absolutely must play a game with him, it should be Nine Men's Morris, but frankly, it would be more appropriate just to thrash the scoundrel soundly.

This leads quite naturally to my next complaint, which is…

Pointless Surrealism

A number of games have come out which eschew the standard SF/fantasy worlds and instead plunge the player into a twisted and disturbing realm of yadda yadda yadda. Let me tell you something about the capital-S Surrealism of the capital-A Art world: it's not just randomness. Real Surrealism seeks to shock the mind into a new awareness of [the human condition | the nature of God | the meaning of compassion | etc.] through the juxtaposition of seemingly unrelated objects and ideas – the key word being "seemingly." Although appearing bizarre and perhaps even nonsensical at first, true Surrealism is informed by an underlying theme.

I haven't seen any surrealism in computer games that could claim such noble goals. Most of it has looked to me like somebody said, "… and when you reach the control room of the Doomsday Machine, there'll be a clown in there! Yeah! That'll be cool!" Surrealism is like prose poetry: easy to do, but extremely hard to do well. "It's surrealism" is not an adequate excuse for a poorly conceived vision in the first place.

Which takes me effortlessly to…

Puzzles Requiring Extreme Lateral Thinking

These are puzzles of the "use the lampshade with the bulldozer" variety. The designer may think he's being funny or even surreal, but he's really just being adolescently tiresome. It's lazy puzzle design – making a puzzle difficult by making its solution obscure or irrational. You can add to the player's play-time by creating ridiculous obstacles, but you're not really adding to his or her enjoyment, and that's supposed to be the point.

Puzzles Permitting No Lateral Thinking At All

You come to a locked door. The obvious solution is to find the key, but it's also the most boring, so maybe the game provides some other way to get it open. But like as not, there's only one solution, whatever it is.

In text-adventure terms, this was known as the "find the right verb" problem – you were dead in the water until you figured out exactly what verb the game was waiting for you to say. Break? Hit? Smash? Demolish? Pound? Incinerate? And a lot of games today have the same problem: an obstacle which can only be overcome in one way. The game doesn't encourage the player to think; it demands that the player read the designer's mind.

In the real world, think of all the things you can do with a locked door:

- Find the key
- Pick the lock
- Force or persuade the person who has the key to open it
- Trick someone on the other side into opening it (maybe just by knocking!)
- Break the door down, burn it, cut it, dissolve it with acid, etc.
- Circumvent it – go through a window instead, or cut a hole in the wall.

The list is limited only by your imagination.

OK, I know this is a tall order. As a developer, it's difficult and expensive to think of all the ways that someone could try to get through a door and to implement them all. Still, now that we have the have the power to create "deformable environments" – that is, your gunshots and explosions actually affect everything in the real world and not just your enemies – it's time to add a little variety to our worlds, to reward players who do some lateral thinking.

Puzzles Requiring Obscure Knowledge From Outside the Game

I owe this one to my friend, the genius puzzle-master Scott Kim (). I didn't think of it until he read a draft of this column and pointed it out to me. This is a cheap trick, and even more irritating than inside jokes. No, I don't know the name of the third track on Sgt. Pepper's Lonely Hearts Club Band, and if it's vital that I know it for the game, then the game is just weird. (Trivia games like You Don't Know Jack are of course excluded from this gripe – with them you know what you're getting into.)

A Switch in One Room Opens a Door In Another Room A Mile Away

Nor does it have to be a door – I mean any item which affects a game obstacle a long way off. Doom was guilty of this a lot, but the worst example ever was in The Hitchhiker's Guide to the Galaxy, an Infocom text adventure. In that game, if you didn't pick up the junk mail at the very beginning of the game, it was unwinnable at the very end. This misfeature is profoundly and pointlessly irritating. With the exception of refineries and nuclear power plants, in most places in the world the knob for a door is – wonder of wonders – in the door. It's another example of lazy puzzle design, making the problem difficult not by cleverness but artificially extending the time it takes to solve it.

Only One of [some large number] of Possible Combinations Is the Right One

More lazy puzzle design. At the end of Infidel, which was another Infocom adventure, you had to do four things in a certain sequence. The number of possible combinations is 4! (four factorial, or 24). There was no clue whatsoever as to the correct sequence; you just had to try them all. Yuck. Yet another time-waster with no enjoyment value.

Kill Monster/Take Sword/Sell Sword/Buy A Different Sword/Kill Another Monster

...or in other words, the canonical RPG experience. You may have heard John F. Kennedy's joke that Washington D.C. is a city of southern efficiency and northern charm. Well, in my opinion most RPG's combine the pulse-pounding excitement of a business simulation with the intellectual challenge of a shooter. I play games of medieval adventure and heroism to slay princesses and rescue dragons; I don't play them to spend two-thirds of my time dickering with shopkeepers. I want to be a hero, but the game forces me to be an itinerant second-hand arms dealer. Earning money by robbing corpses doesn't make me feel all that noble, either.

You Have 30 Seconds to Figure Out This Level Before You Die

With the length of time most games take to load their core modules, this isn't clever or challenging; it's just frustrating. If there's a trick to the solution for which no clues are provided, then it's just another annoying trial-and-error time-waster. If clues are provided, then you need a reasonable amount of time to think them over. The military doesn't charge blindly into unreconnoitered territory – or if they do, they usually regret it. Expecting your player to do it is unreasonable. If you're going to place your player in imminent danger from the very first second she sees the screen, then at least one out of every three of her possible choices should lead to safety.

Stupid Opponents

Another thing I'm tired of is stupid monsters who lumber towards you until you shoot them. This was the Doom technique, and that of a million video games since the dawn of time. Instead of providing you with an intelligent challenge, the game seeks to overwhelm you with sheer numbers. Yawn. Space Invaders may have been brilliant and addictive in its day, but it's time to move on.

None of these ideas are new; it's just that we don't see them that often. Why? Laziness again. Dumb monsters are easy to program. Smart ones aren't. And it's easy to balance a game with dumb opponents. You just figure out the appropriate ratio of monsters to "health" powerups. To make the game harder, you change the ratio. But it's boring. Let's put a little thought into monster design, give our customers a new challenge.

Two other things I'm tired of – these are aesthetic rather than design elements, but I'll throw 'em in for good measure.

Poor Acting

Bad acting is a distraction, no less in a computer game than in a movie theater. It breaks your suspension of disbelief. When a bad actor is surrounded by good actors, it's especially noticeable, and you find yourself praying that their character will be killed off. And most of the acting in computer games is still pretty poor.

Fortunately, this is a problem that will probably take care of itself in the end. Competition will force us to develop some competence in this area. If we can manage to get up to the TV-movie-of-the-week level, I'll be happy. John Gielgud and Katharine Hepburn's talents would be wasted in a computer game, where the point is supposed to be interactivity anyway. It's better to do without acting in a computer game than to include bad acting, and usually cheaper and easier as well.

Neat, Tidy Explosions

Look closely at a picture of a place where a bomb went off. It's a mess. A real mess. Things are broken into pieces of all sizes, from chunks that are nearly the whole object, to shrapnel and slivers, down to dust. And they're twisted, shredded, barely recognizable. Things that are blown up by a bomb don't fall neatly apart into four or five little polygons – they're blasted to smithereens.

I suppose for the sake of our stomachs we'll have to preserve the TV and film fiction that people who die violently do so quickly and quietly rather than screaming and rolling around; but I don't see any need to pretend that high explosives are less than apallingly destructive. Bombs ruin things – lives and buildings. They leave the places they've been shattered and unattractive. Let's tell the truth about them.

Conclusion

Scott Kim tells me that I'm being a bit harsh by labeling some of these misfeatures as "lazy" puzzle design. He points out that puzzle design is hard work to begin with, and unless you're quite familiar with the games of the past, it's easy to make the same mistakes again without knowing it. In addition, a lot of people come into puzzle design from other fields like programming or art, and so don't have much experience at it.

I'll buy that. But now that you have this handy list, at least you needn't make these mistakes, right?

Sprite Based Text

By Mark Overmars

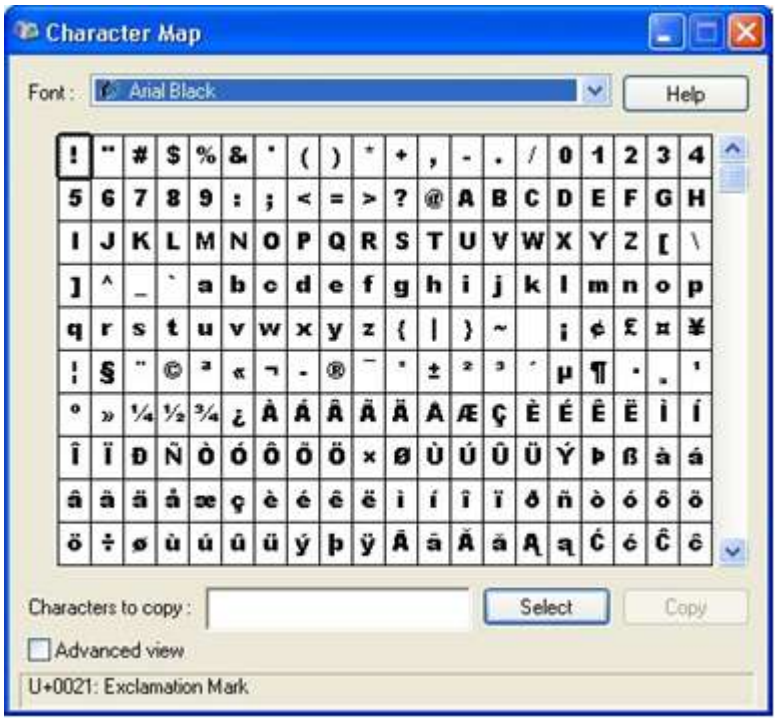
In many games you want to tell something to the user using text. This can range from simply showing the score or some help text to interaction with characters in the game as is common in adventure and RPG games. In Game Maker there are no text objects that display text on the screen but you can use the actions or functions for drawing a text to create such objects.

When you draw text on the screen using the normal actions or functions the drawing is done using the standard Windows drawing routines. These have some disadvantages. First of all, text is in one color. Hence you cannot make nice graphics effects. Secondly, using Windows for drawing the text is relatively slow. Game Maker is optimized for drawing sprites. Finally, the fonts might not exist on the machine where the game is played. (The last problem can be solved by using a **data file** object to store the font and install it.) Hence, it would be useful if we could use sprites instead. Game Maker has one function for that, called draw_text_sprite. This one function is actually very powerful. In this article I will show how to use it.

The function draw_text_sprite uses so-called sprite-base fonts, or bitmap fonts. A sprite-based font is a sprite that contains an image for easy character in the alphabet, that is, each character we need. For example, when we want to use a sprite based font just for displaying numbers, we can use a sprite with the following images.

Because these are images you can draw them in any way you like using all colors you want. When we use a sprite-based font for drawing a text, the system takes for each letter the correct images from the sprite and puts them next to each other. So only sprites are drawn and, as I indicated above, Game Maker can draw sprites very fast. It also shows some of the limitations of sprite-based fonts: They have a fixed size (although you can scale them but that is slow), they cannot be rotated, they have a fixed spacing (distance between the characters) and they use a lot of memory.

Let us look a bit better at sprite-based fonts. As I said above they contain images for a subset of the characters, but in what order? Game Maker assumes that the characters are in the ASCI order and that they form a continuous subset of these. If you want to find out what the ASCI order is, start the character map program in the accessories (system tools). The following form will show:



It contains the printable characters in the ASCI order. So for example, for numbers you can provide the digits 0-9. For uppercase text you can only provide the uppercase letters. If you want all normal characters you need to provide 96 images (and if these are 32x32 images this will cost a total of close to 300 KB of memory, so be careful).

Some words of advice when creating font sprites. First of all, there is no need to use precise collision checking for them. So uncheck this box. Secondly, pick the origin of the sprite carefully because this is used in placing the characters on the screen. If you use the sprite-based font only in a few rooms, make sure to set **load only on use**. Also, if the amount of text is small, better uncheck **use video memory**, because, as indicated above, the sprites can be rather big. One the other hand, if you have a lot of simultaneous text, better make sure the sprite is stored in video memory.

Now let us look better at the function to draw sprite-based text. It has 8 arguments, as follows:

draw_text_sprite(x,y,string,sep,w,sprite,firstchar,scale)

Let us look at the arguments.

x,y: these indicate the position where the text is drawn. Normally this is the position for the origin of the sprite corresponding to the first letter in the text. You can though change this by setting the variable **font_align** to one of the values **fa_right** or **fa_center**.

string: this is the string that must be drawn. You can use the symbol # as a line separator. (Use \# to print the # character.) Characters in the string that are not available in the font sprite will be replaced by empty space.

sep: this indicates the separation distance between lines of text. You can change this value to get the lines close to each other or far apart. Use -1 for the default distance.

w: the width of the text box. The function tries to keep the text inside a box of this width. It will break the text in lines to do this (only at spaces or the - sign). Use -1 to not split up lines.

sprite: this is the sprite that contains the characters of the font as individual images. Note that the width and height of the sprite are used to determine the distance between the letters and lines.

firstchar: this is a character (a sting consisting of 1 character) that indicates the first character in the sprite. For example, use ‘0’ when the sprite starts with the the digits or ‘a’ when it starts with the lowercase characters.

scale: the scale factor used to draw the character images. Use 1 to not scale the images. (Remember that scaling can be slow on certain computers.)

Let us look at an example. Assume we have the above sprite with the digits and assume it is called spr_digits. We will use it to display the score in a much nicer way. Now create an object and in its drawing event put a code action with the following piece of code:

```
{
  draw_text_sprite(x,y,score,-1,-1, spr_digits,'0',1);
}
```

Note that we use score in this function even though score is not a string. Officially you would have to write string(score) instead but the text drawing function automatically converts the number to a string. Place an instance of this object at the correct place in each room (or a persistent one in the first room only) and you are done. (Note that you can use the function only in the drawing event. Never call drawing functions anywhere else unless you understand exactly how the drawing procedure works in Game Maker).

It should be clear now how to use this function for drawing other texts.

If you want to have even more flexibility, like using transparency, you can still use sprite-based fonts but now you have to write yourself a script to draw the text. This is not really hard (all functions required exist in GML) but you need some programming experience for this and it will be slower.

Color and game arcade graphics

Part 1

By Ari Feldman

Get the entire book

Color is extremely important to arcade game graphics design. When used-cor rectly, color can produce a variety of powerful physical and emotional effects in games. Among other things, we can use color to:

Attract the user's attention —Color can make game objects "pop" or stand out to the user.

Alter the user's mood and feelings —Color can alter and affect the user's mood and emotions. For example, bright colors can induce cheer or happiness while darker colors can induce fear. It can also be used to convey cultural or gender-specific messages.

Alter the user's perception of space —Color can add depth and dimension to objects and scenes. Essentially, color can make things seem more "real" to the user and can project 3D properties onto 2D images by manipulating the user's perception.

Create aesthetic appeal —Color can make objects and scenes seem more enticing, which can stimulate the user's interest and enhance their enjoyment.

Show and accentuate similarities and differences —Color can highlight the similarities and differences between game objects. For example, you can use color to emphasize one object and de-emphasize another such as in a menu or title screen.

As you can see, color is an extremely powerful device. However, in order to achieve these results in your own games you must first learn what color is, how it works, how to produce it, and then how best to use it. That's the purpose of the rest of this chapter. There's quite a bit of material, so let's dive right in.

Basic Color Theory

Color exists as wavelengths of light. The sun generates this light and shines it on different objects. All objects, whether they are organic or man-made, absorb some of these wavelengths while reflecting others. Those wavelengths that are reflected reach the retina of the eye and stimulate the brain so that the perception of color is experienced.

The total sum of light (and color) we can see is called the visible spectrum The total number of colors that a computer display can generate is called its color gamut Despite the fact that modern computer displays can produce millions of colors, no computer system can reproduce all of the colors that can be seen with the human eye. This is due to the wide differences of CRT picture tube quality and the impact of room light. Under ideal viewing conditions, scientists estimate that the average human eye can distinguish and perceive anywhere between 1 and 7 million distinct colors. This is pretty important since I mention throughout this book (as do many other authors) that modern computers can generate up to 16.7 million colors. Actually, this is only partially true. The 24-bit RGB hardware color

palette creates 16.7 million *machine states*, but these do not necessarily correspond to 16.7 million unique colors. The average human eye just can't perceive that many colors.

In addition to the concept of color there is the element of *tone*. Tone measures the lightness or darkness of an image and is subjective in relationship to the other colors that are present in an image. Tone gives color its depth and form. It provides objects with shape and definition as the tonal range of an image runs from light to dark and vice versa. Tone is analogous to a color gradient described elsewhere in these pages. Tone has simple rules: the larger the tonal range, the higher the image quality. Meanwhile, the smaller the tonal range, the lower the image quality. Images with poor tone appear flat and washed out, while images with good tone look smooth and vibrant. High color modes tend to exhibit strong tonal ranges, while low color modes exhibit weak tonal ranges.

But wait, there's more! Every color also has three basic properties: *hue*, *saturation*, and *value*.

Hue is the color being described or produced, such as yellow, black, green, etc. Hue is also known as the *local color* of an object as it will look as expected when viewed close up but can appear quite differently when viewed from a distance. For example, a particular shade of green might appear bluish if you move far away from the computer's display.

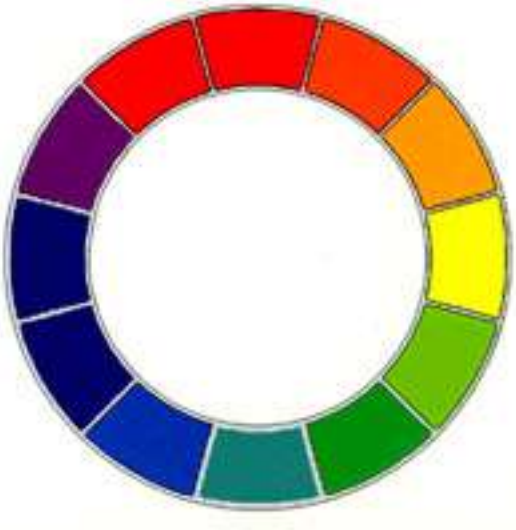
Saturation, also called *intensity* or *brightness*, is the strength or weakness exhibited by a given color. In other words, saturation measures the "purity" of a color. On computer displays, saturation is typically specified as a percentage of light. Highly saturated colors display more brilliance, while less saturated colors display more dullness. For example, 100% black would produce a vivid black, while 50% black would produce a shade of gray.

Value simply measures the degree of lightness or darkness, also known as the *tonal range* of a particular hue.

NOTE: Like RGB, hue-saturation-value, or HSV, is yet another method of selecting computer color. It will be described in more detail in Chapter 8.

Color Mixing

At the heart of color theory, whether it's computer generated or not, is the *color wheel*. Think of the color wheel as a circle that contains different colors waiting to be blended together to produce other colors.



At the most basic level of this model, there are three *primary* or primitive colors—red, blue, and yellow on the color wheel. They are called primary because no other colors can be mixed together to create them. They are essentially unique unto themselves.

In addition to the primary colors, there are *secondary colors*. These colors are orange, green, and purple, and are produced by mixing two adjacent primary colors on the color wheel. For example, red + yellow = orange and blue + yellow = green.

Combining primary colors with secondary colors on the color wheel creates *intermediate* or tertiary colors. Examples of intermediate colors include yellow-orange, red-orange, yellow-green, blue-green, blue-purple, and red-purple. Intermediate colors are also sometimes referred to as radical colors. These colors are frequently found in nature and include such colors as sky blue, olive green, and earth brown.

Once you move beyond mixing primary, secondary, and intermediate colors you experience what are known as complementary colors. Complementary colors lie directly opposite from one another on the color wheel. These colors contrast each other because they share no common colors between them. For example, red is complementary to green, orange is complementary to blue, etc. When you combine complementary colors you effectively "cancel out" the complementary colors and produce *neutral* colors. Neutral colors include grays and browns.

The subtle colors we see in most images begin to become apparent as primary, secondary, and tertiary colors are combined and blended ad infinitum. As this happens, a much more sophisticated color wheel evolves. In it, the primary, secondary, and intermediate colors still exist, but there are many thousands or millions of "in between" color shades that become evident as well. It's this color wheel that's the foundation of the color used by computers and in computer arcade games.

On computers, there are two general methods for combining color together electronically. They are known as *additive* and *subtractive* color mixing.

Additive Color Mixing

Red, blue, and green are the primary colors used in the additive color mixing process. When these colors are projected onto each other in equal parts they produce white. Varying the intensities or mixtures of these colors creates other shades of color. For example, black is produced when all traces of red, green, and blue are removed from an image. All computer displays employ a form of additive color mixing to generate their colors.

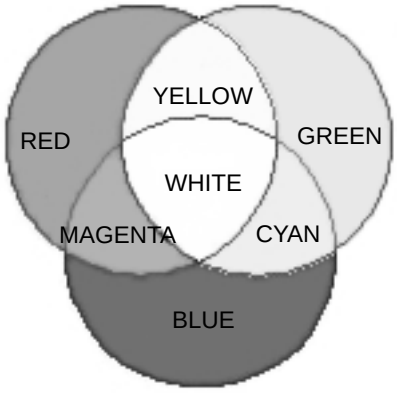


FIGURE 7-2: Additive Color Mixing Diagram

Subtractive Color Mixing

Subtractive color mixing occurs when white light is project on a colored surface and is partially reflected back. The reflected light is the color that we actually see. For example, when sunlight hits a lemon, we see yellow. Unlike additive color mixing, subtractive color mixing isn't based on RGB. Rather it uses cyan-magenta-yellow, or CMY. Mixing these three colors to different degrees produces other colors. For example, black is produced when all three colors are mixed together at 100%. Subtractive color mixing is used almost exclusively in color printing and color photography and is just mentioned here for comparative purposes.

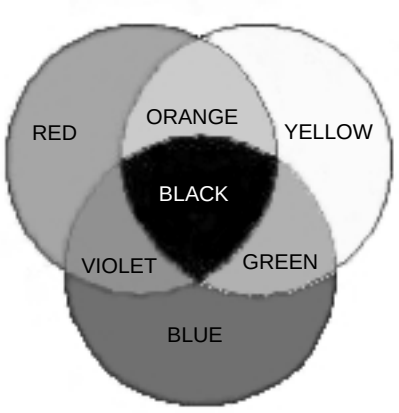


FIGURE 7-3: Subtractive Color Mixing Diagram

Color Temperature

Designers often classify colors by their *temperature*, or their degree of warmness or coolness. Specifically, reds, oranges, and yellows are considered to be warm and exciting, while greens, blues, and violets are considered to be cool and sedating. Warm colors tend to be stimulating, producing the emotional feelings of closeness and friendliness. Cool colors, on the other hand, tend to be soothing, producing the feelings of distance and/or openness.

Understanding these color behaviors can have a major influence on the mood your game artwork projects to the user. For example, a game could use warm colors to represent the heat of an underground lava cave or cool colors to depict a dark forest or a cold iceberg. It's even suggested that warm and cool colors can raise or lower one's body temperature, suggesting a physical connection with the use of color in addition to the emotional one.

Once you grasp the fundamentals of color mixing, you'll be in good shape when it comes time to define your own. Yet, before we learn how to do this, it would be helpful if you understood the hidden language of color as well as reviewed the essential dos and don'ts of color use in arcade style games.

The Language and Meaning of Color

Color is more than just something we see and experience. It has its own language and can be used to apply specific meanings, moods, and symbolism to the images you create for your games. For example, a black object in a game might be used to signify evil or a background scene that incorporates lots of orange might project a sense of warmth. Along the same lines, certain color choices can provoke specific emotions and feelings in the user. For example, a fast-action arcade shooter rendered in various shades of red might add to the sensation of excitement or an adventure game rendered in cool, dark colors might provoke a sense of fear. By knowing a little something about the "hidden" language and meaning of color, you can maximize its use in your game projects.

Consider the examples in Table 7-1 of popular meanings for some of the more common colors.

TABLE 7-1: Meanings of Common Colors

Color	Common Meaning(s)
Red	Danger, urgency, passion, excitement, aggression, heat, love, or blood
Magenta	Imagination or outrageousness
Pink	Friendliness, sweetness, romance, or compassion
Orange	Warning, courage, warmth, or happiness
Yellow	Caution, warmth, brightness, or cowardice
Gold	Illumination or wisdom
Brown	Earthiness or stability
Blue	Attention, dignity, coolness, depression, power, or peace
Turquoise	Refreshing or cool
Purple	Wealth, royalty, mystery, sophistication, or intelligence
Lavender	Romance or fantasy

Color	Common Meaning(s)
Green	Safety, nature, health, happiness, environment, envy, or money
Gray	Neutrality, gloom, practicality, or security
Black	Death, evil, rebellion, strength, or fear
Beige	Neutrality
White	Purity, light, or emptiness

The Cross-Cultural Meaning of Color

Like any other language, color's vocabulary is largely culture dependent. For example, in the Middle East, blue is considered to be a protective color, implying strength; in the west, blue is considered a tranquil color and implies peace. Similarly, green is a sacred color in some cultures and represents greed and wealth in others. It's important to be aware of these distinctions in order to avoid confusing users from different cultures. This is particularly important given the growth of the Internet and how popular arcade games are with players around the world. Even so, these definitions can be helpful in determining how to portray certain characters and objects with color in your games.

Table 7-2 provides some additional examples of the cross-cultural meanings of color.

TABLE 7-2: Cross-cultural Color Meaning Matrix

Color	Western Europe & USA	Japan	China	Brazil	Nigeria	Korea	Middle East
Red	Danger, anger, or stop	Danger and anger	Joy and festivity	Anger and hate	Danger	Anger, danger, evil	Danger and evil
Yellow/ Gold	Cowardice, caution, or warmth	Happiness, grace, nobility	Honor and royalty	Money, gold, and wealth	Sunshine and brightness	Wealth	Happiness and prosperity
Green	Sexual arousal, safety, greed, envy, or environment	Future, youth, energy, or forest	Youth and growth	Hope or wealth	Wealth	Nature, peace, or freshness	Fertility and strength

Color	Western Europe & USA	Japan	China	Brazil	Nigeria	Korea	Middle East
White	Purity, virtue, goodness	Death and mourning	Mourning and humility	Purity and peace	Purity	Innocence and purity	Purity or mourning
Blue	Machismo, masculinity, calm, authority, or coolness	Villainy or cold	Strength and power	Happiness	Calm and peace	Cool or freshness	Protection
Black	Death and evil	Evil	Evil	Death	Evil	Darkness or evil	Mystery or evil

NOTE: The examples presented in Table 7-2 aren't meant to dictate how you should use color in your games. Rather, they are provided to educate you on how some users (both domestically and abroad) may interpret the colors you've chosen from a cultural standpoint. When designing games that may wind up overseas, it's important to consider the sensibilities of the users that may eventually play them.

Do these differences mean that you'll have to rework your existing game artwork to pass muster in different countries? No, probably not. However, you should be aware of the different cultural interpretations of color so you don't accidentally offend users in different countries or improperly communicate the messages or themes associated with your game.

Color and Mood

When used appropriately, color can strengthen the user's interest and stimulate their involvement with your game. Setting the proper mood through your color choices only helps to reinforce this. While the examples in Table 7-3 are by no means definitive, they do provide you with some good starting points on how to best tailor your color selections for different game audiences and scenarios. For example, you wouldn't want to use lots of warm, bright, "happy" colors for a game with a dark or violent theme. Along the same lines, you wouldn't want to use very strong or aggressive colors for a game with peaceful or harmonious overtones.

By applying these meanings to your graphics, you can induce different moods in users as they play your game. For example, graphics rendered in dark hues such as gray can instill a sense of fear or fright in a scene. Similarly, certain shades of blue can set a melancholy tone while orange can transmit a happy tone.

Please refer to Table 7-3 for some examples of when and where to use certain colors to create specific moods and emotions in your games.

TABLE 7-3: Game Mood Color Usage Matrix

Desired Mood	Recommended Game Type	Suggested Foreground Color Combinations	Suggested Background Color Combinations
Happy, upbeat, cheery	Maze/chase games, puzzlers, platformers, and educational games for young children	Oranges	Reds and yellows
Sadness, gloom, despair	Shooters and platformers	Grays	Blues
Harmony, peace, wisdom	Educational games	White	Blue and gold
Evil, doom, death	Shooters and platformers	Blues	Grays
Mystery, adventure, fantasy	Maze/chase games, puzzlers, platformers, and educational games for young children	Purples	Lavender, magentas, pinks, and gold
Purity, romance, sweetness	Maze/chase games, puzzlers, platformers, and educational games for young children	Pinks	Whites, reds, and lavender
Nature, stability, earthiness	Shooters and platformers	Greens	Grays and browns
Technology, futuristic	Maze/chase games, puzzlers, and platformers	Grays and blues	Purple and gold
Excitement, anger, power	Maze/chase games, puzzlers, and platformers	Reds	Blues and oranges

NOTE: For our purposes, foreground colors refer to the colors used for foreground screen objects such as sprites and bonus/pick-up items. Background colors refer to background tiles, scenes, and framing elements. As indicated in Table 7-3, these color choices are just suggestions. Your individual tastes and the type of game will actually dictate the colors you use. Even so, I thought it might be helpful to give you a sense of how certain colors can project certain moods, feelings, and emotions to the user.

Color Perception Issues

As one might expect, everyone perceives color somewhat differently. In some cases, these differences in perception are due to one's upbringing or cultural environment. However, in most cases, biological and physical factors are responsible

for these differences. Of these, age and gender seem to play the most significant roles.

Age and Color Perception

Age can have a major effect on color perception and actually might influence how the user enjoys and experiences your game.

In a nutshell, this is how different age groups interpret color:

Young children —Prefer warm and very intense colors as they instill a sense of happiness.

Older children and young adults —Respond better to bright, warm colors than dull, cool colors.

Adults —Prefer cooler, more subdued colors to warm, saturated colors.

Older adults —Respond better to brighter colors due to degrading eyesight.

Table 7-4 outlines the issues associated with each of these age groups and provides some suggestions on how to accommodate them.

TABLE 7-4: Age Groups and Color Use Suggestions

Age Group	Suggested Color Choices/Schemes
Under 10	Use warm and very intense color schemes. They're friendly, cheerful, and non-threatening to younger audiences. Enhances the "cuteness" factor of most games targeted toward young children.
10-25	Use warm and intense color schemes when you need to attract attention. Don't be afraid, however, to use darker or dreary color combinations with this age group if it makes the overall game experience more realistic.
25-50	Limit the amount of warm and intense colors for adults. They're no longer into candy canes and lollipops. They want action and are particularly interested in maintaining the suspension of disbelief (i.e., realism). Feel free to use plenty of cool, dull, intermediate colors for games targeted towards these audiences. For them, realism counts more than anything.
Over 50	As people age, they begin to experience difficulty discerning between different colors. This is especially true for older adults, as they tend to see colors dimmer than they actually are. As a result, use warmer and more saturated colors for games geared towards older audiences.

NOTE: As always, use the information provided in Table 7-4 as suggestions only.

Gender and Color Perception

Surprisingly, game designers and graphic artists seldom take the issue of gender into account. There's evidence that gender plays a significant role in color appreciation and recognition, which in turn may influence how both men and women perceive and enjoy your game.

Simply put, the research of gender-based color perception is interesting, to say the least. Studies have shown that women prefer cooler colors than men do. Similar studies concluded that men tend to prefer brighter, more intense colors than women do. Moreover, women are able to distinguish and perceive color better than men are.

What does this mean in real-world terms? Well, this research suggests several things, including:

- The possibility that women and girls are more likely to play and enjoy games with soft, cool, or pastel color schemes than men.
- The possibility that men and boys are more likely to play and enjoy games with warm, saturated colors than women.
- The possibility that women and girls are probably more sensitive and aware of color schemes used in games that don't quite match reality. For example, they are probably more likely to tell that a game is using the wrong shades of brown for wood or hair, for example, than most men are.
- The possibility that women and girls may be more likely to have their moods altered by a particular color scheme than men.

Table 7-5 summarizes these gender-specific differences. In any case, don't let this issue stop you from making the game you want to make. Rather, just be aware of it and take it into consideration when you plan and select the colors for your next game project.

TABLE 7-5: Gender and Color Preference Summary

Gender	Warm Colors	Cool Colors	Saturated Colors	Softer Colors	Color Sensitivity
Men	Prefer	Not preferred	Prefer	Not preferred	Low
Women	Not preferred	Prefer	Not preferred	Prefer	High

NOTE: There is no significant evidence that different races perceive colors differently. Instead, age, gender, and culture remain the main determinants of color perception.

Other Important Color Concepts

Before delving further into the other aspects of color use, it's important that you understand some of the lingo and concepts you're likely to encounter throughout the rest of this book, not to mention the software you'll use. Therefore, you should understand such items as:

- Shade
- Tint
- Contrast
- Luminance

Shade

Shades are the dark values of a particular color. Blending a given color with different degrees of black produces shades. Shade plays an extremely important role in arcade game graphics because it gives objects the illusion of realism by simulating how light is reflected and absorbed by different objects. For example, anyone can draw an apple and color it red. However, unless they properly shade it with progressively darker shades of red, it won't look very realistic.

Figure 7-4 demonstrates this concept by showing progressively darker shades of white as you move towards the right.

FIGURE 7-4: Example of Shade

Tint

Tints are the light values of a particular color. Essentially, tint is the inverse of shade. Blending a given color with varying degrees of white produces different tints.

Figure 7-5 demonstrates this concept by showing progressively lighter shades of black as you move towards the right.

FIGURE 7-5: Example of Tint

Contrast

Contrast emphasizes the differences between two colors. For example, both white and black are contrasting colors, as are many complementary colors. Like shade, contrast is an important element in arcade game design when properly used, as it allows you to create excitement and interest with your images.

Contrast is particularly important for backgrounds because without proper contrast, your foreground objects will be difficult to distinguish against backgrounds of different colors and complexities.

Images that exhibit proper contrast are much more pleasing to the eye than images that don't. Figure 7-6 shows how contrast can influence visual appeal. Notice how much better object A looks when compared to object B. This is because object A uses proper contrast.

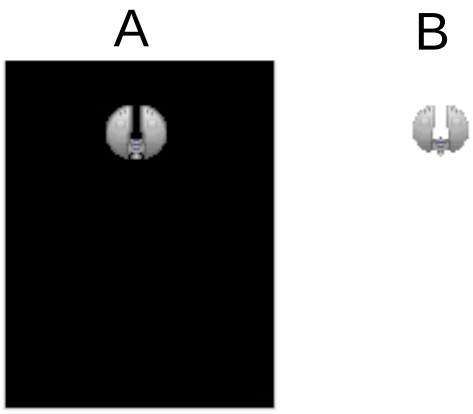


FIGURE 7-6: Example of Contrast

Luminance

Luminance is the energy of a given color. Colors with more luminance will appear brighter than those with less luminance.

Figure 7-7 demonstrates this concept. Notice how object A appears so much brighter than object B. This is because object A has more luminance than object B.

Luminance influences game graphics in various ways. For example, objects with low luminance will be difficult to see against dark backgrounds and vice versa.

FIGURE 7-7: Luminance Example

Smoothing Objects with Color

In Chapter 2, we discussed how screen resolution could cause unsightly stair stepping (called aliasing) to occur along the edges of graphic shapes as they are drawn on the screen. Well, aside from increasing the available resolution, the only other way to improve the quality of images and reduce the impact of aliasing is to use careful color shading. The most common method to accomplish this task is to use what is called *anti-aliasing*. Anti-aliasing uses a variety of algorithms to blend the edges of an object with shades of a similar color. This has the effect of smoothing the edges of the object and makes it appear more natural against the background. Figure 7-8 demonstrates this concept by comparing two graphic shapes. The object on the left is aliased while the object on the right is anti-aliased. Notice the subtle shades of gray along the edges of the object on the right and how much smoother it appears on-screen.

FIGURE 7-8: Aliasing vs. Anti-aliasing

Anti-aliasing is a powerful technique with many, many applications in arcade game graphics. When used properly it can greatly improve the appearance of all of your text and graphic objects. Therefore, you are encouraged to take advantage of it whenever possible.

One thing to understand is that anti-aliasing won't be available to you in every instance. This might be due to a limitation of your graphics program or due to color limitations in the current display mode since there must be extra colors available to produce the smoothing effect. However, you can effectively simulate anti-aliasing by taking advantage of the power of shade and picking similar colors to manually blend an object's edges with the contents of the background.

Volume, Light, and Shadow

Every object has volume, whether it's an airplane, egg, building, or coffee cup. Volume adds the element of depth to objects. In turn, this depth makes objects appear natural and realistic. In order to achieve the illusion of volume in a 2D space, you must understand how to manipulate light to bring out the subtle and dramatic differences exhibited by different objects.

Light is by far the most important element in representing volume. All of the shadows and highlights visible on objects are directly influenced by the origin of the light source that shines on them.

The origin of the light source directly determines the size and location of shadows and highlights cast by a particular object. Shadows are created whenever light is unable to reach certain parts of an object since the object itself blocks it. For example, a building casts a shadow because it blocks all or part of the sun. High lights, on the other hand, are created whenever light is reflected off an object. For example, a metal dome displays a highlight when light is reflected off its surface.

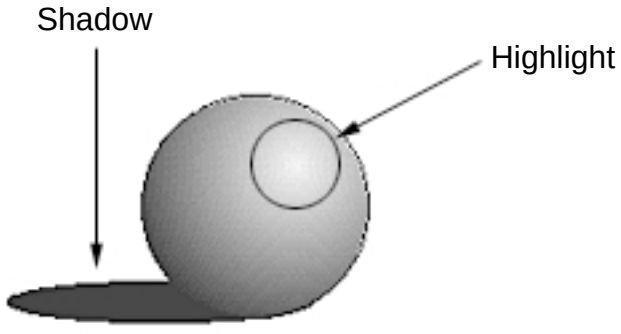


FIGURE 7-9: Shadow and Highlight Example

It's very important to understand that the appearance and positioning of shadows and highlights on objects are not constant. They can change and tend to move along with the light source that created them. Many designers fail to take this factor into account when they design their artwork.

Figure 7-10 shows some examples of this phenomenon.

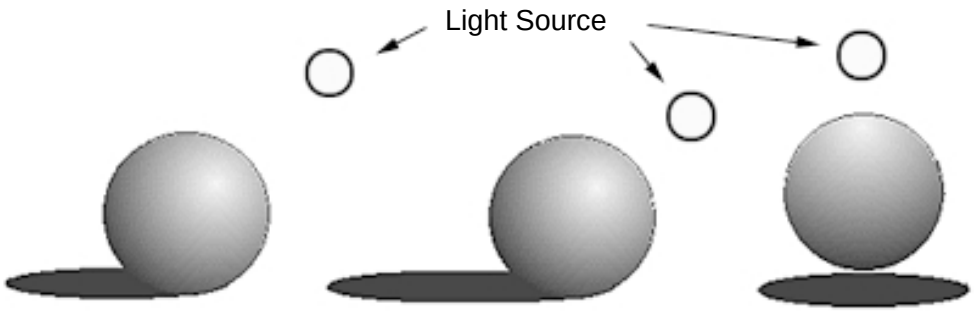


FIGURE 7-10: Light Source Movement and Impact on Shadows and Highlights

Shadows possess an interesting property that highlights do not. Shadows take the general shape of the object that casts them. For example, if the object casting a shadow is round, so is its shadow. The actual size and shape of the shadow will depend, however, on the position of the light source. Light sources that are positioned at extreme angles will cast longer, thinner shadows while light sources that are positioned at slight angles will cast shorter, thicker shadows.

Basic Rules of Light and Shadow

There are several simple rules that you should remember when working with light and shadow. Understanding them will help you to produce convincing shadows and highlights. These rules are:

All shadows an object casts will always appear darker than any part of the object itself.

Always select colors that meet the user's contextual expectations. In other words, don't use a dark color to render a highlight! Highlights should be created using very intense colors such as white or yellow but can also be made up of lighter versions of other colors as well. Similarly, shadows should be created using dark colors such as gray or very dark shades of the primary colors.

Always make sure you properly orient the location of shadows and highlights. For example, a common mistake many designers make is to place shadows and highlights on the same side of an object. This is wrong. Shadows and highlights will always appear away from each other.

The closer the light source, the brighter the highlight on an object. This means that you should draw the highlight with a larger radius and with more intensity to properly reflect the proximity of the light's reflection.

Don't use unrelated shades to render shadows! For example, if you opt to create a shadow using shades of gray, stick with it. Don't suddenly introduce blues, reds, or other colors. This will simply ruin the effect.

Objects with rounded and/or smooth surfaces tend to cast softer, more subtle shadows, i.e., lighter shadows.

Objects with sharp and/or distinct edges tend to cast harder and more distinct shadows, i.e., darker shadows.

Shadows and highlights must always be used together. One cannot be present without the other; otherwise, you'll break the rules of light and shadow and ruin the element of realism they can provide.

The length of a shadow varies depending on the angle of the light source. Steeper angles produce shorter shadows. For example, if light reflects in front or above an object, a short shadow will result. Conversely, slight angles produce relatively long shadows. Be careful, however. If the angle of the light source is too slight, the illusion of depth can be ruined because the resulting shadow will be too long.

Try to limit how you describe shadows and highlights by using a maximum of four to eight shades of a given color. Using any less reduces the effectiveness of the illusion while using any more is simply overkill.

Table 7-6 contains some examples of good colors to use for shadows and highlights.

TABLE 7-6: Useful Shadow and Highlight Colors

Shadow Colors	Highlight Color
Black*	White
Dark gray*	Light gray
Medium gray*	Yellow
Dark blue	Light yellow
Very dark green	See Note
Very dark red	See Note

NOTE: Colors marked with an asterisk (*) denote realistic shadow colors. Other colors can be used to render shadow effects but with diminished realism. Highlight colors work somewhat differently as they always use white or yellow to represent the area where the reflection of light is the most intense.

Notice the closeness of the colors that appear in the gradient. This subtle shading is crucial, as using colors that are too contrasting or dissimilar to each other will ruin the effect.

Radial gradients are a bit more complex to use than linear gradients. Figure 7-13 demonstrates how one might be applied to an object. Notice that the radial gradient assumes the shape of the object to which it is applied.

By subtly blending and shading the colors present in the gradient, we can also achieve the illusion of smoothness. Should we opt to use a rougher method of shading, we can achieve a grainy, textured look as illustrated by Figure 7-14.

FIGURE 7-13: Radial Gradient

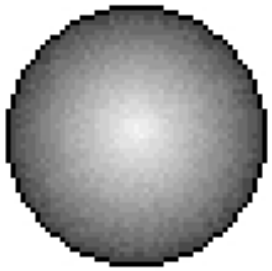


FIGURE 7-14: Rough Radial Gradient

Rules for Gradient Use

Color gradients have simple rules. Follow them and you'll successfully create the illusion of depth and volume in the objects you create. These rules are:

Always select shades of color with sufficient contrast. Gradients tend to be less effective if the shades between colors are too similar to each other. For example, if you can't visibly discern between two colors in a gradient, chances are that one of the colors is too similar to the other.

Apply linear gradients to long and narrow objects. Linear gradients don't work well for wide objects, as there are usually not enough shades present in the gradient to cover the entire surface area.

Apply radial gradients to short and wide objects. They usually don't look right when applied to long and narrow areas due to their circular orientation.

Avoid using too much distance between the next shade in your gradient when shading in objects. It's been my experience that most objects start looking flat after about eight pixels between gradient colors, except at relatively high screen resolutions. If you're trying to fill in a large area and there aren't enough colors in your gradients to cover the area convincingly, simply add more colors to them.

Gradients tend to be ineffective on very small objects due to the physical limitations imposed by screen resolution.

The larger the image, the longer the color gradient should be. Otherwise, the illusion of depth may be jeopardized due to the presence of flat color.

The longer the color gradient, the smoother the overall shading effect will be.

The shorter the color gradient, the rougher the overall shading effect will be.

The smaller and simpler the object is, the shorter the color gradient should be.

Use smoother radial gradients whenever you have a large number of shades in your gradient. Otherwise, use rougher gradients to maintain the illusion of more color being present in a gradient than there actually are.

It's difficult to recommend specific gradient sizes for a given game project. Therefore, I created Table 7-7 as a guide for determining the proper gradient sizes for use in your artwork.

TABLE 7-7: Color Gradient Shade Selection Guidelines

Gradient Size	Comments	Usage Guidelines
2 shades	The absolute minimum needed to produce the illusion of depth. Generally not recommended. Avoid whenever possible.	Best when used to color very small objects.
3-6 shades	Sufficient to achieve an acceptable level of depth. How convincing the effect is really depends on the shades or tints selected. There should be sufficient contrast between colors or else the power of the effect is minimized. Using more shades or colors is preferred, so only restrict yourself to such ranges if you must.	Best when used to color small objects and areas.
6-8 shades	Sufficient to achieve an acceptable level of depth. How convincing the effect is really depends on the shades or tints selected. There should be sufficient contrast between colors or else the power of the effect is minimized. Eight shades are usually enough to produce fairly good results. While using more shades or colors is preferred, this number should work well enough for most applications.	Best when used to color small to medium-sized objects and areas.

Gradient Size	Comments	Usage Guidelines
8-16 shades	Produces a very believable illusion of depth in objects and scenes. As always, you should make sure that sufficient contrast exists between tints or shades. Whenever possible, you should standardize on this gradient size since it makes objects appear very realistic while maximizing the use of available color.	Best when used to color medium-sized objects and areas.
16-32 shades	Produces a very believable illusion of depth in objects and scenes but only moderately better than what is possible with 16 shades. As always, you should make sure that sufficient contrast exists between tints or shades. Be aware that only a few types of objects actually benefit from this size gradient. For example, stone and metals will look more realistic but the effect may be lost on other types of objects.	Best when used to color medium to large objects and areas.
32-64 shades	Produces a very believable illusion of depth in objects and scenes but only moderately better than what is possible with 16 shades. As always, you should make sure that sufficient contrast exists between tints or shades. Such gradients seldom offer much advantage over smaller gradient sizes so it's best to avoid unless you have very specific needs in mind.	Best when used to color large objects and areas.
More than 64 shades	Can be used to create some interesting effects for objects and materials such as metals, plastics, stone, and glass but tends to be overkill for pretty much everything else. Use such gradients sparingly as objects shaded with them to make other objects look flat in comparison.	Best when used to color very large objects and areas.

NOTE: I generally recommend restricting your gradient sizes to 16 shades or less. More than 16 can become difficult to manage. Also, sticking with 16 or fewer shades allows you to use color more efficiently without sacrificing image quality.

While we're still on the topic of gradients, be sure to look at Table 7-8. It contains some examples of when and how to use gradients on different types of game objects to achieve the illusion of depth and realism.

TABLE 7-8: Example Arcade Game Object Gradient Usage

Game Object	Linear Gradient	Radial Gradient
Planet or globe	No	Yes, use coarse gradient unless object is very small. This will provide a more realistic shading effect. Otherwise, use a smooth gradient.
Stone block	Yes, if the block is rectangular or square.	Yes, if the block is oblong. Use a rough gradient to give block a grainy, weathered look; otherwise, use a smooth gradient.
Steel beam	Yes	No
Spaceship hull	Yes, if hull is long and narrow.	Yes, if hull is short and wide. Always use a smooth gradient for this type of object.
Missile/rocket	Yes	No
Terrain	No	Yes, use a rough gradient to simulate different degrees of texture.

NOTE: The information in this table is only meant to give you ideas. It's by no means a definitive list. With both practice and time, you'll find plenty of other creative ways to use and apply your color gradients.

In short, the interplay of light and shadow is one of the most important concepts to understand if you're interested in making your graphics look as realistic as pos sible. Once you master it, you'll be on your way to creating some very impressive

continued in next issue

Featured Interview

Ari Feldman

1. Please introduce yourself for those who don't know you.

My name is Ari Feldman. By day, I manage the online department for a Cable TV trade association in the U.S. and sometimes I draw sprites. Many GM users know me as the creator of SpriteLib/Spritelib GPL, which is a collection of ready-to-use game sprites and tiles.

2. As an artist, what do you think graphically should be in a game? Great sprites, backgrounds….

All of the above. Great graphical elements, while not necessary to make a game fun, do play a major role in how one percieves the quality of a game and they certainly can enhance the overall gameplaying experience.

One should not ignore title screens, menu screens or the layout of these items either.

3. How important are game graphics in a game against the music, story, and game play of a game?

Very important but they are just one part of the entire game. Great graphics "eye candy" do make games standout but without the other elements being there, they can't guarantee a good game.

Therefore, it's a balance. A good game has good design, graphics,
playability and sound/music.

4. What are your thoughts on 3D vs. 2D graphics?

Well, I'm biased towards 2D graphics but I don't think one is better than the other. Rather, they're suited for different purposes. 3D graphics are great for immersive simulations and shooters. They also can be used for other types of games. Meanwhile 2D is better suited for arcade style games, puzzle games, etc

5. When designing graphics for a game, is there anything that influences you during the creation process? Such as music…

Many things influence my work. Most of my influence come from the work of others. Specifically, I study the work of other artists to learn/evaluate their style and technique, their use of color and to formulate ideas on how to do things better and/or differently in my own creation.

Music is never an influence. If I use sound in a game of my own, I tailor the sound to fit the art rather than the other way around. That's how it should work.

6. Is character design important in a game?

Absolutely. Of course, it depends on the game genre involved. For RPG games, platform games where characters are often tied to the game's plot and backstory, it's crucial. For other types of games, it's less important.

7. What game out there do you think has the best character design?

Ironically, I don't play many popular games. I prefer to make them or to play arcade classics via emulators like MAME.

8. What are the dos and don’ts when creating graphics?

Too many to list! One can easily write 20 or 30 pages on what to do and what not to do and the list will vary depending on the type of game you're working on.

9. Other than game maker, what other hobbies and interest are there that you enjoy? Music, movies…

History and Current Events: Pay attention to the world around you. Politics can really provide lots of inspiration for games.

Music: 80s/90s college alternative, Britpop (most contemporary American bands stuck), Shoegazing, 60s Motown, 60s Psychadelic, 80s Two-Tone Ska, New Wave, Punk and Hardcore Punk, etc.

Movies: Too many to list.

Girls: One can never date too many!

Weight Lifting: Being pale and puny is no way to go through life! :-)

10. Ending comments?

Not really. I know a lot but am also always learning and have no problem sharing my knowledge and experience with other GM hobbyists.

Game Reviews

Reviews by the Game Room and others!

Bicman: Gravitized

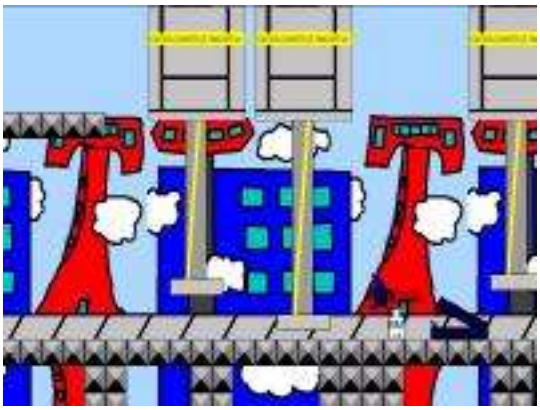
score 6.38/10

Creator: Max Williams

Bicman: Gravitized is the fourth game in the great Bicman series. Bicman is a wite-out you play as, and in this episode, he finally dies. On his way to heaven, an evil pencil knocked him into hell. Your job is to escape and get back home before the evil pencil destroys you! Nice aspects such as particle systems in a lot of it, and some enemies now have health bars. The game is mainly played by reaching the end of the level by jumping, moving and shooting enemies. Their are also some bosses.

The graphics are great. Some are resource pack graphics, some use particles and gradient effects while others are completly hand drawn and very cartoony looking.
The music is excellent and their is even an mp3 or too as the background music. The sound effects are also good.

Overall, a worthy edition to the great range of Bicman platform games.



DOWNLOAD

Undead Fred

score 5/10

Creator: Carl Norris

Undead Fred is another game to add to the popular platform game type.
The Story: In your pastlife your were an exorcist you spent your time down in the dungeons of your castle eliminating ghosts. When you died in a tradgic chess accident your were buried in the dungeons of your castle, your will stated that you must be buried with all of your money. Unfortunately the ghosts seek revenge they steal your money and turn you into the undead. You must get that money back!
So your basic objectives are to collect gold and kill your enemies.

The graphics are pretty good. A mix of originals sprites and ones taken from the GM resource packs. Their are also some very nice backgrounds. Unfortunately, thier is no music in the game but their are some sound effects.

The game is quite challenging and also fun.

Overall, another good platform game but most of it we've seen before

DOWNLOAD

Funky Guy in Gift Center

score 6.77/10

Creator: Kayle Mortensen / MadHatter Production

Funky Guy in Gift Getter is another simple platform game. The object of the game is to collect all the present points to advance to the next game. Because this game is the first version, there is no real story line to it.

Graphics, most are home drawn i'd say and they are pretty average. Their is a snow effect which is kinda strange though because the background has some clouds but also blue sky.
No music in this game and just the occasional sound effect when you die.

Overall, a pretty boring game as you can't even attack but check it out anyway, it may interest you.

DOWNLOAD

Maple Island

score 6.79/10

Creator: ZIX games

Maple Island is a nicely done platformer, with great cartoon style graphics. It has many different nicely drawn enemies, and some collectable things. The game requires a fast computer since it uses some slow transparency effects.

DOWNLOAD

JWS Racer

score 6.6/10

Creator: Jeroen

JWS Racer is a racing game in 3D. These graphics remind me of Star Fox for the NES. You race thre opponents, a tank, a van and a car.

As I said, the game has 3D graphics but don't expect too much detail. But, it is a very good attempt and the parallex background is excellent (mountains, sky, etc)
No sound effects, just two midi's one for title screen and one for the actual in-game.

The game has easy controlls, just the arrow keys.
The game however is too easy for my liking but when I first played, I got losed alot.

Overall, this is a nice racing game and it's good to see people trying 3D stuff.

Trentguy

score 6.86/10

Creator: Kayle Mandelbrotjulia

Trentguy is a highly proffesional puzzle game which has commercial game quality. You control Trentyguy. You must progress through each board by clearing all tiles on it. This is done by touching two of the same type. You begin grey. When you touch any of the tiles, you turn the same color and must touch its match before being able to clear any more tiles. Its match is not only the same color, but also has the same shape on its face. The bar at the top of the screen will show you which tile you need to match, if any. The game is made more challenging as their is enemies to avoid, a timer which must not reach zero or else you lose a life and their are static traps.
Every few levels, you will encounter a bonus level. You cannot die here, but only have 100 seconds to collect every tile on the board. On some bonus boards, there will be blocks. Trentyguy can move these only when he's grey. If you can clear the board before the time runs out, you will gain an extra life. Their are two game types, the childrens version and regular. Obviously, the children version is easier and shorter.

Mine Killer 2D 3

score 6.89/10

Creator: Oddwarg

Minekiller 2D 3 must be one of the best GM games i've ever played. The point is to destroy mines by killing the reactors, so the mine will self-destruct. Then you have to escape through an escape tunnel. If you don't escape the mine, you complete the level stil but lose all your weapons.

The has great graphics and almost none of the sprites have been taken from the packaged ones. It is 2D overhead but they are still good. Their are lots of different enemies and they all look cool. Great sound effects and music too.

The controls are easy too, move with arrow keys, shift to shoot and pg up and pg down to change weapons.
I didn't find any major bugs either.

Overall an excellent game which has a combination of genres. Two player-mode, 20 robots, 20 levels, 5 reactors & bosses, 10 weapons, 4 difficulties and bonus's if you complete the game make

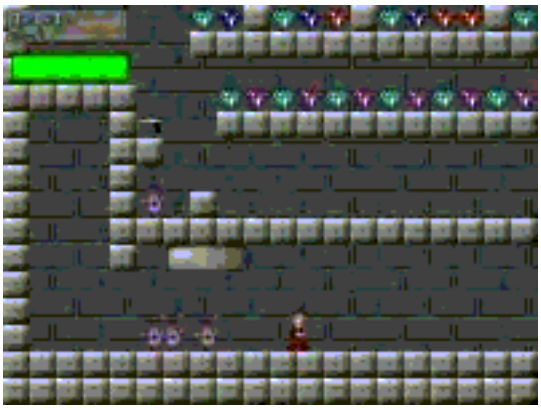
DOWNLOAD

Krod Fallen's Nightmare

score 6.64/10

Creator: Cazacu Andrei

Yet another platform game. Collect diamonds, weapons, and watch out for the enemies. The graphics and all are well done, but the player gets stuck in some places.



DOWNLOAD

Word Wizard

score 7/10

Creator: Morphosis Enter-Active based from game by Gunther Serrano

Great for kids and adults alike. The game is most enjoyable played on two computers via the internet or not! One player games are also fun. A chat function was built in also.

Time passes by as you try to figure out the mixxed up word...you are the Word Wizard!

DOWNLOAD

Rounder (o)

score 7/10

Creator: Morphosis Enter-Active John Hempstead

A fun looking and wacky sounding Tetris! Great for the entire family for all ages!

Very nice game...Mr. Chubigins “Vertigo Games”

DOWNLOAD

TETRIS 3050

score 8/10

Creator: Morphosis Enter-Active John Hempstead

Step into a dirty machine in the year 3050 and play the Classic game like never before!

Tetris 3050 is very professional, but no surprise there since it IS a morphosis game There's nothing really to explain- it is tetris after all- but the atmosphere is what separates this game. The dingy, rusty metal look combined with the foretelling soundtrack is really nice. The LCD afterimage is a cool effect. Good save feature also.

Graphics- 10/10
Sound- 7/10
Music- 9/10
Gameplay- 9/10
Mr. Chubigins “Vertigo Games”

DOWNLOAD


MUSIC
SUPERSTAR
RISE OR FALL
2004 MORPHOSIS ENTER-ACTIVE 2004
JOHN P HEMPSTEAD
HTTP://MEA.INVISIBLEFURY.COM
MADE WITH
GAME MAKER


THE
EDEN PROJECT
...TIME WILL TELL
IF THE WORLDS WILL END
SHORT MOVIE AND VIDEO GAME SUMMER 2004
MORPHOSIS ENTER-ACTIVE HTTP://MEA.INVISIBLEFURY.COM


are you
ready to morph?
morphosis enter-active

Quick Scripts

Add some to your games!

Snap shot then open in MS Paint!

Here you can take a screen shot and MS Paint will open with the screen shot!

```
{
screen_save("picture.bmp");
file_open_read("picture.bmp");
execute_shell("mspaint.exe","/p picture.bmp");
file_close();
}
```

Show a movie!

Add an avi or mpeg to your game!

```
{
show_video('game\game.avi',true,10)
}
```

Name your Game Character!

Very nice I have been waiting for this code for a while. It makes a little window in which the 'player' can put in a name. In where it say's superman thats a name that will be already displayed change it to suit your needs!

put this in the CREATION EVENT of an object:

```
{
global.name = get_string('Name','superman');
}
```

Now to 'SHOW' or 'Draw' the name put this in the DRAW EVENT of an object(or the same object, the first is drawing it in the upper-left hand corner, the second at the position of the object):

```
{
draw_text(view_left[0],view_top[0],global.name);
}
```

OR

```
{
draw_text(object1.x,object1.y,blogal.name);
}
```

Jump Code

Mess around with it to get it the way you like it! Put this in the Step or Creation Event .

```
{
gravity_direction = 270;
if place_free(x,y+1)
gravity = 0.5
else
gravity = 0;
if (vspeed >12) vspeed = 12;
}
```

Send email with input!

Here the user fills out the email address, subject, and message and then sends the email!

```
send_to = get_string('To...','')
subject = get_string('Subject...','')
body = get_string('Message...','')
target = string('mailto:') + send_to + string('?subject=') + subject + string('&body=') + body
execute_shell(target,0)
```

Morphosis “Quick Launch” code

From my program at http://mea.invisiblefury.com

SET APP:

```
{
global.app1 = get_open_filename('Program(*.exe)|*.exe','')
}
```

RUN APP:

```
if (!file_exists(global.app1)) exit
execute_shell(global.app1,"")
mouse_clear(mb_left)
```

Turn based code!

For every 16 pixels traveled, it should subtract a turn point...

(I assume that turnpoints means two players, so I'll try this for the first player)

in creation:

```
{
global.turnpoint1 = 15;
tempX = x;
tempY = y;
}
```

In the step:

```
{
if (global.turnpoint1 > 0)
{
if(abs(x - tempX) >= 16 || abs(y - tempY) >= 16)
{
tempX = x;
tempY = y;
global.turnpoint1 -= 1;
}
}
else
{
sprite_index = "new sprite";
}
}
```

Video Game History

Timeline 58-02

1958 Willy Higinbotham designs the very first video game, "Tennis for Two", played on an oscilloscope at the Brookhaven National Laboratory in NY.

1961 Steve Russell designs "Spacewar!" on a PDP-1 mainframe computer at the Massachusetts Institute of Technology (MIT).

1963 Nintendo Co. Ltd, previously involved in the manufacture of playing cards, moves into the games market.

1967 Ralph Baer designs the first video game played on a television set, "Chase" for Sanders Associates.

1971 Nolan Bushnell and Ted Dabney design the first arcade game based on Russell's “Spacewar!", titled "Computer Space".

1972 Maganavox' Odyssey, designed by Ralph Baer, becomes the first home game console. Nolan Bushnell and Ted Dabney form Atari. They hire programming wiz Al Alcorn, whose first project is to design an arcade game called "Pong". Atari's Pong Doubles becomes the first arcade game to allow 4 player simultaneous play.

1973 Atari's Gotcha becomes the first video game to have a maze, something we'd see a lot of years later.

1974 Atari's Gran Trak 10 becomes the first arcade driving game.

1975 Al Alcorn's dedicated TV game "Pong" becomes Atari's first home video game product. Midway's Gunfight is released, the first arcade game to use a microprocessor instead of hard-wired circuits.

1976 Coleco releases their first home video game product, the dedicated console "Telstar". Fairchild releases the first programmable home video game console, the Fairchild Video Entertainment System, which would later be renamed Channel F. Street price: $170. Nolan Bushnell sells Atari to Warner Communications for a reported $28 million. Sharp Image's coin-op Death Race becomes the first arcade game based on a movie (though it wasn't "licensed"), Death Race 2000.

1977 Atari releases the home video game system Atari Video Computer System (VCS), later known as the Atari 2600. Street price: $249. Coleco releases their first programmable home video game console, Telstar Arcade.
Nintendo releases their first home video game product, the dedicated console "TV Game 6". RCA releases the home video game system RCA Studio II. The system plays games in black and white only. Avalon Hill's "Microcomputer Division" is formed. Milton Bradley releases the first cartridge-based handheld video game system

1979 Atari releases their first home computer system, the Atari 400. Texas Instruments releases the TI-99/4A home computer. It retails for $1499.99. Exidy releases the arcade game "Starfire", the first to allow a player to enter his initials with a sufficient high score. Atari releases the arcade game "Asteroids".

MicroVision. **Data East** is formed. **Infocom** is formed.
Atari releases the first game to use a trak-ball controller, **Atari Football**. Namco's **Galaxian** becomes the first arcade game to use all color sprites. Though Pac-Man would revolutionize the genre only months later, it was Sega/Gremlin's arcade game **Head On** that first employed the "eat the dots" style. Vectorbeam releases the first head-to-head fighting game, the vector-based **Warrior**.

1980 Mattel releases the home video game system **Intellivision** after test marketing the system in California the prior year. David Crane, Alan Miller, Bob Whitehead, and Larry Kaplan form **Activision**, the first third-party software developer. Namco releases **Pac-Man**, which becomes the very first animated main character in an arcade game. APF releases the **Imagination Machine**, a computer add-on for the home video game system APF MP-1000. Nintendo releases the first in a long series of handheld games called **Game & Watch**. Sharp Image's **Star Fire** becomes not only the first arcade game to feature a cockpit-style cabinet (there is also a stand-up style cabinet), but perhaps more importantly it is the first arcade game that allowed you to enter your initials for a high score, beating Atari's **Asteroids** the the punch by mere months.. The first video game "**Easter Egg**" is born when Warren Robinett programs a technique to find his name while playing his Atari 2600 game "Adventure". **Broderbund** is formed. **Mindscape** is formed.
Space Invaders Part II (aka **Space Invaders Deluxe** in the US) becomes the first arcde game to include an intermission ("SOS"). Taito's coin-op **Stratovox** becomes the first game with speech synthesis. Williams' **Defender** becomes the first arcade game to feature a world where things DP Royal Archives - Video Game Timeline

1978 Bally releases the home video game system Bally Professional Arcade. The console would later be sold to Astrovision and marketed as the Astrocade in 1980. APF releases the home video game system M-1000 and MP-1000. Atari releases the arcade game "Football", the first game to institute a trackball controller. Magnavox releases the home video game system Odyssey2. Midway releases the arcade game "Space Invaders", the first video game to keep track of high scores. Automated Simulations (which would later be called Epyx) is formed. Nintendo releases it's very first arcade game, Computer Othello in Japan. Leijac's Space Wars becomes the first arcade game to use vector technology and graphics. Namco releases their first arcade game, Gee Bee.

Atari releases the arcade game **Battlezone**, the first game with an interactive 3D environment and first-person perspective. Stern's **Berzerk** becomes the first game to be blamed for someone's death when in January 1981, 19 year old Jeff Dailey died of a massive heart attack while playing the game. The first arcade game to be designed by a woman (Dona Bailey) is released by Atari: **Centipede**. She got some help from Atari guru Ed Logg.

1981 The very first IBM PC rolls off the production line, with it's 8088 processor running at a whopping 4.77mhz Arnie Katz, Bill Kunkel, and Joyce Worley form **Electronic Games Magazine**, the first commercial publication dedicated to video games. Nintendo releases the arcade game "**Donkey Kong**".Atari releases the arcade game "**Tempest**", the first color vector graphics game. Commodore releases their home computer **VIC-20**.

1982 GCE releases the first vector-based, portable home video game system **Vectrex**. Street price: $200. Entex releases the portable home video game system **AdventureVision**. Coleco releases the home video game system **ColecoVision**. Street price: $175. Atari releases the home video game system **Atari 5200** Super System. The Arcadia **SuperCharger** is released for the Atari 2600, adding graphic and programming power to the console and featuring all new cassette-based games to take advantage of this. Arcadia later renames itself to Starpath.
Mattel releases the **IntelliVoice** expansion module for the Intellivision home system, allowing the system clear, independent speech synthesis and featuring all new games to take advantage of this. North American Phillips releases the "**The Voice**" expansion module for the Odyssey2 home system, allowing the system clear, independent speech synthesis and featuring all new games to take advantage of this. Commodore releases their home computer **Commodore 64**. Emerson releases the home video game system **Arcadia 2001**. Amazin' Software (which would later become **Electronic Arts**) is formed by Trip Hawkins. Interplay Productions is formed by Brian Fargo. By 1985 they would simply go by the name **Interplay**. Sega's **Zaxxon** becomes the first arcade game to employ an isometric "3/4" view.

1981 The very first IBM PC

Vectrex

1983 Mattel releases their home computer **Aquarius**. Coleco releases their home computer **ADAM**, which also integrates with the ColecoVision system. The computer is a dismal failure and if not for Coleco's toy line (notably Cabbage Patch Kids, the company may very well have folded). Street price: $600. Atari announces the Atari 7800, but it isn't released to the public until 1986. Sega's **Astron Belt** becomes the first arcade game to use laserdisc technology. Cinematronics releases the arcade game **Dragon's Lair**, the first animated laserdisc game. Nintendo releases the home video game system **Famicom** in Japan. **Infogrames** is formed. **Origin Systems**, best known for their "Ultima" series of role-playing games, is formed. **Sega** Enterprises is formed, flops, and is ultimately sold off to Bally, a venture that would last just about a year. **Virgin Interactive** is formed. Bally/Midway's **Journey**, capitalizing on the popularity of the rock band of the same name, becomes the first game to use digitized graphics. Bally/Midway's laserdisc game **NFL Football** becomes the first arcade game to accept dollar bills. Atari's **I, Robot** becomes the first arcade game to use 3D filled polygon graphics. Leijac's **Cosmic Chasm** becomes the first home game to be converted into an arcade game. It was a GCE developed game for their Vectrex home console originally.

1984 The home video game market **crashes**, sending dozens of hardware and software game manufacturers into chapter 11. Notable companies that either folded or stopped producing video games: Apollo, US Games, Telesys, Data Age, Spectravision, and 20th Century Fox. Apple releases the **Macintosh** home computer. The speedy little machine with a built-in black and white monitor sported a zippy 7.83mhz processor. Street price: $2000. Atari releases the **Atari ST** home computer. IBM releases the **IBM PC AT** line of home computers. Atari releases the arcade game "**I, Robot**", the first to feature 3D polygon graphics. **Accolade** is formed by ex-Atari and Activision guys Bob Whitehead and Alan Miller. **Ocean** is formed. **Psygnosis** is formed. Bally closes down Sega Enterprises, and Sega of Japan is sold to investors who re-launch Sega in the United States, appropriately titled **Sega of America**. Atari's **Firefox** becomes the first coin-op to use actual movie footage in a game.

1985 Nintendo releases the home video game system Nintendo Entertainment System (**NES**) to a test market in NY, this would become the North American version of the Famicom. Street price: $199. Microsoft releases their first version of **Windows**, v 1.0 Alex Pajitnov designs the PC game "**Tetris**". The first **CD-ROM drive** for personal computers is released, sporting a full "1X" speed. Commodore releases the first in their **Amiga** computer line. Street price: $1295. **Datasoft** is formed. **Titus** is formed. **Westwood Studios** is formed.

1986 Sega releases the home video game system **Sega Master System**. Street price: $199.A ari releases the home video game system **Atari 7800**. It is the first system to eature backwards-compatibility and is compatible with existing Atari 260 cartridges **Cosmi** is formed. **Pectrum Holobyte** is formed. **Ubi Soft** is formed. SNK's **Psycho Soldier** becomes the first arcade game to feature a fully digitized Vocal soundtrack.

1987 NEC releases the first 16-bit (though this is widely disputed) home video game system **PC Engine** in Japan. **Acclaim** is formed and promptly becomes the first independent US publisher for the Nintendo Entertainment System, quite a feat at that time. Atari releases the home video game system **Atari XE**, internally compatible with their 8-bit computer line but marketed as a game console. **Apogee** is formed. **Absolute** is formed by ex-Activision guys Dan Kitchen, Garry Kitchen, John Van Ryzin, and Alex DeMeo. David Crane would later re-join his gang. **Cinemaware** is formed. **Gametek** is formed. **Koei** is formed. **Maxis** is formed. **1988** The first CD-ROM game, "**The Manhole**", is released by Mediagenic. **Codemasters** is formed by ex-Commodore 64 programmers Richard and David
Darling. **Domark** is formed. **Microplay** is formed. Williams' **NARC** becomes the first 32-bit arcade game, utilizing a TI 34010 processor.

1989 Nintendo releases the handheld video game system **Game Bo**y. Street price: $109. NEC releases the video game system **Turbografx-16**, the North American version of the PC Engine. Street price: $189. Sega releases the video game system Sega **Genesis**. Street price: $249. Atari releases the first color handheld video game system Atari **Lynx**. Street price: $149. NEC releases an enhanced version of their PC Engine console, **SuperGrafx**, in Japan. The system would only feature 5 games that take advantage of this enhanced hardware and was never commercially available outside of Japan. Gottlieb's **Exterminator** becomes the first arcade game to use 100% digitized graphics.

1990 Trip Hawkins leaves Electronic Arts to form **3DO**. NEC releases the **Turbografx CD** expansion for Turbografx-16 owners, upgrading that system's to CD-ROM functionality and featuring all new games to take advantage of this. Street price: $399. SNK releases the home video game system **Neo-Geo (AES).** Its insides are the same as it's arcade counterpart, (the MVS), for the first time TRULY bringing arcade games home. At a price.. Street price: $699. SquareSoft releases the first in what will become the best-selling console role-playing series ever: **Final Fantasy**. Sega releases the handheld video game system **Game Gear**. **Microprose** is formed. NEC releases a handheld version of their Turbografx-16 console, completely compatible with its cartridges, **TurboExpress**.

1991 Commodore releases the CD-based home video game system **CDTV**. Street price: $999. Fujitsu releases the **FM Towns Marty**, the world's first 32-bit console. It was a game console based on the FM Towns computer they released in 1989. Nintendo releases the home video game system **Super Nintendo**. Street price: $199. Joe Santulli and Kevin Oleniacz form **Digital Press**, an independent publication for video game collectors of every system. Two major PC entertainment publishers, **Cinemaware** and **Epyx**, close shop for good. **id Software** is formed.

1992 Philips releases the CD-based home video game system **CD-i**. Sega releases the **Sega CD** expansion for Sega Genesis owners, upgrading that system's to CD-ROM featuring all new games to take advantage of this.

1993 Atari releases the first 64-bit (though this is widely disputed) home video game system Atari **Jaguar**. Street price: $250. Commodore releases the **Amiga CD32**, a 32-bit game console based on the Amiga 1200 computer. Street price: $399. Trip Hawkins' 3DO releases the CD-based home video game system **3DO Multiplayer**. The Panasonic version is the first on shelves, Goldstar would follow several months later. Street price: $699. Midway's **NBA Jam** becomes the first sports-licensed arcade game.

1994 The Entertainment Software Ratings Board (**ESRB**) is created, a product of the US government - led by Sen. Joseph Lieberman - to rate video games in much the same way the MPAA rates movies. Sega releases the **Sega 32X** expansion module for Sega Genesis owners, upgrading that system's processing power to 32-bit and featuring all new games to take advantage of this. Street price: $159. Sony releases the 32-bit CD-based home video game system Sony **PlayStation** in Japan.

1995 Sega releases the 32-bit CD-based home video game system **Sega Saturn**. Street price: $399. Sony releases the Sony **PlayStation** in the US. Street price: $299. Nintendo releases the first 32-bit portable video game system, the **Virtual Boy**. The system produces real 3D effects in shades of red. Street price: $179. Sega releases a handheld version of their Genesis console, completely compatible with its cartridges, **Nomad**.

1996 Nintendo releases the 64-bit home video game system **Nintendo 64**. It is the last of the home consoles to use cartridge media. Bandai releases the home video game system @World, also known as **Pippin**. Street price: $499.

1997 Tiger Electronics releases the handheld video game system **game.com**.

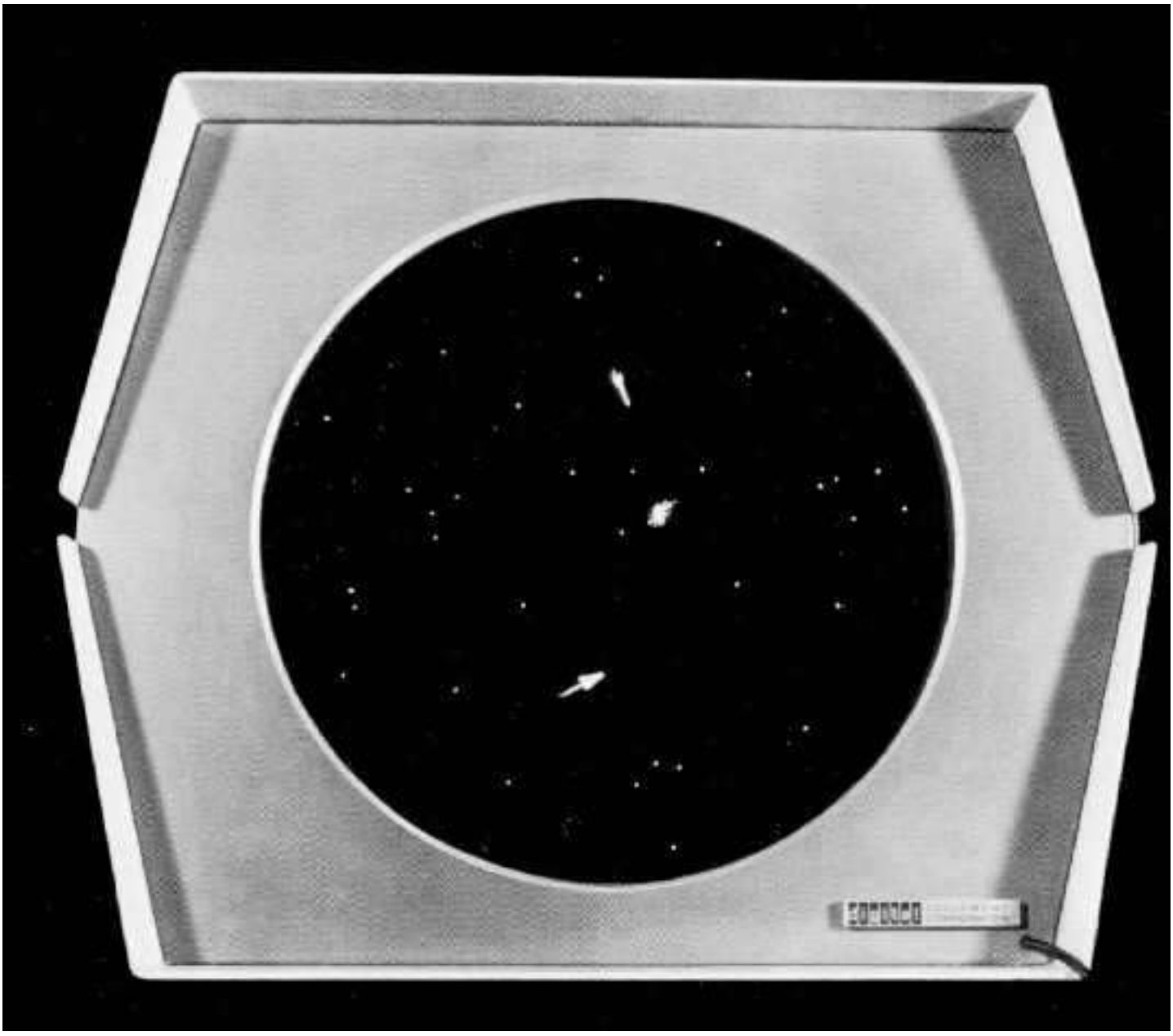
1998 Sega releases the home video game system Sega **Dreamcast** in Japan. Nintendo releases the color handheld video game system **Game Boy Color**. The system is backwards-compatible with existing Game Boy cartridges. World of Atari, which would the following year be dubbed "**Classic Gaming Expo**", is held for the first time in Las Vegas. **Hasbro Interactive** purchases the Atari legacy for a reported $5 million.

1999 Sega releases the **Sega Dreamcast** in the US on 9/9/99. Street price: $199. SNK releases the color handheld video game system **Neo-Geo Pocket**. Sony releases the home video game system **PlayStation 2**.

2000 backwards-compatible with existing PlayStation games. Street price: $299.

2001 Microsoft releases its first home video game system **Xbox**. Street price: $299. Nintendo releases the home video game system **GameCube**. Street price: $199. Nintendo releases the **Game Boy Advance** handheld system. It is backwards-compatible with Game Boy and Game Boy color cartridges, extending the dynasty.

2002 This marks the first year that every video game console in production has an online option; in addition, they're all broadband-ready options! Sega leaves the console industry and becomes a software-only designer. We never thought we'd see Sonic on a Nintendo system, but it happened. Two of the world's most renowned role-playing game developers merge. Square and Enix announced that there would be jointly produced titles as early as 2003.



Interviews

Some of the best with Game Maker

Carl Gustafsson

1.Please introduce yourself for those who don't know you.

My name is Carl Gustafsson and I live in Sweden. I am 29 years old, have been married since 1997 and have two children. Earlier I was pretty active at the Game Maker Forum, trying to help people and answering questions. That is probably why I am being interviewed right now. Or perhaps because of the Beginner's Guide that I wrote?
Sadly I have not been able to be as active on the Forum as before. This is largely because of the time it takes to raise and take care of 2 children, especially since the youngest of them (2 months old boy) was born with some defects, such as a heart problem. Which however will be fixed surgically really soon (March 4th).

2. When did you start using game maker?
I think it was some time late 2001.

3. Why did you start using game maker?
Because it was free, it was about making games and it was good!

4. Have you used any other game making or multimedia programs before?
Only tried briefly. Such as DarkBasic and some other that I have forgotten the name of.

5. What have you created with game maker that you think is just great?
DogFight. That is the only game I have made actually. It is a two-player game that can be played via the Internet.

6. What new project(s) are you working on now?
The Game Maker course at www.gameuniv.net.

7. What is your future plans with game maker?
I really do not know. There are so many things I want to do, but time seems to not be enough. I have also developed an interest in 3D graphics. Hope to be able to join the two (nice sprites in GM).

8. Other than game maker, what other hobbies and interest are there that you enjoy?
As I mentioned earlier, I am very interested in 3D graphics. I specifically try to use Blender () for that and can recommend it heartily! Apart from that, I play the piano a bit.

9. What are your future career plans?
I work at a very small company, writing applications for Pocket PCs, and for "normal" PCs too. I hope that we can grow and get an outcome from it.

10. Ending comments?
I want to say a really great THANK YOU to Mark Overmars for making Game Maker and devoting so much time to us wannabe game creators. =o)

Thank you also to John, aka Morphosis, for interviewing me and for making the Game Maker's Data Mag.
Finally, I would like to welcome people to my web page, hem.passagen.se/birchdale/carl. However it is sadly not updated since last year. I will soon (I hope) move it all over to my newly registered domain, birchdale.net!

Mr. Chubigans

1.Please introduce yourself for those who don’t know you.
My screen name is Mr.Chubigans. I have been in the game making biz for almost 5 years now. My game company is Vertigo Games, and have made games like Helicopter Cacophony and Ore no Ryomi

2. When did you start using game maker?
Around early 2000 or so.

3. Why did you start using game maker
I wanted to make a kart racing game called "Team Fortress Kart Racing," but it ultimately failed. Oh, those were the days...

4. Have you used any other game making or multimedia programs before?I tried Dark Basic, Click & Play, 3D Game Maker...but none have the ultimate power like Game Maker 5 does. None of them.

5. What have you created with game maker that you think is just great?
I think Explodin' Crapola: Helicopter Cacophony 2 is really a showcase of what teamwork can accomplish. It was a joint project with myself and Him, who is an excellent graphic artist.

6. What new project(s) are you working on now?
Ore no Ryomi 2, and planning my next game, even though ONR2 may be my last game. I don't know yet.

7. What is your future plans with game maker?
Hopefully continue with it, but again, life outside of Game Maker is extremely busy. I would love to finish my dream project, In DreAms (no pun intended).

8. Other than game maker, what other hobbies and interest are there that you enjoy?
I'm a Home Theater man. I love tinkering with my equipment, popping in Starship Troopers, and being blown away by the sound. I also do animations, and won a four year scholarship for college with my 10 minute animation called "Superkintetica!"

9. What are your future career plans?Becoming an animator, perhaps creating my own animation show.

10. Ending comments?
Never, ever give up with Game Maker. Trust me, it gets easier

Game Previews

Soon to a PC near you!

There is something wonderful about making sequels, since you know for sure what worked and what didn't in the original game, and what to improve on for the new game. Ore no Ryomi had a small catalog, few unlockables, and no real motivation after buying all the items in the game. Yet somehow, it was very popular.

Working on the original Ore no Ryomi was tough, and took almost two years of off again, on again planning. I was actually finished with the game when the file got corrupted, forcing me to start all over. It was a painful expierence.

But all of that has changed with the sequel, called Ore no Ryomi 2: The Restaurant. I'm actually having fun making this game! I have started from scratch this time, destroying the original GMD file for the first game, and building a more robust engine. People who enjoyed the first game had many requests, and I have exceeded even my own expectations for this game. This is not an upgrade to Ore no Ryomi. This is a true sequel.

One of the biggest additions to the game is the new restaurant management features. It allows the user to advertise in television, radio, newspaper...or even something as cheap as putting flyers on cars. By buying advertisement, your advertise percentage goes up. The further it goes, the more customers you get. By not advertising, your customers will be sparse throughout the day, even if you have purchased all the food items in the game. Its a new mechanism that keeps the game more sim-like.

Another new addition is the game show Iron Cook. Every now and then, the user will get an invitation to appear on the Iron Cook show in Tokyo, where they will compete in certain challenges, such as making 20 perfect beer orders in 60 seconds (see screenshot). Its another fun addition that earns the player money to spend in the catalog.

Improvements over the original are numerous. There are no day limits to meet (like the original game's 40 days). But there are TONS of unlockables- almost 30 different objects- that unlock when you meet certain requirements. The catalog has expanded immensely. There are new emergency events, like fires. The days are shorter, but more action happens. The food menu has doubled to 12 items you can cook, compaired to the 6 in the last game. There are tons of new orders, such as the pizza having 10 different recipies, unlike the original game's 4. There's even a trophy case full of trophys to unlock, such as the award for "100 Perfect Orders. More, more, more are the key words here.

As for the graphics and sound...well, they are also getting an overhaul. Compressed MP3s are being used once again, like Explodin' Crapola: Helicopter Cacophony 2. The graphics are being overhauled, with Morphosis and Him onboard.

Overall, this will be a huge game- one of the biggest I have ever made in my five years of game making. The target date is April/May, with the target game size being under 5 megs. Keep watching for updates on the release date, because I get a good feeling that we may release it sooner.

Here's to making an enjoyable game!

Different characters to choose from.

Music Superstar is a very huge project that I have been working on for months now. It is actually moving much faster than I thought it would! It's one of those games were everything seems to work out just right, unless I don't play music at any clubs, don't train, don't make stickers to promote my band, or don't bother to pay off my loans...hold up one second, that's my ROCK STAR from Music Superstar!

That's right, in Music Super star you have to try to climb your way to the top of the charts. There are many ways to do it and many things to stop you from doing it. You have to get famous, but first you need to train so you can clubs will let you play there which gets you money to create promotional material and may be one day sign a record deal or even make a music video! A lot of things must be taken care of in order for things to fall into place. You have to take care of your appearance, charm, entertainment value, and so much more! It's a hard life to be #1.

Customize your character

Customize your character before you begin their career! Name them, adjust their talent, charm, and more. There are many characters to select from, a Rock Star, a Rapper, a Pop Star, and more! Each character will be a completely different game! Different music, graphics, and difficulty. You know the life a beautiful pop star can't be as hard as the life of a rugged rock star!

The City Streets

The Bank

Shop around and have fun!

Promotion is a key to fame!

Take a vacation and chat online with others on vacation!

Morphtown

In the city of “Morphtown” you can try to get into clubs to play music. If you are not that talented or famous, they may not let you play or they may pay you a few dollars to play. You can also visit your bank to borrow some money or pay back money that you owe. The bankers are pretty rude and will come after you if you don’t pay them their cash. If you want to visit an “on-line” lounge you can. These on-line lounges are actual chat rooms you can use when playing the game if connected to the internet. You can chat with the other players comparing stats and just chilling out.

As with many cities there are dangers in Morphtown. Dangers like getting mugged, prostitution, and drugs. It’s up to you what path you want to take, but I am warning you about these dangers. :)

Money, money, money!

If you have the fame and fortune you have to spend your money you can get a house, a car, a guitar, or even take a vacation. Spend your money wisely, you don’t want to go to the local drug house and blow your cash on drugs for a quick fix! But you can boost your career really fast while taking the risk to get busted by the police.

What ever you buy in the game for the most part will not just sit there and get dusty, you can use you items. For example, you buy a house and want to have a party to increase your fame and connections...no problem, have a party. And if you if you buy a musical instrument you can play it and increase your skills.

Training and promotions

Practice makes is one way to become better, or you can make a bunch of flyers, T-shirts, stickers, or a web site to promote yourself. These items will increase your fame, charm, and who knows what. This stuff cost money and may fail at times, but it’s worth a shot. Who knows you might be able to make a music video.

Get the game

Take a look at my site **http://mea.invisiblefury.com** to check availibility and more game details.

Rock on!

Music Superstar
© 2004 Morphosis Enter-Active
John Hempstead

Looking Good

DVD Case tutorial

If you make a great game and want it to look even better. Make a case for it! That is what I do for most of my pretty good games I plan to sell or give as gifts.

It make the game look so much better when it's packaged up. You can also make a great instruction booklet which fits inside the case.

It's really not that expensive to create one of these. I happen to find some DVD cases, 5 packs at a store (US) called "Dollar Tree" 5 cases for $1.00. I spent around $60.00 on this great deal. I did not find them again on the last visit to the store. You still can find them just in just about any store and the run from $6-8/5 pack.

Printing the page can be done on your home computer or you can go to a copy place and it cost about $1.00/color page.

On my cases I include the title of the game on the front and on the back I put down the features of the game, a partial story, many screen shots and other game related graphics. I also include my logo and copyright information. I may use other logos if needed like the Compact Disc logo, the MP3 logo, DVD logo, or VCD logo. At times I create a brocade, it just looks more commercial. On the side I put the title and my logos.

To create them I use photoshop and/or Illustrator but any graphics program could work!

Here is a "Template" you can have to begin to create your own DVD case!

DOWNLOAD **DVD jpeg TEMPLATE**

* The template is 72 dpi, so PSP and photoshop users make sure you change the resolution to at least 200dpi for print.

Thank you for reading gmdm john