y Jamie McCornack

But before we get started, I must confess a bug in my text. So for you Loyal Readers who've been following this column for a few months...

Synchronous? Asynchronous? My dog ate my homework?

In the November column (Some Sound Advice)...well, I lied. The three paragraphs after the header Synchronous vs. Asynchronous Sound are exactly backward. It's correct in the book (page 32, Tricks of the Mac Game Programming Gurus, Hayden Books, 1995), and I wrote both the column and the book chapter I lifted it from, so I ought to know better, right? And if I know better, you'd think I'd write better.

Synchronous sound is "synchronized" to screen action in the crudest possible way—when synchronous sounds are playing, all screen activities are locked up solid. This is fine if you want an immobile warning sign in the background while your Mac says, "Intruder alert, intruder alert," or, "Coin detected in pocket," but in the real world of game development, you'll want sound and graphic action to be independent of each other (asynchronous) instead of sequential (synchronous).

Folks get it mixed up, thinking "synchronous" must mean the game characters' lips are moving while the sound is playing, but that's not the way it is in the Toolbox. To the Mac Toolbox SndPlay() function, "synchronous" means sound-then-action, and "asynchronous" means action can continue while the sound is playing. 99 times out of 100, asynchronous sound is what you want. And that's the truth.

Sorry about last month. My thanks to Alert Reader MacGregory for pointing out the error of my ways. At least I got it right in the comments to the code(MGWSound1.c, lines 375 to 425).

And now, back to this month's column, which is already in progress...

Clams in Combustion

Though for the most part, the ambulatory clams I've been using in these demos have been well received, one recurring question keeps finding its way to me:

"Love your column, Jamie, and the clams are charming, but how do you go about blowing

them up?”

From the programming standpoint, this is a trivial task. From the standpoint of the art department, this is an interesting challenge. Fortunately, there are resources available, or it would be a hair-pulling cola-swilling allnighter challenge, since animated fiery explosions are extraordinarily difficult to draw by hand.

ETHICS ALERT!!! THE FOLLOWING TWO PARAGRAPHS HAVE NOTHING TO DO WITH PROGRAMMING OR ART DEVELOPMENT TECHNIQUES!

I am unconcerned about this demo’s impact on society. As excellent as it might be, I doubt it will inspire anyone to skulk down to Cap’n Jack’s Seafood Emporium and commit a copycat crime. Yes, blowing things up is a staple of the computer game genre, and yes, if you’re going to do it, you should do it well. Still, to quote john calhoun, nobody ever lost sleep from writing a nonviolent game. The techniques herein can be used in appropriate or inappropriate context. Are you blowing up asteroids before they can plunge into the atmosphere and smoosh all creatures here below? Neato! Are you incinerating University of Alabama cheerleaders? Then even if you’re an Auburn fan, I’d suggest you ask your shrink to lighten up a bit on your medication, okay?

Even these clams…look, there’s no such thing as a clam with legs, certainly not with legs in orange trousers, and if it wasn’t for the beach scene, most folks wouldn’t even recognize it as a representation of a clam. Yet when I showed the finished GoodbyeWorld program to one of my partners, she said, “Awwww,” and didn’t think it was funny at all. So imagine we’re developing the game of Help Claudia Clam Find Her Pearl, Ages 3-6, and quicker than the player can say, “What does ‘Beware of Minefield’ mean, Daddy?” Claudia goes off like a drummer for Spinal Tap. Think a bit about the impression you’re making. Okay, back to the column.

So the project manager says, “Can it be done?” and the lead programmer says, “Piece of cake, we’ll have the clambake code done by the end of the day. Have the art department whip up a PICT sequence for an exploding fireball, oh, eight frames long, 64 by 64 pixels, 256 colors, standard system pallet.”

So the project manager passes the request on to the art department, finishing with, “…and we need it by the end of the day,” and the art director has a good laugh and says, “Yeah, right. Who is this really? You sound just like our project manager,” and says she’ll see what they can do but don’t get your hopes up.

A little later she calls back and says, “Cliff and Lisa say they can have it done by five A.M. on their Macs at home, if they get started on it right now, but they want Friday off to make up for it, you want to approve that?” and the project manager says, “Yeah, all right, whatever it takes to get the job done.” So Cliff and Lisa leave the office and go somewhere for a leisurely lunch, and do some shopping, and go to the gym to play some racquetball and swim a few laps, and later at dinner who should walk into the restaurant but the program manager. “It’s going okay, we’ll have this job wrapped up before dawn, we had to eat something and there isn’t time to cook,” Cliff says wearily. “We sure appreciate what you’re doing,” says the project manager, picking up the check, “This one’s on the company.” Lisa doesn’t say anything. She’s trying not to laugh, because back at home, she has a copy of Pyromania!

Pyromania! and Pyromania2 ($199.95 each, VCE Inc., 818-367-9187) are CD-ROM collections of 640 x 480 (and up) 32-bit color PICT files of various explosions, fires, fireworks, sparks, and smoke, shot against a black background by a genuine Hollywood special effects studio. Explosion sequences run from 40 to 80 frames as a rule, while fire sequences run up

around 200 frames. These aren’t video captured, they are scanned from 35mm motion picture film—they are sharp and clear and there are no strange artifacts. They look like what you see in the movies, and I don’t mean Ed Wood productions.

Explosions run the gamut of ground, air, zero G, flak, and assorted shockwave explosions. For our spontaneous clambustion demonstration, we’ll use a medium size zero G fiery explosion. Here is a typical frame, of which there are over a thousand per disc, clipped slightly to fit on this page:

f your monitor is set to Millions of colors, this looks gorgeous, and at Thousands, it looks terrific. At 256 colors, well, it looks a bit spotty. That's one of our jobs, to optimize this explosion for 256 colors.

This frame is ZG02.127 from a series that runs from ZG02.100 to ZG02.157. In explosions, things happen pretty fast and changes are dramatic from frame to frame, but we don’t need every frame of that great big explosion to blow up a modest little clam.

Back home, Lisa asks Cliff to make a PICT with nine 64 x 64 boxes laid out in a square, while she finds her Pyromania! disc. Cliff opens Zeus (Delta Tao Software, 408-730-9336), which is his favorite paint program (despite its quirks, it has a bunch of cool features that appeal to a guy like Cliff), and whips up this grid:

ote that the boxes are 64 x 64 inclusive of the lines. If the art covers a line, no problem, just erase the line. The lines are only there to guide the artists. However, the art cannot go past a line, or even into the line of the neighboring box (which is why the lines are two slightly different colors), or is will show up in the wrong sprite face when the program is run. Anyway, one of the reasons Cliff likes Zeus is because of its clear and accurate dimensioning tools, so it only takes five minutes to come up with this grid.

Lisa sticks the Pyromania! disc in the slot, and they pore over the QuickTime preview movies to find an explosion they like. Yep, ZG02 look like the cat's pajamas, so they open its PICT folder, where thumbnails of the frames are laid out in chronological order.

"The first frames are boring, and the last frames are boring," says Cliff, "so let's pick the eight best frames out of the middle."

"Seven frames," says Lisa, "If we skip the first six frames, and then take every seventh frame until we have seven frames, we'll be done before Blockbuster closes."
"Why seven frames?" Cliff asks, "they said they want an eight frame sequence."
"Because the last frame should be blank," says Lisa. "It'll make the programming easier if they have a frame with nothing happening, the explosion's done with and the clam is history."

They open ZG02.127, which looks like the biggest image, at 25% scaling and measure it. Nope, 84 pixels at the widest. Try ZG01.127. Good, 54 pixels for the main explosion, which leaves room for those bits of clam shrapnel. Create a ClaBoomƒ folder on the hard disk, save this PICT at 256 colors (art department talk for 8-bit color), name it ClaPop127 (slightly smaller than ClaBoom)…

…and do the same with frames ZG02.106, .113, .120, .134, .141, and .148. Then they open ClaPop127 again, select white as the fill color, and use the paintbucket tool to change the background from black to white.

“That’s going to leave us a lot of handwork around the edges,” Cliff says, so they move the precision selector on the paintbucket tool from Exact down a little toward Vague (that’s one of the reasons Cliff likes Zeus) to get rid of some of the almost-black pixels.

o they do a bit of handwork with the pencil tool, erasing a few pixels that don’t fit their artistic tastes, and copy-and-paste the seven frame sequence into the grid, saving the result as BoomSprites.

“Now we make the mask,” says Lisa, “and we’re done. Sure hope we’re done in time to rent a video.”

“Piece of cake,” says Cliff, making literary recursive history by waving this very CD-ROM in the air—the January ’96 issue of Inside Mac Games—and saying in melodramatic tones, “I have here…the Mask of Zorro!”

Sure enough, ZorroTable is here in the code folder for this month’s column. It is a Photoshop color table with 255 black entries and one white entry. Cliff and Lisa quit Zeus and open BoomSprites in Photoshop. They go to the Mode menu, select Color Table…, load ZorroTable, and wallah!

“Why are the first three frames all speckled?” Lisa asks.

“Because there was a lot of white in the early stages of the explosion. We can fill them in with the brush or pencil, it’ll just take a jiffy. And we have to erase the grid lines, but if we erase them by hand where they touch the artwork…

“…we can get rid of the grid with the paintbucket. Then we go back to Mode and select Bitmap, 50% Threshold and save it as BoomMask. There, easier done than said.”

“And we’re outta here,” says Cliff, grabbing his hat.

“Uh, let’s think about this for a minute,” says Lisa. “We’ve made an explosion sprite set in record time, and it’s right to project specs, but we’re talented and creative artists, right? I mean, that’s why they pay us the big bucks…

[pause for laughter here, and wait for those with computer graphics careers to catch their breath]

“…and this is not as good as we can make it. Even though we’ve dumbed it down to 256 colors and seven frames, this explosion still borders on the photorealistic. The clams, on the other hand, are cartoons. It’s not going to look right. You know how Wile E. Coyote is always getting run over by trucks in the Road Runner cartoons? Well what if after the truck went by, they switched to a photograph of what a coyote really looks like after a truck has run over it and…”

“Yeah, I hear you, you’re right,” says Cliff, “but we can still be out of here in ten minutes, because I have another of Jamie’s cluts here…”

“Who’s this Jamie you’re talking about?” asks Lisa.

“Brilliant, brilliant man,” says Cliff, with sincere respect in his voice. “Anyway, he has a clut called HotSwapTable with only 16 colors on it. There’s one black entry, 31 white entries, and 16 entries each going from bright yellow, through orange, and down to deep red. Sure, he stole the idea from Video Fusion, which has a Black Body color set you can apply to video, but there’s no denying this is an inspired application of…”
“Fine, enough fawning, we haven’t got all night. How does it work?” Lisa asks.
“Open a copy of BoomSprites in Photoshop, select Color Table from the Mode menu, load HotSwapTable, and click Okay,” says Cliff.

“Hey, cool!” says Lisa. “Not brilliant, ’cause I see some white speckles in this one too, but definitely cool.”

Cliff shrugs. “Yeah, some of the real dark pixels come out white. We can fix them with the eyedropper and pencil tools. Then all we have to do is convert it to the System color table.”

“So go back to the Mode menu, and select System from the Color Table…”

“Nah,” says Cliff, “I tried that once, and I ended up right back where I started. First convert it to Mode RGB Color, then convert it back to Indexed Color. Choose 8-bit resolution and System Palette, and save it as CartoonBoomSprites. You end up with exaggerated fire colors, using the same color table as the clams and the beach, so there’s no color conversion lag in the animation. BoomSprites and CartoonBoomSprites use the same mask, so let’s turn them both in and let the project manager choose a favorite.”

“Allllrighty then,” says Lisa, “what movie do you want to see?”

Meanwhile, back at the Programmer’s Pit…

If you got through the October installment of this column, the programming here will be a breeze. If you didn’t, well, it’s all available in Chapter 0 of Tricks of the Etcetera Etcetera and I hear there’s going to be an IMG Back Issue CD-ROM set available (probably mentioned elsewhere in this mag).

Open the GoodbyeWorld1.c source code file. To make the new stuff more obvious, the program has been simplified from recent HelloWorld versions; a single clam and no mood music.

We have some new constants, kBoomFace0 through kBoomFace7. Note we use the word “face” in this context to describe a particular member of a sprite set, rather than that thing on the front of a head. The animation of the explosion is only different from the animation of

the running clam in one significant way: the explosion does not move, so ShowExplosion(), unlike ShowClam(), does not require a union of the rect where it is and the rect where it's been.

kShrinkFactor sets how much smaller the fireIsAt rect is than the clamIsAt rect, and since we want fireIsAt rect to be bigger (64 x 64) than clamIsAt rect (32 x 32), kShrinkFactor is a negative number.

Both of the explosion sprite sets are in the resource file; set rFireFacesID to 141 for the cartoon version, 142 for the photo version. Oh yes, and we've added rExplodeSndID to our 'snd ' resources, and changed rHelloSnd to rGoodbyeSnd.

The new variables are what you'd expect; fireFacesCPort, fireMasksPort and some new rects to locate the proper faces, etc.

Explode() is nearly identical to LipSynch(), except the sprite faces selected show an exploding fireball instead of a talking clam. Note the last face selected is blank—a lazy way to erase the final frame of the explosion. If Macs still had 128K of RAM like they did a dozen years ago, I'd do something more sophisticated.

Why are there two blank frames in the explosion art? I thought I might want to leave some smoking rubble on the beach, but we found the Cajun Style Blackened Clam to be unappetizing.

SayGoodbye() gives us an orderly transition from LipSynch() to Explode(). The one slick thing we do in SayGoodbye() is to set the rect variable fireIsAt to be equal to clamIsAt, so the explosion will happen where the clam is, and then expand fireIsAt from a 32 x 32 pixel rect to a 64 x 64 pixel rect via the Toolbox routine InsetRect(). InsetRect was developed to shrink rects (such as windows when you close them), but it expands rects quite nicely when fed negative numbers. So we "inset" by -16, which pushes each edge of the rect 16 pixels outward.

Goodbye, World! See you next month.