Volume 5 Number 2 February 2009

THE AIF COMMUNITY NEWSLETTER

Contents

Mission Statement

Inside Erin is written and published by people who enjoy AIF. It is done for fun, but we also have some goals that we seek to achieve through the newsletter:

1. To encourage the production of more quality AIF games by providing advice from game developers, and by offering constructive criticism that is specifically relevant to AIF.

2. To encourage activity and growth in the AIF community. We aim to generate a constant level of activity so that there aren't long periods in which people can lose interest in AIF.

3. To help document and organize the AIF community. This is done through reporting on games and events, as well as by helping to organize community-wide activities such as competitions and the yearly Erin Awards.

A Letter From the Editor

Purple Dragon

Welcome to another exciting installment of Inside Erin. I'm sorry that we're a bit late again this month. I've recently started using a different program to edit the newsletter, and while it is much more powerful than what I was using before, it has also turned out to be a bit much for my rather feeble intelligence. On the bright side, with the new program I was able to do a couple of things that I couldn't while using the old one. Most notably, links in the text will now work like they're supposed to. Both hyperlinks that will take you to the vast world wide web beyond our walls, as well as internal links like the contents and the "continued on…" text that will now take you to where it's, well, continued on. Hopefully this easier navigation will make up for the wait.

In other news, now that our *Programming Erin* feature is over, we are starting a new one in its place. The name of the new feature is *Coder's Corner*, and the format will look very familiar to those of you who were following along with *Programming Erin*. There are a couple of differences, which I explain in more detail at the beginning of the feature.One change that takes place with the beginning of the new feature is that I am no longer writing the Inform 7 segment. Dudeman has shown through his I7 AIF extension, as well as other things, that he knows the program almost as well as I do. Well, maybe even as well as I do. Oh, who am I kidding, he blows me away, and I'd like to think that I'm smart enough to know what I don't know, and to step aside for someone who knows the things that I know I don't know, you know? Err, anyway, all that is to say, we welcome Dudeman to the staff and leave the new feature in his very capable hands.

Also in this issue we have an article by the great Lucilla Frost, talking about issues that come up in writing games. The fact that these issues came up while she was writing her new British Fox game is something that I'm sure a lot of you are as happy about as I am.

So as always, come on in, kick off your shoes, relax and get comfortable as you peruse this month's offerings. I hope you enjoy yourself, and I'll see you next month. ◆

This Month In AIF

by BBBen

This month the deadline for the threesome comp arrived. Sadly, there was only one entry (by rip_cpu) so I had to call the comp off. I saw this coming a while ago so I'm not too bothered (this saves me some work, after all) but it is a bit of a pity for rip_cpu, who actually had his act together enough to put together a good game in hopes of putting it up against the competition. I'm not really sure what went wrong with the whole thing, but I suspect it's just down to the lack of enough active authors around the place. There were about five other authors interested that dropped out along the way, too, so maybe having such an extended deadline led people to drop out easier along the way. Oh, well, at least this means I haven't stolen too much life away from the mini-comp!

In spite of the comp flop, there was actually a bit of life around the place lately, with a new game release and a new version of another game. There was also a decent amount of discussion

Continued on page 3

I am a cheat and a thief. No wait, let me explain. When it comes to writing games I'm no programmer. I've tried, boy have I tried. Many a time I've poured over the manual and tutorials but I just can't seem to learn from them, I just can't connect what they're trying to say with what I need to do.

To get an idea of what I need to do I need to see it in context. So I rummage around in other people's games to work out how it fits together. Hopefully I know enough by now, because it seems more people are putting passwords on their work, and even the smallest intro seem to be locked to the outside world.

Now don't get me wrong I understand why people do this. First, they worked hard on their game, trying to work out a challenging game with epic plots and sparkling characters, only to have people open the game up and bypass the problem solving elements. Or second, they don't want people stealing all their hard work.

For the second one, there is no way I'm going to change your mind, I understand and respect your decisions, though I think there something satisfying in others commenting on how well something is written. As for the first, just look at the forums and boards, there are a large number of people who don't or won't play the game. They just want to get to the change or the sex scene or the mind control scene, working out how to get there just isn't part of how they play the games. Then again you find it in real world games anyhow.

And for the record I normally attempt to play the game through straight first then I'll see if the game will open, and poke around inside. And I'm inspired, I never directly steal.

If you really must keep the game locked then there are other ways you can help, like the tutorials provided in this very newsletter. Or provide code snippets that demonstrate some function of the system, like those Ehlanna has provided for RAGS (Ehlanna's also working on a set of tutorials but it's in the early days). Or like the mighty Chris Cole you can just provide the password once enough people have had a chance to enjoy the game. After all, this might encourage more people to try their hand at game writing and the more people involved the better.

Continued on Page 3

It's been a busy month for the game authors on the Collective!

Ehlanna has released a game called Debutante, in her words:

"A rather short game that grew a little! Started off as an entry for a competition in the TFGames forum, which restricted the number of rooms (which explains why it is all so 'cramped'). It took a little longer than I hoped, due to RL intruding, so I tossed in some additional bits and bobs."

The thread following the development is:

http://hypnopics-collective.net/viewtopic.php?f=11&t=14548

And the game can be downloaded at:

http://hypnopics-collective.net/cpg132/displayimage.php?pos=-83909

Darstan has released a game called Bodywerks which he describes as:

"It contains a lot of mind control, is focused on women only and, as a warning to those who don't like it, also contains bodily modifications of various kinds...

Continued on Page 4

This Month in AIF, Continued from page 1

on the boards (though not all that much really worth reporting on). One subject I thought I might bring up, however, was that the author of the 1989 proto-AIF game *Wuz* apparently saw a review of the game that was hosted on AIFGames.com and was so outraged as to demand its removal, or the game to be taken off the site. Sexton, the owner of AIFGames.com decided to remove the game on the (I feel correct) grounds that reviews can't be censored just because the reviewed authors don't like the review. The issue is a little bit more complex than just that so if you have an opinion, go and check out the thread on the AIF Game Announcements board of the AIFGames.com forums.

New Games

The Pizza Boy by Gray64 for ADRIFT 4.0, released 4th January 2009 (withdrawn for beta testing before re-release on the 13th of January). You are a pizza boy with a delivery for a house with a mad scientist and his beautiful creation…

Deadly Climax 0.8 by madquest8 for ADRIFT 4.0, released 23rd January 2009. The latest update of *Deadly Climax*, with "two new targets, and some minor tinkering... together with some spelling and punctuation work".
But now for a game writer that I think more people should know about. ◆

TFGames, Continued from page 2

In the 90's there was a band called Whale that I'm quite a fan of. No wait, there is a point to this. You see they sang in English but were from Sweden, and used this fact to play around with their lyrics, prodding English in ways it really shouldn't go.

As for the point of this, well there is a TG game creator who does a similar thing with their games, Sally73. I can't put my finger on it, but there's something different about how the games are structured that makes them unique. Maybe it's just a more European sensitivity which adds that certain "je ne ces pas". Now, they're not perfect, none are finished so they don't have a final polish, and they can be a little more adult than most, but I recommend that you all check them out.

Now to the job that I get pay for, activity on the TFGames forums.

A new year and a new term have bought a burst of activity to our little corner of the Internet. Finally, after a steward's inquiry the Halloween competition winners were announced:

1st *Haunted House* by Maelyn
2nd *The Theatre* by TestZero
3rd *I* by Nandi Bear
4th *Trick or Treat* by Sally73

Because of the closeness of the third and forth, Lunablue has kindly provided an extra prize for fouth place. Well done to all, especially Lunablue for organising it all, and don't forget about the next competition in March.

There was a little burst of activity about the form of the games themselves. Some demeaned the lack of non TG transformation in games, whilst others discussed what people look for in games. At the same time, those of us who create games wondered if people noticed or appreciate the Easter Eggs and extra touches some of us add to our games (on evidence so far, no one does).

But the real bonus is a slew of new games, most of which are still in early beta, but which promise greater things to come. In a continued cross-pollination between TFgames and Hypnopics three games crossed over, the excellent *Bodywerks* (RAGS) by Darstan, *Dominated* (RAGS) by MadisonX and *Demon Town* (RAGS) by Orcha. Also TinaB gave us a peek at the first part of *House of the Future* (RAGS).

On the TFGamesSite itself Sally73 gave us the first assignment for *Career Change* (RAGS) whilst TG Mage gave us the start of *Space Mall* (RAGS) and Vengance1701 allowed us to wonder around the world that will be *Bikini Beach House Carwash 2* (RAGS).

And a whole load of authors promised us a whole load of new games soon.

Well that's me for another month, if you have any comments don't hesitate to contact me. ◆

Collective, Continued from page 2

BodyWerks was heavily influenced by a lot of other people's ideas, some of them you might recognize. The primary idea is based on a series of stories on the MC stories archive, other parts and characters were influenced by the images, stories and games posted here in the Collective."

Development thread:

http://hypnopics-collective.net/viewtopic.php?f=11&t=14623

Game Download at:

http://rapidshare.com/files/182380105/Bodywerks102.rag

MadisonX posted a game called Dominated which is described as:

"Dominated sees the player join a criminal Cartel that deals in slave trading. The player will be able to abduct women, buy / sell slaves, and train them. The basic concept was taken from games like Slave Maker. The game has RPG elements, with stats and skills for the player and NPCs. The PC will work at Cartel facilities, and also at home in their own private Dungeon. So, for example, the PC can capture an unwilling target, train her, and sell her. Or, the player could purchase a cheap slave, re-train her, and sell for a large profit."

Development thread:

http://hypnopics-collective.net/viewtopic.php?f=11&t=14703

Download at:

http://www.4shared.com/dir/5952135/5146de6c/sharing.html

Go to the RAGS section in the games folder to find it.

Orcha has released Demon Town:

"This game is very explicit, and not intended for underage persons. Some violence is also involved. It includes a couple transformations (TG is one of them) and some MC content as well."

Discussion thread:

http://hypnopics-collective.net/viewtopic.php?f=11&t=14673

Download at:

http://hypnopics-collective.net/cpg132/displayimage.php?pos=-85539

In other game developments...

Tinab has posted an early version of her House of the Future game here:

http://www.4shared.com/file/80800514/76f12c1e/HOTF02.html

Dragoon93041 posted that he is working on his first game called Mystic Slaver:

"The concept is capturing women and subjecting them to various actions, raising one of the three stats: pleasure, pain, and confusion. Pain and pleasure are both self explanatory but confusion involves making their minds cloudy and making it harder to resist. The endgame is breaking the women, changing them into one of 7 forms, depending on the stats."

Development can be followed in this thread:

http://hypnopics-collective.net/viewtopic.php?f=11&t=14696

And lastly, Xiriel has posted that we can expect Act II of Jandarrow sometime in February with some luck!

http://hypnopics-collective.net/viewtopic.php?f=11&t=13688&p=360593#p360593

Have a great month! ◆

Dear Mortal Men and Women,

As you recall, last month I was describing how, while commuting by rail, I became aware that a woman on the train (who was not me, strangely enough) had become the object of infatuation of a man who had not gotten the courage up to talk to her, or perhaps had just decided that he was going to admire her from afar in perpetuity. This runs counter to everything I stand for. Love – and even lust – should be acted upon, nurtured. So I intervened on their behalf. After speaking to the woman for a moment, I beckoned the man to join us.

“Me?” he mouthed upon my indication to him.
I nodded and said, loud enough for other passengers to hear, “of course, yes, you. Come here. We want to ask you something.”
Fortunately, the row in front of us was empty, so I flipped the backrest so the seats faced us, forming a group of four seats together. The man brought his briefcase and, looking tentative only for a moment, sat down with us. We exchanged a round of ‘good mornings’ and fell momentarily silent. He looked at Sonia for a moment, then over at me, giving me a questioning look.

I glanced at Sonia. She had a tight-lipped, off-putting nervousness about her. Her hands were on her lap, but her fingers were balled together. She was trying not to look at me or at the man. This seemed like it was going to be a challenge.

Only a beat had passed with nobody speaking, and I was about to dive in when the man broke the ice himself, directing this comment to me: “You know, you look like Kathleen Turner.”
“Really? You think so?” I considered the appearance I’d taken – the young side of middle-age, straight sandy hair – I think I’d been aiming more for Emma Thompson, but I’ll go along with Kathleen Turner, too. “She’s really beautiful. Thanks,” I said.
“I was thinking you look like Emma Thompson,” said Sonia, chiming in.
I grinned – maybe this wasn’t going to be so hard after all – and then asked: “How about her,” indicating Sonia, “who does she look like?”
“Honestly, I… Uh… I’ve never seen anyone like her. I…”
“You’re smitten, aren’t you.”
He blushed, but didn’t answer what wasn’t really a question. She blushed right along with him. It was time to get them talking to one another.

“What’s your name?” I asked him.
“Eric,” he responded. “What are yours?”
“I’m Veronica,” I answered, then pointedly left it for Sonia to introduce herself. I offered my hand and he took it, grasping it with appropriate warmth and firmness.
“Sonia,” she said simply and following my lead let him shake her hand.

I could feel the difference between his handshake with me and the one with Sonia. It was not really any longer, but it was obvious to me that he was cataloging the feel of her skin, and trying to communicate beyond a simple greeting. Their eyes locked, hers looking inquisitive while his were intense. I smiled, though neither of them saw. Mortals are easily molded, especially when they’re young and horny. She had already decided that she was smitten as well, and the bright smile that lit up her face went all the way toward erasing that chilly demeanor that Eric had complained of just a few moments earlier.

Volume 5 Number 2 February 2009

For the rest of that train ride I facilitated their getting acquainted, playing referee to their conversation, running through the basic facts of their lives. The details of their careers and families aren't pertinent to this description, though certainly the airing of those details was necessary for them to build a foundation for a relationship.

When we got off the train, Sonia split off in another direction to go to work, while Eric and I walked together through the station. He fingered his iPhone (he'd tapped in Sonia's phone number just a moment earlier) like it was a treasure. "Hey … thank you for helping to introduce me to Sonia, Veronica," he said, "I don't know how you knew or why you did that, but it was amazing how easy you made it. I'd been trying to figure a way to break the ice for a long time."
"You're welcome, Eric," I said. "As for why, well, it's just a thing I like to do. Can I ask something of you?"
"Name it."
"Promise?"
"Sure."
"Well, this is going to sound fairy tale-like… I want your and Sonia's first sexual encounter, when there is one."
He paused, suddenly looking at me askew. "Um, what?"
"I'm serious. Listen, I'm a collector of initial sexual experiences between couples. They're the most poignant by far. Yours and Sonia's will be a wonderful addition – I think you'll have a natural connection."
"But… how…?"
"Listen. I'm not going to have this discussion with her – women take it a little differently – but you can and should. There are a few different ways you can fulfill this promise. You can invite me to participate – very few people do that, and I don't think I'd accept with you guys; you can videotape it for me – I promise not to put you on the Internet; or you can each write a description of what happened. That's what most people choose to do. But it has to be detailed. Anyway, think about it."

We reached the door to the station and found that we had to go different directions. After saying our farewells and our 'see you on the train tomorrows' he walked off, clearly happy but a bit confused and worried as well.

Well, this seems a good place to break off for now. This story continues, but it will have to wait for future months.

Until then, I wish you all wonderful love.

Aphrodite

How never to finish an AIF adventure

(without ever stopping working on it)

By Lucilla Frost

The subtitle is because the easiest way never to finish an adventure is to start playing a MMORPG, in my case Lord of the Rings Online. Which was great. And you'll never finish *that* either.

I've been working on British Fox and the Nationalist Conspiracy since December 2006 and it is beginning to look like I may someday finish it, (Chapter 1 is in playtesting as I write, although I don't intend to release it in parts). However, I may not as well and I thought I would illustrate one the problems with an example of the way I write, which may resonate with some of the rest of you. This will also provide a teaser for the game itself…

One of the criticisms of the ADRIFT version of British Fox and the Celebrity Abductions was:

"Another issue is the occasional lack of descriptions for objects listed in the main room description. If you say there's an alarm clock in the room, let me look at it! That just irritates me."

On that dreadful day I swore that no one would ever make the same criticism of me again, and unfortunately I then started using TADS which not only allows you to describe everything, it allows you to describe everything in a variety of states, and to add little bits of code and amusing quirks to every single object in the game.

So, part of the new game takes place inside the secret base of the enemy organisation which, it becomes clear as it is explored, is a bit of a 'cut-price' villain base. In the interests of completeness, and because I thought of a couple jokes to go with it, I decided to include a bathroom, shared by the villains. Hence the description:

The room has all the hallmarks of a shared bathroom, there are a great many bottles of cleaning products on the lip of the bath, the side of the sink and the windowsill, there are several towels and facecloths, there is a rota pinned to the wall above the sink and nothing is clean.

Alright as far as it goes, but several objects are listed there. The bath, the sink, the windowsill (implying a window), assorted detritus and a rota. We'll start with the rota…

A smudged piece of paper stuck to the wall with blu-tac indicating when the various guests can use the bathroom. "

And the player is bound to want to read it.

readdesc - 6:00 to 6:30 - Libra 6:30 to 7:00 - Norsefire 7:05 to 7:10 - Combat Eighteen 7:20 to 7:50 - Manticore 8:00 to 9:45 - Purity 10:00 to 10:30 - Captain England 12:00 to 12:15 - Captain England 16:10 - 16:25 - Captain England 20:05 to 20:20 - Captain England 10:30 to 11:30 - Purity

Thus, in case you haven't met them, letting you know who the people you are likely to meet might be, and a faint hint of what their personalities are like. Then, in case you haven't got the joke…

"Hmmm...looks like as well as being a prick, your boyfriend is a wanker too," says White Rose.

Since the rota is just a bit of paper it's not reasonable to just forbid the character from taking it away, so it's a movable item, but it's mentioned in the room description so that has to be altered so that it reads differently if the rota is still stuck above the sink. The same for the description of the rota which describes it as blu-taced to the wall. And it has to be altered so that it doesn't appear as an item in the room as well as part of the room description. Unless it's no longer attached to the wall.

Alright, what's next? Well, it's a bathroom, so even though it's not mentioned in the room description it had better have a toilet.

It's a toilet. It's clear that no one takes responsibility for cleaning it and typically for a house with men in it the toilet seat is up.

Alright, I could leave it at that, but with that description someone is likely to want to close it, so I have to create a toilet seat which can be opened and closed and which changes the description of the toilet. And some pervert is *bound* to want to 'use' the toilet so I'd better prevent that since it's not one of *my* perversions. And there isn't actually a useVerb in the base TADS definitions so I'll have to create it. (Fortunately, I did that earlier in the day because of something British Fox has in her bedroom so I can just copy a bit of code). And, for completeness sake, I'll write code for flushing the toilet as well. That should be all. Oh, no, wait, what if someone wants to put something *in* the toilet. Well, it's not actually unreasonable so I'll write some code for flushing documents away, but given the variety of objects in the game (e.g. The Britmobile) I don't want to have to decide individually which ones can and which ones can't fit into the toilet so I'll just add a message for everything else.

The toilet could be used to dispose of paper items, but nothing else will fit.

It's a bit of a short cut, but I want to eat sometime today, and I don't actually want some player dropping the car keys into the toilet bowl just to see if I coded it.

Next, given it's a bathroom I'll move onto the bath itself. Continuing the theme of poor hygiene it is described thus:

The bath was once white enamel, with white tiling around it. Now it's grey and has mold growing on it above the ring ingrained into it. The plug hole is full of matted hair. There is a shower fitting above the taps and a curtain to pull around the bath to keep the steam in.

I'm not entirely sure of the grammar, and my spell checker doesn't like mold – I thought it was mould if you were shaping something and the stuff that grows on damp surfaces was mold. Wonder what my beta-testers will make of it. Unfortunately, the description of the bath includes a shower, a ring, a plughole, taps, a curtain and mold (or possibly mould). So that's six more descriptions needed. Seven if you want both taps but I think I'll just describe 'taps' and direct you to the same description whether you want to look at the hot tap or the cold tap. I'll give you the plughole description.

The filter seems to be missing so this is just a dark and odoriferous hole, full of hair and other disgusting matter. British Fox shies away from it, it looks like some kind of mutated sewer beast could come crawling out of it any minute. There's no sign of an actual *plug, but then, it probably wouldn't stop anything climbing out anyway.*

Not sure about 'filter' – what *do* you call the bit of metal with the holes in that allows small stuff to be washed down the drain but prevents you getting your big toe stuck (or, for that matter, prevents you washing dirty great spiders away). Notice the cunning way I avoided having to write a description for another object by saying there isn't a plug.

Now, however horrible it is, it's certainly possible to get into the bath so it's going to need some work to allow you to do this. Fortunately there's a bath in Navy Fox's room at The Institute so I can borrow some code from there, and by defining it as a bed and using Sir Gareth's bed.t much of the heavy lifting is done. Make sure you're allowed to 'stand' in it as well as lie and sit and ensure that the preposition is 'in' rather than 'on'. On the other hand, what if someone wants to actually take a bath. Well, they can't. But I'd better tell them why not, especially since you can use the baths and showers at The Institute (which does at least mean that most of the code is already written).

Even Navy Fox wouldn't be willing to bathe in this *bath.*
While taking a shower here is a marginally less unappealing prospect than taking a bath it's still out of the question.

I *could* write code for filling the bath with water (especially since it's already done for Navy Fox's bath) and have alternate descriptions, but instead I'll just explain why you can't fill the bath.

<<actor.sdesc>> turns the taps and some slightly brownish water spills out. She makes a face and shuts the tap off.

(<<actor.sdesc >> copes with the possibility that a character other than British Fox might be turning the taps).

So, surely the bath is finished. Oh, wait, there's a curtain which some joker is bound to want to pull. So that's code for opening, closing and pulling the curtain. If I was obsessive I'd also write a verb for 'drawing' the curtain, alternatively I could add 'draw' as a synonym of pull (although that might create some odd quirks if there's anything else in the game which you need to pull). (*If* I was obsessive?) And another description for the bath itself…

Thankfully, the view of the bath is obscured by a shower curtain.

Now we're getting somewhere. Really just the sink to do, and that's just a cut down version of the bath.

Not quite as disgusting as the bath, the sink would still benefit from an hour or two's work and a bottle of bleach. The enamel is splashed with toothpaste, shaving foam, soap and hair. A mirror is attached to the wall above the sink.

Well, I've managed to avoid mentioning anything new except for the mirror. I could just have 'enamel' as a synonym for the sink itself, although I've used it in the description of the bath as well, either way I don't think it deserves a new object. There are, of course, taps even though they're not mentioned, and a plughole. These would need to be disambiguated from the bath taps and the bath plughole, so I'll create the taps since 'sink taps' is perfectly reasonable but I don't like the sound of 'sink plughole' and 'bath plughole' so I won't bother with the sink one at all. This also means I don't need to worry about a plug for it either. The code for the taps can be cannibalised straight from the bath along with a more or less identical reason for not filling or using the sink.

<<actor.sdesc>> looks at the sink and makes a face. Cleaning up can wait until she gets home.

We're on a roll. The new object (the mirror) is the work of but a moment.

The mirror is misty with condensation and gives only a distorted reflection, British Fox is tempted to wipe it clean, but she's not sure what might end up sticking to her hand.

We can see the end in sight now, although we did mention bottles of cleaning stuff in the main room description.

Every available flat surface is scattered with bottles and tubes of soap, shampoo, deodorant, shaving foam, toothpaste, mouthwash, moisturiser, shaving balm, lotion, aftershave, shower gel, hair gel, shaving gel, sunblock, nail polish remover, cleansing fluid, hair dye... Clearly, no one actually uses anything up, *when it gets to half empty they just get a new bottle. Notable by its absence is any form of bleach or cream for cleaning the bathroom itself.*

Of course, now the player, just to see if I've *really* made all that stuff, is going to x soap, x shampoo, x deodorant…. However, I'm smarter than they are (I hope) and all these are valid synonyms for the bottles so they just get redirected back to the same

description. It does mean that you can x hair polish gel, or x shaving nail cleansing sunblock and still get the same, but I can live with that.
That's when I remember the window. Windows are a pain and are notable by their general absence in my games. Obviously, all my characters are photo-sensitive troglodytes and like living in the dark. However, I've mentioned it now.

A frosted glass window. The sill and frame were once white but are now grey and covered in the ubiquitous mold.

Unfortunately, now it's there you can look through it. Fortunately, it's frosted glass so this just gives a 'you can't see through the frosted glass' message. (Well, it doesn't automatically give that, I still have to code it, but it's pretty straight forwards). Unfortunately it being a window you should be able to open it. Fortunately, I can determine that it's painted shut. Unfortunately, it's only glass which even *I* could break, let alone British Fox. Now, I could explain that she won't commit criminal damage, but this *is* the villain's base. So I give in, and write breaking the window into the code, with different descriptions depending on who's doing it.

British Fox punches her fist through the glass, shattering it. A fresh breeze (well, as fresh as you get in Leeds), blows in.
"Good idea," says Ayesha. She picks up a towel, places it across the window and hits it hard with her elbow. The glass, which was clearly installed before modern safety regulations, breaks into shards and a fresh breeze (well, as fresh as you get in Leeds) blows in.

Towel? A towel now? We'll come back to that. With an open window you finally *can* look through it, so there's a description for that, along with an alternate description of the window itself once it's broken. A debate with myself ensues about whether to allow the character to climb out the window, which will probably entail creating a new room to land in when you jump down from the first floor (that's second floor to you Americans), so I decide the window is just too small to get through. Although I do have to create a new verb for 'climbing out' a window, just so that I can tell you you can't do it. Finally (for the window) I decide that the glass itself needs a separate description given it can exist in two states ('broken' and 'still part of the window').

Foolishly I just mentioned a towel, but since they're also mentioned in the main room description I was going to have to cover them anyway.

Towels and face cloths are hanging over the edge of the bath and sink, shower curtain rail and radiator. Others are on the floor. Some are even hung over the towel rail.

Fuck fuck fuck!!! A radiator, a shower curtain rail and a towel rail. Well, none of these should be a problem. If the towels were proper items they would probably have to be some sort of surface or container which can only hold towels, but since I'm just leaving the towels as decorations as opposed to objects the railings can be the same.

A rail for holding towels. There are even a couple on it, but they're clearly not clean.
A radiator for heating the room although it's primary use seems to be for holding towels.
A rail running around the bath at head height. It looks fairly precarious and one wouldn't want to hang anything heavier than a face flannel over it. Which does seem to be what it's mostly being used for.

And I think we're finally done. A few tweaks, like code for trying to actually *clean* some of the stuff, or pulling the shower curtain off the wall and it's finished. None of this has been terribly complicated, unlike some of the villains' own rooms (say Purity's in which I decided to code what happens if you kill her favourite stuffed animal while she's in the room, and all the knock on effects of 'what if she defeats British Fox?' 'what happens if she's defeated and isn't available for her big scene later in the scenario?'). It's taken all day and 383 lines of code for 1 room, 19 other objects and 2 new verbs which serve no function in the game. Nothing can happen there, the bad guys don't go in, there's no information that's of any use (apart, possibly, from discovering that your boyfriend's a wanker)…. Given that the room could, without detriment to the game, have been described as:

The villains' bathroom is a disgusting mess. Everything is filthy, it's obvious no one has cleaned anything for weeks, nothing is put away and the whole thing is a health hazard which makes British Fox want to contact the Environment Protection Agency immediately.

Estimated completion date for British Fox and the Nationalist Conspiracy is probably sometime in late 2011.

xxx
Lucilla

How Much Is Too Much?

By Purple Dragon

Welcome to the second installment in my "How ___ is too ___" series. Actually, when I wrote the one last month, I had no intention of turning it into a series, and I still don't, but after reading Lucilla's article, which you have hopefully just got through reading, I couldn't resist adding a few observations and opinions.

We have all heard comments, or even made or felt like making them ourselves, like the one Lucilla quotes at the beginning of her article. We have all played games where the environment is sparse to say the least, where the room descriptions are short, and even then, few if any of the objects listed are available to be examined, much less interacted with in other ways. While this does not seem to bother some people at all, there are others who seem nearly outraged by it. Although I will not immediately dismiss a game that does this, I still believe that anything mentioned in a room description should have at least a basic description of its own. I think this is an opinion that many people in the community would agree with, and though it is a bit of a subjective matter, most can seem to agree when a particular game has broken this unofficial rule. But what about the other side of the coin? How much is too much?

This is a much trickier question. How well implemented should the environment actually be? How much detail should there be? A lot of people would be tempted to answer, "As much as possible," but I don't really know if that is the best answer. Do players really want to deal with a fully implemented environment? How much work can an author do on this aspect of a game before the work becomes counterproductive?

Let's take a look at Lucilla's bathroom example. This is one of the best, if not THE best, most thoroughly implemented bathrooms that I have ever seen in a game, but has she handled every possible option? Of course not. Take those bottles sitting all over the room for example. All the room description tells us is, *"there are a great many bottles of cleaning products"* so at the least, we should be able to examine "bottles" and "cleaning products." Of course, then we find out some more specific information about what is there. Lucilla handles this situation by making all these new objects synonyms for the cleaning supplies, which I think is more than fair, but we're not talking about fair here, we're talking about a fully functional environment.

In such an environment, we would need to be able to examine each of these things individually. In fact, examining really wouldn't be enough if we are trying for full realism. We would also need to be able to take them all, use them all, etc. Wash up with the soap, brush our teeth with the toothpaste, gargle with the mouthwash, paint our nails, dye our hair, etc., etc. What happens if we turn on the water in the tub and pour in some shampoo? We should get bubbles right? What about the toothpaste? Probably no bubbles. So we have to figure out which of the cleaning supplies produce bubbles, and how much.

This is, of course, a ridiculous example. At least I hope you found it a bit ridiculous since I think you have little to no chance ever seeing that level of detail in any game. But come on now, be honest, if an author actually went to the trouble of doing all that work would it even matter? Would you really want to spend your game time picking up every bottle of cleaner in a bathroom and dumping into a tub to see how many bubbles you got? Keep in mind that this isn't a puzzle, there is no reward for figuring it out. Once you have a bathtub full of bubbles a beautiful girl doesn't jump out of it or anything. This is just detail for detail sake.

I think it's safe to say that very few game players would want this level of detail. I don't say none here, but very few. This level of detail is liable to get people complaining about busy work and pointless puzzles when the intention was just to make the environment fuller. In the end, I think it would annoy nearly as many people as not having enough objects, and it would certainly be a lot, lot more work for the author. As is so often the case, I believe that the best choice lies somewhere in the middle. So here are my opinions on what a game should have in general.

1. I do think that any object mentioned in a room description should also have a separate description. If you walk into a kitchen and it mentions a sink, then the game should also reply to "X sink."

2. Under most circumstances, I don't think there is any benefit to going more than two levels deep with any object. Here is what I mean by that. Just like some of the examples that Lucilla gave, the kitchen sink might mention other objects. For instance, the faucet, the drain, etc. Then the description of the faucet might mention the spout, the handle, etc. I think it's more than ok to make these synonyms of the faucet, and in reality, it would probably be ok to make the faucet and drain synonyms of the sink. In this way, the player isn't typing something and getting annoying or untrue statements like "I don't see any faucet," but the author isn't forced to describe the lawn down to every single blade of grass (so to speak).

No matter what authoring system is being used to create a game, synonyms are pretty easy things to handle, and just using this one trick will make a game feel fuller, even if it isn't.

3. I don't think it's always fair to cut down on the number of object you need to describe by just not mentioning them in the room description. What do I mean by that? Let's say that you are putting a bathroom in your game. You have no desire to go the route of Miss Frost's fully functional bathroom, but you do have a really good idea for a hot shower scene (or a cold shower scene, or maybe a bathtub filled with jello…hmm…sorry, got distracted there for a minute, what was I saying?). Oh yeah, just because you don't mention a toilet or sink in the room description, doesn't mean you shouldn't put them in. Have you ever been in a bathroom that didn't have a toilet and sink? Even if there were some reason that you really didn't want it to have one, you would still need to spend the time to explain that reasoning to the player, which would take at least as much time as just adding it. The bottom line is that bathrooms have toilets, kitchens have refrigerators, bedrooms have beds, etc. If it's common sense that something is supposed to be in a room, then put it in. You don't have to go to any extreme lengths, but some basic object descriptions don't take all that long. A lot of times I have a handful of objects in my games that aren't mentioned anywhere at all. It isn't important to the game, serves no practical function, but since I would expect it to be there, I add it. Of course, not everyone is going to agree on what SHOULD be in any given room, but just adding a few of the more obvious ones will go a long way.

4. My last point isn't so much about what should or shouldn't be there as it is a piece of unsolicited advice to anyone who is reading this and thinking about writing a game. Do NOT save this for the last thing. The temptation is to create all the objects you really need in the game, create the characters, write the sex scenes, and then go back and fill in the details. The problem with this approach is that the details are usually the most boring things to write. Saving them till the end, when I guarantee that you will be at least a little tired of the game as a whole anyway, there is a much better chance that they will just never get in there at all. When you are several weeks or months into working on a game, and you have something that is 95% done and plays all the way through, the words, "good enough" start sounding mighty nice.

Of course this is all just my own opinion. Don't agree with it? Feel free to write up your own and send them into me, I'll print them next month, (seriously, I will) but that is how I see matters. Games without enough objects feel bare, devoid of life. But too many objects can be simply overwhelming. This is, after all, a game that we're talking about. I've never in my life squeezed toothpaste into my bathtub to see if it bubbles, I certainly have no desire to do so in a game. ◆

Welcome to Coder's Corner. Those of you who have been following along with Programming Erin will find the format familiar, and our goal is basically the same, to compare the different game authoring systems. There are a few differences between the two features however. The biggest one is that while Programming Erin followed a single (very simple) game from start to finish, the new feature will focus on one standalone task each month. About the most continuity that you are likely to see is possibly a two part issue on some of the more complicated topics. Other than that, each month will be its own task.

We encourage you to get involved by submitting tasks of your own. If you need immediate help on a game you're writing, your best bet is probably still to go to one of the boards and ask there, but if you have something that you've been wondering about, might want to use in a future game, or would just like to see compared between the five systems, then send it in. If you're curious about it, you can almost bet that others are as well.

That's enough chit chat, on with our feature. This month's topic is timed events. In other words, how to make things happen at certain times or for a specific number of turns. I'm printing out the assignment that I sent to the staff, just like I gave it to them. Not only did they demonstrate the techniques, but they came up with a variety of scenarios in which to do it. My hat's off to them, and I hope you enjoy it.

"The player starts at the top of a flight of stairs. The only apparent exit is down the stairs. If he does not go down the stairs within 3 turns, the stairs tilt, forming a slide that deposits him at the bottom. The wall closes, blocking the stairs/slide. He is now in a room with no visible exits, and the walls start closing in. He has 10 turns to stop the walls or find a way out before being crushed to death."

TADS 2 Segment by A. Bomire

This month, we are discussing timed events. This is an extremely useful tool in any game, not just AIF. In TADS, timed events are known by two names: fuses and daemons. A fuse is a timed event that takes place at a specific time, just once. Think of it like the fuse on a bomb. A daemon is a timed event that takes place over several turns. A good example is the atmospheric messages you see with some characters. TADS has several functions that you will make use of when starting, stopping and checking timed events:

notify (object, &event, turns) - starts timed *event* after *turns* number of turns; if *turns* is zero, starts the *event* now and executes it again every turn until the *event* is stopped.
unnotify (object, &event) - stops timed *event*
getfuse (object, &event) - returns the number of turns until timed *event* executes; if *event* is not currently active, then it returns nil.

(Note: the ampersand (&) is important, as it tells TADS not to evaluate the *event*, but to just refer to it. This way, it won't execute at the wrong moment.)

Using these three functions, you can implement just about any timing function you can think of. While with creative programming, you can probably get away with just one master event for your entire game, TADS is flexible enough to allow you to have several fuses/daemons running at the same time. (There is a limit, however, on how many you can run. I believe it is 256, which is more than enough.)

The event is just a property of some object in your game. I usually define it for the object to which it refers. But, you can define your event for just about any object in your game. For some games, I'll create a master object that holds all of my timing events.

Let's see an example. I'll create the prototypical bomb, and an event describing what happens when the bomb explodes:

```
bomb: fixeditem
  location = BombRoom
  sdesc = “bomb”
  ldesc = “It’s a bomb, and the fuse is lit! “
  noun =  ‘bomb’
  boom =
  {
    “The bomb explodes, and you die. “;
    die();
  }
;
```

The “boom” property you see above is the timed event. Right now, it does nothing. At the appropriate part of the game, I will “light the fuse” by starting the event. In this case, I will set it to go off in 5 turns:

```
notify (bomb, &boom, 5);
```

Should the player manage to defuse the bomb, I can stop the bomb from exploding using *unnotify*:

unnotify (bomb, &boom);

Suppose the bomb had a visual timer that the player can examine. I can use *getfuse* to determine how much time is left until the bomb explodes:

```
"There are <<getfuse(bomb, &boom)>> seconds left. ";
```

Now, for a more extensive example. In this case, the player is exploring a haunted mansion, carefully making his way from room to room. He comes to a stairway leading down to the basement, and begins going down. After three turns, the stairs collapse, turning into a slide which catapults the player into the basement. Of course, if the player goes into the basement on his own, this doesn't happen. Let's create a room which will be the stairway. Upon entering the room, the timer begins (using TADS *enterRoom*). If the player leaves, the timer ends.

```
Stairway: room
 sdesc = "Stairway"
 ldesc = "This creepy stairway leads down into darkness.
    The stairs creek with every step you take. "
 down =
 {
  "You exit the stairs into a damp basement. You look
  around yourself, and are startled when the stairs
  behind you collapse into a ramp. You study it, finding
  it too steep to climb. As you puzzle how to exit the
  basement, another sound startles you. With a horrible
  grinding noise, the walls to either side of you begin
  moving closer and closer. If you can't figure out how
  to escape, you'll soon be crushed between them! \b";
  notify (Basement, &crushed, 10);
  notify (Basement, &backgroundNoise, 0);
  unnotify (self, &collapse);
  lever.isActive := true;
  return Basement;
 }
 collapse =
 {
  "\bWith a groan, the stairs collapse, forming a wooden
  slide. You scramble to keep your feet, but you quickly lose
  your footing and slide down the ramp. With a bump, you
  find yourself in a damp basement. You stand and dust
  yourself off. You examine the stairs, which are still a
  wooden slide, trying to figure out how to get out again
  when you are startled by yet another sound. With a
  horrible grinding noise, the walls to either side of you
  begin moving closer and closer. If you can't figure out how
  to escape, you'll soon be crushed between them! \b";
  notify (Basement, &crushed, 10);
  notify (Basement, &backgroundNoise, 0);
  lever.isActive := true;
      Me.travelTo(Basement);
 }
 enterRoom(actor) =
 {
  inherited.enterRoom(actor);
  notify (self, &collapse, 3);
 }
;
```

You'll notice that I refer to three different timed events: *collapse, crushed, and backgroundNoise*. The *collapse* event collapses the stairs and sends the player into the Basement. At the same time it starts up two other events: *crushed*, which will crush and kill the player within 10 turns, and *backgroundNoise* which will display messages of the walls creaking and squeaking as they slide closer. Since the player can enter the basement in two ways (by waiting until the stairs collapse, or by going down the stairs into the basement), I am starting these two events in two different places. Now, to actually create the Basement:

```
Basement: room
  sdesc = "Basement"
  ldesc = "This damp and dank basement is featureless, with
```

```
   the exception of a dark brown wood lever against the wall.
   <<getfuse(self, &crushed) ? “Well, there are also the walls
   which are closing in on you. There is a stairway that leads
   upwards, but it is currently collapsed into a steep ramp.” :
   “There is a stairway leading upwards.” >> “
  up =
  {
   if (getfuse(self, &crushed) )
   {
    “The collapsed stairway is too steep to climb. “;
    return nil;
   }
   else return Stairway;
  }
  crushed =
  {
   “\bThe walls, which have been steadily getting closer and
   closer, are finally touching you. You press against them
   mightily, defiantly attempting to gain a few inches of
   space, but it is to no avail. With a last CLANG, the walls
   crush you. “;
   die();
  }
  backgroundNoise =
  {
   “\bWith a grinding noise, the walls inch closer and closer. “;
  }
;

lever: fixeditem
  location = Basement
  sdesc = “lever”
  ldesc = “Situated in the middle of the wall is a wooden lever,
   which is currently <<self.isActive ? “pushed upwards” :
   “pulled downwards”>>. “
  noun = ‘lever’
  adjective = ‘wooden’ ‘dark’ ‘brown’
  isActive = true
  verDoPull(actor) =
  {
   if (not self.isActive)
    “The lever is already pulled downwards. “;
  }
  doPull(actor) =
  {
   “You pull on the lever. The dampness of the basement has
   caused it to stick and you have to struggle to get it to
   move. Finally, though, it slips downwards. Immediately,
   the walls stop their encroaching movement and begin
   sliding back into their original position. After a few
   moments, they CLICK into place. At the same time, the
   stairway returns to normal, allowing you exit from the
   room. “;
   self.isActive := nil;
   unotify (Basement, &crushed);
   unotify (Basement, &backgroundNoise);
  }
  verDoPush(actor) =
  {
   if (self.isActive)
    “The lever is already pushed upwards. “;
   else
    “Don’t do that! Do you want to be crushed? “;
  }
;
```

You may notice that the above scenario is a continuous repeating loop. When the player enters the stairway, the countdown begins for the stairway to collapse. Whether the player enters the basement or the stairway collapses to dump him into the basement, he has 10 turns to pull the lever and stop the walls from closing in. The only way out is the stairway, which again starts the process over. In a real game, the player would be given the option to return from whence he came before entering the stairway.

TADS 3 Segment by Knight Errant

This month we’ll be addressing timed events in TADS 3. TADS 3 has several ways for dealing with timed events. If it’s a timed event involving NPCs, it may be best to utilize the NPC’s Agenda. If it’s a repeating timed event, a Daemon would be the option to use. However, this month Purple Dragon has given us simple timed events. They’re triggered and then some turns later they fire, much like a fuse burning down before the dynamite explodes. This case is exactly what the Fuse class was designed for.

Of course, before we can get to our timed events, we need to set our scenario. That means we a PC, rooms and intro text. Since we already covered this in *Programming Erin*, I’ll just skim over it. For more detail, read the June 2008 issue of *Inside Erin*. First, the intro text, the starting room, and the PC:

```
        gameMain: GameMainDef
  initialPlayerChar = me
  showIntro()
  {
   "<center><b>Breakdown</b></center>\b
   You were a normal person once.  Just a few weeks ago you laughed at the nutcase on
the street, mumbling to himself about things only he could see.  You never imagined it
would happen to you.  Still, it could be worse.  The mind has a wide range of defense
mechanisms for traumatic events, right now it’s just taken itself off the hook and told
reality to call back later.  Of course, that all depends on if you can find a way out of
your own head.  ";
  }

  showGoodBye()
  {
   "<.p>Thanks for playing!”;
  }
;

startRoom: Room ‘Small room’
  "The concrete walls of the small room give a distinctly <q>institutional</q> feel to
your surroundings.  To the south, a steel staircase leads down into the dark.  There’s
nowhere else to go, but the brown-red splatters on the stairs do nothing to ease your
nervousness. "
down=downStair
;

+ me: Actor
  "You are you.  You know all about yourself, but your mind shies away from providing
any more information.  "
;
```

A word of warning, this is not going to be a happy game. Luckily, it's short. Now that we have the basics, let's add the stairway. In TADS 3, passage objects like stairs, doors or tunnels are represented by two objects, one in each of the two rooms. In this case, they'll be downStair and upStair. I've coded them like this:

```
          + downStair: StairwayDown 'bloodstained metal stair/staircase' 'metal
          staircase'
   "The metal staircase leads down into the dark.  It's shiny surface is marred by
brownish-red splattering."
  noteTraversal(traveler)
  {
   "The staircase creaks and groans as you climb down.  Old rusted bolts fall and clatter
to the floor.  Just as you consider turning back, the upper end of the stairway breaks
free from the wall.  You leap off the stairs as the old metal collapses into a pile of
twisted beams.  ";
  }
;

++ driedBlood: MultiInstance, Decoration 'brown red blood/splatter' 'brown-red
splatter'
  "It doesn't appear to be paint.  "
  notImportantMsg = 'Something about the splatter makes you wary of taking a closer
look.  '
  locationList = [slaughterHouse, upStair]
;

slaughterHouse: Room 'Slaughterhouse'
  "This large concrete room is barely illuminated by the flickering of an old flourescent
bulb.  The floor is slightly sloped downward.  At the lowest point, in the center of the
room is a small drain.  More dark red stains spike out from around it, surrounding the
drain like a starburst.  The heap of scrap metal that used to be the staircase is piled
in the corner."
  up=upStair
;

+ upStair: StairwayUp -> downStair 'bloodstained metal stair/staircase' 'metal
staircase'
  "The twisted metal beams that used to be the staircase lie in a heap.  They resemble
the ancient bones of some long-dead giant.  "
  canTravelerPass(traveler)
  {
    return nil;
  }
  explainTravelBarrier(traveler)
  {
    "The staircase is completely destroyed. ";
  }
;
```

Much of this code will resemble the Door code from Programming Erin, but there are a few new methods here. noteTraversal() is a method called when the RoomConnector is being used. In this case, all it has to do is provide a description of travel, but it's also possible for it to change the game state. In the upStair, canTravelerPass and explainBarrier do exactly what they say. In this case, canTravelerPass simply returns nil (meaning no), but it's also possible for it to be conditional, allowing travel if something is true and disallowing it if it is not.

Now the player can get down to the slaughterhouse and get stuck, but in case he's less then eager to go down there, we'll set up a timed event forcing him to go. TADS 3 provides a roomDaemon for rooms. Normally this is used to fire an atmosphereList, to provide “atmosphere messages”. In this case, however, we can utilize it to trigger forcing the player to move. Here’s the new room definition.

```
          startRoom: Room ‘Small room’
              “The concrete walls of the small room give a distinctly <q>institutional</
          q> feel to your surroundings.  To the south, a steel staircase leads down into
          the dark.  There’s nowhere else to go, but the  brown-red splatters on the
          stairs do nothing to ease your nervousness. “
               turns=0
  roomDaemon()
  {
    turns++;
    if(turns==3)
    {
     “<.p>The room shudders as if in an earthquake.  With a loud rumble, the whole room
tilts, sending you tumbling down the stairs.  The metal staircase creaks and bends as
the room shifts, finally breaking free of the wall and collapsing into a heap.  As the
dust settles, you pull yourself to your feet.  <.p>”;
      me.moveIntoForTravel(slaughterHouse);
      slaughterHouse.lookAround(me, true);
    }
  }
  down=downStair
;
```

The only new code here is slaughterHouse.lookAround. This simply triggers the same result of a Look command in the room slaughterHouse on behalf of me. We need to do this explicitly because moveIntoForTravel simply moves the PC without triggering any normal travel mechanisms.

Now that we’re in the slaughterhouse, the next part of our scenario is the standard “walls closing in” scenario. Honestly, we could accomplish this task with the same roomDaemon, but what fun would that be? Instead, we’ll use a normal Daemon. Since we have two ways to enter the room (going down the stairs, or being pushed by the roomDaemon), we’ll need to call it in both places. We’ll use this code: `slaughterHouse.shrinkRoomID = new Daemon(slaughterHouse, &shrink, 1);` This will initialize a Daemon in the slaughterHouse object as the property shrinkRoomID. The three properties of the Daemon are the object the Daemon is associated with (slaughterHouse), the property the Daemon will trigger (shrink), and the time interval it will be firing on (1 means it fires every turn, beginning on the next turn). So every turn after the player enters the slaughterhouse, it will trigger the shrink method. Here is the shrink method:

```
          slaughterHouse: Room ‘Slaughterhouse’
  “This large concrete room is barely illuminated by the flickering of an old flourescent
bulb.  The floor is slightly sloped downward.  At the lowest point, in the center of the
room is a small drain.  More dark red stains spike out from around it, surrounding the
drain like a starburst.  The heap of scrap metal that used to be the staircase is piled
in the corner.”
  up=upStair
  shrinkRoomID=nil
  shrinkTurn=0
  shrink()
  {
    switch(shrinkTurn++)
    {
      case 0: break;
      case 1: “<.p>A loud grinding sound comes from all sides.  “; break;
      case 2: break;
      case 3: “<.p>You may be crazy, but somehow the room looks smaller than it did at
first.  “;  break;
       case 4: “<.p>Concrete dust and fragments fall from the edges of the walls.  It
looks like the room is shrinking.  "; break;
      case 5: "<.p>The moving walls push the metal staircase fragments forward with a
loud clatter.  "; break;
     case 6: break;
     case 7: "<.p>The room grinds against each other, shifting inward.  You should get
moving.  "; break;
     case 8: break;
      case 9: "<.p>The walls move in further, leaving you only inches of room.  ";
break;
     case 10: "<.p>The walls and ceiling press on you from all sides, crushing you into
a dark red smear.  "; finishGameMsg(ftDeath, [finishOptionUndo]); break;
    }
  }
  down=hole
;
```

Finally, we have the switch statement. Switch statements are a good replacement for multiple nested if/else statements. In this case, every time the switch statement is called, it increments shrinkTurn by 1 and compares the result to each of the cases. The first case that matches will be triggered, and the program will continue until it reaches a break statement.

Now we've set up the timed puzzle, we need to implement a solution. We mentioned a drain in the room description, so we'll let the player smash it open with a metal beam from the destroyed staircase.

```
          + drain: Component 'drain' 'drain'
  "It's a small drain.  "
  attackCount = 0
  dobjFor(AttackWith)
  {
      verify(){if(gIobj!=metalBeam) illogical('That\'s not strong enough. '); else
logical;}
    action()
     {
     if(attackCount==0)
         {attackCount=1; "You smash the ground with beam, cracking the concrete. ";
exit;}
     else if(attackCount==1)
        {attackCount++; "You smash the ground again, breaking a hole in the concrete
leading down into the blood drain.  "; hole.makeOpen(true); drain.moveInto(nil);}
     else{"The hole is already big enough.  ";}
     }
  }
;

++ hole: HiddenDoor 'hole floor' 'hole in the floor'
  "You've smashed a hole in the floor, just large enough for you to fit through.  "
  destination = darkRoom
  noteTraversal(traveler)
  {
    "You escape through the floor before the room smashes you, but without knowing what
horrors you have yet to face."; finishGameMsg(ftGameOver, [finishOptionUndo]);
  }
;

+ upStair: StairwayUp -> downStair 'bloodstained metal stair/stairs/staircase' 'metal
staircase'
   "The twisted metal beams that used to be the staircase lie in a heap.  They
resemble the ancient bones of some long-dead giant.  Most of them are too heavy to be
```

```
useful<<metalBeam.moved ? '.' : ', but one metal beam is small enough to carry. '>>
<<metalBeam.discover>>"
  canTravelerPass(traveler)
  {
    return nil;
  }
  explainTravelBarrier(traveler)
  {
    "The staircase is completely destroyed. ";
  }
;

+ metalBeam: Hidden 'large heavy piece metal beam' 'metal beam'
  "A large metal beam.  "
  iobjFor(AttackWith){verify(){logical;}}
;

darkRoom: DarkRoom 'dark room'
  "It's dark."
;
```

The metal beam is of the Hidden class, which means it's not available until it's discover method is triggered. In this case, we trigger it when the player examines the remains of the staircase. When the player has the beam, he can use it to attack the drain. The hole is a HiddenDoor class. Like the Hidden class, a HiddenDoor is not available until it's opened. On the second attack, we open the door by calling hole.makeOpen(true). Since the hole is a Door, we have to create a destination. In this case, I've made a simple dark room. However, the end of the game is triggered when the PC enters the hole, so the destination is never reached.

Inform 6 Segment by 'trix

Coding timed events requires some kind of function that runs every turn, either to update its state or to count down to the timed event. The entry points that Inform 6 provides to do this are daemons, timers, `each_turn`, and `TimePasses`.

`TimePasses` is simply a function that a game can provide that will be run at the end of every turn. `each_turn` is a property routine, that is run every turn for any object in scope that provides it. Daemons are specified by giving any object a `daemon` property. Daemons are run even on objects that are out of scope, but only after they have been explicitly started, and until they are explicitly stopped. Timers are like daemons, but instead of running every turn, they wait a specified number of turns and then run once.

The first requirement in this month's exercise is a staircase which, after a fixed number of turns, changes into a slide. That sounds like a good situation for a timer. Here's the starting point for our gamelet.

```
Constant STORY "THE STAIRCASE OF SINISTER DOOM";
Constant HEADLINE "^An example.^";
Include "Parser";
Include "VerbLib";

Object stairway "Stairway"
 with description "You are at the top of a long flight
     of stairs.",
    d_to traproom,
    u_to "There seems to be no way out upwards.",
 has  light;

Object -> stairs "stairs"
 with name 'stairs' 'stair' 'step' 'steps' 'staircase' 'stairway' 'flight' 'of',
   description "A long, slightly sinister looking flight
```

```
     of stairs, leading down into darkness.",
   before
   [;
    Climb: <<Go u_obj>>;
   ],
   time_left,
   time_out
   [;
    if (player in stairway)
    {
     print "With a frightening creak of gears, the staircase
      suddenly judders and starts to move under your feet.
      In barely a second, the stairs flatten into a slide,
      and you find yourself falling helplessly downwards
      into the unknown.^";
     PlayerTo(traproom, 2);
    }
   ],
 has  scenery;

Object traproom "TBW"
 with description "TBW"
 has  light;

[ Initialise;
 location = stairway;
];

Include "Grammar";
```

If you read Programming Erin, or you've written any Inform 6 before, that should all be pretty clear to you. The `time_out` property is the code to be executed when the timer runs out, and the `time_left` property is there to hold the number of turns left while the timer is running. The `PlayerTo` routine is the easiest way to move the player to a different place. Calling `PlayerTo(someroom,2);` moves the player and prints the room description as if the player had entered the room in the normal way.
You can compile and run it as it is, but the timer will not activate. Timers and daemons need to be explicitly started in the code. Since we're in the staircase at the beginning of the game, we can do that in `Initialise`.

```
[ Initialise;
  location = stairway;
  StartTimer(stairs, 2);
];
```

This will set the timer for the `stairs` object to go off after 2 turns (this actually means it will go off at the end of the third turn, slightly unintuitively).
The next task for this month is the old impending walls trap, which we're going to put in the presciently named `traproom`. You could view the impending walls as something like a timer, whereby the player gets crushed to a pulp when the time runs out. But timers only fire once. For the crushing walls, we need continual updates about the state of the walls to convey the threat to the player. So we're going to use a daemon.

```
Object traproom "Deadly Sinister Room of Doom"
 with description "You find yourself in a defiantly featureless room.
     Even the way you came in has unaccountably vanished.",
   daemon
   [;
    --(self.time_left);
    switch (self.time_left)
    {
     9: "You hear an ominous creaking from the walls.";
     8: "The creaking of the walls grows louder.";
     7: "The walls seem to quake slightly.";
     6: "The walls shake, and clouds of dust billow out
       from the corners of the room.";
     5: "Suddenly you realise that two opposing walls have
       begun encroaching on the room.";
     4: "The room visibly shrinks as the walls close in.";
     3: "You are getting very short of space.";
     2: "You barely have room to move.";
     1: "The walls are pressing in on you.";
     default:
       deadflag = 1;
       "The walls slam together with a final ~BOOM~, which
       you never hear.";
    }
   ],
   before
   [;
    Listen: "You hear an ominous creaking.";
   ],
   each_turn
   [;
    StartDaemon(self);
    self.each_turn = NULL;
   ],
   time_left 10,
 has  light;
```

Programmers might like to notice the unusual syntax of the I6 switch statement.
In the daemon, we manually count down the time left to the player's impending doom, and each turn, we print a threatening message, finally ending by killing the player. The daemon is started by the room's `each_turn` property. This will be executed the first turn the room is in scope (which will be the first turn the player is in the room). It would be executed on subsequent turns as well, but the line `self.each_turn = NULL;` removes that routine from the object so it won't be run again.
Since we keep mentioning the walls, we'd better put some in the room.

```
Object -> walls "walls"
 with article "the",
   name 'wall' 'walls' 'impending' 'opposing' 'stone',
   description "They look like ordinary stone walls... OF DOOM!",
   before
   [;
    Push, PushDir: "You cannot move back the walls. They are
     controlled by ancient mechanisms... OF DOOM!";
   ],
 has  scenery pluralname;
```

The player is certainly under some threat now. Reluctantly I must provide her with a means of escape. This month I'm only going to save bona-fide IF fans, by putting an object in the room that is a clue on how to escape.
The way to stop the daemon is a routine called (ahem) `StopDaemon`. You can stop and restart daemons as the mood takes you.

```
Object -> blackrod "black rod"
 with name 'black' 'rod' 'rusty' 'star',
   description "A three foot black rod with a rusty star on one end."
;
```

```
Object wellhouse "Inside Building"
 with description "You are inside a building, once probably some kind
      of well house for a spring of some size and description.
      It's not much to look at now, but at least it's better
      than that room with the impending walls... OF DOOM!
      ^By the way, the way out is to the south.",
    s_to
    [;
     deadflag = 2;
     "Ah, fresh air at last! Now to find my hacksaw
       and get back to murdering. ";
    ],
    out_to [; return self.s_to(); ],
 has  light;

[ XyzzySub;
  if (location==traproom)
  {
   StopDaemon(traproom);
   PlayerTo(wellhouse, 2);
  }
  else if (location==wellhouse)
  {
   PlayerTo(traproom);
   traproom.time_left = 0;
   StartDaemon(traproom);
  }
  else "Nothing happens.";
];

Verb 'xyzzy'  *  -> Xyzzy;
```

So let's recap on what we've learned.
`TimePasses` is very simple, but not that useful since it can be provided only once per game.
`each_turn` is useful for having things happen when the player is nearby.
Timers don't rely on scope, but they don't run while they're counting down, only when they hit zero.
Daemons run every turn, and don't rely on scope, but they do have to be explicitly started and stopped.
And finally, it's fun to crush people with gradually encroaching walls in IF as well as in real life.

Inform 7 Segment by Dudeman

Hello everyone. Welcome to my first entry into the Coder's Corner which also happens to be my first entry into Inside Erin. I will be covering the various topics covered by this feature using Inform 7, a very nifty coding language that has been gaining some attention recently. So if you are learning Inform 7 or are just reading this to get an idea of what the language is like, I hope you find my segments helpful and enjoyable.

Now this month's topic is timed events. Many times in IF, or more specifically AIF, we would like certain things to happen, but only for a certain amount of time or turns. These are what we call timed events. Now there are a few different forms of timed events and a few slightly different ways to handle them so I am going to show you how you could go about doing them in a very small, short sample game. It's very bare minimum, but should give you an idea of how you can go about doing something similar in your own game.

Now, let's start off by going over what we want to happen in our game. For a starting scenario, let's say that the PC has gone into a haunted house in search of his friend (who is of course a hot girl, this is AIF after all) who went in on a dare and didn't come back out. Searching through the house, the PC comes upon a bedroom and when he enters, the door slams shut locking him inside. Now the only way out of the room is for the PC is to go down a mysterious staircase and towards an eerie light coming from a creepy stone cellar below. For simplicity we will start out our game here and we can set it up using basic skills from past coding tutorials like the "One Night Stand Tutorial" and "Programming Erin" in previous issues of the newsletter.

```
After printing the banner text:
say "[line break]It has already been a good hour since your best friend, Julie, ventured into the old Baker house. The two of you had heard stories every since you were kids about the gruesome murders that took place there decades ago and how it has been haunted ever since. Julie was always convinced that it was just a story parents told their kids to keep them from playing around in the old dilapidated wooden house. It was this reason that Julie wasn't scared at all when she was dared to go into the house and bring back something from the master bedroom to prove she was brave enough to enter. However, you've never been so sure that the stories weren't true and got worried when she didn't come back after so much time. It took all the courage you could muster, but you final worked up enough to go into the old house and look for her.[paragraph break]When you finally reach the master bedroom having not seen or heard a trace of Julie your worry really begins to grow. You call out her name and look around the dusty old room, but no luck. Just as you are about to leave the room and look elsewhere, you suddenly hear the bedroom door slam shut behind you. You try with all your might to open the door, but it's no good. With panic setting in, you take a good look at the old bedroom once more to see what you can make of your situation.".

A Locked Master Bedroom is a room. The description of the locked master bedroom is "The master bedroom to the old Baker house is as spooky as you imagined it to be. There is no artificial lighting in the room and only a thin layer of moonlight from the cracks in the boarded up window allow you to barely make out the features of the old, abandoned room. However, in the darkness you suddenly notice a dim light coming from inside the open closet door which had previously been shut. Looking closer, you can see the light is coming from the bottom of a stone stairway at the back of the closet. You cannot see where it leads, but it appears to be the only way out of the room with the doorway locked shut.".

A Dusty Stone Cellar is below the locked bedroom. The description of the dusty stone cellar is "You are in a small, dusty cellar illuminated by several burning torches hung high out of reach on the thick stone walls which cover all four sides leaving no visible exit to the room.".
```

Now, for the first timed event. Let's say we want the player to go down to the cellar, one way or another even if he doesn't do it himself after a few turns have passed. With something like this, it's best to use an "Every turn" rule which is a type of rule that is automatically checked after every turn that it applies. Luckily, Inform keeps track of how many turns a player is in a location, so in this case, we can use an every turn rule to check if the player has been in the locked bedroom for 3 turns and if so will automatically move him down to the cellar.

```
Every turn while the player is in the locked bedroom:
if the player has been in the locked bedroom for 3 turns begin;
say "Hearing a strange noise coming from the stone staircase, you slowly approach it with caution to see if anything is there. However, as you reach the top of the staircase, the floor under you suddenly gives way and the staircase straightens to form a slide which you roughly fall down until you come tumbling into the room at the bottom. Before you can even get up, you hear a loud slamming noise and look up to see the doorway you just came through gone and replaced with nothing but a large stone wall.";
move player to the dusty stone cellar;
end if.
```

Now we also want a similar event to happen if the player voluntarily goes down to the cellar and then stop him from going back up either way. We can do that with a simple before rule for when the player goes to the cellar and defining that there is no room above the cellar that the player can move to. While we're at it, let's go ahead and put the PC's friend, Julie, down in the cellar too.

```
Before going to the dusty stone cellar:
say "You don't much like the idea of going down such a mysterious staircase in such a spooky old house, but you can't think of anything else to do. As you walk down, the light gets brighter and brighter until you finally enter a dusty stone cellar. However, as soon as you step into the room you hear a loud slamming noise and look up to see the doorway you just came through gone and replaced with nothing but a large stone wall.".

Above the dusty stone cellar is nowhere.

Julie is a woman in the dusty stone cellar. The description of Julie is "Julie has been your friend since you were kids and even you are surprised by how hot she grew up to be.". The initial appearance of Julie is "Julie is here with you in the cellar, obviously very scared and confused just like yourself.".
```

Now we come to our second timed event. Let's say we want the walls to start slowly closing in on the PC after he enters the cellar. Unless he can find a way to stop it, the walls will surly crush him after a short period of time, we'll say 10 turns. Now we can do this one in the same way we did before, but let's try doing it a little different by using a scene. Scenes are a good way to track time in a game because they introduce a unique starting and ending point that we can use regardless of what room the player is in unlike the previous method. By using an every turn rule, we can also print messages every turn reminding the player about what is happening.

```
Walls Closing In is a scene.
Walls Closing In begins when the player is in the dusty stone cellar.
Walls Closing In ends when the time since Walls Closing In began is 10 minutes.

When Walls Closing In begins:
say "'Oh Joe!' Julie cries as runs up to you and hugs you tight. 'I was so scared, I've been down here for so long and I couldn't find a way out.'[paragraph break]'Don't worry, I'm sure someone will come for us. We just need to wait until..,' you try to reassure her before you are interrupted by a loud rumbling noise and notice two opposite walls slowly moving in on the two of you. It would seem waiting for help isn't an option and the two of you need to find a way out fast!".

Every turn during Walls Closing In:
if the time since Walls Closing In began is less than 7 minutes begin;
say "[one of]The walls continue to creak and slowly move inward reminding you that you have only a short time until the walls crush you and Julie to death.[or]Julie clings tightly to you as the walls continue to close in. You better think fast if you are going to save Julie and yourself.[or]The sound of the stone's grinding as they move makes it hard to concentrate on anything but the thought of your impending doom.[or]'Oh Joe! I'm so scared,' Julie cries as the walls continue to close in on the two of you.[at random]";
otherwise if the time since Walls Closing In began is less than 9 minutes;
say "The walls are now so close that you and Julie have less than a foot to move around. If you don't do something really fast it is going to be too late.";
end if.

When Walls Closing In ends:
say "The walls continue to close in on the two of you until you barely have any room to move. You barely have time to say goodbye to Julie before you are slowly crushed to death.";
end the game in death.
```

You might want to note how I used "minutes" instead of turns on a few of those rules. In Inform a minute is by default 1 turn unless you manually change it. When talking about time you have to use minutes instead of turns, but it's the same thing so it's pretty easy. Also, you might notice the [one of][or][at random] format used in the every turn rule. This is a way of varying what text is displayed and you can learn more about that by reading the Inform 7 documentation chapter 5 part 6.

And with that taken care of we are left with the final missing piece. We need something that will stop the walls from moving and crushing the poor PC and his friend. Since this is an AIF newsletter, let's make it that when the PC and Julie have sex. As for why that would stop the walls? Well, let's just say that the ghost haunting the house was a lonely love starved person before they died and the embrace of two young lovers has moved it enough for it to spare the lives of our two heroes. We could go deeper into the plot if the game was longer, but we'll leave it at that for now.

```
Fucking is an action applying to one thing. Understand "fuck [something]" as fucking.

After fucking Julie:
say "'Julie,' you say taking her in your arms. 'I have something I've always wanted to tell you but was too afraid to do it before. I love you!'[paragraph break]'Joe, I love you too. I always have!' she cries. Forgetting your surroundings, you pull her in tight and give her a big kiss. As if consumed by passion, the two of you quickly strip out of your clothes and make love for the first time. In those few blissful minutes, you completely forget the impending doom threatening both your lives, but when you finally finish you both realize that the walls have stopped moving as you two were having sex. On top of that, the doorway you came through has suddenly reappeared. Not wanting to take any chances, you and Julie quickly grab your clothes and run up the stairs and out of the building.[paragraph break]You aren't sure exactly what just happened, but as you look at Julie panting next to you still butt naked, you can't help but smile at how things turned out.";
end the game in victory.
```

There we have it, the game is complete. Although of course if this were a real game we would want to add a lot more to make it more entertaining and filled out. None the less, I hope this gives you a few ideas of how you could incorporate timed events into your game using Inform 7 or at the very least had a good time reading it.

ADRIFT Segment by BBBen

Here's the intro: You are the great space adventurer Striker Starbright, and you have saved the beautiful android damsel SX1 from her cruel alien master. She has nothing to offer you in thanks except her robotic affections, and since it has been a long time since your last shore leave, you accept with gusto.

Okay, so we want the player to have three turns to kiss SX1 to kick off the action, or she will kiss him and start it up herself. Start off by creating a task called "kiss SX1". You won't need to do much with this task, just make it work in the relevant room (for this example we'll say it's a bedroom) then put in the following text for the "Message to show on completion" box:

> You and SX1 come together and your lips meet in a passionate kiss. After some initial fooling around she wraps her pale-blue thighs around you and whispers into your ear, "Striker, you are just so… `ERROR! Malfunction in servo mechanism. Leg spread mode cancelled. Reset model for system maintenance.`"<br><br> Suddenly you feel a crushing pain as SX1's legs suddenly begin contracting around you. You'll need to find some way to shut her down, and fast – or you're going to be broken by her mighty thighs!

Now create an event called "force kiss". First, on the "Timings" tab there's a section labelled "When should event start:". For this first timed event set that to "Immediately", then set the event to last between [3] and [3] turns. Then go to the "advanced" tab and down the bottom set it to "[Execute task] [kiss SX1]". Now the task will trigger in three turns whether the player does it or not, but it will not trigger if the player has already kissed her, because the task is not repeatable.

Next up we want the player to have ten turns to flip the switch on the back of SX1's head before he is crushed. There are probably a few ways to do this, but actually it can be done really easily with two tasks and an event.

First create a task called "thisendsthegame" (or whatever – just something the player would never type). In the message box type "You have been crushed by SX1's legs! You are dead!", then select the room for this task to be completed in, and in the "actions" tab create the action "Ends game with status Kills the player".

Next create a task called “flip switch”, set it to be completed in the right room and have it display the message “You flip the switch, and it shuts SX1 down.” You don’t need to do any more with this task for now.

Then create an event called “Mighty thighs”. Only show descriptions if in the appropriate room, and make sure that the event should last between 10 and 10 turns. Next, where it’s labelled “When should event start:” set that to “After task [kiss SX1] is complete”. Then go to the “Advanced” tab and up the top make it say “Event can be paused if [flip switch] is [Completed]”. Down the bottom select “[Execute task] [thisendsthegame]”.

Now for a little bit of colour, go into the “descriptions” tab and you’ll see two boxes that say “Display this [0] turns from event finish”. Set the number on the first one to 7 and the second one to 3. Then in the one that will trigger 7 turns from event finish, put the words “Her legs are getting tighter! You’d better work fast!”. In the box that will trigger 3 turns from event finish, write “You can feel yourself being crushed! You don’t have much time left!”. You don’t need to put anything in the box labelled “What to display on event finish” because there is already text in the “thisendsthegame” task that will display on the event finish.

That should cover it. Obviously you can put more depth and complex gameplay into this scene fairly easily, but I’m just showing you the basic mechanics. For instance, you could have the player have to rummage around in SX1’s hair to find the switch, and you could have SX1 say computer-stuff in response; those would be easy enough to implement, and I’m sure you can figure that kind of thing out for yourself. ◆

If you can write game reviews, articles, opinion pieces, humorous essays, or endless blather, we want you. Contact the Editor for suggested content or just write what you want and send it to us.

Submitting your work to Inside Erin:

Please direct all comments, articles, reviews, discussion and art to the Editor at aifsubmissions@gmail.com.

Editor:

Purple Dragon has written six AIF games including *Archie's Birthday - Chapter 1: Reggie's Gift, A Dream Come True*, and *Time in the Dark.* He has received one Erin award and been nominated for several others.

Staff:

A Bomire is the author of several TADS AIF games, including *Dexter Dixon: In Search of the Prussian Pussy*, *Tomorrow Never Comes* and *The Backlot*. His games have won numerous awards and Erin nominations. He was the co-recipient of the Badman Memorial Lifetime Achievement Award in 2006.

A Ninny is an AIF player, author of four AIF games and frequent beta-tester. His *Parlour* received an Erin for Best "One Night Stand" game in 2004 and his most recent game, *HORSE* walked away with three Erins at the 2007 awards show.

BBBen is an author of a number of Adrift AIF games. His games have received numerous Erin awards and nominations and first place in A. Bomire's 2004 mini-comp. He was also the recipient of the 2007 Badman Memorial Lifetime Achievement Award.

Bitterfrost is a longtime IF/AIF player working on his first (and last) game, *How I Got Syphilix*.

Dudeman has released one game and is working on a second. He has also released an impressive Inform 7 sex extension to help make it easier for others to write games of their own.

Knight Errant is an AIF player who has released one game and is currently working on a couple of others.

'trix has released one game, *Casting*, which was written in Inform 6, and is sporadically working on another in TADS 3.