game reviews game development interviews code tutorials

GM data mag
morphosis enter-active

start a game company
Read the easy steps to start your own company!
page 70

game design sequence
Grande artistic goals and mountains of code...ready?

...a good rpg?
What makes one a good one?

interactive storytelling
Holding their interest

diversity
Man - Women - Boy - Girl, there are so many out there!

reviews
Shortline Express - Guitar Trainer and more!

GMDM Issue #4

issue 4

Contents

Credits

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

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

Other contributions to GMDM include and get a big THANKS!
CGW
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, and more!
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”

Morphosis: John Hempstead

From the editor

Welcome to the 4th packed issue of Game Makers Data Magazine.
It’s been a pretty good summer and I have spent more time this year away from my computer at nights and closer to fires roasting marshmallows! But, I did not forget about my passion, game design, graphic design, and GMDM! I plan to have a fall issue out and will begin working on the 5th issue when I return from vacation. When this issue you are reading now is released I’ll be swimming with the sharks! Enjoy Game Makers Data Magazine!

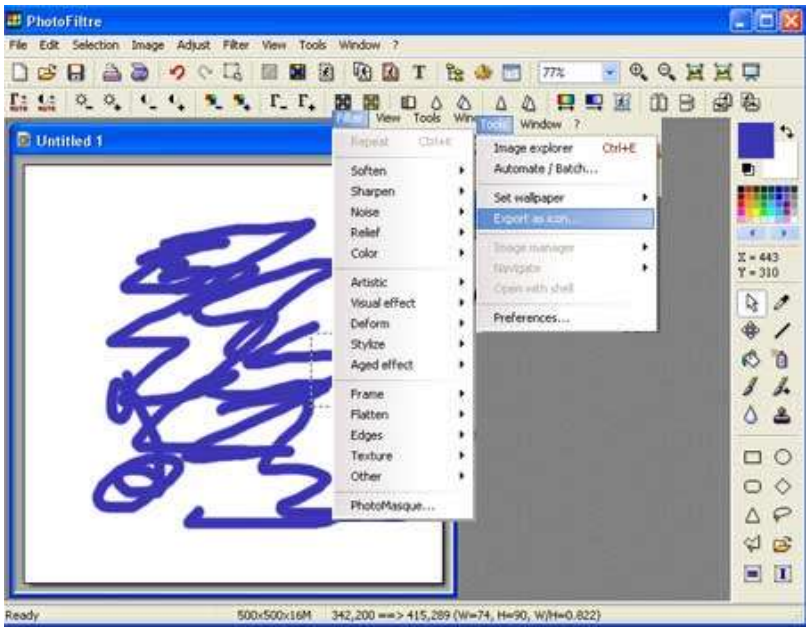
Morphosis Enter-Active
www.meagames.cjb.net
www.morphosis.filetap.com
http://mea.invisiblefury.com
John

GM Tools

Tools to use with game maker

FREE

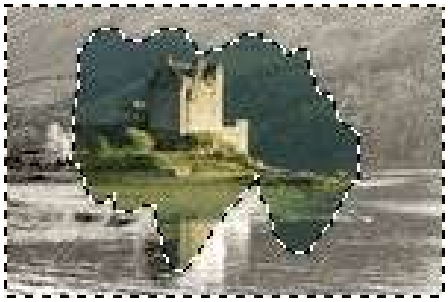
http://www.photofiltre.com/



PhotoFiltre is a complete image retouching program. It allows you to do simple or advanced adjustments to an image and apply a vast range of filters on it. It is simple and intuitive to use, and has an easy learning curve. The toolbar, giving you access to the standard filters with just a few clicks, gives a robust look.

Filters

Its wide range of filters allows novice users to familiarize themselves with the world of graphics. You can
find the standard adjustment functions (Brightness, contrast, dyed, saturation, gamma correction) and also artistic filters (watercolor, pastels, Indian ink, pointillism, puzzle effect).
filters to be discovered !



Vectorial selections

PhotoFiltre uses two types of vectorial selections. The first type uses automatic shapes (rectangle,
The second type corresponds to the lasso and polygon. They both allow a customized form by drawing a shape by hand or using a series of lines. Every selection can be saved into a seperate file, to be used later on

Tool bar

The toolbar is primarily made up of drawing tools, such as pipette, displacement cursor, fill bucket, aerosol, brush, drop of water (blur), cloning stamp, smudge (finger) and magic wand.

Other functions
Image browser
Scanning of images using a TWAIN compatible device (scanner, webcam, photocamera)
Transparency management for GIF images and exporting them to icons (16, 256 or 16 millions of colors)
Advanced text effects (rotation, shading, bevel)
Serveral types of contours and textures

...I have a game!

Now show others!

When it comes to making a game available for online play, there are numerous avenues you can take. At this point, no one can really tell if one method is better than another. So try one or try them all.

You may choose to go through one of the services that provide direct access to games via the Internet. These include companies such as TEN (Total Entertainment Network), MPath`s Mplayer, Dwango, XBand, and MPG-Net.

Some Internet service providers (companies selling connections to the Internet) also are starting to provide areas for online game play. An example of one such provider is EarthLink Network and its "Multiplayer Entertainment" site. VR-1, which initially will offer its games in Europe, eventually plans to bring its content (which could be developed internally or by third parties) to the US using this avenue.

Another option is to go through a service that connects players via one or more of the major online services, such as America Online, CompuServe, Prodigy, Delphi, and Genie. Taking this route are Engage, which also plans to offer games via the Internet; Aries Online Games (the publishing arm of Kesmai), which also offers games via Internet providers such as EarthLink; and The ImagiNation Network (acquired by AOL), which will make its new 3D game environment called CyberPark available initially via AOL, eventually adding other online and Internet service providers.

If a developer has the money, it can set up its own network--which is exactly what Origin is doing for Ultima Online. The 3DO Company (Redwood City, CA) also took this route with its first Internet-based game, a role-playing fantasy called Meridian 59. Or if you don`t want to deal with any of these options, you can include the ability to let players establish their own Internet connections with a program called Kali. Kali began as a DOS program that tricked computers on different networks into thinking they were on the same LAN. There are now OS/2, Windows 95, and Macintosh versions of Kali. You can test the software for free, but if you decide to use it, you need to order and pay for an official copy.

For more information, check out these services and game developers` Web sites. Those of you interested in playing, keep in mind that none of these is free (although some trials are free); some services charge by the hour, some by the month, so be sure to check before subscribing.

Accolade: www.accolade.com
America Online: www.aol.com
Aries Online Games: www.ariesgames.com
CompuServe: www.compuserve.com
Dwango: www.dwango.com
EarthLink Network: www.earthlink.com
Engage Games Online: www.gamesonline.com
Genie Interactive: www.genie.com
ImagiNation Network: www.inngames.com
Interworld Productions: www.homeworld.net
Kali: www.axxis.com/kali
Meridian 59: meridian.3do.com
MPG-Net: www.mpgn.com
Mplayer: www.mplayer.com
Total Entertainment Network: www.ten.net
Ultima Online: www.owo.com/uo.html
VR-1: www.vr1.com
XBand: www.xband.com

Tools of Drama

The Three Act Structure

Randy Littlejohn

In a dramatic presentation the pattern of human conduct is developed within the framework of a particular structure or dramatic form, which, despite passing innovations, has persisted over thousands of years. The study of this structure is the next step in understanding the principles of drama.

Dramatic structure is the destruction and restoration of the balance of forces. Simply, it is the process of getting into, and then back out of, trouble. Examine any compelling story and you will find that at the outset an equilibrium exists; the potentials of struggle may be present and even boiling under the surface, but the trigger has not been pulled. During the presentation or interaction the balance is destroyed. At the conclusion of the drama balance has been regained. It may be a balance of forces completely different from that found in the beginning, but a balance is present.

This balance-imbalance-balance structure is divided into five parts. The parts are:
- Exposition
- Complication
- Climax
- Resolution
- Conclusion

These five parts fall into three acts in the following way:

Act One
Act One is composed of exposition: the initial situation is described. Time, place, and the social and psychological aspects of the situation are set forth for the information of the audience or participant. The characters are introduced and the audience is given everything necessary to understanding their reasons for being. The theme is introduced, perhaps as foreshadowing, so that the spectator is aware of all the forces that will lead to conflict. Above all, the exposition must catch the interest of the audience. First you have to get their attention. The exposition leads to the inciting action. The inciting action is the moment of destruction of the balance of forces — the trigger being pulled. Sometimes it is called the inciting moment or the overt act, meaning that it is the clear, visible action which incites the struggle.

Act Two
Act Two is composed of complications leading to a climax: once the balance of forces has been disturbed by the inciting action, the storyteller goes about the business of getting her characters into trouble. The complication is the body of the drama. It is the bringing together of the protagonistic and antagonistic forces in a series of more and more important crises in the struggle. The development of the conflict continues with increasing fury until it can go no further without resolution.
The climax is the high tide of the drama. From the spectator's or participant's standpoint, it is the high point of excitement. From the standpoint of conflict, it is the point at which the protagonist and antagonistic forces arrive at an impasse that allows no other solution but to finally resolve the difficulty. This moment is often a seemingly unsolvable problem.

Act Three
Act Three is composed of the inevitable unwinding of the conflict, governed by the turn the conflict takes at the climax, leading to a conclusion. During the resolution the tension drops somewhat, in that the audience thinks that it is able to forecast the final result, though not the method of reaching it. This unwinding must be handled without any loss of interest. Surprise and more suspense are the tools to solve this problem. Often a false climax is followed by the true climax, which is then followed by the true resolution.

Finally, at the conclusion, the questions of the audience are logically and finally answered. In contrast, a conclusion can also simply be an emotional pay-off, as in the final scene of *Star Wars*, which explains nothing, but communicates the victorious return of our heroes. Whether or not the conclusion is satisfying comes back to how we relate to the characters. How do we feel about the protagonist and her goal? If the protagonist has a clear and compelling goal that we can all relate to, if we care about the protagonist and feel that she has fought an admirable and tough fight against a worthy adversary, then we will experience a purge of emotions when the protagonist finally succeeds, or fails.

The idea of an interactive computer story probably at first knee-jerk implies that many of the tools used to enhance a story with dramatic elements are now in the hands of the player instead of the writer or designer. In other words, the point of view, order of settings, and order of the story events, are now all at the whim of the "audience". Potentially lost are the balance-imbalance-balance over-structure, and the five-part (exposition, complication, climax, resolution, and conclusion) sub-structure of drama. Decreased control of these tools equals a dramatic problem.

A way that we can achieve both a non-linear, free-choice environment and keep our five-part dramatic structure is by assigning the non-linearity to the micro-level, while maintaining a scripted structure at the macro-level. That's a mouthful, I know. For simplicity sake, let's say that areas in an action/adventure game in which the participant is allowed to freely explore are made up of two elements: the environment and NPC's. This means that the dramatist has two ways to communicate information to the participant: through environmental design and events, and through the actions and dialogue of NPC's.

One approach to achieving dramatic structure while maintaining free exploration is to create environments and NPC's that are informed by the five-part dramatic structure.
A way to achieve both is to group possible environmental events and possible NPC actions and dialogues into five libraries: exposition, complication, climax, resolution, and conclusion. In other words, the participant can explore the various environments at will and encounter the NPC's at will, but as long as we are working from the exposition library, for example, no matter what happens, the events and NPC actions will be about the time, place, the social and psychological aspects of the situation, the introduction of characters, their reasons for being, the introduction of theme, and foreshadowing conflict. When the inciting incident has been enacted, and it is time to begin on the conflict library, no matter where the player goes, no matter who the player encounters, the environmental events and the actions of NPC's will be about conflict, and so forth. In this way we maintain a large degree of non-linearity, while maintaining enough control to guarantee our dramatic structure.

Suspense
The basic task of the dramatist consists of capturing the attention of the audience and holding it for as long as required. If the audience fails to concentrate on what is happening from moment to moment, on what is being said and done, all is lost.
The creation of suspense underlies all dramatic construction. Expectations must be aroused, but never, until the last, wholly fulfilled; the action must seem to be getting nearer to the objective yet never reach it entirely before the end. Above all, in order to maintain interest, there must be constant variation of pace and rhythm. There are many kinds of suspense: suspense may lie in a question like, 'What is going to happen next?', or in 'I know what is going to happen, but how is it going to happen?' or, indeed, 'I know what id going to happen and I know how it is going to happen, but how is X going to react to it?' Suspense can also be aroused by a quite different type of question, such as, 'What is it that I see happening?' or by the question 'these events seem to have a pattern; what kind of pattern will it turn out to be?' One thing, however, is certain: some sort of basic question must emerge fairly early in any dramatic form so that the audience can settle down to the main element of suspense. At its most basic suspense depends on the existence of at least two possible solutions to the problem.

The human attention span is relatively short. One major suspense element is not enough to hold an audience's attention throughout the course of a story. Beyond the main question or theme or story arc, the rise and fall of subsidiary arcs, arising from subsidiary suspense elements, must be superimposed. For instance, while our main interest is held by the question of why Planet Alderon was targeted, at the same time, but in a much shorter time span, we are eagerly asking ourselves how the princess now being questioned was involved and whether she had anything to do with the final action. The main suspense element inspires subsidiary suspense elements. There is an element of suspense needed for each scene or section of the action, superimposed on the main suspense element of the work.
Secondary questions, goals, or problems could be a part of the set design. For instance, a body is found; how was the character killed? The answer seems to be inside a cave, but is the cave safe to enter? What about the giant footprints leading into the cave? The machine at the entrance of the cave seems to be part of the answer, but what does it do? Or even more simply, 'The path doesn't look too safe. What is beyond that next corner?' A study of theme rides in amusement parks would offer many examples of the creation of suspense in set design.

In order to insure moment-by-moment interest, there must be a third, purely local, micro-level element of suspense at any given moment in a well-devised story — the line of dialogue or single detail of business the characters are engaged in at that moment. Good dialogue and good moment-by-moment action is unpredictable. Predictability is the death of suspense and therefore of drama. In addition, a character who never says a line which is arresting, witty, amusing or interesting, will have great difficulty in catching the audience's sympathy or, conversely, loathing.

If we are careful to design our "stages" in an evocative way, and if we populate our stages with unpredictable and interesting NPC's, we can get the player to wonder 'What is going to happen next?', or 'I suspect I know what is going to happen, but how is it going to happen?' or even 'What is it that I see happening?'.

Creating Intelligent Creatures

Donna Coco

What are looks without brains? That`s the question many a game developer is posing these days. With incredible graphics becoming more the norm than the exception, developers are turning to Artificial Intelligence (AI) to distinguish their titles from the hundreds vying for consumers` attention.
True, AI in games is nothing new. But the amount and sophistication of the AI showing up in games is, as is the growing trend to use AI to make characters appear more interesting and believable as "living" creatures. Driving these trends is the ever-increasing processing power of computers, in both microprocessors and 3D accelerator chips. As more processing power becomes available at lower costs, computing the graphics portion of games becomes less taxing, freeing up processing power for other features, such as AI.

"Generally, AI hasn`t gotten a lot of attention in the past, but I`ve seen that change in the last few years," confirms Steven Woodcock, a software engineer for Real3D (Orlando, FL) who also has an extensive background in AI and acts as a consultant. Additionally, he has run sessions on AI at the Computer Game Developers Conference (CGDC) for the past two years. "I ask a lot of the participants at my AI sessions at the CGDC how much CPU they get and how many people are working on the AI in their game. Just last year, the answers were often `less than a percent of CPU` and `Joe works on the AI half time.` This year the CPU percentage was pretty decent, and many companies have dedicated AI developers."

Perhaps one of the most intriguing games in development that`s using a hefty amount of AI is Galapagos from Anark (Boulder, CO). In this game, players guide a four-legged creature called Mendel through the 3D texture-mapped worlds of Galapagos, solving puzzles and helping Mendel eventually escape from this world. Here`s the amazing part: Mendel is an artificial organism with the ability to learn, adapt, and interact with his environment and the player. Some refer to this type of AI as artificial life; Scott Collins, development engineer at Anark, prefers to call it by its more scientific name, adaptive technology. "A neural network is an example of an adaptive technology," notes Collins. "Genetic algorithms are a form of adaptive technology as well."

But Anark didn`t use either of these technologies for its AI. For Mendel`s "brain," Collins and his coworkers developed their own version of an adaptive controller called Non-stationary Entropic Reduction Mapping (NERM). The NERM controllers accept inputs and produce outputs that are translated into Mendel`s behavior. It`s also self-organizing, which is how it learns. Collins explains, "Self organizing means that there`s nothing outside of it that dictates its final form. For example, take a dog and give it praise and Scooby snacks, or scold it, and based on your [input], the dog will figure out how to behave to minimize its scoldings and maximize the affection. It`s self-organizing. Our adaptive controller receives feedback error and, based on this error, what it tries to do is converge in a way to minimize this error."

In fact, the player doesn`t have to interact at all with Mendel for him to learn; he`ll do it all by himself. "If you take a newborn Mendel, it`s actually very amusing. He`s flexing his limbs and falling off ledges and dying all the time. But if you watch him during his first hour, you`ll notice how he dramatically changes from infant-like movements to more mature actions."

Actually, says Collins, Anark set out to create an adaptive-controller technology that could be used in many applications, including industrial. Galapagos is just the first. "Frankly, more of NERM`s commercial value is outside [of entertainment]. There`s a variety of methods we could`ve applied to this game, but the engineering world is more critical," says Collins, explaining that to compete on that level, Anark had to take a different approach, which resulted in the NERM technology.

That`s not to say the technology doesn`t fit well in games. There are definite benefits to using adaptive AI versus a more traditional rules-based AI in games, notes Collins. "You end up with a more human-like opponent," he says. "A quick example would be when you have an enemy in a shoot `em up, and it uses traditional AI. It`s rules-based, as in `if within a certain range, then do this.` The thing is, these behaviors are very easy to uncover. A human can explore those patterns very quickly and figure out how they work and how to beat the machine.

The reason an adaptive technology would be interesting here is there`d be enemies observing the behavior of the player and changing in response to the player`s play-style."

Adds Collins, "That`s why network games are becoming so popular. The human opponent is so interesting." Naturally, Collins finds Galapagos interesting, too. (The title should ship sometime this fall.) And because of the nature of Galapagos, it`s never the same game twice. "It`s quite addictive," says Collins.

Incredible Creatures Already on the market in the UK and expected any day in the US is Creatures from CyberLife (Cambridge, England), a spin-off of Millennium Interactive. This title lets players breed and raise Norns--cute little two-legged creatures with big eyes, long ears, and tails.

CyberLife`s technology, also named Cyberlife, is a proprietary system that uses neural networks. "What we`re doing is modeling biological building blocks that are then combined to create biological systems," says Toby Simpson, producer. "Our neural net is not like textbook neural nets. Neural nets were originally a biological metaphor, but they were highjacked by mathematicians. We stayed with the biological approach, where you have a huge set of chemicals, and no one individual neuron knows the big picture."
Norns begin as children and then progress to adolescents, adults, and finally, seniors. During their lifetime, they can breed, creating offspring with genetically different systems. Norns also make choices and learn from their mistakes. These creatures learn best and live longest, though, when assisted by the player, who can teach them right from wrong as well as help them with a number of tasks, such as breeding, finding food, and avoiding danger. If cared for, a Norn has a typical lifespan of about 15 hours. But because players can help create generations of Norns, the "game" can last much longer.

Upon hearing various stories about the Creatures, it`s easy to understand how people can get so attached to these simulated organisms. Relating one story, Simpson tells how two Creatures followed each other around all the time, much like best friends. Then one died. "The other just stood next to the body until it died, too," says Simpson. "To this day we have no idea what happened. We can`t, obviously, get inside their heads. But we can logically come up with an idea, and we think it committed suicide. The links in that Creature`s head between being alive and the other object were so substantial that it killed itself when the other Creature died. Those are the sort of things--the glimmers of life--that make this worthwhile."

Your Pet Computer Fin Fin--a charming "animal" that`s half dolphin, half bird and was created by engineers at Fujitsu--is another character recently introduced to the market that uses AI to appear more lifelike. The AI in Fin Fin on Teo, the Magic Planet is based on what Michael Pontecorvo, director of technology at Fujitsu Interactive (San Francisco), calls believable agents. "It`s good for modeling internal-state and decision-making processes that lead to externally believable behavior. The engine executes a set of productions that work from a goal to a solution through activating multiple plans. It doesn`t do the planning; it uses them. We pre-build the plans, and it applies those in an expert-system-like fashion," he says.

Technically, Fin Fin does not learn; the technology used for Fin Fin is not adaptive. Pontecorvo explains, "The only thing Fin Fin really does [that`s learning] is there`s a variable that`s familiarity. He becomes more familiar with you over time as you attend to him. There`s a sensor that`s in your range; it`s a smart sensor and an audio and proximity detector. So up to a meter away, it can detect the presence of a person. As much as you attend to that system, Fin Fin will become more familiar with you."

Fin Fin has at least a couple more releases to come, adds Pontecorvo. Plans are to add to Teo, the planet on which Fin Fin lives, and to include more creatures. He also notes that they are working on a completely new game, which isn`t slated for introduction for at least two years. (Although Pontecorvo did not specify that this game would use adaptive technology, he did say that Fujitsu is working on such an implementation.)

NPCs With Character

Creating commercial titles in which the star players are based on AI is a recent development in the gaming world. A more traditional use for AI is to give non-playing characters (NPCs) intelligence and, on some level, personalities. However here, too, developers are beginning to push AI in new directions, employing more sophisticated AI than ever before and getting some fascinating results.

Dungeon Keeper from Bullfrog (Austin, TX) is an excellent example. In this game, the player takes the role of a bad guy, says Peter Molyneux, designer, producer, and programmer. (Note: Molyneux recently left Bullfrog to start his own company, which was unnamed at press time.) During the game, players build an elaborate, complex dungeon and fill it with monsters.

In Dungeon Keeper, AI is used to give the monsters character. Molyneux says it`s difficult to categorize the type of AI employed. "What we`ve had to do is take the traditional science methods, such as rules-based and neural nets, and throw away about 50% of that because it`s too processor-intensive and memory hungry to use. [The game can] have upwards of 300 creatures all processing intelligence at once. I`ve been working on AI routines since 1988, since my first game, called Populous. And all the games I`ve done since then have been hybrids and improvements to that AI. So it`s based on that."

Each monster has its own intelligence as well as individual personality. There are about 15 features designed into each monster, says Molyneux, including sight, memory, curiosity, fear, anger, sadness, happiness, and sociability. "The interaction of those variables then creates the individual personalities," says Molyneux.

Molyneux uses one of the monsters, a giant fly, as an example. "Flies are very curious," he says. "They will go out and explore, and they`ll remember where they`ve explored and communicate that to the other creatures and formulate maps of their environment. But they`re also very cowardly. So if there`s an enemy creature nearby or a corridor that`s completely dark, they will literally run away--and they will communicate that fear and caution to the other creatures and modify how they think of that area of the dungeon."

As the "boss," the player can affect the monsters` behavior by scolding and praising them. "For example, you can slap these creatures, and that causes them pain, and they remember those slaps," says Molyneux.

The monsters also will interact with each other, regardless of whether the player is present or not. Says Molyneux, "You can turn on the game and have a dungeon full of monsters, go out, and come back after an hour, and find they`ve explored areas and built things. They will get angry with you, too, if you don`t pay them or give them enough food. They might put graffiti on the wall or destroy the dungeon. If one of their buddies dies, they`ll get sad. There are a fairly broad range of emotions."

This May, Psygnosis (Foster City, CA) introduced Sentient, another title that uses AI to make its NPCs seem more human-like. Sentient is a sci-fi adventure in which the player is a medical technician sent to a space station to investigate an outbreak of radiation sickness.

On the space station are 60 NPCs, notes Julian Hicks, producer. "They all have objectives and missions and things they want to do. You try to persuade them to help you do what you want to do. But the NPCs tend to carry on about their own personal goals. And whether you`re there or not, they will have conversations and pass on bits of gossip and knowledge to each other."

.

Within the NPCs, there are a range of factions, notes Hicks, citing engineers and scientists as examples. Then there are groups with similar moral allegiances. ("The engineer who believes in what the scientists stand for," notes Hicks.) And then each NPC has a personal style and attributes.

The AI used to create these characters is based on a combination of techniques, says Hicks. Some is decision-based, he says, but it`s not that simple. "The decisions are derived based on stats, but we`ve also introduced a random element. Again, we`re trying to capture that human essence.

"The fact is, in most computer role-playing games, the NPCs don`t give any semblance of intelligence," he continues. "Say you`re in a bar and you throw your beer at the bartender one day. The next day you go back, and he`s just as happy to see you. That shouldn`t happen."

It won`t in Sentient. Each NPC has its own memory in terms of where it has been and what it has learned, says Craig Grounsell, one of the programmers. So if the player isn`t nice to a character, that character will remember that and treat the player with equal disdain. In fact, the NPC may pass on ill feelings about the player to the other members of its division, who also will then treat the player poorly. Additionally, NPCs have "feelings" about each other, which will determine whether or not they pass information to various characters.
The ultimate, adds Hicks, will be when a person is playing an online game and comes upon a character and can`t figure out whether it`s a real person or an NPC.

Dinos That Think

At DreamWorks Interactive (Los Angeles, CA), inspiring fear in the player is the ultimate goal. These folks are currently developing a game called Trespasser: Jurassic Park based on The Lost World: Jurassic Park movie and are using AI for the dinosaur characters. "We want you to look at the animal and feel like it`s thinking," says Seamus Blackley, producer. (Note: this title isn`t expected to ship until sometime 4th QTR. For more details on the CG in Trespasser, see the June CGW feature, "Interacting with Dinosaurs.")

Like most of the others, Andrew Grant, the title`s AI programmer, has used a "mishmash" of AI techniques to achieve this goal. "The way we work it is, every animal has its own little brain, which consists of a bunch of sub-brains that are each in charge of one important activity--eating, drinking, showing excitement, running away. These sub-brains are competing for attention in the way the animal expresses itself. So depending on how the dinosaur is feeling and its environment, different [sub-brains] will be dominant. For example, if there`s a T-Rex and it sees a goat, then its eating brain will come to the fore. If it`s hungry, then that will take over and the T-Rex will try to eat the goat. It`s a complicated system of balances, sort of a fuzzy logic sort of system."

Grant goes on to explain that if a Raptor sees a goat but also a T-Rex, its eating brain may want the goat, but its run-away brain is probably scared of the T-Rex and will cause the Raptor to scram. "But let`s say the Raptor hasn`t eaten in days and is really hungry. And it hasn`t had a bad experience with the T-Rex lately. If the T-Rex is farther away than the goat is, the Raptor might go for it," says Grant. "That`s really hard to do. I`d rather have it err on the side of running away usually, but what I`m shooting for--and what`s the hard part--is getting to the case when in the right circumstances, it will go after the goat anyway."

Sound amazing? As game developers see it, they`re only beginning to explore what`s possible with AI. In the next few years, 3D accelerators will take over not only the rendering process but also the geometry pipeline, freeing up more processing cycles without even counting the increased power of the microprocessor itself.

"It is inevitable that AI will get a lot more popular," says Molyneux, creator of Dungeon Keeper. "What we`ve done is to create these graphically rich worlds, and we`ve done quite an amazing job of it. And that`s all well and good, creating beautiful graphics. But unless you get some good AI routines that fill the game with reasonably intelligent characters, you`ll have a rather hollow experience."

Woodcock, AI consultant, agrees. "The last few years we`ve been working on the big 3D problem. Well, we pretty much have that solved now--basically--we know how to do it and do it well. So now the question is, what do you do to make your game different from everybody else`s?"

There are 60 non-playing characters (NPCs) in Sentient from Psygnosis. And thanks to the AI in this game, these NPCs tend to carry on about their business, with or without player interaction.

A Taxonomy of Computer Games

by Chris Crawford

Thousands of computer games are commercially available on a variety of hardware configurations. These games present a bewildering array of properties. Many show close similarities. Most possess some unique design feature. Given this large sample of games, we can learn a great deal about game design by establishing a taxonomy of computer games. A taxonomy would illuminate the common factors that link families of games, while revealing critical differences between families and between members of families. A well-constructed taxonomy will often suggest previously unexplored areas of game design. Most important, a taxonomy reveals underlying principles of game design. In another field of study, Charles Darwin's meticulous taxonometric work while on the Beagle led almost inevitably to his development of the theory of evolution. While we cannot hope that taxonometric work in computer game studies will be so spectacularly productive, it certainly seems worth the effort.

I will insist on an important qualification: I do not claim that the taxonomy I propose is the correct one, nor will I accept the claim that any correct taxonomy can be formulated. A taxonomy is only a way of organizing a large number of related objects. If there were some organizing agent, some underlying process that created the group of objects, then we could reasonably expect to be able to find a single correct taxonomy embodying the central organizing principle in its structure. For example, the wide array of living creatures on this earth did not arise by chance; this array is the product of natural selection. Natural selection is a reasonable, understandable, nonarbitrary process. Therefore, there is only one reasonable taxonomy for life on earth, the taxonomy that embodies the principles of natural selection. In the shape of an airplane we can see the principles of aerodynamics; so too in a taxonomy of living creatures can we see the hand of natural selection.

Such is not the case with computer games. The field is too young, the sample too small, for whatever organizing principles there may be to have asserted themselves. The games we now have are more the product of happenstance than the inevitable result of well-established forces. Without a wide array of games there is little opportunity to choose between games; without choice there can be no natural selection. It is therefore impossible for us to devise a single, absolute taxonomy. Many taxonomies are admissible. Indeed, attempting to construct several alternative taxonomies is a useful way to examine the common traits of computer games. I am not so ambitious; I shall be happy to propose just one taxonomy. I divide computer games into two broad categories: skill-and-action ("S&A") games (emphasizing perceptual and motor skills) and strategy games (emphasizing cognitive effort). Each major category has several subcategories.

SKILL-AND-ACTION GAMES

This is easily the largest and most popular class of computer games. Indeed, most people associate all computer games with skill-and-action games. All arcade games are S&A games and almost all games for the ATARI 2600 are S&A games. This class of games is characterized by real-time play, heavy emphasis on graphics and sound, and use of joysticks or paddles rather than a keyboard. The primary skills demanded of the player are hand-eye coordination and fast reaction time.

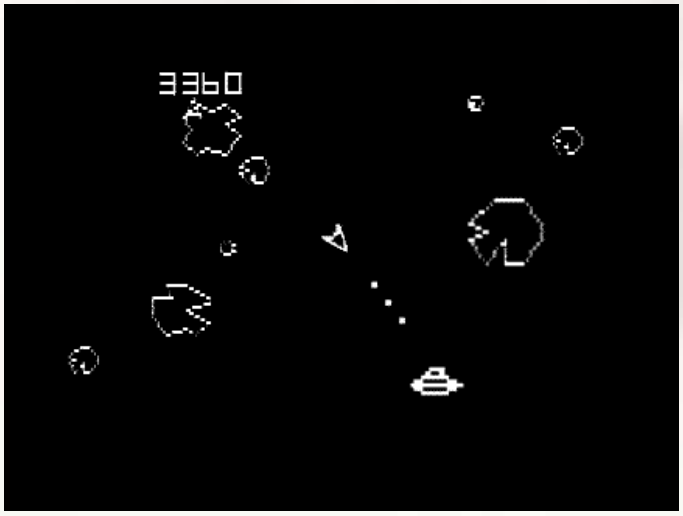
I group skill-and-action games into six categories: combat games, maze games, sports games, paddle games, race games, and miscellaneous games.

Combat Games

Combat games all present a direct, violent confrontation. The human player must shoot and destroy the bad guys controlled by the computer. The challenge is to position oneself properly to avoid being hit by the enemy while shooting him. These games are immensely popular; they are Atari's forte. There are many variations on this theme, most arising from variations on the geometry of the situation or the weaponry of the opponents.

STAR RAIDERS and SPACEWAR can be compared on these bases of geometry and weaponry. In both games the player files through space in a rocket ship and engages enemy spaceships in real-time cosmic dogfights. STAR RAIDERS presents the conflict in first-person geometry (that is, the television screen shows the same scene that the pilot would see.) SPACEWAR uses much the same weaponry and mechanisms with one crucial difference: the geometry of the game is third-person rather than first-person (that is, the player sees his own and his opponent's spaceships from a distance.) The difference in result is obvious to anyone who has played both games. The first-person game is more exciting and compelling than the third-person game. Unfortunately, the first-person geometry is so technically difficult to execute that it has been implemented on only a few games. Most games use third-person geometry.

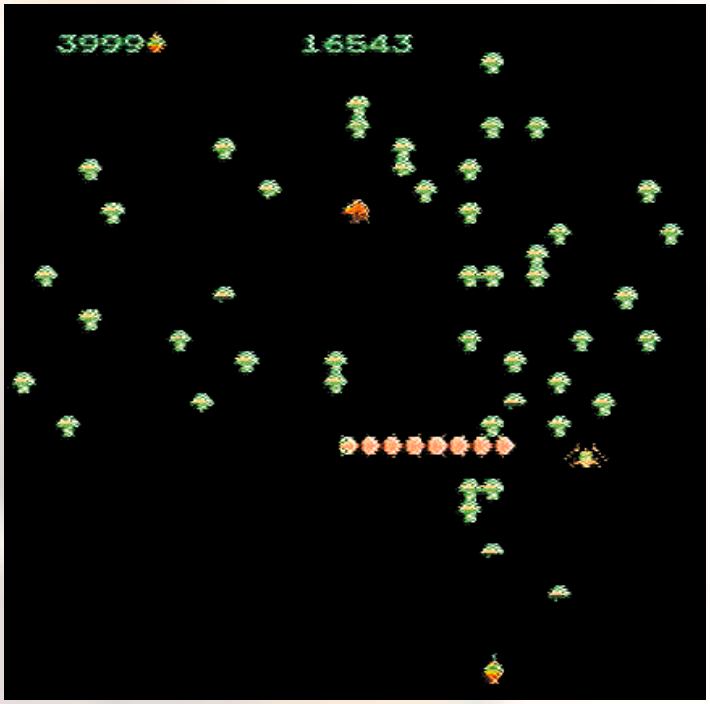
ASTEROIDS is a shoot-em-up game that uses the same space environ that STAR RAIDERS uses. The primary difference between the two games is in the nature of the opposition. The enemy in ASTEROIDS is not a small number of intelligent opponents armed with weapons identical to the player's; instead, the enemy is a large number of stupid rocks armed only with their ability to destructively collide with the player.



MISSILE COMMAND is another combat game with several interesting twists. First, the player must defend not only himself but also his cities from descending nuclear bombs. Second, the game is a purely defensive game in that the player never has the opportunity to attack his enemy. Third, while shots in other games are very rapid events, the shooting process in this game is slower and takes time to develop because the missiles must fly to their targets before detonating. Because the time between firing and impact is so long, the player must plan his shots with greater foresight and make use of multiple explosions. Thus, although this is a skill-and-action game, there are more strategic elements involved than in many games of this category.

SPACE INVADERS (trademark of Taito America Corp.) is one of the most successful combat games of all time. It was one of the first smash hit games and contributed to the upsurge of popularity of computer games that began in 1979. While STAR RAIDERS and ASTEROIDS give the player great mobility and MISSILE COMMAND gives him none, SPACE INVADERS gives the player limited mobility in one dimension only. As in ASTEROIDS, the player must face a multitude of rather stupid opponents who can win by touching the player (landing); in addition, as in STAR RAIDERS, the monsters also shoot back. The monsters march back and forth across the screen, slowly descending onto the player. As the player kills more and more monsters, they march faster and faster. This gives the game a hypnotic accelerating tempo. SPACE INVADERS is definitely a classic.

The success of SPACE INVADERS has spawned a whole series of copies and derivatives. There are a large number of copies whose only goal was to cash in on the success of the original game. There are also several genuine derivative games. For example, GALAXIAN (trademark of Midway) is a simple variation on SPACE INVADERS. Individual invaders peel off and attack the player with more ferocity than the docile monsters of the original game. CENTIPEDE; is also a derivative of SPACE INVADERS; it is different enough to be a new design, but the internal game structure is very similar to the original. The invaders have been grouped into a segmented centipede; their side-to-side motion is bounded not by the edges of the screen



There are many, many other combat games. BATTLEZONE and RED BARON are two first-person combat games utilizing vector displays. Other combat games include CAVERNS OF MARS, YAR'S REVENGE, CROSSFIRE (trademark of On-Line Systems) and DEFENDER (trademark of Williams).
You may wonder why so many combat games are set in outer space. There are three reasons. First, space is easy to depict and animate with a computer---all the designer need do is draw a blank screen with a few white dots for stars. Second, space is not encumbered by the expectations of the players. A designer encountering problems can always concoct some super-duper zapper to solve any design problems with the game and nobody can object that it is unrealistic. Earthbound games constrain the designer to look reality squarely in the eye---such a tiresome burden for a "creative" mind. Third, space is an intrinsically fantasy-laden environment that encourages suspension of disbelief because it is unfamiliar to its audience. Combat games have always been at the heart of computer gaming. Players never seem to tire of them; it appears that they will be around for a long time to come.

Maze Games

The second subgrouping of S&A games is the set of maze games. PAC-MAN (trademark of Namco) is the most successful of these, although maze games predate PAC-MAN. The defining characteristic of the maze games is the maze of paths through which the player must move. Sometimes one or more bad guys pursue the player through the maze. Some maze games (MAZE CRAZE for the ATARI 2600 is a good example) specifically require that the player make his way to an exit. Other maze games require that the player move through each part of the maze. DODGE 'EM is an early example of such a game. In either case, the number, speed, and intelligence of the pursuers then determines the pace and difficulty of the game. PAC-MAN has a very carefully balanced combination of these factors. The pursuers are just slightly but by mushrooms randomly scattered across the screen. Numerous embellishments (spiders, fleas, and

scorpions) extend the game considerably. TEMPEST is a three-dimensional first-person derivative of SPACE INVADERS using vector graphics. The amount of design attention that SPACE INVADERS has attracted is a tribute to the game's originality, appeal, and durability. slower than the human player; their intelligence and number make up for this. The overall pace of the game makes it difficult for the player to fully analyze the positions of the five pieces in real time.

Any successful game is certain to attract copies, variations, and derivatives, and PAC-MAN is no exception. One of the first such games for the ATARI Home Computer System was the first edition of JAWBREAKERS (trademark of On-Line Systems). This game, now removed from the market, clearly demonstrates the difference between structural changes and cosmetic changes. Structurally, it is indistinguishable from PAC-MAN. The play of the game is almost identical to that of PAC-MAN. Cosmetically, there are a number of differences: the pursuers are faces rather than ghosts; the player is a set of teeth rather than a head with mouth; the maze is laid out differently; the sounds are different. This game provides a good example of the methods that can be used to copy games while attempting to minimize legal problems.

Another PAC-MAN derivative is MOUSKATTACK (trademark of On-Line Systems). This game shows some structural changes relative to PAC-MAN. The player is again pursued through a maze by four computer-controlled creatures, but the basic scenario contains a number of embellishments. First, merely passing through every point in the maze is not enough; some points, randomly chosen by the computer, must be passed through twice. Second, the player is allowed to fight back against the pursuers in a very different way (setting mousetraps). The strategic and tactical effects of this counterforce capability yield a game that plays rather differently. Finally, there is a very interesting two-player game that allows both cooperative and competitive strategies. In MOUSKATTACK we see the basic structure of PAC-MAN with a number of embellishments and extensions that produce a distinct game.

The appeal of maze games can be attributed to the cleanliness with which they encapsulate the branching structure that is a fundamental aspect of all games. The reader will remember from Chapter One that a game has a tree structure with each branch point representing a decision made by the player. In a maze game, each branch point is neatly depicted by an intersection in the maze, and the options available to the player are visually presented as the paths available at the intersection. Thus, a maze game presents a clear visual representation of the branching structure of the game.

Even more fascinating is the looping structure possible with maze games. A player can return to an intersection in the maze many times. Yet, each time he does so, the options he has take different meanings because the other maze-inhabitants have moved in the interim to a different pattern of positions. In this way, a small number of displayed intersections can represent a huge number of branch-points in the game-tree. The analogy with a computer program, in which a small number of program instructions, through looping and branching, can address a large number of specific cases, is striking.

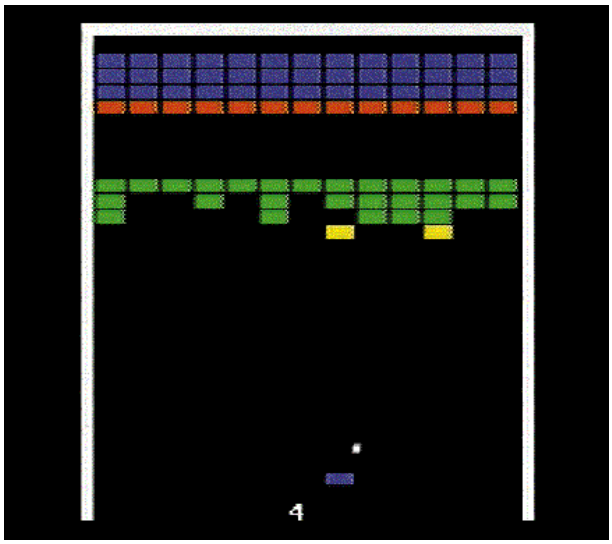
Sports Games

These games model popular sports games. They are anachronisms derived from the early days of computer game design when computer games had no identity of their own. People without original ideas for games fell back on the sports games as models around which to design. This also served a useful marketing purpose: why would a conservative consumer buy a game with a title and subject completely alien to his experience? Better to offer him a game he is already familiar with. Thus we have games based on basketball, football, baseball, soccer, tennis, boxing, and others. All of these games take liberties with their subject matter to achieve playability. The most enjoyable aspects of the computer game have very little to do with the real game. This is fortunate, for a slavish attempt at replication would have produced a poor computer game. Only by substantially altering the original games were the authors able to produce a decent design. Even so, sports games remain the wallflowers of computer gaming. I suspect that sports games will not attract a great deal of design attention in the future. Now that computer games have an accepted identity of their own, the need for recognizable game titles has diminished.

Paddle Games

I use the title "Paddle Games" to cover the PONG-based games. PONG is certainly one of the most successful and fertile of game designs, for it has many grandchildren and great-grandchildren. The central element of the game, that of intercepting a projectile with a paddle-controlled piece, has been used in endless variations. The original PONG pitted two players in an electronic version of ping-pong, hence the name. BREAKOUT was a solitaire version that required the player to chip away at a wall with the ball. The player received points for each brick destroyed. SUPERBREAKOUT introduced variations on this theme with moving walls, extra balls, and other tricks. CIRCUS ATARI introduced parabolic trajectories for the projectiles and a complex moving wall of balloons. WARLORDS; took the genre even further; up to four players (one in each corner) defend brick castles against a projectile bounced around the field by their shield-paddles.

In these games, the player uses the ball as a weapon to batter; in other paddle games the player must only catch the ball, or many balls, rather than deflect it. AVALANCHE is one such game. In this game, the player is at the bottom of the screen and large numbers of rocks are failing; each one must be caught with the player's piece. The game becomes quite frantic as more and more rocks fall at a faster and faster pace. Another game, CHICKEN, (trademark of Synapse Software) expands on this theme by replacing the rocks with eggs and making each one hatch on striking the ground, forcing the player-hen to jump over it as she moves about.



The paddle game-system is a very simple one; although I doubt that it has much development potential remaining, I am hesitant to pronounce such a durable old system dead.

Race Games

Some computer games involve a straightforward race. Most of these games allow the player to move at constant speed, but extract time penalties for failure to skillfully negotiate an assortment of hazards. Thus, a player in the APX skiing game DOWNHILL must avoid the trees and rocks; the player's score is based on his time to complete the course. MATCH RACER by Gebelli Software is a car-racing game with oil slicks and obstacles. NIGHT DRIVER is a car-racing game featuring a first-person view of the road. One problem with all of these games is that they are not true games but puzzles, for there is no real interaction in a race between a player and his opponent. Indeed, it is difficult to identify the opponent in these games.

A more involved variation on the race game is DOG DAZE by Grey Chang. This is a true game, not a puzzle. It presents a two-player competitive race game with variable goals and asymmetric obstacles. Each player has a dog; hydrants pop onto the screen at random locations; the players must race to be the first to touch the hydrant, thereby claiming it as their own. Players may not touch hydrants owned by their opponents on pain of being temporarily paralyzed. The game has many interesting twists and turns without being overly complex; it demonstrates that the race game can be a flexible vehicle of game design.

Miscellaneous Games

My taxonomy is flawed; there exist a number of games that do not fit into this taxonomy very well. The first I will mention is DONKEY KONG, (trademark of Nintendo) a game that looks vaguely like a race game with intelligent obstacles. FROGGER (trademark of _____) is another game that defies classification in this taxonomy. It could perhaps be called a maze game with moving walls or obstacles, but the fit is poor. APPLE PANIC by Broderbund Software also defies my taxonomy. In some ways it is like a maze game and in some ways it is a combat game. The pace of the game is oddly slow. I don't know what to call this game. The fact that these games do not fit my taxonomy does not bother me overly much; I certainly don't want to create ad hoc categories for individual games. I am content to wait and see other developments before I create new categories or revise old ones.

STRATEGY GAMES

Strategy games comprise the second broad class of computer games. These games emphasize cogitation rather than manipulation. I do not mean to imply that S&A games are devoid of strategic content; some S&A games do indeed have a strategic element. The major distinguishing factor between strategy games and S&A games is the emphasis on motor skills. All skill-and-action games require some motor skills; strategy games do not. Indeed, real-time play is rare in strategy games (this is changing; LEGIONNAIRE from Avalon-HIII is a notable real-time strategy game). Strategy games typically require more time to play than S&A games. Strategy games are nonexistent in the arcades; they are rare on the ATARI 2600; they are almost exclusively restricted to personal computers. I divide strategy games into six categories: Adventures, D&D games, wargames, games of chance, educational games, and interpersonal games.

Adventures

These games derive from one of the oldest computer games, called "Adventure". In these games the adventurer must move through a complex world, accumulating tools and booty adequate for overcoming each obstacle, until finally the adventurer reaches the treasure or goal. Scott Adams created the first set of Adventures widely available for personal computers; his software house (Adventure International) is built on those games. The Scott Adams games are pure text adventures that run in a small amount of memory, so they do not need disk drives; they are also readily transportable to different machines. A short time later Ken and Roberta Williams built On-Line Systems with THE WIZARD AND THE PRINCESS (trademark of On-Line Systems), an adventure that presented pictures of the scenes in which the adventurer found himself. The game itself was not particularly new; the innovation was primarily the use of graphics. Both firms have expanded their lines with more games using the systems they pioneered. Most of these derivative games are structurally similar to the originals, differing in detail, polish, and size.

The next variation on the adventure theme was the giant adventure, of which there are several. TIME ZONE by On-Line Systems is one of these. These giant adventures use multiple diskettes to link together a gigantic adventure. As the player solves the puzzle in one environment he moves on to another environment on another disk. The games are structurally identical to earlier games; the only difference is one of magnitude. They take many weeks of play to solve.

A new variation on the adventure game genre is DEADLINE (trademark of Infocom), a detective adventure with a number of interesting twists. Its heritage as an adventure is evident in its lack of graphics and its use of an excellent sentence parser. This adventure puts the player in the role of a detective attempting to solve a murder. The game is played in a real-time mode that adds to the interest and challenge of the game. The player searches not for treasure but for information with which to solve the murder. This game shows the potential of the adventure system in that the same system can be used, with the storyline and goals altered, to appeal to a different audience.

One of the most clever adventures ever done is Warren Robinett's ADVENTURE on the ATARI 2600. This adventure follows the same basic format as all adventures, except that it uses absolutely no text. Instead, the user moves through a series of rooms presented in rather simple graphics. Although the graphics and input schemes are radically different, the basic feel of the adventure system has been successfully retained. SUPERMAN, HAUNTED HOUSE, and GALAHAD AND THE HOLY GRAIL by Doug Crockford are all derivatives of this game.

Adventures are closer to puzzles than to games. As discussed in Chapter One, puzzles are distinguished from games by the static nature of the obstacles they present to the player. Adventures present intricate obstacles that, once cracked, no longer provide challenge to the player. It is true that some adventures push closer to being games by incorporating obstacles such as hungry dragons that in some way react to the player. Nevertheless, they remain primarily puzzles.

D&D Games

A completely independent thread of development comes from the D&D style games. Fantasy role-playing was created by Gary Gygax with Dungeons and Dragons (trademark of TSR Hobbles), a complex noncomputer game of exploration, cooperation, and conflict set in a fairytale world of castles, dragons, sorcerers, and dwarves. in D&D, a group of players under the guidance of a "dungeonmaster" sets out to gather treasure. The game is played with a minimum of hardware; players gather around a table and use little more than a pad of paper. The dungeonmaster applies the rules of the game structure and referees the game. The dungeonmaster has authority to adjudicate all events; this allows very complex systems to be created without the frustrations of complex rules. The atmosphere is quite loose and informal. For these reasons, D&D has become a popular game, with endless variations and derivatives.

D&D first appeared in the mid-70's; it didn't take long for people to realize that it had two serious limitations. First, the game needed a group of players and a dungeonmaster, so it was impossible to play the game solitaire. Second, the game could sometimes become tedious when it required lengthy computations and throwing of dice. Many people recognized that these problems could be solved with a microcomputer. The first company to make a D&D style computer game available was Automated Simulations. Their TEMPLE OF APSHAI program has been very successful. They also market a number of other D&D-style games.

So far, however, few games have been marketed that truly capture the spirit of D&D. There are several reasons for this. First, most D&D-players are young and don't have the money for such packages. Second, the adventure games have slowly absorbed many of the ideas of the D&D games. There was a time when we could easily distinguish an adventure from a D&D game with several factors.Adventures were pure text games, while D&D games used some graphics. Adventures were puzzles; D&D games were true games. Adventures were by and large nonviolent, while D&D games tended to be quite violent. Lately, we have seen adventures taking on many of the traits of D&D games, so that it is now harder to tell the difference between them.

An ideal example of this phenomenon is ALI BABA AND THE FORTY THIEVES (trademark of Quality Software), a game with the basic elements of both adventures and D&D games. The player must search through a large maze to find and rescue a princess, but on the way he must fight monsters and thieves. The player, as Ali Baba, possesses personal characteristics (dexterity, speed, etc.) that are reminiscent of a D&D game, but he must explore the maze as in an adventure. For these reasons, I feel that this game cannot be classified as either an adventure or a D&D game, but rather is a solid example of the merging of these two genres into a new class of games, the fantasy role-playing ("FRP") games. This suggests that we will see more such games combining the "search and discover" aspects of adventure games with the "defeat opponents" aspects of D&D games.

Wargames

A third class of strategy games is provided by the wargames. Noncomputer wargames as a gaming form have a long heritage. Commercial wargaming goes all the way back to the 1880's with an American wargame design using wooden blocks. The British have long had a dedicated group of wargamers using miniature models of soldiers and very complex rules. Their games, called miniatures games, have grown in popularity and are now played in the USA. But the largest segment of wargamers in recent years has been the boardgamers. This hobby was founded in the late 1950's by Charles Roberts, who founded the Avalon-Hill Game Company and created such classic games of the 60's as BLITZKRIEG, WATERLOO, and AFRIKA KORPS (all trademarks of the Avalon-Hill Game Company). During the 1970's a new company, Simulations Publications, Inc., turned board wargaming into the largest segment of wargaming.

Wargames are easily the most complex and demanding of all games available to the public. Their rules books read like contracts for corporate mergers and their playing times often exceed three hours. Wargames have therefore proven to be very difficult to implement on the computer; we have, nevertheless, seen entries.

The computer wargames available now fall into two distinct groups. The first group is composed of direct conversions of conventional boardgames. COMPUTER BISMARK, COMPUTER AMBUSH, and COMPUTER NAPOLEONICS (trademarks of Strategic Simulations, Inc.) are examples of this group of games. These games illustrate the folly of direct conversion of games of one form to another. They parrot successful and respected boardgames, but are themselves not as successful. Because they attempt to replicate boardgames, they are, like boardgames, slow and clumsy to play.

The second group of computer wargames are less slavish in their copying of board wargames. My own EASTERN FRONT 1941 is generally considered to be the best of this lot, primarily because of its graphics and human engineering features. Many of the games in this category are experimental; hence the successes are outnumbered by the failures. Avalon-Hill's first entries into the computer wargaming arena were such experiments. My own TANKTICS game is an early experiment that once was the most advanced commercially available wargame (it was the ONLY commercially available wargame when I first released it in 1978). It is now generally regarded as a mediocre game. It can safely be said that computer wargaming is not a well-developed area of computer gaming. For the moment, computer wargaming is too closely associated with board wargaming in the minds of the public and most designers; until it can shake free from the constraints of boardgames and, establish its own identity, computer wargaming will evolve slowly.

Games of Chance

Games of chance have been played for thousands of years; their implementation onto computers is therefore quite expectable. They are quite easy to program, so we have seen many versions of craps, blackjack, and other such games. Despite their wide availability, these games have not proven very popular, most likely because they do not take advantage of the computer's strong points. Furthermore, they lose the advantages of their original technologies. These games demonstrate the folly of mindlessly transporting games from one medium to another.

Educational and Children's Games

The fifth category of strategy games is that of the educational games. Although all games are in some way educational, the games in this set are designed with explicit educational goals in mind. This group is not heavily populated as yet, perhaps because the people interested in educational uses of computers have not yet concentrated much attention on game design. The Thorne-EMI puzzles are good entries in this field, and APX sells a collection of very simple children's games that have some educational value. Several of the classic computer games are educational: HANGMAN, HAMMURABI, and LUNAR LANDER are the three most noteworthy of these early educational games. SCRAM (a nuclear power plant simulation) and ENERGY CZAR (an energy economics simulation) are two of the more complex programs in the educational games field. My favorite entry to date is ROCKY'S BOOTS (trademark of The Learning Company), a children's game about Boolean logic and digital circuits. The child assembles logic gates to create simulated logical machines. This game demonstrates the vast educational potential of computer games. Educators are becoming more aware of the motivational power of computer games; with time we can expect to see more entries of the caliber of ROCKY'S BOOTS.

Interpersonal Games

I have been exploring a class of games that focus on the relationships between individuals or groups. One such game explores gossip groups. The player exchanges gossip with up to seven other computer-controlled players. The topic of conversation is always feelings, positive or negative, expressed by one person for another. Adroit posturing increases popularity. Similar games could address corporate politics, soap-opera situations, gothic romances, international diplomacy, and espionage. Although the category is undeveloped, I believe it is important because it addresses fantasies that are very important to people. Many other art forms devote a great deal of attention to interpersonal relationships. It is only a matter of time before computer games follow a similar course.

CONCLUSIONS

This concludes the description of my proposed taxonomy. Obviously, this taxonomy has many flaws. This is primarily because the basis of division is not any grand principle but is instead historical happenstance. There is no fundamental reason why wargames should be treated any differently than D&D games. Yet, both game systems evolved separately and are historically quite distinct. Similarly, the creation of an educational games category is my response to the efforts of educators to create educational games. With the passage of time, market forces will assert themselves, and a more organized and consistent taxonomy will become possible. People have tried to create educational games, so we now have them. My taxonomy is a patchwork because the set of available computer games is a patchwork.

This taxonomy suggests a number of observations about the state of game design with computers. For example, it should be obvious that there are very few basic scenarios for skill-and-action games, each scenario taking one category. The archetypical game in each category spawned a whole family of imitators, variations, and improvements. Moreover, the archetypical game in each category was seldom the big moneymaker; instead, the archetypical game was followed by several successor games that improved on it until one game hit the nail on the head. Thus we have COMBAT leading to SPACE INVADERS in the combat category, DODGE 'EM leading to PAC-MAN in the maze category, and PONG leading to SUPERBREAKOUT in the paddle category.

Another lesson that arises from this taxonomy is that the Analogy games are still in a very poorly-developed state in comparison to the S&A games. While S&A games have fairly clear-cut categories that make sense, the categories in strategy games are less satisfying and the distinctions between categories are muddier. This ambiguity suggests that much creative opportunity remains in the strategy games field.

A taxonomy reflects the body of material it attempts to organize. The state of computer game design is changing quickly. We would therefore expect the taxonomy presented here to become obsolete or inadequate in a short time. New taxonomies must be created to reflect the changes in the marketplace in the next few years. For the present, however, the proposed taxonomy can provide us with an organized way to view the menagerie of games while suggesting new areas to explore.

Game Design Sequence

by Chris Crawford

Game design is primarily an artistic process, but it is also a technical process. The game designer pursues grand artistic goals even as she grinds through mountains of code. During the process of developing the game, she inhabits two very different worlds, the artistic world and the technical world. How does one manage the integration of such dissimilar worlds? In short, how does one go about the process of designing a computer game? In previous chapters I have touched on some of the questions related to this process; I have also laid down a few precepts. In this chapter I will suggest a procedure by which a computer game could be designed and programmed.

The procedure I will describe is based on my own experiences with game design, and reflects many of the practices that I use in designing a game. However, I have never used this procedure in a step-by-step fashion, nor do I recommend that any person follow this procedure exactly. In the first place, game design is far too complex an activity to be reducible to a formal procedure. Furthermore, the game designer's personality should dictate the working habits she uses. Even more important, the whole concept of formal reliance on procedures is inimical to the creative imperative of game design. Finally, my experience in game design is primarily with personal computers, so my suggestions are not completely applicable to arcade game designers or home video game designers. I therefore present this procedure not as a normative formula but as a set of suggested habits that the prospective game designer might wish to assimilate into her existing work pattern. With these important qualifications in mind, let us proceed.

CHOOSE A GOAL AND A TOPIC

This vitally important step seems obvious, yet is ignored time and time again by game designers who set out with no clear intent. In my conversations with game designers, I have many times discerned an indifference to the need for clear design goals. Game designers will admit under close examination that they sought to produce a "fun" game, or an "exciting" game, but that is more often than not the extent of their thinking on goals.

A game must have a clearly defined goal. This goal must be expressed in terms of the effect that it will have on the player. It is not enough to declare that a game will be enjoyable, fun, exciting, or good; the goal must establish the fantasies that the game will support and the types of emotions it will engender in its audience. Since many games are in some way educational, the goal should in such cases establish what the player will learn. It is entirely appropriate for the game designer to ask how the game will edify its audience.

The importance of a goal does not become obvious until later in the game design cycle. The crucial problems in game development with microcomputers are always problems of trade-offs. Everything that the game designer wants to do with her game costs memory, and memory is always in short supply with microcomputers. Thus, the designer must make trade-offs. Some game features can be included, and some must be rejected. At two o'clock in the morning, when you must face the awful decision of rejecting one of two highly desirable features, the only criterion you will have for making this painful choice will be the goal you have established for the game. If your goals are clear, your decision will be painful but obvious; if your goals are murky, you may well make the wrong choice, and whatever you choose, you will never know if your decision was correct.

How do you select a proper goal? There is no objective answer to this question; the selection of a goal is the most undeniably subjective process in the art of computer game design. This is your opportunity to express yourself; choose a goal in which you believe, a goal that expresses your sense of aesthetic, your world view. Honesty is an essential in this enterprise; if you select a goal to satisfy your audience but not your own taste, you will surely produce an anemic game. It matters not what your goal is, so long as it is congruent with your own interests, beliefs, and passions. If you are true to yourself in selecting your goal, your game can be executed with an intensity that others will find compelling, whatever the nature of the game. If you are false to yourself, your game will necessarily be second-hand, me-too.

There are situations in which it is not quite possible to attain the purity of this artistic ideal. For example, I would not claim that only immature, childish people should design games for children. Nor would I suggest that good shoot-'em-up games can only be done by shoot-'em-up personalities. The realities of the marketplace demand that such games be written, and it is better that they be written by mature professionals than by simpering fools. Such emotionally indirect games, however, will never have the psychological impact, the artistic power, of games coming straight from the heart. Once you have settled on your goal, you must select a topic. The topic is the means of expressing the goal, the environment in which the game will be played. It is the concrete collection of conditions and events through which the abstract goal will be communicated. For example, the goal of STAR RAIDERS apparently concerns the violent resolution of anger through skillful planning and dexterity. The topic is combat in space. The goal of EASTERN FRONT 1941 concerns the nature of modern war, and especially the difference between firepower and effectiveness. The topic is the war between Russia and Germany.

Most game designers start off by selecting their topic, with their goals subordinated to their topic. Indeed, they commonly describe a game under development by its topic rather than its goal. When I tell other designers that I am working on a game about leadership, I am met with quizzical expressions. Is it a space game, or a wargame, or a dungeon game, they wonder; they seem satisfied when I tell them it's a game about King Arthur. It is a serious mistake to subordinate the goal to the topic. Although your initial flash of inspiration may focus more on the topic than the goal, you must have the determination to take control of the design and impose your own goals onto the topic rather than allowing yourself to be swept away by the momentum of the topic.

Selecting a good topic can be time-consuming, for each potential topic must be carefully examined for its ability to successfully realize the goals of the game. Many topics carry with them some excess emotional baggage that may interfere with the goals of the game. For example, my most recent game design effort uses the Arthurian legends as its topic. My goal in the game is to examine the nature of leadership. I found the Arthurian legends to be a compelling vehicle for this goal. Unfortunately these legends contain a strong component of male braggadocio, the vanquishing of opponents by brute force. This theme directly contradicts some of the points I want to make with the game, thus weakening the utility of this topic for my ends. I find the legends so powerful and so malleable that I am willing to accept and work around this potential pitfall.

RESEARCH AND PREPARATION

With a goal and topic firmly in mind, the next step is to immerse yourself in the topic. Read everything you can on the topic. Study all previous efforts related to either your goal or your topic. What aspects of these earlier efforts appeal to you? What aspects disappoint or anger you? Make sure that you understand the mechanics of the environment your game will attempt to represent. Your game must give the authentic feel, the texture of the real world, and this can only be achieved if you firmly understand the environment of the game. While researching EXCALIBUR, I studied the history of Britain during the period AD 400-700. I found little in the history books that was harmonious with my goal of depicting the nature of leadership. But in the Arthurian legends I found recurring themes more closely related to my goal. You may well find yourself adjusting your goals as you perform this research function; such erratic decision-making is an embarrassing admission of poorly-defined goals, but reflects an honest willingness to adapt to the exigencies of the topic-environment. It is a departure from the ideal in which I have sinfully indulged myself many times.

During this phase it is critical that you commit little to paper and above all, WRITE NO CODE! Take long walks as you contemplate your game. Cogitate. Meditate. Let the goal, the topic, and the facts gleaned from your research simmer together in the innards of your mind. Weave them together into a whole. Take your time with this phase; impatience now will lead to mistakes that will kill the game. I give myself at least three weeks to develop a game idea in this stage before proceeding to the next step. With EXCALIBUR I expended several months on this stage. During this time I kept my fidgeting hands busy by writing an opening graphic display that had little relevance to the final game.

You will generate during this phase a great variety of specific implementation ideas for your game. They will not all fit together neatly---like any hodgepodge, they will require much sorting and rearranging before they can be used. You should not wed yourself to any of them. A large collection of candidates for implementation is a useful resource during the design phase. A laundry list of implementation ideas that must be included is a liability. Indulge yourself in creating implementation ideas, but be prepared to winnow them ruthlessly during design.

For example, I recently designed a corporate politics game in association with another person. During the research and preparation phase, we came up with a long list of clever ideas that we wanted to into the game. We had agreed that the game would have a feminist point of view without being preachy. We wanted to have a demanding boss, tough projects, deadlines, brownie points, one male chauvinist pig, neutral males, neutral females, family and home obligations, mentors, and the competition for the big promotion. We managed to include almost all of these ideas in the final design. We were not able to integrate the family elements into the game. Every design we created failed to do justice to our desires. In the end, we had to discard this desirable element.

DESIGN PHASE

You now have a clear idea of the game's ideals but you know nothing of its form. You are now ready to begin the concrete design phase. Your primary goal in the design phase is to create the outlines of three interdependent structures: the I/O structure, the game structure, and the program structure. The I/O structure is the system that communicates information between the computer and the player. The game structure is the internal architecture of causal relationships that define the obstacles the player must overcome in the course of the game. The program structure is the organization of mainline code, subroutines, interrupts, and data that make up the entire program. All three structures must be created simultaneously, for they must work in concert. Decisions primarily relating to one structure must be checked for their impacts on the other structures.

I/O Structure

I prefer to start with the I/O structure, for it is the most constraining of the three. I/O is the language of communication between the computer and the player; like any human language, it is the funnel through which we must squeeze the avalanche of thoughts, ideas, and feelings that we seek to share with our fellow human beings. I/O will dictate what can and cannot be done with the gains.

I/O is composed of input and output. Unlike human languages, the two are not symmetric. The computer has two means of output to the human: graphics on the screen and sound. In the future, we may see more exotic devices for output for games, but for the moment these are the two most common. Graphics are the most important of the two, perhaps because we humans are more oriented towards vision than hearing. For this reason, many game designers devote a large portion of their energy towards the design of quality displays. Indeed, some designers go so far as to design the display first and let the game develop from the display, as extreme an example of goal-less design as ever there could be.

Don't make the common mistake of creating cute graphics solely to show off your ability to create cute graphics. Graphics are there for a reason: to communicate. Use graphics to communicate to the user forcefully and with feeling, and for no other reason. Plan functional, meaningful graphics that convey the critical game information while supporting the fantasy of the game. Don't use graphics tricks as a crutch for a bad game design. If the game is dull and boring, no amount of graphics gift-wrapping is going to fix it. The worst examples of this mistake are the games that alternate boring game segments with cute but meaningless graphics displays. Use of sound should follow the same rules: use it to tell the player what's going on in the game. The only place where striking but uninformative graphics and sound can be useful is at the beginning of the game, and then only if they help to establish the mood or tone of the game.

Storyboards are a graphics design tool that tempt many game designers, for they are a well-developed technology from the film industry. They are not appropriate to games, because storyboards are an intrinsically sequential technology. Games are not sequential, they are branching tree structures. The game designer who uses an intrinsically sequential tool risks having her designs made subtly sequential. The tool shapes the mind of its user; the saw suggests that we cut wood, and the freeway suggests that we drive wherever it takes us, not where we choose to go. In like manner does a storyboard impress its sequentiality upon our games.

Devote special care to the input structure of the game. The input structure is the player's tactile contact with the game; people attach deep significance to touch, so touch must be a rewarding experience for them. Have you ever noticed the tremendous importance programmers attach to the feel of a keyboard? Remember that players will do the same thing with your game. A case in point is provided by the games JAWBREAKER and MOUSKATTACK (trademarks of On-Line Systems). In both games the joystick entry routine admits an unfortunate ambiguity when a diagonal move is entered. This gives the player the impression that the joystick is unresponsive. I have seen players slam down the joystick in frustration and swear that they would never play the damn thing again. Remember this well as you plan your input structure: will your input structure frustrate and anger your players?

The input structure lies at the heart of a fundamental dilemma all game designers must face. An excellent game allows the player to interact heavily with his opponent, to invest a great deal of his personality into the game. This requires that the game offer the player a large number of meaningful options, enough options that the player can express the nuances of his personality through the choices he makes. Yet, decisions must be inputted, and a large number of options seem to require an extensive and complicated input structure, which could well be intimidating to the player. Our dilemma, then, is that an excellent game seems to require a hulking input structure.

The dilemma is resolved through the designer's creativity in designing a clean input structure that allows many options. This does not come easily. Many schemes must be considered and rejected before a satisfactory solution is found. Yet, such a solution is often possible. In designing SCRAM, a nuclear power plant game, I faced the following problem: how can a player control an entire nuclear power plant with only a joystick? At first glance, the task seems hopeless. Nevertheless, the solution I eventually discovered works very well. The player moves a cursor through the plant display. With the cursor adjacent to a piece of controllable equipment, the player presses the joystick button and pushes the stick up to turn on or increase power, and down to turn off or decrease power. The system is simple and easily understood once the player has seen it.

There is a general solution, at the theoretical level, to the dilemma of option richness versus input cleanliness; I call this solution "the webwork". To design a webwork game, we start with a small number of pieces. We then define a relationship that applies to all pairs of pieces. The set of relationships between pieces constitutes a webwork. The webwork can easily become quite complex, yet few pieces are required to create the webwork. In general, the number of pairwise relationships is equal to N*(N-1), where N is the number of pieces. Thus, four pieces can generate 12 pairings, 8 pieces can generate 56 pairings, and 16 pieces can generate 240 pairings. With fewer pieces to manipulate the player faces fewer I/O problems without sacrificing a rich set of relationships in the game.

Backgammon illustrates the simplicity and power of webwork games. Backgammon has only 30 pieces and 26 positions for them to occupy. The relationships between pieces are fairly simple and are expressed through the ability to move and bump. Yet, on any given move, each piece has an offensive, defensive, blocking, or blocked relationship with most of the other pieces on the board. This is partly because almost every other board position in front of the piece can be reached, given the right die roll. It is no accident that the length of the playing area (24 steps) is exactly equal to the maximum die roll. It had to be that way to squeeze all of the pieces into range of each other, thereby maximizing the number of significant pairwise relationships.

Most webwork games rely on spatially expressed webworks; these are easy to depict and easy for the player to visualize. Few games have non-spatial webworks; my own GOSSIP is one such game. Curiously, GOSSIP uses a spatial webwork for its internal computations even though the game webwork is non-spatial. This may imply that game webworks are intrinsically spatial; it may equally well imply that I cannot shake my mind-set free from spatial webworks.

The choice of input device is an important design decision. I maintain that a good game designer should eschew the use of the keyboard for input and restrict herself to a single simple device, such as a joystick, paddle, or mouse. The value of these devices does not arise from any direct superiority over the keyboard, but rather in the discipline they impose on the designer. Simple input devices go hand-in-hand with simple input structures. Complex input devices encourage complex input structures.

The I/O structure is the most important of the three structures in a computer game, for it is the face of the game that the player sees. It is the vehicle of interaction for the game. It is also the most difficult of the three structures to design, demanding both human sensitivity and complete technical mastery of the computer. Give it the care it deserves.

Game Structure

The central problem in designing the game structure is figuring out how to distill the fantasy of the goal and topic into a workable system. The game designer must identify some key element from the topic environment and build the game around that key element. This key element must be central to the topic, representative or symbolic of the issues addressed in the game, manipulable, and understandable. For example, in EASTERN FRONT 1941, I started with the enormous complexity of modern warfare and extracted a key element: movement. Movement dictates the dispositions of the military units. Moving into an enemy's position initiates combat with him. Moving behind him disrupts his supplies and blocks his retreat routs. Moving into a city captures it. Movement is not equitable with all aspects of war; it is, instead, the key element through which many other aspects of war are expressible. It is easily manipulable and immediately understandable.

A more difficult design challenge came from the game GOSSIP. This game addresses social relationships. The enormous complexity of the subject matter and the intricate twists and turns of human interaction together suggest that the subject is beyond treatment in a game. After much thought I was able to isolate a key element: the "statement of affinity". One way or another, many of our social interactions boil down to one of two declarations: a first-person statement of feeling ("I rather like Sandra"), and a third-person statement ("Well, Tom told me that he doesn't like Sandra one bit"). The key element encapsulates the grander array of human interactions rather well. It is easily manipulable; indeed, it is quantifiable. And it is quite understandable. The isolation of the statement of affinity as the key element of human interaction made possible the game GOSSIP.

The nature of manipulability assumes tremendous importance to the success of the game. The key element must be manipulable, but in a very specific set of ways. It must be expressively manipulable; that is, it must allow the player to express himself, to do the things that he wants or needs to do to experience the fantasy of the game. For example, in a combat game, shooting is almost always a key element. If the player's freedom to shoot is heavily restricted, the player cannot live the fantasy. At the same time, the manipulability must be concise. To use the combat game example again, if the player is required to declare the amount of gunpowder to be expended on each shot, he may well find the manipulability a hindrance to the game. The manipulability must be meaningful to the fantasies of the game. Finally, the manipulability must be focused: the options from which the player chooses while manipulating the key element must be closely related. For example, in the game GOSSIP, the key element (statement of affinity) assumes a linear sequence of values ranging from hatred through love. ENERGY CZAR violates this principle by requiring the player to choose from a large, disconnected set of options. Menu structures and use of the keyboard both arise from unfocussed key elements.

Many games employ multiple key elements. For example, most combat games include both movement and shooting. This is not necessarily bad; if both key elements are kept simple, or if one key element retains primacy, the game can be successful. However, too many key elements violating too many of these principles will rob the game of its focus.

Your main problem with creating the I/O structure is overcoming constraints; your main problem with creating the game structure is realizing possibilities. Your previous work with the I/O structure defines the limitations on the structure of the game. You can take more liberties with the internal structure because the player will not directly encounter it. For example, for the game TACTICS I developed a very complex combat algorithm that realistically calculates the effects of armor-piercing shot. The complexity of this algorithm would have confused the player had I tried to explain it. But the player does not need to understand the internal workings of the algorithm; he need only grasp its effects. I therefore did not feel constrained to design a simple-minded and intuitively obvious algorithm. Concentrate an providing enough color to guarantee that the game will convey the authentic feel of reality. Keep your sense of proportion while adding details. It will do your game no good to provide exquisite detail and accuracy in one sphere while overlooking the most fundamental elements in another sphere.

A very common mistake many designers make is to pile too many game features onto the game structure. In so doing, they create an overly intricate game, a dirty game. As I discussed in Chapter 4, dirt is undesirable; a game is a structure that must fit together cleanly and well, not a brushpile. Dirt creates a second problem not mentioned in Chapter 4: it gums up the I/O structure of the game. For example, the long-range scan feature of STAR RAIDERS does provide some nice additional capabilities, but it adds another keystroke to be memorized by the player. That's dirty input. Fortunately this problem is overridden in STAR RAIDERS, because the fantasy puts the player at the controls of a starship, and so the player finds the intricacy of the control layout a supporting element of the fantasy rather than a hindrance. In most games, you may well be forced to give up nice elements in the game structure in order to maintain the quality of the I/O structure. On the other hand, you may be forced to go back and change the I/O structure to incorporate a game feature you are unwilling to abandon. If you do so, do not simply tack on a now command; rethink the entire I/O structure and modify it so that the new command fits well with the rest of the I/O structure.

Designing the game structure is emotionally very different from designing the I/O structure. While designing the I/O structure, the designer must thread a precarious path between the Scylla of expressive power and the Charybdis of expressive clarity, even while the storms of hardware limitations toss her design to and fro. While designing the game structure, the designer finds herself on a placid sea stretching flat to the horizon. The challenge taunting her now is "Where do you go?"

Program Structure

The program structure is the third object of your design attentions. This structure is the vehicle which translates the I/O structure and game structure into a real product. One of the most important elements of the program structure is the memory map. You must allocate chunks of memory for specific tasks. Without such safeguards, you may end up expending excessive quantities of memory on minor functions, and having insufficient memory remaining for important tasks. Definitions of critical variables and subroutines are also necessary. Finally, some documentation on program flow is important. Use flow charts or Warnier-Orr diagrams or whatever suits your fancy. This book is not primarily concerned with programming; if you need guidance on program development, consult any of the many excellent books on program development.

Evaluation of the Design

You now have three structures in hand: the I/O structure, the game structure, and the program structure. You are satisfied that all three structures will work and that they are compatible with each other. The next stop in the design phase is to evaluate the overall design for the most common design flaws that plague games. The first and most important question is: does this design satisfy my design goals? Does it do what I want it to do? Will the player really experience what I want him to experience? If you are satisfied that the design does pass this crucial test, proceed to the next test.

Examine the stability of the game structure. Remember that a game is a dynamic process. Are there any circumstances in which the game could get out of control? For example, if the game has money in it, could a situation arise in which the player finds himself the owner of ridiculously large amounts of money? In short, does the game structure guarantee reasonable upper and lower bounds on all values? If not, re-examine the game structure carefully with an eye to structural changes that will right the situation. If you have no other options, you may be obliged to put them in by brute force (e.g., "IF MONEY > 10000 THEN MONEY 10000")

Now probe the design for unanticipated shortcuts to victory. A player who can find a way to guarantee victory with little effort on his part will not derive the full benefit of your game. Insure that all unintended shortcuts are blocked so that the player must experience those processes that you want him to experience. Any blocks you place must be unobtrusive and reasonable. The player must never notice that he is being shepherded down the primrose path. An example of obtrusive blocking comes from the game WAR IN THE EAST (trademark of Simulations Publications, Inc). This wargame deals with the Eastern Front in World War 11. The Germans blitzed deep into Russia but their advance ground to a halt before Moscow. To simulate this the designers gave the Germans an overwhelming superiority but also gave them a supply noose whose length was carefully calculated to insure that the Germans would be jerked to a dead halt just outside Moscow. The effect was correct, but the means of achieving it were too obvious, too obtrusive.

The last and most crucial decision is the decision to abort the game or proceed. It should be made now, before you commit to programming the game. Do not hesitate to abort the game now; even if you abort now you will still have I earned a great deal and can say that the effort was worthwhile. A decision to give up at a later stage will entail a real loss, so give this option careful consideration now while you can still do it without major loss. Abort if the game no longer excites you. Abort if you have doubts about its likelihood of success. Abort if you are unsure that you can successfully implement it. I have in my files nearly a hundred game ideas; of these, I have explored at length some 30 to 40. Of these, all but eight were aborted in the design stage.

PRE-PROGRAMMING PHASE

If the game has made it this far, you are now ready to commit your ideas to paper. Until now your documentation has been sketchy, more along the lines of notes and doodles than documents. Now you are ready to prepare your complete game documentation. First, commit all of your design results from the previous phase to paper. Define the I/O structure and the internal game structure. The tone of this documentation should emphasize the player's experience rather than technical considerations. Compare this first set of documents with your preliminary program structure notes; adjust the program structure documents if necessary.

PROGRAMMING PHASE

This is the easiest of all the phases. Programming itself is straightforward and tedious work, requiring attention to detail more than anything else. Seldom has a game failed solely because the programmer lacked the requisite programming skills. Games have failed to live up to their potential because the programmer did not expend enough effort, or rushed the job, or didn't bother to write in assembly language, but in few cases has talent or lack of it been the crucial factor in the programming of a game; rather, effort or lack of it is most often the responsible factor. If you place all of your self-respect eggs in the programming basket, I suggest that you get out of game design and work in systems programming. Otherwise, write the code and debug it.

PLAYTESTING PHASE

Ideally, playtesting is a process that yields information used to polish and refine the game design. In practice, playtesting often reveals fundamental design and programming problems that require major efforts to correct. Thus, playtesting is often interwoven with a certain amount of program debugging.

Sometimes playtesting reveals that the game is too seriously flawed to save. A nonfatal, correctable flaw is usually a matter of insufficiency or excess: not enough color, too many pieces, not enough action, too much computation required of the player. A fatal flaw arises from a fundamental conflict between two important elements of the game whose incompatibility was not foreseen. You must have the courage to trash a game with such a fatal flaw. Patching after the game is programmed can only achieve limited gains; if the game is badly deformed, abortion is preferable to surgery.

If playtesting reveals serious but not fatal problems, you must very carefully weigh your options. Do not succumb to the temptation to fall back on a quick and dirty patch job. Many times the problem that is discovered in playtesting is really only a symptom of a more fundamental design flaw. Be analytical; determine the essence of the problem. Once you have determined the true nature of the problem, take plenty of time to devise a variety of solutions. Don't rush this process; sometimes the ideal solution comes from an unexpected angle. Choose a solution for its promise of furthering the faithfulness of the game to your goals. Do not opt for the easiest solution, but the solution that best meets your goals.

For example, while designing EASTERN FRONT 1941, I ran into a severe problem with unit counts: there were far too many units for the player to control conveniently. After wasting much time trying to devise ways to shrink the map or directly reduce the number of units, I eventually stumbled upon zones of control, a standard wargaming technique that extends the effective size of a unit. The inclusion of zones of control in the game not only solved the unit count problem; it also made the logistics rules more significant and gave the game a richer set of strategies. I set out with the narrow goal of reducing the unit count, but I found an improvement with much broader implications. If your initial design was well-developed (or you are just plain lucky) the game will not face such crises; instead, the problems you will face will be problems of polish. All of the little things that make a game go will be out of tune, and the game will move like a drunken dinosaur instead of the lithe leopard you had envisioned. Tuning the game will take many weeks of work. For the short term you can scrimp on the tuning while you are working on other problems, for tuning the game requires delicate adjustments of all the game factors; any other changes will only throw off the tune. Therefore, defer final tuning work until the very end of the polishing stage.

There are actually two forms of playtesting. The first is your own playtesting done in the final stages of debugging. The second form comes later when you turn over the game to other playtesters. The salient-difference between the two lies in the nature of the bugs exposed. Your own playtesting should reveal and eliminate all program bugs (arising from flaws in the program structure) and many of the game bugs (arising from flaws in the game structure). The game you give to the playtesters should be free of program bugs; they should discover only bugs in the game structure. There is no point in showing an incomplete game to playtesters, and indeed there is a danger in contaminating their objectivity by showing them a version of the game too early. But the time will come when you feel that the game is very close to completion, and your own stock of ideas for improvements is dwindling. This is the time to show the game to a few select playtesters.

Playtesters must be selected and used with great care. You cannot simply grab a few friends and ask them what they think of the game. You need playtesters who possess a deep familiarity with games, playtesters who can analyze and criticize your game with some basis of experience. Ideally the playtesters would themselves be game designers, for they would then share your appreciation for the trade-offs essential to good game design. You should also know the player well, both his personality and his game taste. You should never use more than five or six playtesters. A surplus of playtesters only insures that you will not be able to assess carefully the reaction of each playtester.

A variety of other systems have been used for playtesting. Most rely on gathering large groups of "real people" and assessing their reactions to the game. I have little respect for such systems. Although they are scientific, objective, and democratic, they seldom yield useful design information, for consumers make lousy critics. The suggestions they make are inane and impractical; they don't know enough about computers or games to make practical suggestions. Such methods may well work with detergent and shaving cream, but I very much doubt that any great movie, book, or song was created through market research of this kind. I will concede that such methods can prove to be a useful way to guide the mass production of cheap games by designers of limited talents; this book is not directed to persons of such a mentality.

The playtesters will need a preliminary manual for the game. It need not be a finished product any more than the game itself---just enough orientation information to get the playtester going with the game. Make sure that there is enough in the manual that the playtester doesn't waste time critiquing problems of the game that will be solved by the manual. Do not sit down with the playtester in advance and coach him through the game; you will only contaminate his objectivity. The playtester's first reaction to the game is your best feedback on the success of the manual . Let the playtester experiment with the game for perhaps a week before you meet with him. Do not ask the playtester to keep lengthy written records of play performance; he won't do it. Instead, include in the manual a few suggestions about potential problems that worry you. The most for which you should ask in writing is a simple record of game options selected and subsequent scores.

Schedule along interview with the playtester after he has had enough time to digest the game. Come to the interview prepared with a set of standard questions that you ask all playtesters. Do not lead the playtester's answers and don't solicit praise. Your job is to find flaws; accolades come later. While it is more scientific to use a third person to conduct the interview (thereby assuring more honest answers), this imposes a middleman between you and your playtesters. I prefer to get the information directly from the playtester. I also prefer to take a very negative tack during the interview, encouraging the playtester to criticize the game along with me and to suggest means of improving it.

Playtesters' criticisms are difficult to evaluate. Most criticisms must be rejected for a variety of reasons. Some are incompatible with your goals; some are not achievable in the-memory space remaining. Some are reasonable, but would require major software surgery incommensurate with the gains offered. Do not hesitate to reject 90% of the suggestions made. The remaining 10% are right; waste no time implementing them. How do you tell the good 10%? This is the stuff of wisdom; I certainly don't know.

The final stage of the design cycle is devoted to polishing the game. The polishing stage is actually concurrent with the later stages of playtesting and may involve several iterations with the playtesters. This stage is critical; the designer has been working on the game for a long time by now and the luster of the new design has worn off. It is now only a big job that should have been finished months ago. The playtesters love it, the publisher loves it and wants it right now, and the designer is sick of it. The urge to dump the damn thing is overpowering. Resist this urge; press on relentlessly and polish, polish, polish. Keep testing the game, fine-tuning it, and adding tiny embellishments to it. Once it's out the door, it's gone forever. Every single game I have done has followed the same pattern: I polished the game until I was sick of it and never wanted to see it again. When at last I sent the game out, I rejoiced; I was free of that dog at last. Within a month I was regretting my impatience and wishing I could have a chance to clean up that one embarrassing bug that I had never noticed. Within three months my regret had turned into shame as I discovered or was told of many more bugs. I have programs out there whose patrimony I hope never becomes widely known.

One of the last tasks you must perform before releasing the game is the preparation of a game manual. Manuals are frequently given short shrift by just about everybody associated with computer games. This is a serious mistake, for the manual is a vital element in the overall game package. A computer has many limitations; some can be overcome with a good manual. Much of the static information associated with a game can be presented in a manual. The manual is also an excellent place to add fantasy support elements such as pictures and background stories. Finally, a well-written manual will clear up many of the misunderstandings that often arise during a game.

You must write your own manual for the game, no matter how poor a writer you are, and even if a professional writer will prepare the final manual. The attempt to write your own manual will increase your respect for the skills of the professional writer, making it more likely that you will have a productive relationship with the writer. Writing your own manual will also provide feedback on the cleanliness of the game design. Clumsy designs are hard to describe, while clean designs are easier to describe. Finally, your own manual will be a useful source document for the professional writer. You should be prepared for the writer to throw out your manual and start all over---a good writer would rather create a new manual than polish an amateur's crude efforts. You must cater to the writer's needs, answering all his questions as completely as possible. Only a close and supportive relationship between designer and writer can produce an excellent manual.

POST-MORTEM

Once the program is out, brace yourself for the critics. They will get their filthy hands on your lovely game and do the most terrible things to it. They will play it without reading the rules. If it's a strategic game, they will castigate it for being insufficiently exciting; if it's an S&A game, they will find it intellectually deficient. They will divine imaginary technical flaws and speculate incorrectly on your deep psychological hang-ups that led you to produce such a game. One critic of mine concluded that TANKTICS was obviously slapped together on a rush schedule; actually, the time between first efforts and final publication was five years and two months. Another roasted ENERGY CZAR (an energy economics educational simulation) because it wasn't as exciting as his favorite arcade game. Don't let these critics affect you. Most critics are far less qualified to criticize programs than you are to write them. A very few critics with the larger publications are quite thoughtful; you should pay attention to their comments. With most critics, though, you should pay heed only to views shared by three or more independent critics. Remember also that even a good critic will roast you if your goal is not to his taste.

The public is another matter. If they don't buy your game, you lose two ways: first, you or your employer make little money on the game; and second, you don't reach as many people with your message. It doesn't matter how beautiful your message is-if nobody listens to it, you have failed as an artist. One failure is nothing to worry about; every artist bombs occasionally. Two failures in a row are bad; three should initiate a serious reconsideration of artistic values. Are you willing to be a noble and starving artist, or a somewhat wealthier artisan? Look within your heart, long and hard. If deep down inside you know that you met your goals, then ignore the critics and the public.

Good RPG/Adventure Games

dlubarov@yahoo.com

What aspects of RPG/adventure games make them so addicting? What qualities do they behold that makes them one of the most popular genres in the world of gaming? The answers to these questions are not simple ones. They require good logical thinking and a clear understanding of the human psyche.

Three factors play a role in whether one will stay glued to a game. The first is implied; the quality of the game play. The second is how many unique, interesting challenges there are to keep the player busy. The third is whether there are obstacles which would make the player jump up, scream, and start thrashing his monitor.

A well designed RPG has good game play naturally. How? There is one thing other than the general components of a good game. That is the sense of growing, of accomplishing, of gaining power. Here is where the human psyche comes in. If a human of early time were to build a shelter for protection from animals, he would be more likely to survive and reproduce. Likewise if he built a fire to drive off small critters. Now, I'm sure you're wondering how this relates to game play. Well, the same would apply if a medieval knight obtained a large stock of gold (money is power) or if he got stronger armor. Because of evolution, those who like building forts would soon make up a large portion of the population. This can explain why a child likes building forts at the beach. The medieval knight example explains why consumers like aspects of role playing games. As a designer, you must use this knowledge to your advantage.

Now you understand the logistics, but the question becomes how to implement them in your game. I will provide the simple answers, but to get further you must study it yourself. The player should be able to progress in two fields: finance (including possessions) and experience. He should be able to collect gold or some form of currency from conquering foes, completing quests, and selling positions. He should be able to save up for possessions or acquire them from enemies. He should then be rewarded separately for combat. I know I'm describing every other commercial game, but there is sense in it. If you are more ambitious and want something unique you could try implementing these aspects in a game with a futuristic theme.

Okay, you've got the engine down. But something's missing. Ah, there's nothing to do! Imagine your favorite RPG with one room and one monster. This is a limited environment. The player should be able to set goals for himself and have them set for him, and take steps to progress towards them. Defeating a single monster would get boring when done repetitively. The player can get gold, but in order for him to be happy about it he needs to have something to spend the gold on. One of the most essential qualities of an RPG is length. Let us look at a somewhat bad example, Zelda: Links Awakening. It is not strictly an RPG, but it will do. When you defeat the boss in the egg, you eventually are forced to shut off your Gameboy and have nothing to do but defeat him again from where you last saved. A good RPG has no "Game Over. You Win". If you must have a small number of challenges, at least make them harder near the end. That way if the player still wants to play, he still has something to work toward. Many online RPGs use player against player combat to achieve this.

There is one more factor that will compel the player to stop playing your game, and that is obstacles that are too hard, or any otherwise aspects that could cause frustration. Here are some bad examples:

-In Runescape, if you accidentally click on a pumpkin or Easter egg, you could lose a valuable item. Likewise if you click on someone your level in the wild.
-In Final Fantasy III, near the end, if you didn't bring someone who knows X-Zone, you will not be able to defeat the tow monsters and must let yourself die, then backtrack a long way.

There are many other situations. Here are some tips on how to avoid them:

-Warn the player of possible consequences for doing things.
-Don’t make the consequences for dying or anything else too harsh.
-Don’t let the player waste finance on an item they don’t need.
-Consider letting the player sell items for the same as the buying price.

You should take these matters into account when doing game design or planning. I hope to see some good RPGs from the community.

Feedback? Questions? Comments? Complements? Rants? Marriage proposals? Death threats? Send it all to me at dlubarov@yahoo.com.

Online Games

Donna Coco

Everyone knows that the Internet and multiplayer games are a natural fit. Even when computers and networks weren`t adept at handling numerous people playing games simultaneously, developers and creative players found ways for multiperson interactive play. So now that computers are more powerful and network connections have improved, it`s only fitting that an increasing number of developers are interested in staking a claim in this growing market.

Of course, knowing that the Internet is a good conduit for gameplay is quite different from creating a successful online game. The good news is, there are some similarities between developing a general game and developing for online. For example, just like with non-networked games, the graphics portion of online games typically runs on the players` systems. Only strategic information is sent over the network--who moved what piece where, etc. In fact, developers stress that online games need to be as compelling graphically as any other game. They still must choose whether to go with real-time 3D polygons or pre-rendered bitmaps and sprites. And artists use the same 3D modeling and animation, image editing, and other tools they would to develop any game.

Indeed, many developers are taking games originally created for individual play on the PC and making them network- and multiplayer-capable. These include the typical range of games, from fast-action shooting to racing to war/strategy to flight simulations to board and card games. Other companies, such as Kesmai Studios (Charlottesville, VA) and Interworld Productions (Fairfax, VA), are developing games only intended for online play.

Regardless of the game`s origin, there are several factors that developers need to consider when going online. One seemingly obvious factor is that multiplayer games have several people controlling a single environment. Says Richard Lawrence, third-party producer at Kesmai, "In a single-player game, you [the player] own the machine, so we can make a lot of assumptions. You never depend on anyone but yourself for information. So if you read a value from a database--say, the weight of your plane--everything else can halt while you wait for that information. But in a multiplayer game, you can`t make everyone else freeze because this guy has a problem getting information. So you make a request ahead of time to read the weight of the plane and asynchronously wait for that to appear. If it doesn`t, you keep this one player from flying. Or, you can pre-cache [all the information a player] might need. But again, the essential difference is that no single player has control over the environment."

Mark Vange, chief technology officer for VR-1 (Boulder, CO), an online game developer/service provider, says his company eventually plans to create games that can support as many as 2000 players, which introduces a number of concerns. "For shrink wrap, all I have to do is create a game where one person is in command, and the story only has to support that one person. But when you have 2000 people playing together, each one has to be entertained. And other human players are doing things I can`t predict. So there has to be enough of a framework that all those people can be entertained."

Starr Long also knows from experience the difficulties of designing a game for thousands of players. He is producer and director for Origin`s (Austin, TX) Ultima Online, a soon-to-be-released fantasy role-playing Internet game based on a long-running single-player game of the same title (Origin is working on the ninth installment of the single-player series). Long says his game, too, will support as many as 2000 players simultaneously. "We had a lot to consider when taking Ultima from a single-player to a multiplayer environment," he says. "In keeping with the Ultima name, we had to figure out some way to have a plot. Origin is known for making games with richness and depth of plot. But that`s difficult to do with multiplayer games. In a single-player game, you can have one person going places and doing certain tasks; the game evolves, builds to a climax, and ends. But you can`t pace a multiplayer game. With 2000 people playing simultaneously, you can`t time how all those people will progress."

To deal with the plot issue, Ultima developers decided to center the game around two major characters. "Neither character is good or bad," notes Long, "but both have definite motives that conflict with each other and adventures and quests that are associated with those two people."

Additionally, the two major characters will be played by real people. In fact, one will be Origin founder Richard Garriot, who will play his long-time character Lord British. "This will be the first opportunity Richard has had to actually play that character," notes Long. "So that`s how we solved the plot problem. It will be an ongoing plot with mini-climaxes and different adventures, but it will never really end."
With many people new to the concept of online games, developers also should consider that some potential players may be intimidated by the thought of interacting with people they don`t know. Plus, that fear may be compounded if a person isn`t familiar with the game. "This is actually a common problem," says VR-1`s Vange. "It can take some people months to come out of their shells."

One way to deal with this is to create what Vange calls a "rookie arena"--a place where newcomers can practice playing without the pressure of competing against established players. Says Vange, "This arena would vary depending on the game, but it would be an area where the rules are more forgiving and you`re not earning points toward your overall stats. It`s a place to explore and learn before going into full-fledged battle, where some people will be very [experienced] and some could be mean."

On a more technical level, another important issue to consider is latency: because information is sent over such complex networks and players` computers can vary greatly in power and connection capabilities, developers have to think about how to deal with the varying amounts of time it can take for packets of data to get to their destinations. Mark Jacobs, president of Interworld Productions, says his company considers latency issues from initial conception. "We`d love to have very low latency--under 150msec--but we`re counting on 500msec. We keep that in mind right away. You have to sit down and go into what you can realistically do from the start."

To manage latency, most developers use some form of predictive motion and interpolation. Kesmai`s Lawrence explains, "Our [air-combat] game Air Warrior is a huge game of anticipating requests, making guesses ahead of time as to what the player will do, and never interrupting the suspension of disbelief. If a player makes a request of `what do I see,` I can`t stop the guy`s plane because I haven`t received that information. So there`s a lot of interpolation built in: the game guesses forward based on data it had previously. If I had a packet that said another guy was crossing my path going 300mph, my game will continue to draw him at that rate if I don`t get a new packet."

Lawrence continues, "When I get another packet, I have to be careful not to interrupt the reality, which is what people call warping. That`s when there`s a plane here, and suddenly it appears over there. If the change is within a defined set of accepted error, I`ll interpolate the path of the plane to say it turned that way naturally, so the plane will angle off to the right maybe two frames after it normally would have."

Over at Origin, Long says they designed Ultima Online to absorb a latency of 500msec without an obvious degradation to play. But they also employed other methods for combating latency. Says Long, "For example, it doesn`t matter how fast your connection is for things like combat, because it`s controlled by the server. You select your target, and then the server automatically does the attacking for you. Otherwise, if you have a faster connection, you can maybe hit someone 10 times, compared to the slow guy who can only hit maybe five times. We`re avoiding an outcome based solely on speed."

Another technical issue to resolve is the architecture of the game: mainly, will it be client/server or peer-to-peer? In client/server, the client (the player`s computer) handles display of graphics and connects to a main server, which acts as the traffic cop and controls the overall action in the game. In peer-to-peer, players connect directly to each others` computers. "The problem with peer-to-peer is the more clients, the more connections you must form, so your chances of having a bad connection increase," says Lawrence. "If you have eight players, you have eight people depending on being in a good communications state. If anyone has low latency, it will damage the progress for everyone."

Matthew Ford, producer and game designer at game developer Accolade (San Jose, CA), adds, "Usually, the problem with peer-to-peer is that everybody has to move at the speed of the slowest computer, which can decrease the speed of the game. In a client/server model, if you have a slow machine, it only hurts you."

Additionally, certain types of games need a client/server architecture. Game developers refer to these as "persistent-universe" games, in which the game environment continues to exist regardless of who is or isn`t playing. To maintain the game, the server must run 24 hours a day. Origin`s Ultima Online, for example, is a persistent-universe game.

Interworld creates both persistent-universe and non-persistent games. "With Dragon`s Gate and Magestorm, you go in, create a character, and everything you do from that moment is recorded," says Jacobs. "In a game like Splatterball, [in which opponents fire paint balls at each other], you have an ongoing score, but there`s no role playing [so nothing else is persistent]. In Castles, you go in and play matches and play until someone wins."

Kesmai`s Air Warrior is somewhat of a hybrid persistent-world game. Kesmai runs Air Warrior in monthly cycles. During that month, the game is persistent, so players could find a completely different scene between the time they log off and back on to the game. Explains Lawrence, "There are three countries fighting each other in Air Warrior, and everyone gets the same capabilities. Except every night a different group of people might be playing. Some nights, you might log on and your country is beating up on everyone else. Another night, you might log on and only 10 other players from your country are there, and the other countries have you surrounded."

At the end of the month, a "winner" is proclaimed and the entire game is reset. Everyone starts at ground zero again. Still, notes Lawrence, there`s no point in which Air Warrior "the game" has been beaten. "It doesn`t happen. We don`t want people to finish playing," he says.

Although various types of online games will undoubtedly be popular, many developers feel persistent-universe games--specifically, those that are role-playing--hold the most potential. Jacobs of Interworld says, "Persistent universes are the way to go. After awhile, most people don`t care that much about the game anymore. They care more about the social interaction."

Adds Accolade`s Ford, "With persistent universes, you have an identity. Every time you check in, your possessions and skills are maintained. You can play week after week, and you see your status move."

Accolade`s first Internet-capable game, Deadlock--a futuristic strategy game in which players are one of seven alien races vying for control of a new planet--is not persistent-universe. But Accolade`s current project is, notes Ford. And because the game will in theory exist "forever," designers have to take a somewhat different approach when developing the game. Says Ford, "Creating these types of games is about creating a world that has lots of possibilities. You put this dungeon here and these monsters there so that people can get some experience. Then that castle over there may be tougher. You set up a world that would be fun to play in, then let the players in and see what they do."

Designers also must figure out methods for allowing all players the chance to complete all the game`s tasks. For example, if one player slays a dragon, how do you deal with the fact that others may have wanted to tackle that mission? Because people come and go but the world remains, such scenarios will happen. A number of games deal with this issue by re-populating after a certain period--monsters, treasures, everything tied to quests is re-stocked. Additionally, developers usually architect the game so they can regularly add new content--a new "land" or "planet," for example--to keep people interested. Says Origin`s Long, "Basically, you want people to log off only because they have to sleep or eat."

Like other designers, the creators of Ultima Online had to figure out how to make their world dynamic. But they decided to take a different approach. To start, Origin discarded what gamers refer to as the "room mentality": players enter a room (or cave or whatever), complete a task, and leave. The problem with this format, says Long, is that eventually skilled players will complete all the tasks.

"Instead," he says, "we developed an intelligent resource system and virtual ecology that are linked to each other. For example, let`s say there`s a cave. The cave will generate a need for a monster to live in it, so the game will create a dragon to live in the cave. Well, the dragon needs something to eat, so it will look for deer in the forest outside its cave. But say someone kills the deer. Now the dragon needs food but will have to start looking further away for it, perhaps in a nearby village. The game is never told to make the dragon attack the town of villagers; it`s what people do within the game that sets events in motion."

People will have to get used to playing in such an environment, notes Long. A player may leave the game for a week, come back, and find his/her village was eaten by a dragon and all new people now live there. However, the game will incorporate means to help players in such situations, including nonplayer characters who can tell gamers what happened or help direct them in their quests.

Whether taking on the role of a wizard and casting spells or taking flight in a 3D plane to fight for a virtual country, people are drawn to online games for the challenges and, more importantly, for the social aspects of games, note developers. As Vange of VR-1 says, "People will come back for the other people who are playing." Now it`s up to developers to create the content that will lure people in to play.

Legends of Kesmai was created by online developer Kesmai Studios. In the role-playing Legends of Kesmai, players take on a persona and explore this ever-growing fantasy world of creatures and treasures.


GAME PLAYERS
Wanted: Skilled, Fun, Talented and Competitive person to play FREEvideo games and even earn cash. Duties include: Having fun while using your skills when playing a really wide variety of games and entering fun tournaments where you play with other players. Hours: You decide that. Earnings: You also decide that. Start work today!
www.gameland.com
BIRD WASHER
PROTOTYPE TESTER
PRISON POOL CLEANER
CARRIER
DETAILER
CLEANING
WARHEAD TESTE
HOE MAKER
ROLL OF QUARTERS
gameland
Wanna play?
www.gameland.com
Play...It's FREE!
When asked who referred you
say morphosisgames@aol.com

Diversity In Gaming

Phil LoPiccolo

In case there was any doubt about it, a new poll from the Entertainment Software Association (www.theESA.com) shows that computer gaming is not just for the boy next door anymore. In fact, the survey reveals that computer game players are becoming increasingly diverse, thanks largely to women, who now make up a larger percentage of gamers than boys age 6 to 17.

ESA's research indicates that more than half of the US population plays computer games. And while men over 18 still represent the largest segment (38 percent), women 18 and older now account for 26 percent of the audience, whereas boys 6 to 17 represent just 21 percent. Add the fact that girls 6 to 17 comprise another 12 percent, and we find that female players of all ages total as large a portion of the gaming audience as men.

What has led to a wider audience and, in particular, the greater participation in gaming by women? According to ESA president Douglas Lowenstein, "This diversity is being driven by advancing technology and the introduction of more titles in more genres, which provide more entertainment choices for all players, regardless of age or gender." He also notes that the top three factors players cited when making game purchasing decisions are quality of the graphics, price, and 3D graphics content.

So how can game publishers reach this wider audience? Should they ride the graphics curve while developing more titles that appeal to women? On one hand, there's little doubt that males and females have dramatically different tastes in games. For example, ESA's survey shows that the vast majority of boys and men play sports and action games, while girls and women overwhelmingly prefer card games, puzzles, and board games. Moreover, there's growing evidence that males and females are dramatically different, in general. The latest research indicates that even our DNA is far more different than previously thought. Indeed, based on newly discovered differences between human male and female chromosomes, geneticists say that the genomes of men and women differ by as much as 2 percent. What's astounding about this finding is that men may be more similar to male chimpanzees than they are to women (although this would not shock women who know me), and women may be more similar to female chimpanzees than they are to men.

On the other hand, despite such obvious differences, there's more diversity within gender groups than between them, at least in terms of gaming preferences. In fact, the ESA poll reveals that 30 percent of all gamers play at least three genres of computer games. What's more, even though players say they choose games on the basis of genre and technology, they will "cross over" to play any type of game, if they can relate to it on some level. "The most important aspect of a game is its ability to connect with the player," Lowenstein contends. "While there may be games with great graphics and sound, if the player isn't drawn in by some aspect of the game, he or she won't play."

One sure way to connect with players would be to develop more interesting characters and story lines. In fact, the ESA poll shows that this aspect of gaming was the fourth most cited reason for buying games, after graphics, price, and 3D content. Of course, an ideal way to achieve that would be to expand the types of game narratives and include more contributions from women, who now make up only about 10 percent of the game development community, according to the International Game Developers Association (www.igda.org). Perhaps the best strategy for further expanding this $16 billion market is simply to get more people involved in creating games that everyone can enjoy. After all, while men and women may differ genetically by 2 percent, we're still 98 percent alike.

Interactive Storytelling

Randy Littlejohn

Game developers are beginning to realize that mindless, violent action, and fantastic special effects supported by ever advancing hardware will not hold the interest of core gamers forever. After certain point, advances in resolution and sound won't be enough to increase sales. The added effect will be negligible. In addition, mindless games without good characters and narratives will never attract a wider market, despite photo-realistic decapitation and volcanic eruptions of blood. Sure, we can always count on x-number of boys coming up through the ranks who will buy x-number of units. But why should we be satisfied with this small market when there is a much larger market to be nurtured and exploited? As a result, some have begun to reconsider the importance of story and character development.

A rising concern is, "How do we graft a story to our action game?" Story means linear...right? The whole idea of a story is opposed to the idea of interactivity…right? The basic concern is "How do we make an effective interactive story?" So what does effective mean in terms of interactive storytelling? There are two basic ingredients. These are intuitive interface design and compelling stories. In this article, I will address one of the two ingredients, the development of compelling interactive storytelling.

My definition of a compelling story is one that grabs and holds the attention of the audience. It must move and excite them. It must take them on an emotional roller coaster. Finally, it must make them feel like they have had a worthwhile experience at the conclusion.

What is it that engages and holds on to us in stories, interactive or not? What makes a story compelling and satisfying? An art form has evolved to deal with these issues. The name of this art form is "drama". Though the word "drama" is thrown around a lot, very few could accurately describe it. So before we begin to explore how the principles of drama can be adapted to create compelling interactive entertainment, we must first briefly review what drama actually is. After a brief overview, we will explore specific tools and suggest some ways to adapt them.

What is Drama?

There are many generalized descriptions of drama, which is actually a body of arcane knowledge compiled over thousands of years. The main points of agreement are that drama is a story of human conflict communicated by means of speech and action to an audience. Moreover, that which depicts human conflict will command attention and interest. Therefore drama uses the innate human interest in conflict to engage an audience for the purpose of communicating a theme. The theme must be something that we can all relate to.

In a dramatic presentation, conflict is expressed through visible action. Of course game designers understand the need for action. But to make a project compelling, the reason for the action is more important than the action itself. The forces that cause the action are what excite the audience, making the action believable, and holding the audience in rapt attention.

What are the reasons for action? In life and in drama, the study of the human being resolves itself into an evaluation of the motivation that provokes action. Whenever there is a balance of forces in our lives, we prefer to not act. However, when there is an imbalance of forces, and the motivation to restore balance is strong enough to overcome this basic inertia, some kind of action is taken.

The motivation to act lies in our wishes, needs, and desires. When any obstacle stands in the path of the resolution of these motivations, conflict occurs. In its barest form, a dramatic work all comes down to a character, or a group of characters, that we empathize with because they want something that we can all relate to wanting, and antagonistic forces that opposes the fulfillment of our want. The clash of these opposing forces results in dramatic action.

Human motivation can arguably be divided into four basic drives: desire for response, desire for recognition, desire for adventure, and desire for security. These are the motivating forces that control the actions of all humans.

Desire for Response: the need every human being feels for intimate contacts with others -the desire for companionship and fellowship- can be fulfilled by a dramatic work in at least two different ways. It can be a social institution. People seldom go to the theatre or to the theater by themselves. Of course massively multiplayer games hook into this aspect of dramatic presentations.
In a more universal sense, an interactive dramatic work can satisfy the desire for response by providing the participant with a chance to partake in the drive to resolve the conflict with others. The participant who is caught up in the imaginative whirl of the work feels a fellowship and an intimate personal contact with the dramatic characters — empathy, in other words. Instead of just the immediate thrill of a firefight, we also gain the desire of the participant to achieve a positive response from the characters by his or her actions. Thus, we have just made ordinary action more compelling.

Desire for Recognition
By way of the dramatic work we may enjoy all of the recognition denied us in life: fame, influence, authority, reputation, and renown. Drama is peopled with fabulous or fantastic characters to identify with. Traditionally we vicariously enjoy the homage given kings and heroic warriors. In the interactive realm we can directly receive the plaudits of a grateful society for bringing the bad guys to task. We can feel firsthand the rush of victory after a battle that would be much too dangerous in real life. If we choose to follow an outsider or anti-hero, we get the chance to feel much more special, unique, or unusual than in mundane life. But this only works if we have based our venture on the basic premise of drama: In its barest form, a dramatic work all comes down to a character, or a group of characters that we empathize with, because they want something that we can all relate to, but very difficult obstacles stand in our way. If we don't care about what the characters want, or if what we want is too easy to get, it won't move us. That is, it won't be fun.

The Desire for Adventure
No one's life is so complete that he or she doesn't desire vital new experiences beyond the possibility of attainment in ordinary life. The dramatic work is a land of action and adventure. We get to enjoy the thrills of romance and conflict that is frequently denied in life. We may grapple with the problems of a falling dynasty, or stand casually, blaster in hand, and then thwart the alien mob. We are the ones who get to protect the weak and destroy the wicked. (Or rid the cosmos of weak-minded inferiors).

The Desire for Security
In most dramatic works the hero emerges triumphant. When we identify with the hero we vicariously pass through the trials, the struggles, the crises, and remain reasonably sure that our cause will win out. This accounts for the popularity of films with happy endings. When we indulge ourselves with interactive entertainment we experience this firsthand. Some will say, "How immature! Life isn't like that." Of course, but most people do not go to the films or buy a video game to prove their maturity or to see life as it is. Life is complicated and our control of it is minimal. In our times not only our security but the security of life on earth is threatened. The feeling of helplessness in the face of it all is an every day fact of life. But in the dramatic work we get to indulge our emotional and imaginative sensitivity, to be stimulated and diverted, and to see life as it "ought" to be — more secure.

There are of course many other reasons that we seek out a good story, interactive or not. We may seek great intellectual as well as emotional values. They comment upon life and its problems, and perhaps pose specific argumentative propositions. A dramatic work can also provide deep aesthetic and artistic experiences. However, when all is said and done, the great attraction of a well done story lies in the opportunity to participate imaginatively in the dramatic action. A dramatic work can perhaps survive without art or intellect; it cannot survive without emotion.

Drama is a work that encourages empathy, but even more than that it promotes pathos — the quality that arouses feelings of pity, sorrow, and compassion. When a drama is successful, the audience is suspended in an altered state of hyper-awareness and emotion. The principles of drama are what make stories compelling.

Economy is the essence of clarity

Drama is an art form, and as such is a method of concise, powerful communication. In watching a film or a television episode we have declared our willingness to have something communicated to us. We are conditioned to think of a television or cinema screen as space within which significant things are being shown; we will therefore try to arrange everything that happens within this space into an understandable and significant pattern. Hence, anything that is unnecessary or does not contribute to that pattern will be seen as an intrusion, an irritant.

The dramatist limits and controls her imaginative flight within a well-defined dramatic structure. Her prime purpose is to project her interpretation of life clearly and forcefully, so that the experiences of the characters may become the experiences of the spectator. To do this successfully, the dramatist must follow the universal artistic process in adapting life to the stage. It is a process of informed simplification and refinement. The key steps in the creation of a work of art are:

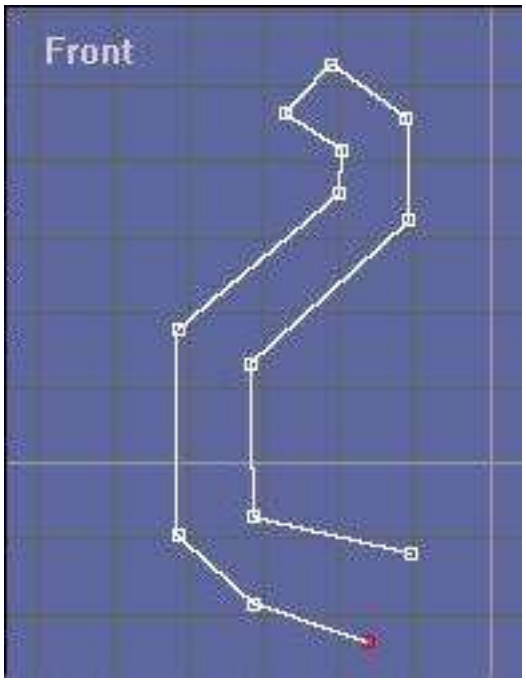
·Selection
·Rearrangement
·Intensification

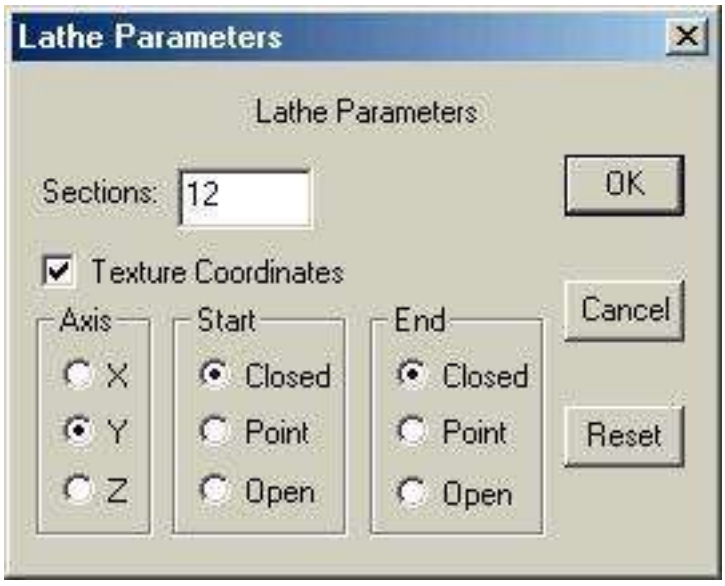
By careful selection, the playwright chooses the conflict, theme, characters, and situation that communicate the playwrights meaning. By rearrangement, they create a dramatic and exciting sequence. The playwright may intensify by highlighting certain characters and subordinating others. They may emphasize particular ideas to the exclusion of others. The Playwright may develop some situations fully and trace others only lightly. The meaning and the power of the drama will depend upon the elements the playwright intensifies. Finally, highly selected dramatic characters are placed in highly selected dramatic situations.

The headline that proclaims "Space Ray Destroys Planet Alderaan" tells of an exciting action. It is compelling in and of itself, but only briefly — the reader wants to know more. What drove someone to do this? How did it happen? What are the results of the action? The drama is the concise tale of the background of the climactic action stated in the headline. It traces in an exciting and clear fashion the interplay of the forces that ultimately drove the destruction of a small, peaceful planet. A dramatic presentation is the story of the struggle and conflict that caused the final action.

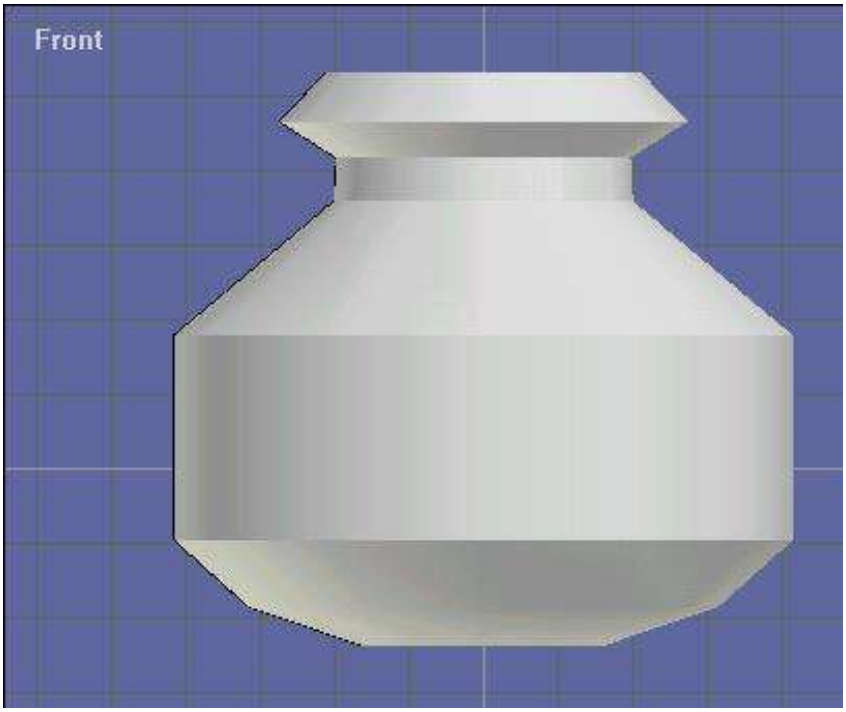
To begin, select the button. Make sure you are in the Front or Side view. If you're not, click View->Front at the top of the form. Now you must decide what kind of pot you would like to draw. I chose for a standard pot shape.

Draw it in the blue area of your screen. Make sure you draw it on the left side of the Y-axis.
When you've done that, select the spline you have drawn and go to Build->lathe. you will see the following:

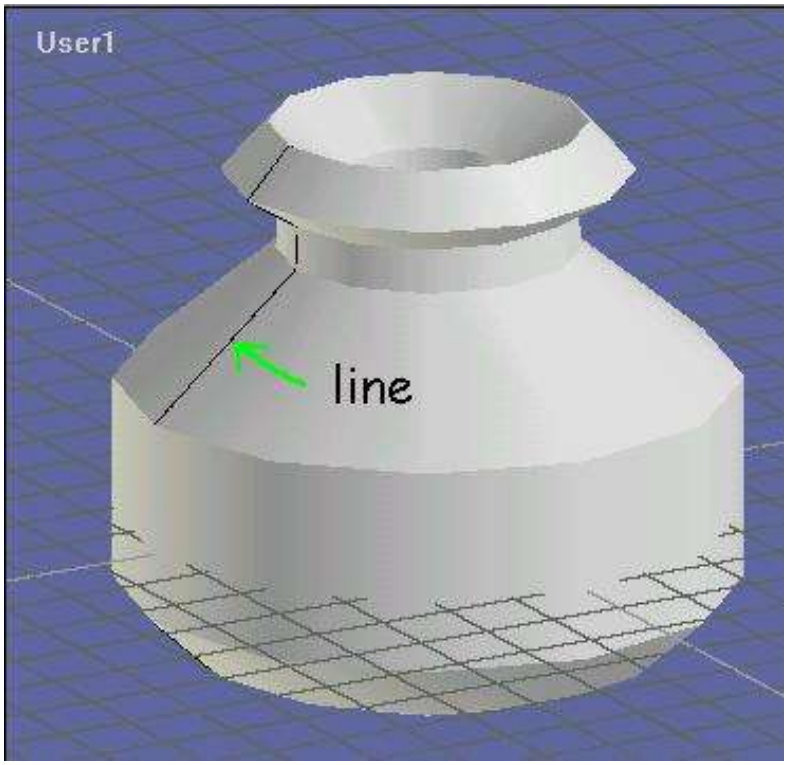




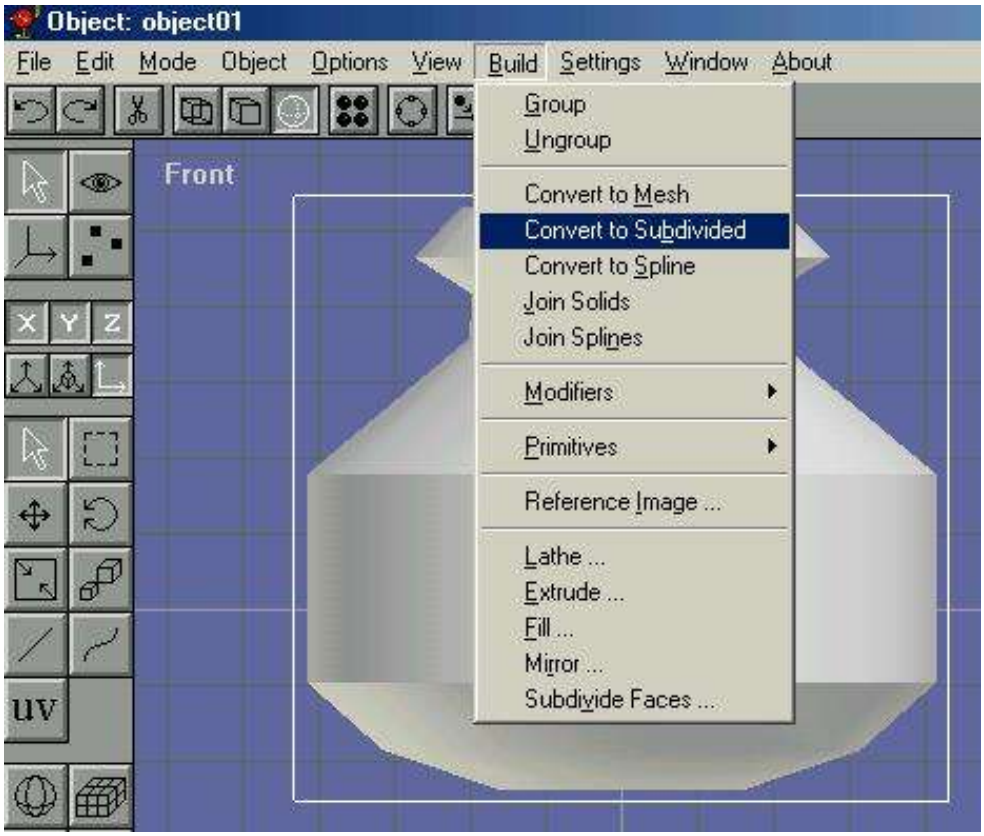
You now have a few choices. How many sections do you want the pot to have? The more sections you enter here, the better your pot will look, but the bigger the file will be. In this case ten or twelve sections are enough, because we are going to smoothen the pot later on. your pot should now look something like this.



Now press 'Ctrl R' or click the button at the top of your screen. As you can see a green circle has appeared on your screen. If you press down the left mouse button in the circle, you can drag the cursor to rotate your view up-down and left-right. But if you want to roll your view, you must start dragging outside the circle. As you may notice, the grey letters in the left upper corner saying "front" or "side" will be replaced by "User1" or "User2" or something like that. That is because now you are not viewing your pot from the front or side any more. To zoom in and out, use the middle mouse button (or press alt and use the right mouse button). To scroll up, down, left and right, you can use the right mouse button.
When you've rotated a bit, you might have noticed a black line.



That is the spline we started with. Select the spline (it will turn white) and delete it. You won't need it any more. If you've accidently deleted your pot instead of the spline, don't worry. Go to Edit->Undo (or use Ctrl Z).
Now we will go to the Front view again. Move your mouse cursor to the 'User1' in the left top corner of your screen, and click. You will see a list of views you can select. Click on Front. Now you're back in the Front-view.



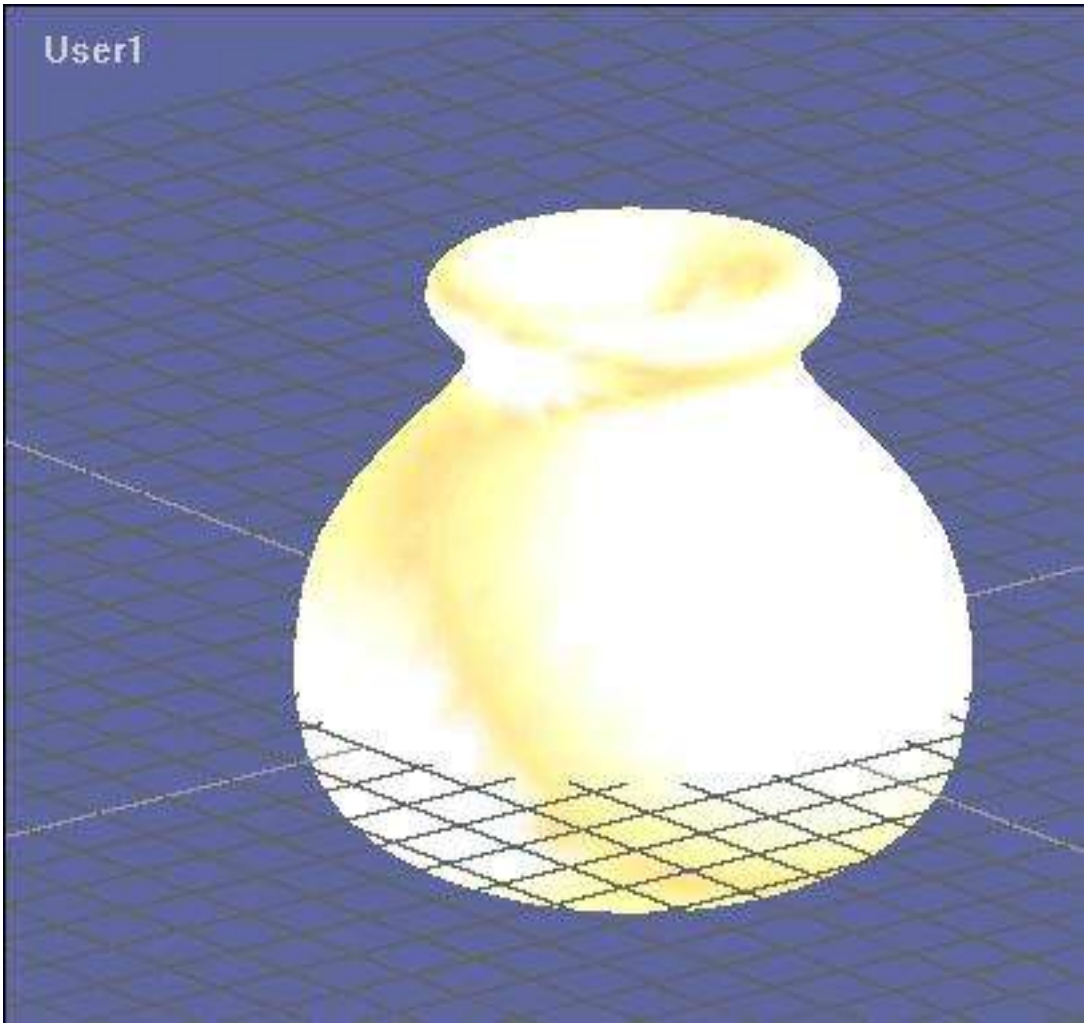
Now we will smoothen up the pot a little bit. Select the pot. Click on Build->Convert to subdivided.

Don't use Subdivide faces, because this will make your pot unnecessarily complex. Your pot should look something like this.

Now let's make a material. We dont want an old silly snow-white pot, but something better.

Click on the button. 

a list will appear on the left side of your screen. Doubleclick on the [New] button, and you will see something like this: You can fill in how shiny the material must be, and what color. Set it to an orange-like color, and click OK. then select the pot, and click [apply].



Game Design Master Class

Simon Donkers

All pictures are taken by Frans Peeters,

On the 9th of June 2004 professor Mark Overmars and Arno Kamphuis gave a master class Game Design at the University of Utrecht. Within this story you will find some more information about the master class. Also you will find some exclusive information about GM 6.0. About 40 students visited this master class and also several computer science teachers accompanied them. The day consisted about several presentations by both Mark Overmars and Arno Kamphuis and also a practical part where you can design your own computer game. On the button of this page you will see some pictures taken during the day.

The master class was given mainly to get people thinking about Game Design. Within the master class Arno Kamphuis gave at the beginning a presentation about the history of computer games. Also he gave some information about games which where successful. After that Mark Overmars gave a presentation about what makes a successful game. In this also some successful games where discussed. Within this he also asked at the students to name up a few good games in their opinion and it was surprisingly how much Mark Overmars knew about those games. After that we had 3 hours the time to create our own game keeping the game design points in our heads and with advice of Mark Overmars himself. During this Mark Overmars also showed us Game Maker 6.0. He showed what new things changed within the interface. What new things where added, which things where changed. Some exclusive information about Game Maker 6.0 can be found below within the article . During this all we also had some time to ask Mark Overmars some questions about Game Maker. Such as why there is no 3D game maker. Why he is working on Game Maker. How Game Maker is built up. More below.

The master class started with an in-depth story about the history of computer games. Starting his presentation with the question: what was the very first game (Space Invaders) and then going up on the chronological ladder with Pong (first commercially published game) and so on until the present. The story was basically a summary of different games which where brought out and different consoles. I am not going to repeat the whole story here but a simple google search found me this article which seems to be about the same: You can find this at . But also he added a few interesting things such as, in which year has Nintendo started: (a)1928 (b)1889 (c)1985 of (d) 1824. And a picture of the first person to make the ultimate highscore within Pacman.

After this very interesting history lesson Mark Overmars started telling about what makes a good game. The basis of his story can be read in the tutorial by Mark Overmars available from the Game Maker website. I highly recommend reading through that tutorial because the presentation was pretty much the same story. An interesting thing was that halfway of the presentation he divided the students in groups of 2 each and game them an amount of small colored blocks and a few dices and a piece of paper with the goal to create a game. A few examples he gave for games

where: try to make a tower of the colored blocks which is as high as possible and which can carry a dice on top. An other idea he gave was to make a game like paper, stone and scissors. Red wins from blue, blue wins from green and green wins from red. Every player gets 4 stones of each color. Now both players take one stone out and show it to the other. The first round is random however with the second round you know which stones are removed from the game so with some good tactics you can now predict which stone is the most likely to be pulled by the opponent. Now it was the job to think up a good game yourself with just a few colored stones and some dices. This gave some very interesting results. I believe Mark Overmars was planning to put a few of these games online.

The afternoon of the master class was reserved for the actual programming of a game. For this we used Game Maker 5.3. Because some people had never worked with Game Maker before most people where busy using the which can be found on the Game Maker main page. Also during these hours we had some time to talk to Mark Overmars about Game Maker and Game Maker 6.0. Mark Overmars showed us Game Maker 6.0. The first change you see is that the image of the button add sprite is changed. Within this alpha version the image was Pacman . A more important change was that in the data tree you will no longer have datafiles. However you now do have font resources. Within this resource you can select a font from your system, select the size, color, bold, italic..., and all those things. Then when you create an executable the font is added with the given settings.

An other change is that within the Game Options (which changed name within this alpha release) you have a new tab page to add data-files. Also you have a few more options for selecting the screen resolution and how to handle screen resize. Also within the sound resource you now no longer need to select the number of buffers. The program does this automatically. You can select some nice effects such as the sound panning, and also add special effects though the sound file such as echo and a few more.

Also Mark has shown us what the GML code was for the available from the Game Maker website. The code itself I guess was about a 100 lines of code for the drawing. Unfortunately I only had a glimpse look over the code but it does appear quite difficult to use the new 3D functions. Mark Overmars also was so kind to explain why there is no 3D game maker. To make a 3D version of Game Maker will require some complex coding for the engine but he is capable of doing so he said. However to use the program there are quite a few things which the game designer, meaning you, should do. For instance you should make the models, you need to arrange for the lightning, you need to arrange for the settings of the camera motion, you need to define how to do collision event…. Basically the programmer needs to define a lot more things, which are game dependent so they can not be preconfigured. Currently when you add an object to the game then you do not need to do much for it. If you add a collision event then it will be executed nicely.

However to handle collision events in 3D requires a lot more calculations. So a simple collision event will drastically decrease your game speed. So you need to define the accuracy of the collision, for which collision this event applies and a lot more things such as this. This will make Game Maker much harder to use and as Mark pointed out he is trying to make with Game Maker a user-friendly program which is easy to use for the end user and which does not require much learning to use while it is still capable of making a variety of different things. So no 3D Game Maker. Currently the functions added do not support for things like dynamic lightning and collision detection so to make a 3D game you will require a lot of hard work.

Also a lot of people claim Game Maker to be slow. Mark Overmars gave us a very simple example. Game Maker can handle about 1.000.000 lines of GML a second (This is an average. It depends on the functions you use and the machine you got). Now a game runs with a room speed of 30. So you still got a speed left of 33.000 lines of code per step. If you got a ball with 100 lines of code in the step event and you got 500 of those balls bouncing through the game you already got 50.000 lines of code. So the game will slow down. However if you program your game good then it is very well possible to create a smooth running game within Game Maker.

At the end of the day Arno Kamphuis gave a presentation on what can be expected within the future for games. Within this presentation he spend a lot of time about the high levels of the current consoles and also did a look at the X-Box 2, PS 3 and the new N-Gage and the Game Boy DS. He mainly focused on the speed increase and the possibilities these consoles will give game developers. Also he spend some time on the new video card and he shown some very nice smooth very detailed animations which can be made by any PC with a N Videa G-Force 5 video card. Also he gave some information about the way a video card works. How does polygon and pixel shading work and what will be the future.

Overall it was a very good learning experience and a very enjoyable day. I have learned a lot about Game Design and Game Maker 6.0. Perhaps I will write a few more stories about Game Maker 6.0 based upon the things I have seen and have talked about with Mark Overmars but that is for another day. Below you will find a few pictures taken at the master class. I am the guy in the brown t-shirt. Also have a look at Mark's very cool laptop, which you see me using because there weren't enough PC's. Also notice the very cool ability to have the ability to turn the screen 180 degrees and close it so you can have it as a paper with touch screen. . P.S. Nintendo was started in 1889.

New Tools New Tricks

Mike Swanson

Artists working in the video game industry have seen many changes over the last few years. One of the most noticeable of these was the large-scale transition from 2D to 3D about six years ago. Fueling further change more recently are high-powered consoles and PCs that can handle procedures that were once available only to film-effects professionals. Cloth simulations, shaders, and complex facial animation rigs were not to be found in most games two years ago—now they are common. Subsequent console generations will see more simulations, such as muscles, hair, volumetric effects, depth of field, logical motion blur, and cloth with collision detection.

Several factors are driving the technological changes forward so quickly. Software companies are developing middleware for game development that allows artists to view and modify art assets in game engines without having an internal engineer develop specific tools to do so. Developers of 3D and 2D software now offer game-specific tools within their packages. Off-the-shelf programs now exist for the creation of character musculature, automatic level-of-detail foliage, markerless facial motion capture, and accurate automated voice sync. Colleges and universities are now teaching game development within their curriculums, and their graduating students should theoretically be plug and play within the game industry.

Gone are many of the limitations that once prevented game artists from realizing their full visions. But gone too is the usefulness of many past techniques. For example, the smaller game company artist will now need to become a generalist and learn many more programs and techniques in order to accommodate the rise in technology. And the larger game companies will likely have to create specialized art departments that mirror those of large film effects companies.

An example that follows a film model might be a game character that is first created by concept artists. The character is then passed on to a modeler, then to a character rigger, then to an animator, and finally ends with an art technician whose task is to integrate the character with the game. The pipeline involved in getting that character moved through all these different stations will necessitate specialized tools written by a technical director. Many of these pipeline roles did not exist even two years ago within game studios.

Taking it to the next level

As we master the next generation of consoles and PCs, the visual style will be elevated several levels, and the public perception will be that all games should have the same high-quality look. The games that will succeed will be the ones that stand out visually with compelling, entertaining storylines and puzzles to solve. In order to facilitate making more complex games that will touch the emotions of the player, game artists will be required to do more rapid prototyping of work—for instance games created in animatic form before actual production starts with full-scale storyboards, color charts, emotional graphs, and so forth.

The background of game artists is changing as well. Film artists are becoming increasingly common within the gaming industry—animators, TDs, and modelers are crossing the digital divide and cross-pollinating with seasoned game artists. This crossover from films to games is allowing for the rapid exchange of ideas, pipelines, and the creation of new art authoring technology. While this is good for game development, many of the older artists are feeling pressured to keep up with the film artist and the younger generations who have lived entire lives with computers and art creation software.

The competition is fierce; and students coming out of school with the latest software packages under their belts, as well as professionals who are making the leap from the film industry to the game industry, are setting new expectations for artists in the gaming field. In order to meet these challenges, today's game artists must be willing to look ahead. They must be ready to accept the new technologies and techniques that will not only help them accomplish their tasks more efficiently, but make them more valuable workers in a changing marketplace. ..

Indie Opportunities

Phil LoPiccolo

Contrary to popular belief, the independent game-developer community is not made up entirely of nerdy teenage boys who spend more time stealing other people's code than creating their own. A comprehensive new survey from Acacia Research Group (www.acaciarg.com), called "The Amateur/Independent Game Development Tools Market," shows that while this group contains its share of absolute beginners, it is made up largely of professional adults who are well educated, experienced, and willing to devote serious time and money to develop games for a rapidly expanding and diversifying market.

To conduct the survey, Acacia polled independent game developers on the Web sites they tend to frequent, including gamedev.net. The results showed that the majority were males over the age of 21, that nearly three-quarters had attended "institutes of higher learning," and that a third had game-development industry experience. The data also revealed that approximately half the respondents spend between 10 and 30 hours per week on game projects that earn 20 percent of this group between $500 and $50,000 per year and another 5 percent between $50,000 and $100,000 per year.

When it comes to paying for development tools, more than 90 percent said they purchased software over the past five years, and nearly 40 percent spent from $500 to more than $5000. However, on the downside, nearly 30 percent said they pirate development programs, and 25 percent admitted to stealing more than $5000 worth of such tools.

How do they justify stealing? Some say it is excusable because they're not making money from their work. In fact, those who make the least steal the most. Other pirates feel it's acceptable to use stolen software when the vast majority of features in a program go unused. For example, stealing Photoshop for pixel art is fine, some say, because buying such an expensive package wouldn't make sense for such limited use.

Despite the lost revenues from pirating, Acacia's five-year projections show total sales of development tools—including both content and coding software—to the independent game-development market will rise steadily from roughly $30 million this year to more than $55 million by 2008, a compound annual growth rate of 16 percent. Several factors are contributing to the rise of the independent game-development market. First, the Internet has enabled information exchange between developers and provided a ready-made production pipeline for even the most under-funded projects. Second, there has been steady growth in the tools for creating game content—and these have become more accessible to non-professionals, both in terms of price and usability. Third, and perhaps most significant, there are more opportunities to develop content for a greater variety of games than ever before, even for the emerging "serious games" now being designed for education and training purposes.

Independent game developers would do well to seek out these new opportunities, as well as use the new development and distribution tools available to them. And tool vendors would be wise to look beyond the tapped-out high end of the market for new customers and meet the needs of this emerging market by providing users with relevant tool sets at affordable prices.

Make Games...Make Money?

GMDM Feature

Greggman

Do you know what it really takes to make a video game? Do you know why a game costs $60 or even $80. Making a game takes a ton of work.

Sometimes I think I should try to teach a class in it in high school. Creating a game might be something that some students might want to do and having a class about it would allow them to experience what it really takes to make a game. It's not just about having fun, it's about lots and lots of work. It's about reports, schedules, budgets, tradeoffs, teamwork. All the things I wasn't taught in high school or college for that matter.

Pitching

Most people think that a game starts when someone has a great idea for a game. The problem is that almost everyone in the industry and every game player thinks they have a great idea for a game. Someone has to be convinced that your idea is the one idea that should get done. This is done by pitching the game whether you're an internal team (a team that is internal to the game publisher) or an external development team (a team not owned by any publishing company but that does products on contract for a publisher.)

To pitch a game you have to create pitch materials. The better your materials the better your chance of getting your game approved. Usually the minimum materials are a small report 2 or 3 pages describing the game briefly. If you can't describe the game briefly then you are unlikely to be able to keep the attention of the people you are trying to sell the game to. Most of them are not game players. Another common pitch material is the storyboard. Storyboards attempt to show the game with pictures. Good looking storyboards definitely make an impression over those teams that don't have them. Even better than storyboards is an actual demo of the game.

Time and Money

What many people don't realize is that the game they pitch must be able to be done in a certain amount of time within a certain budget. Lets say you wanted to make a 3D Fighting game with 20 different characters and 1 background for each character. An intro video introducing the game and a video ending for each character when that character wins the game. How much time and how many people is that going to require.
Lets guess that each character will take 1 month to create in 3D and animate. Each background also takes 1 month. The 3D programming we guess will take 1 year. The intro video will take 4 months and each ending video will take 1 month. Add it up.

·1 month * 20 characters = 20 months
·1 month * 20 backgrounds = 20 months
·1 year of programming = 12 months
·1 intro video = 4 months
·1 ending video * 20 characters = 20 months

That's 20+20+12+4+20 = 76 months. In other words, if one person could do all the work by themselves it would take them 6 years to make the game. Of course 6 years is too long to take to make a game. If you started today, in 6 years the video game systems that people have at home would probably have been replaced. Instead of a Sony Playstation they'd have a Sony Playstation 2 or maybe even 3 and your game would have no market anymore.

Most publishers would like a game to take about 1 year. So if you wanted to get your game done in a year you're going to need at least 7 people. 76 months / 7 = 10.8 months or almost 1 year.

How much do 7 people cost for a year? Well an artist can cost anywhere from $30,000 a year to $100,000 a year depending on their experience. A programmer from $40,000 to $100,000. Lets just guess and assume you get 6 artists for $45,000 each and one programmer for $65,000. That's $45,000*6 + $65,000 = $335,000. But wait, people need benefits like health insurance, they need supplies like paper and pencils. They need a place to work like an office with a desk, a phone and a chair. You also have to pay certain taxes in addition to the taxes that each person on the team pays. All that adds up to around 30% of their salary. So, $335,000*30% = $100,500. Your total cost is now $335,000+$100,500 = $440,000.

Okay, now you need equipment and software. Each artist and programmer needs at least one computer. A reasonable computer with monitor will cost at least $3000. Artists need software and 3D software can be very expensive. Lets say you decide to use 3D Studio Max. That's $3500. They may need a copy of Photoshop or some other painting software which is about $600. Your programmer will need a an editor $200, and a development system, $30,000. So the total for equipment so far is
·7 machines * $3000 = $21000
·6 3D Studio Max * $3500 = $21000
·6 Photoshop * $600 = $3600
·1 development system = $30000
·21000+21000+3600+30000 = $75600
Total so far, $516,500.

Now lets say you ask a publisher for $516,500 and they agree to give it to you to make the game. What did you forget? Well some things that come to mind, music and sound effects for one. Also, your schedule probably didn't take into account all the communication that needs to go on between team members so they are all working as a team. Do you need someone to lead the team? Do you need a art director to organize the artists and make sure that all the artwork in the game has a consistent look? You could ask one of your 6 artists to do it but then they will be busy managing the other artists and won't have as much time to get their work done. Who is going to pay the bills, do the payroll, order the equipment and software. Whoever does it will have less time for working on the game. What about a network? Are you going to have a network so that people can share there work with each other without having to use lots of floppy disks?

Lets add one more artist as art director and because they are the art director they command a higher salary of $60,000. You also hire a producer or manager to both organize the team and pay the bills and manage the other money matters. (Maybe you don't like the idea of hiring a manager and instead you want to manage. Now your time is taken up by managing so you are going to need to hire someone else to do the work you no longer have time for. Either way it's going to require another person). You need to contract out for music and sound effects. That can easily cost $60,000 to $80,000.

Lets add that in.
·1 art director = $60,000 + 30% for rent, insurance, taxes, supplies, ... = $78,000
·1 producer = $40,000 + 30% for overhead = $52000
·Music and sound fx = $70,000
·2 more machines = $3000 * 2 = $6000
·1 more 3D Studio Max = $3500
·1 peer to peer network = $4000

New total = $4000+3500+6000+70000+52000+78000+516500 = $730,000

Lets say you ask for $730,000 from a publisher and they give it to you. You now have enough money to pay your team for exactly one year and no more. If you forgot something tough luck. If it takes 16 months instead of 12 you're going to go hungry for 4 of those months or your going to have to re-negotiate with your publisher and they are going to want something in return for your failure to deliver your game within the time and budget you originally promised. They might for example lower your royalties or they might demand a part of your company. They might ask you all to take a 50% pay cut until you finish.

Lets take a look at royalties. Most games by external developers are done on an advance against royalties arrangement. That means that the $730,000 they gave you is an advance against your royalties. Maybe you got 15% royalties and the game sells for a suggested list price of $49.95. You don't get 15% of $49.95. You get 15% of net so if the list price is $49.95 the wholesale price is probably 45% of that or $22.48. If this is a Sony Playstation or Sega Saturn game they both charge around $8.00 per disc sold as a licensing fee so the net price is $14.48. 15% of that is $2.17. Your team gets $2.17 per unit sold. You got an advance of $730,000. $730,000 / $2.17 = 336,406 units. You must sell 336,406 units before your team will see any more money than they already got. Not very many games sell 336,406 units. Maybe only the top 10 games on any platform.

Another issue that comes up here is the feeling that the publisher is being greedy. The typical point of view of the developer, you, is that you are going to do all the work and they are getting 85% of that $14.48. You feel like you should get more. I know I often felt this way. Here's the other point of view. From the view of the publisher they put in $730,000 and probably several $100,000 more on marketing and plus they also need to pay sales people and marketing people and producers etc. Lets say they spent a total of $1,500,000 on your game. What have you spent? You've spent $0. They are risking $1.5 million dollars on you. If you or your team fails they are out $1.5 million dollars. On the other hand you risk nothing. If you fail you already got from them $730,000 dollars. That hardly seems fair. The reason they get all the money is that they are the people taking all the risk. That actually brings up another point, if you want a better deal, lower their risk. For example if you develop the game entirely on your own and then once it's finished you go to them and they decide to publish it you can usually get a much better deal. The reason is that they don't have to risk as much money. Of course they still have to risk all the money they will spend of advertising and duplication and distribution and sales. Unfortunately most people can't make a product on their own. It takes too long and too many people.

Design

Design is going to be different for different types of games. For the type of game I like to make, action games or action adventure games, I personally believe the best way to design is by storyboard and sketches. I've seen teams make huge documents 300 to 400 pages long for their games and I personally don't think it works. Nobody wants to read a 300 document. Instead you probably need some kind of outline just so you can make sure you've got everything listed. Then you need to design each world and each character and each object. Each item will need two basic things, a visual design and a behavioral design. The visual design would be designed by the artists. This is one way to get your artists involved in the game. Give them a basic idea of what you want to do with the game and then give them a couple of days to go off and sketch settings or characters or objects. Then have a big meeting and decide which of their ideas you want to actually use in the game. Once you've chosen, have the artists make much more detailed color version of those items. You see this type of thing in movie production. The #1 reason you need this is you need to make sure everybody understands what everybody is trying to make and a visual picture is your blueprint. In other words, "make what you see in the picture".

The perfect example of this is the original Star Wars. If you look in to the making of Star Wars you will see lots of paintings by a guy named Ralph McQuarrie. I used to think those painting were made after the movies since they looks so close to scenes in the movies but actually the opposite is true. Mr. McQuarrie drew those paintings and then from those paintings people made the movie. If you think about it you can see why this is so important. It takes lots of people to make movies and video games and without images like these nobody will have the same idea for how to make their particular piece of a level or scene.

Secondly you need behavioral design. This is best done with sketches. In the movies this would be the black and white sketches that show each scene and camera angle. In a game these would be sketches that show each item and character and all their moves and behaviors with notes giving details for things like timing, speed, distance, power etc. While working in Zombie Revenge at Sega I saw hundreds of these sketches. Every motion needed for every character had a sketch describing the motion BEFORE it was created.

Levels also need to be sketched. These should look like blueprints or top down maps that show where each item/door/character etc should be. Having worked both ways I personally believe levels should be laid out on paper by game designers and THEN those designs should be handed off to artists so the artist can build the level based on the game designers blue print. This lets the game designer make sure the level is designed to be fun, fair, not frustrating, etc and lets the artist make it look beautiful. Some of you are going to think you can just jump into a map editor or 3d program and start creating a level. I could happen but I've never seen a really good level come out this way on time. The problem is without a blueprint you have no idea where you are going or when you're done. You'll just keep noodling and noodling until you get bored and start working on something else. If you have a blueprint you'll have a specific goal in mind. You'll know when you are finished and when you are not. You'll know what other things need to be created for the level before the level is even built.

The successful companies where great products are made on time the designer is at the top of the heap. From the designers the game is made. That means they must be good people capable of leading, of creating designs that are possible, of not creating frivolous un-thought-through designs that the team implements and then have to be thrown away. They need to be aware that from their designs, thousands of dollars will be spent implementing them and that bad decisions from them will cost lots of money and possibly the entire project.

Programming

Programming is getting both harder and easier than ever. Easier because today's systems are powerful enough that you can buy an engine for them. Rendeware, Alchemy, etc. Harder because now there are so many more parts then their used to be. Physics, shaders, networking, hard drives, encryption, multiplayer modes, rumble paks, gameboy advanced options, co-processors, AI, etc etc etc. It used to be that a couple of programmers could do the entire game. Now if you look at the job listings each position is specialized. Server side network programmer, client side network programmer, physics programmer, sound programmer, sound tools programmer, 3d engine programmer, game programmer, AI programmer, 3D tools programmer, content management programmer.

Art

Art creation for games has totally changed over the years as well. When I started in games in the late 70s, early 80s there was no Photoshop or Maya. To get graphics into a game we would sketch the graphics on graph paper and then write down all the positions of the pixels on the paper and type that information into the computer by hand. Now we have tools like Photoshop for 2D graphics, Premiere, After Effects and Vegas for video production, Maya and 3D Studio Max for 3D graphics, digital cameras, scanners, etc.

At the same time, Pac-Man was 16x16 pixels and would take only a few minutes to make today but today's games have levels consisting of thousands to millions of polygons. A 3D character is 2000 to 20000 polygons and has a bone system from 15 to 80 bones. Each vertex on the character has 1 to 4 hand set weights for how much each bone influences it and the bones often have to have constraint systems or functions applied to them so they don't bend the wrong way. Textures have to be drawn to cover a character, often separate textures for different parts. And, as opposed to just a couple of years ago when we only had to make a color texture, we now have to make bump map textures, reflection map textures, glow textures, and a host of others.

Making a character like Pac-Man took just a few minutes including animation. Making a character like Solid Snake takes months. A few days to make the model, then all the textures, setting up the bones, connecting and weighting all the vertices and then animating. Pac-Man just opened and closed his mouth. Maybe 4 frames total. Today's characters have thousands of frames of animation and even though the computer tweens for us we demand so many moves that it takes months to make them all.

And that's just the in game characters. We need artists for the levels. Artists for all the glue, the title, our logo, the score bar or hud, particle effects. We have video scenes in most games today that often require a whole separate art team.

Another issue is that for some systems the artists need to create LODs (Level Of Detail) models. In other words they might create a character out of 3000 polygons for looking at when she's up close to the camera and another out of 300 polygons when she's far away and only 1 inch high on your screen. This saves processing time and allows the game to run smoothly but requires the artists to make more models and textures. There also MIPs which are textures that are smaller in resolution used when an object is being drawn far from the camera. Without them things would be slow and also flicker a lot. Often they are generated automatically but sometimes they need to be hand edited if the auto-generation doesn't do a good job or if you want special effects.

On top of that, many systems require the artists or designers to make separate models for collisions for the entire level. These models are similar to the 3D models used in the level but are much simpler with no details and no textures. Still, it's lots of work and often they have to be tagged with all kinds of attributes like "this thing is deadly" or "this thing is sticky". And it's not just collision geometry marking what's solid. Trigger boxes need to be added that mark things like when the player steps inside this area have the boss appear or start a dialog or cutscene or blowup something. Others might be there only to help direct the camera.

And then, often the AI in the system requires lots of data that we ask the artists to put in. In the simplest case it's just to put an object in the 3D program to mark where a game object will start but it can go up from there from having to draw the paths the objects will follow or having to add special AI geometry that marks where an AI character can walk, where he can't, where he should jump or step down, etc.

Just like programming art has started to get specialized. In the past a couple of artists did it all. Now we have specialty artists. Storyboard artists, 3D modeling artists, texture artists, lighting artists, animators, 2d artists. I'm sure like the movies we may even have camera artists soon.With all that art some companies have even gotten to the point that they have special software to try to keep track of it all.

And it's only going to get worse with the next generation. Today to make a belt buckle on a character an artist just draws it into the texture by hand. On the next generation games, for example Doom 3, the artist actually models the belt buckle in 3D, textures it with a metal like texture and then tools take over and extract a color texture and a normal map so that in the game it still looks like a 3D belt buckle complete with highlights, reflections and shadows.

Another thing is you are not going to be able to just hire any old art student or buddy that draws cool pictures of anime characters. Making good art for games is not the same as making good paper art or even movie CG art. We've got serious limits. In a CG movie some artist might use procedural hair and skin for a character and might use a 1024x1024 texture just for the character's eyeball. They might use nurbs or subdivision surfaces, something the current consoles are not really up to. They might have a million polygons in their model and all 202 human bones were as a game might have 3000 polygons and 15-30 bones and a limit of 1 256x256 texture for an entire character.

Finding artists that can make good stuff with those limits is very difficult. As an example of some artists that can, the Gran Turismo 3 artists and the Metal Gear Solid 2 and 3 artists. All of those games have relatively low-tech graphics engines but the artists are so good those games look incredible.

Sound and Music
In the past sound and music in games were pretty much a last minute thing. Once the game got 1 or 2 months from shipping the company would hire a sound or music person or both to quickly fill the game with beeps, buzzes and background music. As games get bigger and bigger this is no longer the case. Now, many games have almost full orchestral scores and thousands of sounds and we still have extremely limited memory so getting all those sounds and music to fit requires more than just musical skill.

To give you an example my friend on the Jak II team said for one language they had 4000 files of dialog! They did 9 languages so that's 36,000 files!!!! Someone has to go through all 36,000 and make sure they are all correct, all finished, all in place. And that is just the dialog. They still had music and sound effects.

And, speaking of memory, there is never enough room so you'll need programmers to make up systems to get you all those sounds. In Gex, Gex has like 400 quips or more. I don't remember how long the actual list was but we only had memory for 1 quip at a time and so I had to write a system that would try to load a relevant quip, while we were spooling music off the CD at the same time. You can not play the sound until it's loaded and a relevant quip is one that Gex would say when he attacks a particular enemy or sees one or gets hit by one. In other words, you can't wait until Gex does his move and then load the sound and play it. It might be 1 or 2 seconds before that sound is read to play.

Another example. In Crash Team Racing there were tons of quips as well. Those quips were played directly off the CD as CD-XA audio so the sound programmer had to make a system to put all those sounds on the CD and queue them up so they could be played quickly when they were needed.

For a while in the mid 90s musicians thought Redbook audio, playing music directly off the CD, was going to be the end all be all for game music. They could use all the tools at their disposal to make the highest production value music ever. The industry quickly figured out that red-book or streamed music only fits a very small, very limited set of games. For one that's because that kind of music takes time to switch and time to queue up so it's not easy to change the music instantly during game play in response to a power up or to the mood going from safe to dangerous. For another many of today's games need to spool data off the disc while the game is playing.

That's impossible if you are using Redbook audio and can be problematic if you are using streamed music. So, most games use a some kind of structured music format. The problem or issue is that just like it's hard to find a good artist that can make good art for your game within the limits of the game system, it's just as hard to find a good musician that can make good music within the limits of your music system. They might get only 512K or 1Meg in which they have to fit all the instruments for the music that will be played during the current stage AND all the sound effects used in that level. Try taking a look at the average size of a and you'll see 512K is not very much memory.

And all the rest

What else can I add in here. Well, there's a month or 2 of bug testing. If you are doing an international game you will have people testing in all languages from all over the world and you will need to give them each a version. Many of you probably have DVD burners so you are probably aware that making a DVD takes time as just transferring 5 gigs of data over your network to the machines with the burners.

Having versions for playtesting as well. Playtesting is testing to see if the game is playable. Maybe you test it and find that no one can figure out how to open that door or that the green key is under the 3rd box on the left and so you have to go back and redesign your game so that people don't get stuck and frustrated.

There's also often a huge hit to your schedule for things like a version to take to E3 or other industry trade shows. You might need to make versions for the press so they can have their reviews appear the same time your game hits the store shelves.

Now-a-days you might also have to deal with pre-piracy, having the press for example, leak your game to the internet. Having all that work you see above taken for free by some thieves and so you have to spend more work trying to prevent them.

I'm sure I've forgotten quite a few things but hopefully this gives you some idea of why games take lots of people, years to make, and millions of dollars.

Game Reviews

Reviews by John Hempstead and GM Showcase!

Age of Farming

score 3.5/10

Rodrigo Uribe

I am a big fan of simulation games and looked forward to playing this. After starting it up I realized I should have taken a second language, since I only know English….I think, and the game is in Spanish.

As for the game, there needs to be some more audio elements such as background ambient music, atmospheric sounds, and more interaction sounds for the buttons.

There was no help for the game but I just played around with it to see what can be done. You can build a barn, crops, and a watering tower. There may be more but I was not that interested in continuing the game.

The graphics are simple and easy to see. The angle of the barn and water tower when placed on the fields do not look as if they are in the correct perspective.

I would honestly say this is a good start to a game that can be pretty good with some work. Good job and I would say try this game.

Alien Takeover

score 5.5/10

Urisoft

Ten years ago, when the first unidentified radio signal came to one of our spaceships, we finally understood- we're not alone in the universe!...So that's the reason the people of earth are throwing sticks at me!

When I started the game up I was a little impressed with the title screen and menu options. After I started the game I was not that happy with it. But after completing a few levels I kept playing! It is actually pretty fun.

Not the best graphics and sound but the game is a little fun packed with many progressive levels! Try this one.

Defender 3: The Legacy

score 7/10

LEef

If you like the Defender games you will enjoy this one. The graphics are great, the particles bring nice special effects to the screen, and the sound is classic!

The game play is smooth and easy to navigate. I say try it and defend!

DOWNLOAD

Doom 2d: Knee Deep In The Dead

score 4.5/10

Smiley

This sidescroller is not that bad and not great. The graphics are ok and the user interface could be improved. It's a pretty fun game with all of the crazy beast and weapons. Give it a shot!

DOWNLOAD

Duke Nukem: Ready For Action! L.A. Meltdown

score 5.5/10

Smiley

If you are a fan of Duke then try this game out!

The graphics are pretty good with exception to the yellow bullets, they don't match the detail of the game. The sounds are basic and more ambient sounds could improve the overall mood of the game.

Again, give this game a shot!

Guitar Trainer

score 7.5/10

Creator: Yabuti

I now can wipe the dust off of my guitar and join Radiohead! This program is very cool!

It's a great way to learn to play the guitar, it's simple to use, easy to read, sounds great, and has fun things to keep you going like a quiz!

If you have a guitar and it's not tuned up, don't worry there is a tuner guide to help!

If you want to learn the guitar I recommend you get this!

Return to the Home Planet

score 4/10

Creator: 875-Company

An interesting maze like game. The graphics are simple and could be improved on. The game lacks music and sound which made me loose my interest in finding my way home.

The game play is a little awkward but it's easy to get use to. I would say try this out if you enjoy maze games.
this perhaps my favourite GM game ever.

DOWNLOAD

Shortline Express

score 7/10

Creator: Ron Coffen

Here you build a railroad system that safely guides trains and their payloads to the correct destination terminal, earn as much cash as you can, and last as long as you can. And I forgot, have a lot of fun doing it!

This game is pretty good! A Great intro screen that transitions into a nice menu. The graphics and sound are good too. The game play is easy to use. I recommend this game for tycoon lovers!

DOWNLOAD

3d Engines and You!

By John “projectdotl2004_2005” Aljets

With all the hype surrounding such engines as *Xtreme3D* and *Kingspace*, you are usually required to get a good engine to have a good 3D game made with *Game Maker*. I have spent many long days and nights searching for an engine that could quench my thirst for professional-quality games without learning C++ or DarkBASIC. I have yet to find such an engine. But throughout my quest for greatness among 3D developers, I have found many high-quality engines that managed to keep me busy for a while. Several are ones that you probably already know and use yet some are hard to find or not even finished yet! Here is the rundown on the best engines that I could find:

1.Xtreme 3D- It’s like an orgy for your eyes. Developed by Xception, this engine has everything you would want. It only has one problem; it uses a .dll file to run the engine. Some people like .dll’s, I don’t. Features include mouse-look, fire effects, lens flare, and is compatible with .md2, .md3, and .3ds model types. Find it at http://home.tiscalinet.de/xception/.

2.GROM 3D- My baby. It is only in the development stages, but it already looks pretty good. I plan to use it in one of my upcoming games. I don’t know whether I’m going to release it or keep it for my self. 3.Kingspace- It looks like Gadget 3D (See below) except it uses a mouse-look and allows for crouching. Pretty nice fps and doesn’t use a .dll file. The only problem is that it can’t have textured walls or floors. One of the best engines out there. Find it at http://www.geocites.com/stephanboyer/s2/sc631.htm.

4.Gadget 3D- It takes me back to the days of *DOOM* or *Quake.* If you’re nostalgic for the old days then…don’t use this engine. It has many good features but if you want a good engine see below. Features include textured walls and 3d walls.

5.Gadget 3D Extreme- Here is the engine you should use if you want to make an old school-style First Person Shooter. It is a lot like the one above except it allows for jumping and the walls are easier to texture. You could make a FPS in half the time it would take you with the original version. You can find it and the original version at http://www.geocites.com/freegadgets.

6.GM Irrlicht- A port of the great C++ Engine *Irrlicht* for GM. I have to say that this is the best engine ever. It reminds me of the *Quake III* Engine. Maybe that because it uses .bsp and .md2 files. I know that the *Quake III* Engine didn’t use .md2 files but it did use .bsp files. It looks just like a professional engine. Xception has done it again. Find it at http://home.tiscalinet.de/xception/.


HELP!

Quick Scripts

Add some to your games!

Move the background!

Here you can move the background with your number pad keys!

```
{
// Fundamental 4 direcitonal movement
if(keyboard_check(vk_numpad9))
{
background_y[0] -= 2;
background_x[0] -= -2;
}
if(keyboard_check(vk_numpad7))
{
background_y[0] -= 2;
background_x[0] -= 2;
}
if(keyboard_check(vk_numpad1))
{
background_y[0] -= -2;
background_x[0] -= 2;
}
if(keyboard_check(vk_numpad3))
{
background_y[0] -= -2;
background_x[0] -= -2;
}
if(keyboard_check(vk_numpad8))
{
background_y[0] -= 2;
}

if(keyboard_check(vk_numpad2))
{
background_y[0] += 2;
}
if(keyboard_check(vk_numpad4))
{
background_x[0] -= 2;
}
if(keyboard_check(vk_numpad6))
{
background_x[0] += 2;
}
// Move using analog stick!
if(joystick_exists(1))
{
background_x[0] += joystick_xpos(1) * 4;
background_y[0] += joystick_ypos(1) * 4;
}
}
```

Isometric Grid Based Movement

Simon Donkers

```
//////////////// ISOMETRIC GRID BASED MOVEMENT ////////////////////
//
// Copyright Simon Donkers 6-8-04
// www.simondonkers.tk - simondonkers@hotmail.com
// the arguments are: behind is recommandable value
// argument0 is horizontal grid
// argument1 is vertcal grid
// argument2 is speed (Use a speed for which both argument0 devided
// by this as argument1 devided by this will give an integer number!!!
//
// This script will generate a correct grid based movement for an isometric game
// make sure to use the built in grid with arguments0 and 1 and also make sure
// that the character is alligned at the begin
//
////////////////////////////////////////////////////
if (argument0/argument2)mod(1)>0 then show_debug_message('Illigal speed used!');
if (argument1/argument2)mod(1)>0 then show_debug_message('Illigal speed used!');
if ((x)mod(argument0)=argument0/2 and (y)mod(argument1)=argument1/2) or ((x)mod(argument0)=0 and
(y)mod(argument1)=0) then
{
speed:=0;
image_speed:=0.2;
image_single:=0;
if keyboard_check(vk_left) and place_free(x-argument0/2,y-argument1/2) then
{
hspeed:=-2*argument2;
vspeed:=-1*argument2;
sprite_index:=spr_nw;
image_single:=-1;
}
if keyboard_check(vk_right) and place_free(x+argument0/2,y+argument1/2) then
{
hspeed:=+2*argument2;
vspeed:=+1*argument2;
sprite_index:=spr_se;
image_single:=-1;
}
if keyboard_check(vk_up) and place_free(x+argument0/2,y-argument1/2) then
{
hspeed:=+2*argument2;
vspeed:=-1*argument2;
sprite_index:=spr_ne;
image_single:=-1;
}
if keyboard_check(vk_down) and place_free(x-argument0/2,y+argument1/2) then
{
hspeed:=-2*argument2;
vspeed:=+1*argument2;
sprite_index:=spr_sw;
image_single:=-1;
}
}
```

Design Document

FPS Shooter

Here is a great Design Document!

Design History

This is version 1.20 of the document which began on January 15, 2004.

Version 1.10

For example, you can use Version 1.10, 1.20, etc.

1.Changed platforms for game.
2.Graphics are now 32 bit.

Version 1.20

The story has now been rewritten.

1.Details of story now changed.
2.Enemy character is now a humanoid model.

Game Overview

Type of Game This game is a 3D shooter with….

Game Ideas

This game has been in the process of creation for many years. The idea has been tossed around for the past 5 and has been revised many times. We felt that it was the proper time for such a release.

Location

The game takes place on an uninhabited island. You are surrounded by rocky ledges and water for as far as the eye can see.

Players

You will control the main character in the story from a First Person perspective. You will begin the game with a rifle and additional weapons are available throughout the level.

Main Objective

Your objective is to get off the island.

Game Overview This game takes a slightly different approach to the development of First Person Shooters in that instead of blasting your way through a level destroying everything in site, you must also figure out a way to get off the island.

Features

General Features

·Large terrains
·3D graphics
·32-bit color
·Several types of enemies

Multiplayer Features

·N/A
·Might implement in version 2.0

Editor

·No world editor at this time but planned for version 2.1.
·Free levels on the Net.

The Game World

Overview

An island that is uninhabited and without an obvious means of escape.

Key Locations

·There is a cave on the south side of the island that has health.
·The creek that runs throughout the island is the quickest way to travel.
·There is a boat at the bottom of the lagoon. A repair kit is also inside it.
Objects

·A boat for getting off the island.
·Food such as bananas that give you energy.
·Old health kits from a ship wreck.
·Weapons.

Graphics

Rendering / 3D Engine

The renderings will be 3D with polygonal models based on the Half-Life model format. It will be a FPS view without the ability to move the camera in any way. The game utilizes the TrueVision 3D engine which is based on DirectX 8.

Game Characters

Main Character

There is only a single main character in the game.

Enemies

There will be several enemies that you'll encounter:
Mummy
Dracula
Etc.

Weapons

Types

There are several types of weapons on the island including a rifle, handgun, machine gun and laser gun.

Musical and Sound Effects

Music

The game will use wav files created with ACID.

Sound Design

The sound effects include basic information about the weapons being fired and the foot steps when walking.

Appendix ABC

Any additional information…. Ideas include:

User InterfaceThe basic user interface will consist of 3 menus, …

Character Rendering and Animation

For characters, we plan to use MD2 files …

Start a game company

Yes you can!

Everyone who plays games ends up thinking about it -- wouldn't it be great if you could have your own game company where *your* designs came to life? Where you could be your own boss? Where you could show all those other pissant dumbass game developers how a *real* game is made?

Regardless of the motivation, you still need to get through the nuts and bolts of getting something going. Here are some notes from my own experiences.

I'm not a consultant, accountant, or lawyer, so don't take this as concrete advice, and obviously a lot of stuff is going to depend on your own situation and location (e.g. the laws in Europe or even California are quite different than the laws of Georgia).

Note: this is a "living" document. Please drop me an e-mail or post in our forums if you have a question, comment, etc. and I'll try to clarify and update this document as a resource for others as a result.

Getting Started
Before you run out to found the company of your dreams, you really have to sit down and ask the most basic question possible: what will your company do? All too often game developers start a company for nebulous reasons such as "to become independent", "to be my own boss", "to get rich", or "to work on cool stuff". Those aren't business plans, those are warm-and-fuzzy hopeful benefits.

You need a solid idea in your head of exactly what your company will be doing. Are you making small downloadable games? Are you looking for contract work? Are you looking for a publisher funded deal? Are you going to do a small independent massively multiplayer online game?

Your dream needs specific detail before it can actually become reality. Throw aside the hazy "it would be great if" goals, and concentrate on what you'll be doing once the ball gets rolling. What game are you going to write, and who is going to buy it, and who is going to fund it?

The Game
And here's where I waffle -- I can't tell you what kind of game to write, or who to write for, or what platforms. If you want to write a puzzle game for Linux or an arcade game for the Acorn, go ahead. No one knows what's going to be a huge success or a huge flop until someone tries. Throughout this article I'm going to assume that you know what you want to do, so I'm going to stick to detailing some of the background work required to take your idea and make it into something you can sell.
One thing I want to mention though: name your game carefully. There are a lot of people out there with a lot of opinions on how you should name something, but I just go by some general rules:
·make it unique to minimize trademark or copyright conflicts
·make it unique to elevate your chance of securing a first-level domain name
·make it memorable and easy to spell. Cute names with arcane spellings cause confusion for others trying to discuss your company.
·make it evocative so that it's obvious what you do or what kind of business you're in

Business Formation
You have an idea for a game, you've got some friends that are willing to work with you, so now you need to make a business entity. Why? Well, you *could* just make a DBA ("doing business as"), which makes the business synonymous with you personally, but that has quite a few accounting and legal drawbacks in the United States, so I'll assume that you're going to incorporate.

Incorporation
Incorporation can be a confusing process. In a nutshell incorporating creates a new legal entity that is the business, separate and distinct from you as an individual. The business can survive your departure or, hell, death. At least in theory. By forming one of the corporate variants (C-corp, S-corp, and LLC) you derive some potential taxation and legal benefits -- the exact benefits depend on you, and for more information on that I recommend you talk to a lawyer, accountant, or do a lot of research using a search engine or by reading one of the excellent books available from Nolo Press.

I've personally used an S-corp for everything. An LLC works as well, and has a couple advantages over an S-corp, but by and large early on the important thing is to make a business separate from you personally. A corporation also allows multiple owners in the business, which is probably what you want when forming a business with some friends.

Partnership Agreements
Of course, starting a business with some friends can be tricky. Invariably someone is going to leave the company, either for personal reasons, financial reasons, or whatever -- he or she will find that starting a company isn't all it's cracked up to be and they'll end up moving on. Handling that smoothly is going to be very important for the long term viability of your new company.

I would recommend putting a "Buy Sell Share Agreement" in place *as soon as possible* after your company's formation. A lawyer can draft one, but you can also use the book [insert book here] and then take that mostly completed form to a lawyer for final review.

The important thing about the buy/sell share agreement is that it cleanly and clearly describes what happens to someone's ownership under many different situations, including ones you don't think about -- what happens to Bob's shares if he dies in a car wreck? What happens to Linda's shares if she divorces her husband? Dave got an offer to sell his shares to a friend -- can he do that? Erica is quitting -- can she just keep all her shares and stay an owner even if she never works another day in her life? The company is going to fire Tom -- what happens to his ownership?
A buy/sell share agreement will let you handle all those situations while minimizing acrimony since everyone will have a good idea, starting out, what the restrictions are on their ownership.

Accounting
I like accountants. In my experience, they're generally reasonably low cost and they offer a wealth of advice when it comes to managing finances to minimize the impact on your taxation situation. Unlike lawyers, they really don't seem to find it fun to make up how many hours they're billing you -- presumably there are enough accountants around that they realize pissing you off will make you move on to a competitor.

Find a low cost accountant that services the needs of small businesses, and use them for tax preparation and advice on your finances. They may also be able to handle your payroll services, but if they can't someone like PayChex or ADP works really well to keep you compliant with all the state and federal withholding requirements, and they don't cost that much.

A book keeper who manages your receipts and sets up your internal accounting system can help a great deal as well, especially for getting things rolling, but isn't strictly necessary. If you're anything like me, however, keeping track of receipts, stubs, old checks, invoices, etc. is a pain in the butt.

Licenses and Taxes
In order to run your business legitimately there are going to be a host of licenses and taxes you'll have to pay. There are usually corporation fees you pay to your state; a business license from your county and possibly city; sales and withholding accounts from your state's department of revenue (and possibly from your county and city as well); etc. There are a *ton* of little things like this you have to track, so it's best to check the Web sites for your state, county, and city and see what they mention on these topics.

Luckily for me both Cobb County and the State of Georgia have excellent small business Web sites.

Office
A common question for small teams starting up is "Should we get an office?" There's no hard and fast right answer to that, because there are advantages and disadvantages. The disadvantage, of course, is that an office is an extra expense. Depending on the local economy it can cost anywhere from $0.50/sq.ft./month to $10+/sq.ft./month, and figure you'll need about 150 sq. ft. per employee. Not only that, but most building managers prefer 3-year leases, and very few young game companies can accurately look that far into the future. If the company is a success, you'll probably want to move to bigger offices before the lease expires. If the company fails, you'll probably want out of the lease before it expires. You'll also have additional expenses for phone, Internet, utilities (if they're not included in the lease), insurance.

The primary advantage of an office is that it forces the team to gather in a central location and work -- it's enforced discipline. It's also a great way to get everyone together to discuss designn, technical, and business issues. That said, it's a fixed cost that a lot of cash strapped teams would like to avoid, and with IRC, instant messaging, e-mail, and phones, it's possible for everyone to communicate without being in the same location. If there's a spare basement, garage, or bedroom for the team to gather in, that works too.

This can change rapidly once you start having meetings with outside entities like prospective clients, publishers, distributors, and investors. I'm generally opposed to trying to raise money before you have anything to show, so I would still argue that until you *need* an office, you should avoid it unless you have money to burn.

Insurance
There are a ton of insurance policies you'll probably need to take out. Errors and Omissions (E&O) insurance is often required from partners; workman's comp insurance might be required by the office building; there's health insurance as well. For health insurance, it's usually *not* cost effective to get a group rate right off the bat, it's probably cheaper for everyone to get a good personal insurance policy (or extend existing benefits from their previous employer through COBRA or something similar) and get reimbursed until the company is larger and more liquid, at which point you can have a bunch of different insurance agents come in and do the dog and pony show for you.

Web Hosting and E-Mail
A modern computer oriented company can't survive without e-mail and Web hosting. There are a zillion different Web hosts out there of varying quality and cost, finding one shouldn't be that difficult. As a general rule look for one that is low cost; offers phone support in case of an emergency (I've had some very bad experiences with outages that prevented me from contacting the support admins when there was no phone support available); mailing list management; a good Web interface; and reasonable storage and bandwidth rules (if you get Slashdotted, you don't want your site getting ripped down or end up paying $1000 in overage costs).

I highly recommend *against* setting up your own mail and Web servers internally. This puts you in the security driver's seat, which means you'll have to monitor all the security updates for all the software packages you run. You'll need to put in solid power, and establish a rock solid backup rotation. Most Web hosts are colocated in high quality data centers with redundant connections to multiple Internet backbones and steady power -- a small office can't compete with that, especially off of a DSL or cable connection. Finally, keep in mind that you don't want to be competing with external visitors for bandwidth! If you're trying to get e-mail but can't because some fan site just posted your URL, you're going to be mighty annoyed.

Your lifeblood is going to be the Web and e-mail. Business contacts and partners are going to learn about you and contact you through the Web and e-mail, and it is absolutely vital you put the maintenance and operation of those in the hands of professionals devoted to that task 24/7.

Naming Your Business
The rules that apply to naming the game apply to your business as well.

Finances
Ooooh, the part everyone skipped ahead to -- how to get money! Unfortunately, it's not easy, so let's go over the basics.

Nest Egg
Before you even think about starting out on a new game company and devoting your life's savings to it, you need to assess what your life savings are. How much money can you torch to pursue your dream? Figure out that amount *first* and when you run out, that's it, pack your shit up and go home because you gave it your best shot, and it's not more noble to go down with the ship. Bankruptcy will haunt you like the ghost of someone you killed, dig?

Burn Rate
Now you know how much you have -- so figure out how much you're going to spend. Compute all your one time startup costs -- and don't forget the little things like blank CD-Rs, legal pads, potential travel expenses, long distance charges, cell phone bills, sodas for the office, etc. etc. -- and then figure out how much you're going to spend per month. Subtract the startup costs from your nest egg (see previous section) and then divide by your estimated monthly burn rate.

That's how many months you get to survive. Period. If you got $5000 in startup costs, you need $3000 a month to live on and oeprate your business with, and you got $35K saved up, that means you have 10 months to income going. That isn't much time.

The magical date when you theoretically run out of cash is your point of must return -- you need to do everything possible to get cash flow positive by that time or you need to hang it up and try again later.
I
nvestment and Bank Loans
There is something about starting a business that makes people immediately assume other people should assume more financial risk than the owners. "Ooooh, oooh, how do I get a business loan? Is it true that the government will give me a grant?! What about investors, how do I get investors?! I can expect a publishing deal, right?"
It's not that easy. Money trees don't just grow up overnight, and the cash fairy typically doesn't exchange a copy of your business plan for a a million dollar check just because you stuck it under your pillow at night.

The reality is that if you want someone to finance this project, it's going to be you. There are three notable exceptions to this:

- Your business has been profitable for several years, at which point you can probably get a decent loan or line of credit from a bank. But by that point you probably don't need the loan.
- You make it your goal to hunt down outside investors/publishers. This means focusing your entire company on finding money, *not* on making a product. This has worked for some in the past, if you're a very good salesman and preferably have something to show off in your little dog and pony show. Unfortunately this route has led to ruin for a lot more companies that spend all their savings trying to convince others to give them money.
- You stumble across a rich relative or friend who is willing to finance you. There is a severe downside to this if things don't work out -- your relationship with a friend or relative will go to crap if you lose their money. No matter how confident you are in your success, that's a risk that you have to think about really heavily.

Of course, once someone hands you cash there is an expectation you will return it at some point. If you do not return that money, bad things can happen. If the money is a loan, then there's a chance you'll either have to declare bankruptcy (which is bad, bad, BAD for your karma and future credit) or spend a good chunk of your future earnings paying back creditors (who have already taken your house, cars, and big screen television).
If the money you have is from an investor you're somewhat protected, because they didn't loan you the money, they invested in your company. This still isn't risk free, because there's always a chance they can initiate a shareholder lawsuit against you personally and claim gross negligence.
Even if you have success things can get ugly. With a bank loan you simply repay it and you're done. But with an investor, they'll own a portion of the company and can make things complicated -- this is why it's vitally important that you have a buy/sell share agreement among all owners in place before any investments occur. The last thing you want is to have your Uncle Fred sell his 20% stake at a 100% profit to some guy you've never met but "has some ideas about how to do games".

Point being that using other people's money to invest in your own vision is very risky and has long term ramifications, even if you're successful.

Payroll
At some point you'll need to start paying yourself a salary and maybe dividends, royalties, bonuses, or whatever. You need to do this by the numbers to avoid an unhappy confrontation with your state department of revenue, state department of labor, or the IRS.

The easiest way to handle payroll is to use a payroll processing service like PayChex. For a small fee they handle all the withholding details that you need to deal with, and you just tell them "I want to pay such and such gross for this employee every N weeks". You can change the amount and frequency any time you want, and they handle all tax forms and filings, including quarterlies and end of years.

Now, you could save some money by doing all this yourself, but it violates one of my rules of "let the experts handle it". Stick with your core competency, and if your core competency is not accounting, then you'd probably do well to avoid payroll management.

That said, if you have more time than money, then there's no harm learning all the details and doing it yourself.

Hardware
You're making games, ergo you will need hardware. Unless you plan on doing some massively cutting edge technology, you can get by with inexpensive computers just fine. I'm talking $600 PCs from Dell -- they'll have all the specs you need for development. Laptops are nice if you plan on traveling a lot or doing demos, but otherwise they're a luxury -- only get them if you can justify the cost (then again, they're under $1000 these days for okay ones, so the extra cost isn't that bad).
You'll also want uninterruptible power supplies and probably a dedicated file server as well. The file server can be very low end hardware, and it's going to store your assets and source code repository. You'll want to have a CD-R attached so you can make backups, and if you can justify the cost, put in a mirrored RAID subsystem (which is fairly inexpensive these days -- often just the cost of a second drive).
The short list of must-have computing equipment will probably look like this:
· workstations w/ monitors
· (optional) laptop for demos, travel
· file server w/ mirrored RAID and CD-R/DVD-RW for asset and source code repository
· 100 MBit switch (for an office environment)
· firewall/router/wireless station for each networked area
· uninterruptible power supply for each machine
· network cables (do *not* buy these from CompUSA or Staples, you can get them for 75% cheaper, even after shipping, from on-line places like NewEgg.com)
· printer/FAX/copy/scanner all-in-one device like the Brother MFC9700
· telephones

Software
Good professional software development tools are incredibly expensive. The Adobe Creative Suite is $1000, Visual Source Safe is several hundred dollars, Visual Studio.Net is over $1000, Microsoft Office Small Business XP is a few hundred bucks, Maya/3DSMax are several thousand dollars, etc. etc. You can get creative if you're young and a student (or teacher) and try to get academic pricing on some of these, but check the academic pricing qualifications carefully before going that route (for example, even if you're a student you may not be allowed to use a particular package for commercial purposes).

You *could* warez that stuff, but that would be immoral, bad, and hypocritical. If you can't afford the software, then don't use it, especially since there are cheap or free alternatives available out there. Instead of Adobe Creative Suite look into using something like GIMP; instead of Visual C++ look into using Borland C++ Builder or GCC/Dev-C++; instead of using Maya think about using Blender and MilkShape; instead of Visual Source Safe or PerForce look at cvs, BitKeeper, or subversion; there are a lot of options out there if you're willing to settle for 75% of the functionality for a fraction of the price.

And if you're just starting out and given the choice of stealing software; spending too much money on software; or using software that's just good enough but is cheap or free, the last option seems to make the most sense.

Sales

Let's assume that somehow you manage to survive long enough to make a product, you'd like to find some way to get it into the hands of others. There are basically two common methods available: retail boxed distribution, and electronic software distribution.

Retail Distribution

Retail boxed distribution is where you have a CD or DVD-ROM, case, maybe a manual and some cover art, and, if you're feeling really glitzy, a box. You package this together and try to convince people to buy it. Ideally you get into storefronts like GameStop, Electronics Boutique, and, if you dare to dream, Target and Wal-Mart. The last two aren't going to happen for you if you're an indie, so don't bother.

Realistically, boxed retail distribution isn't worth it for a small self-funded indie. You're out an initial cost of goods, and your margin is crap. So there's more money up front but, ironically, less money for you to make since everyone else wants to get their cut.

Electronic Distribution

Electronic software distribution (ESD) is the most sensible way to get your product out there, assuming that it's small enough. A 200MB download is going to tax your internet bandwidth a lot, especially if you have a popular demo.

But the advantages are many -- instant availability for buyers, ability to update the latest version without forcing a new print run on media, zero initial cost, and no one is skimming profit with the possible exception of your merchant bank account (for credit card processing).

The most significant disadvantage for ESD (other than the aforemention Internet bandwidth costs) is that potential customers have to visit your Web site in order to see your game. They'll rarely stumble across it or impulse buy it like they would if they were at a CompUSA or Electronics Boutique.

So a lot of this boils down to numbers -- does the increased exposure of retail distribution offset the lower take per unit compared to direct sales? For most indies -- no. But sometimes a unique set of variables will conspire to make retail make sense, but I've never seen this work out positively for an indie.

If you decide to go the ESD route you'll need to make sure you have a high quality file host for your demos. This can become extremely expensive. Using an inexpensive service such as GetAFile.com still can be expensive ($2/GB). If you're making a big game with lots of content, and which has a similarly large demo, the costs can skyrocket -- one SlashDot article can put you in the red.

Let's say your game is distributed as a 200MB installer. That means each download will cost you $0.20. The general rule of thumb is that 1% of downloads will result in a sale, so that you'll hopefully make one sale for every 100 downloads, which is $20 in bandwidth costs right there -- ouch. If you sell your game for $30, the bandwidth costs alone are going to consume the vast chunk of any potential profit, assuming you at least get your 1% sell through -- one good SlashDot hammer and you're screwed, since massive hits like that tend to get curiousity seekers more than potential customers.

There are a few ways you can offset this. The first is to partner with the myriad file hosting services that are available. This can take the load off of you somewhat, but it has the severe downside of dissuading potential customers that don't like "Free registration required" download sites.

The second option is to get a fixed bandwidth connection to the net. This may mean hosting a file server of your own on a dedicated T1, which can be very expensive. Some services, such as CBeyond, charge about $500/month in exchange for 1.5MBit bidirectional connection, always on, with VOIP on 5 phone lines, voice mail, etc. If you're seriously interested in CBeyond, drop me a line, I could use the referrals...

Instead of a T1 you can lease bandwidth on a colocated server, say 1MBit of fixed bandwidth which might cost you $250/month or so. The primary problem with a fixed bandwidth connections is that they get overloaded. If you're getting light traffic, then you're fine -- a 1Mbit connection can deliver a 200MB file in around 25 minutes, so as long as you're not averaging more than about 50 downloads a day, you're okay. But if you get hammered suddenly, a lot of people will see 5K/sec downloads, which usually results in canceled downloads and losing potential customers.

One compromise is to do both -- use a low bandwidth fixed connection (256Kbit to 1MBit) server for the bulk of transfers, but when you start to see your bandwidth climb have a script automatically start redirecting to a service like GetAFile.

Marketing
Marketing is crucial, but it can also be a major distaction and possibly even a negative if you screw it up. The two biggest marketing problems I see with new game companies are opposite extremes -- either no marketing, at all, so people don't know they exist, or they go overboard on the hype when there's nothing else to show. The latter problem is the result of a combination of overenthusiasm -- hey, we have nothing to show, but *we're so excited we want to talk about ourselves all the time!!!!!* -- and a need for validation.

Buzz, word of mouth, hype -- whatever you want to call it -- is crucial to selling your games and getting the notice of industry partners like portals, publishers, distributors, game review sites, etc. The more people know about you and your product, the better.

The problem is when people know about you and you have no product. They'll get excited at an announcement or a game design spec or a new Web site or some screenshots, but at some point *marketing only matters if you have something to sell!*

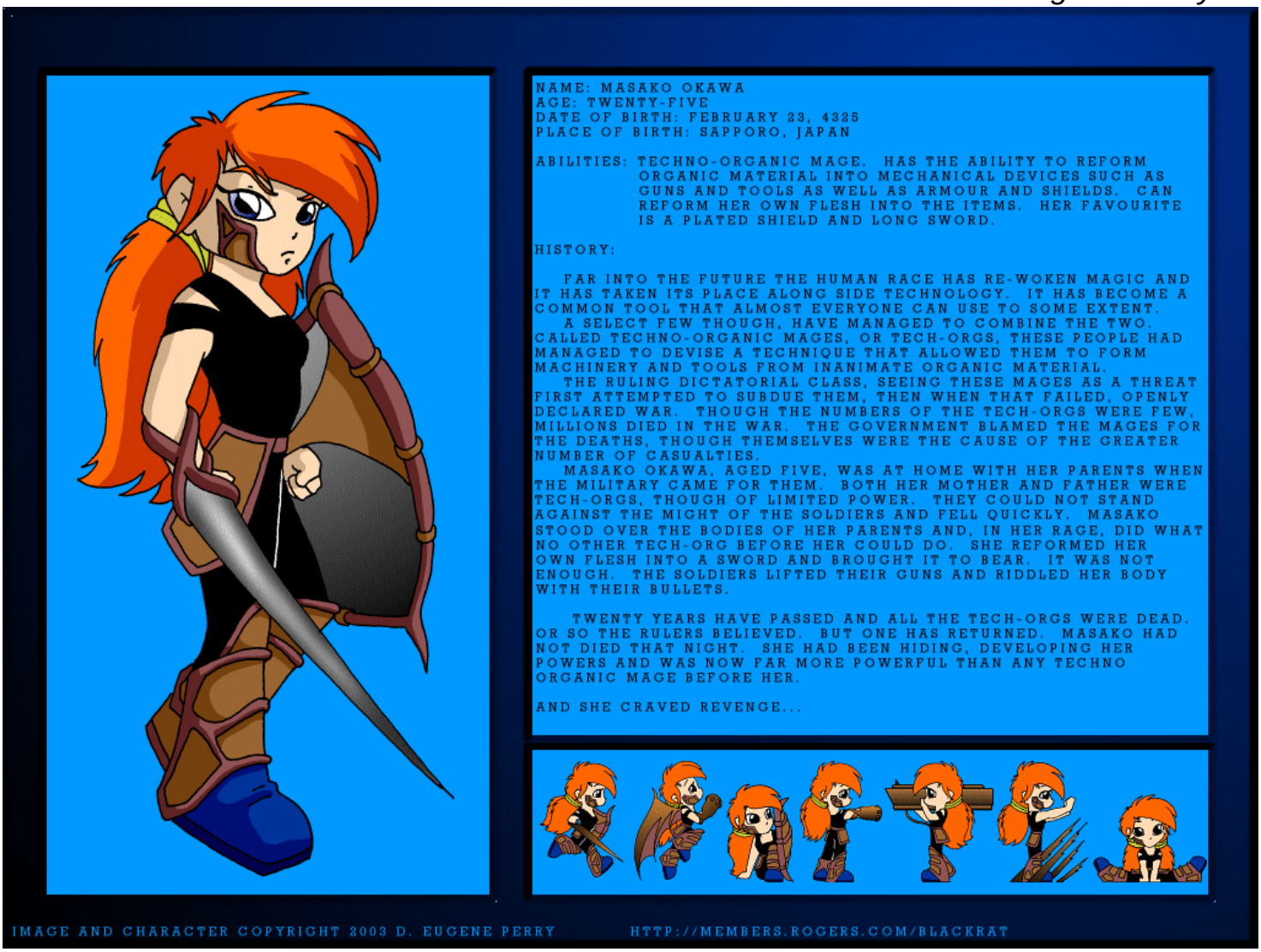
You need to time your marketing blitz with the relesae of your product. Build anticipation about the same time you're going into beta. Get people interested so that the next time they hear about your company or your product they'll think "Oh yeah, I remember hearing about that, I need to check that out". And when you release your product, it will be the "Oh, cool, I remember checking that out, now I want it!" phase.

And keep people coming back! If people hear about you once then hammer your Web site only never to show up again, you've lost a vital opportunity. You need them to return over and over and establish yourself as an important site to visit. This keeps you in the forefront of people's minds (again -- only relevant if there's something tangible for them to give a shit about). Having a Web site go stagnant almost always makes it seem like the company just died or the owners got bored and wandered off.

Summary
Starting a game company is relatively easy but it can seem daunting at first because there are so many things that have to come together in order to become legit. This article is an attempt to at least make up a check list of the things that you'll have to go through early on as you start to ramp up. It's not a guarantee of success -- in fact, it's not even related to the odds of success -- it's only a general outline of the *process* of starting a game company.

Summer 2004 Graphics Award

D. Eugene Perry



2nd Place

dinogamerco@aol.com
www.dinogamer.com

submit your work

* Create a character and send a jpg or gif file (Under 25k)to morphosisgames@aol.com (Subject Line: "Character Contest")
* Include your name and Character information
* Deadline October 15, 2004