INTRODUCTION

Greetings, Manorheads, and thank you already for the fantastic contributions you will have already made in the future!

This virtual leaflet is intended to help you the non-Inform-7-programmer write a design document that a) covers all of the necessary bases & b) can be translated into Inform code by Ryan or myself with a minimum of sobbing. It'll be fun!

PART ZERO: A THING YOU SHOULD PROBABLY KNOW REAL QUICK

Response to this project has been *fantastic* (yay thank you, personally!) and we're going to have a *lot* of rooms to translate into Inform code and make them all play nicely together in the same project.

To make this process easier (which means you'll get to play the game sooner!), Ryan and I decided that the most *basic* stuff -- things like the name of the room, what objects are in it, their descriptions -- might as well be written in standard usable Inform 7 code.

This maybe sounds a little daunting, but here's an example of the kind of thing I'm talking about:

The Kitchen is a room. A spatula is in The Kitchen. The description of the spatula is "It's just a spatula, it's not going to hurt you."

For anything complicated, just explain in standard English, as clearly as possible, what you would like to have happen -- and enclose this explanation within square brackets [], Inform's comment symbol. So, for example, you might write:

[The player has seventeen hit points. Every time they interact with the spatula in any way, they lose a hit point. No message is ever displayed to indicate this. When their hit points reach zero, display the words "YOUR MAMA ALWAYS TOLD YOU NEVER TRUST A SPATULA"; end the game.]

Inform can do *a whole lot of things,* and odds are good that it can implement the cool idea you've had. We do, however, think it'd be useful for you to have a sense of the sort of things it was designed to do and expects people to do all the time, in case there's a feature you didn't know you could utilize. This document attempts to explain a bunch of that stuff, in addition to providing basic syntax.

BIG NOTE: If you genuinely don't want to write even the tiniest sentence of Inform 7 code, that is also fine! Just send us all of the same information written in non-Inform-7 standard English (and let us know if English is not your first language and you want grammar assistance!)

PART ONE: ROOMS AND HOW THEY DO

OPTIONAL EXPLANATION OF WHAT A "ROOM" IS IN PARSER IF

If you're unfamiliar with parser IF semantics, a "room" can be an actual room like a kitchen or the bathroom of a honeymoon suite, or it can be a conceptually self-contained space like a graveyard or the northwest corner of a winding sewage labyrinth.

For our purposes, the primary distinguishing features of a room are:

1) A room is one continuous space that can be fully explored without the player "going anywhere else," i.e. into a separate room with its own description and contents. (Example: if the player is in the bathroom and becomes very tiny and enters the medicine cabinet, and there's a bunch of stuff in there that wasn't accessible from the bathroom, what you've got is two rooms, Brenda.)

2) When the player is inside of a room, they can interact with everything else inside that room, and they cannot interact with anything outside that room. So, when you're in the parlor: you can look out of the window, you can examine the fireplace in the far wall, maybe you can stand on a chair and lick the ceiling fan, but you can't read the diary that's upstairs in the bedroom. (Which makes a lot of sense.)

Other than that try not to worry about what a room is! I'm sorry for even bringing it up.

YOUR ROOM SPECIFICALLY

By now, you should have received your prompt and know what sort of room you are writing (reminder to let us know if you hate your room prompt and we'll come up with a plan B). In order to avoid potential conflicts, we'll give your room a special code name, such as Y83.

Its actual name will be up to you! Feel free to be as prosaic as "Train Station Bathroom," as baroque as "Alastair J. Cragne's Secret Museum of Skin-Bound Tomes and Small Bones," or anywhere in between! (Please do stick within your prompt though.)

Once you've decided on a name for your room, it will help us immensely if the first lines of your design document look a lot like:

Y83 is a room. The printed name of Y83 is "The Horrible Bedroom".

The "printed name of" line ensures that the player will only ever see "The Horrible Bedroom," never "Y83." Whenever you refer to your room -- such as, when you fill it with objects, how exciting! -- please use its code name to ensure there are no conflicts.

THE ROOM DESCRIPTION

Whenever the player enters your room for the first time, or types LOOK inside of it, they will be able to read your lovingly handcrafted description of the room and its contents, e.g.:

What wallpaper remains on the wall is a bright turmeric yellow that threatens to rip your eyeballs out of your face in much the same way that it itself has been torn, vis-a-vis violently, with bloodied fingernails. An empty brass bed frame dominates the space; underneath it, you spy a water-stained journal.

The door back out to the hallway is east.

To include your description, simply add this line to your document:

The description of Y83 is "What wallpaper remains on the [... blah blah blah prose prose prose ...] hallway is east."

(Except with your own room code and description of course!)

*** IMPORTANT NOTE ABOUT PARAGRAPH BREAKS ***

If you want to make Ryan's and my life easier while including paragraph breaks, please type [paragraph break] instead of manually typing the paragraph break. Thank you!

*** END IMPORTANT NOTE ABOUT PARAGRAPH BREAKS ***

*** EVEN MORE IMPORTANT NOTE ABOUT QUOTATION MARKS ***

Inform uses quotation marks to differentiate text you've written to appear in the game from text that is code. If you want quotation marks inside of these quotation marks, please use single quotes (apostrophes) and Inform will automatically print them as quotation marks. Example:

The description of Vanessa is "She looks you right in the eyes. 'What? Whatcha looking at?' she asks you."

Using more double quotes inside of the first double quotes will *break everything* in a way that can be *hard to track down.*

*** END EVEN MORE IMPORTANT NOTE ABOUT QUOTATION MARKS ***

ON ENTERING YOUR ROOM

It's entirely possible that you will want some text to appear when the player enters your room for the first time. Maybe you want to draw their attention to one particular object, or freak them out by having something creepy happen, or, I don't know, y'all are the ones with the great ideas here. I'm just the best practices document.

Here is how you make this thing happen:

Before looking in Y83 for the first time, say "For a moment or two, you hear a frantic scratching from inside the wall, then it fades away as quickly as it began."

DOING FANCY STUFF WITH YOUR ROOM DESCRIPTION

Depending on what's happening inside of your room, you might want the room description to evolve -- for example, if your room is slowly filling up with water, or if the player has noticed that the dolls on the shelves are moving by themselves, or what have you.

If you'd like your room description to change based on some criteria, please explain using your clearest explanation skills inside of two square brackets [] and including *the trigger for the change* and *the text you want altered*. Some examples:

[After the player has been in X92 for twenty turns, change "The door to the hallway is quite unusual and novel." in the room description to "You can't believe how much you hate this door now."]

[After the player has examined the giveaway doll, change "The dolls sit silently on their shelves as though they are watching you" to "The dolls are definitely watching you and have been this whole time, Chuckles" in the room description.]

If there's any other fancy thing you can think of for your room name or description that I haven't mentioned here, drop a (clear and detailed) explanation inside of two square brackets and we'll try to make it work!

BONUS: DESCRIBING THE PLAYER

To maintain tone/style/continuity within a room, *and* because we think it's fun, we are allowing/forcing all of you to write your own description of Naomi Cragne that the human in your room will get to read if they examine her.

This can be a physical description, a state-of-mind description, a complete non-sequitur -- the more variety, the better! Here's the syntax:

The description of the player is "[when the player is in R72]You are as good-looking as ever, not counting the day you married Peter, because on *that* day five of your friends and your sister the professional beautician spent the entire morning Doing Things To You until you resembled yourself far less than you resembled the anthropomorphized concept of Female Beauty Herself.[paragraph break]On most days, you look just fine, though. You look fine now."

PART TWO: THE LIFE-CHANGING MAGIC OF ADDING OBJECTS

Objects! This is where your room really starts to come together, when it's so full of stuff that you can't see the floor anymore and you're like "Wow, why do I own all this stuff, what terrible choices have I made with my life?" (Okay, maybe some personal feelings about having to move soon are bleeding into this otherwise impersonal document. Soldiering on!)

Adding a generic object is very, very easy. Simply type:

A brass bed is in Y83.

Note that this is a *brass* bed and not simply a bed; it's best if your objects have very specific names that you think will differentiate them from other people's objects, so Inform doesn't get confused. (If you and someone else both manage to include, say, a "tea-stained weatherbeaten Hepplewhite chair" in your room, don't worry, we are very impressed and we'll sort it out on the backend.)

To describe an object, type:

The description of the brass bed is "It looks uncomfortable, now that the mattress is gone."

Now there is an object in your room called a brass bed, and when the player examines it, they will see the text "It looks uncomfortable, now that the mattress is gone." Super simple, right?

Maybe too simple? Don't worry, objects can get approximately as fancy as you want them to. Let's start with some standard Inform object types.

*** IMPORTANT NOTE ABOUT OBJECT DESCRIPTIONS ***

If you include an object but fail to include its description, examining the object will yield the incredibly unexciting text "You see nothing special about the [whatever]." When this happens, it means you've missed out on an opportunity to provide Spooky Eldritch Atmosphere, and that Ryan & I are Sad. For these reasons, we would *really* prefer that you describe every object you include, if possible.

*** END IMPORTANT NOTE ABOUT OBJECT DESCRIPTIONS ***

CONTAINERS

Containers are everybody's favorite because they contain stuff! A treasure chest, a cereal box, a human heart, if you can put things inside of it and/or take them out, it's a container, Potsie!

To add a container and its contents to your room, type:

A gilded puzzlebox is a container in P52. The gilded puzzlebox contains a scratched silver orb.

Inform 7 is set up to understand a few commonly utilized properties of containers. A container can be:

-- Closed or open
-- Openable
-- Locked or unlocked
-- Lockable
-- Enterable
-- Opaque or transparent (an object can be seen through a transparent closed container but not an opaque closed container).

To define your object's properties, include them in your object declaration, e.g.:

A gilded puzzlebox is a closed locked opaque portable enterable container in P52.

Also, containers can contain other containers:

The scratched silver orb is a container. The scratched silver orb contains a rune-engraved key.

I don't know how far down this rabbit hole goes, so please exercise caution with your containment powers.

SUPPORTERS

A supporter is any object on top of which you might place another object. Very often in parser IF this means a desk, shelf, or table, but *of course* you are not limited to mundane furniture items. (For example, technically a supporter could be a naked person off of whom other people eat sushi, but I can't recommend that because it's not very Anchorhead and sounds highly unsanitary.)

A supporter and its supportents (a perfectly cromulent word) are declared in much the same way as a container:

A blood-smeared altar is a supporter in N31. On the blood-smeared altar is a supporter called a naked person. On the naked person is a piece of salmon nigiri.

If you want to indicate that a supporter can be sat upon, describe it as enterable:

The Wailing Chair of Sorrows is an enterable supporter in C05.

CLOTHING, ACCESSORIES, & OTHER WEARABLE ITEMS

Inform has a lot of default behaviors to handle the concept of wearability in a way that pretty much makes sense. The short version is, you can define an object as wearable, and the player will be able to put it on and take it off. (You can also describe non-player characters as wearing clothes, and if they're coded to do so, they can take clothes off and put clothes on by themselves.)

Wearable objects are defined thusly:

A grey hooded robe is wearable in N31. The naked person wears a subtle anklet.

FOOD

If you define an object as food, the player (and NPCs) will be able to eat it (or, at least, *try* to eat it).

The salmon nigiri is food. A beating human heart is food in H73.

PEOPLE AND ANIMALS

Players of parser IF games tend to expect more complicated interactions to be possible with a living creature, especially if it's a human. For this reason, it takes more effort to get a person or animal to feel like a satisfying and rewarding object than it does for, say, a bucket of Chapstick.

It's up to you whether you want to deal with the hassle of creating life! If you do, though, here's the first step:

Tavish MacHagard is a man in P77. Annette Cragne-MacHagard is a woman in P77. An innkeeper is a person in P77. Maru is an animal in P77.

Please note: declaring that an object is a person or animal will *not* in itself give it human or animal behaviors! What it *will* do is keep Inform from objecting that your NPC is not a person when you try to put clothes on it or have the player talk to it. (More on that later!)

FIXED IN PLACE VS. PORTABLE

One of the most important qualities an object can have is whether it's portable, like a weathered talisman, or fixed in place, like an abandoned sarcophagus. In parser IF terms, this means the player can TAKE portable things and carry them from room to room, but things that are fixed in place stay right where they are.

If you don't define portability yourself for a given object, Inform will attempt to guess, using a fairly sane metric:

-- Most objects are portable by default.
-- Supporters are fixed in place by default.
-- Scenery (see below) is *never* portable, even if you say it is.
-- People and animals are *never* portable and have their own default failure message for TAKE. (Which you can change, of course!)

The safe way to guarantee an object will be fixed in place or portable, of course, is to go ahead and define it:

The weathered talisman is portable. The abandoned sarcophagus is fixed in place.

SCENERY AND THE AUTOMATIC ITEM LIST

So, let's say we've been adding objects to the Horrible Bedroom like object-adding champions, and we've implemented everything mentioned in the room description. (Which is a really good thing to do!) Here's the code we added:

Some violent yellow wallpaper is in Y83. A garishly papered wall is in Y83. An enterable supporter called a brass bed frame is in Y83. A water-damaged journal is in Y83.

Here's what the player will see when they enter the room (or type LOOK):

The Horrible Bedroom

What wallpaper remains on the wall is a bright turmeric yellow that threatens to rip your eyeballs out of your face in much the same way that it itself has been torn, vis-a-vis violently, with bloodied fingernails. An empty brass bed frame dominates the space; underneath it, you spy a water-damaged journal.

The door back out to the hallway is east.

You can see some violent yellow wallpaper, a wall, a brass bed frame and a water-stained journal here.

Check out that last paragraph: Inform has automatically added a list of every object in the room to the room's description. This feature is great for ensuring the player can see everything there is to look at, but maybe you the *author* would prefer to have more control over how the text in your room displays.

One step you can take is to mark any objects *that the player is never expected to pick up* as scenery. In this example, the wall and the wallpaper both make very good candidates. (So does the bed frame, but for the sake of the next example, let's leave that right how it is.)

Adding the line:

The wall is scenery. The wallpaper is scenery.

removes these objects from our automatic list of items, leaving:

You can see a brass bed frame and a water-stained journal here.

If you're not happy with the entire list of non-scenery items appearing in the last paragraph of your room's description, please leave us a note in square brackets [] near your room description.

Some examples:

[The only object mentioned in the item list should be the brass bed frame.]

[The wallpaper should not appear in the item list until the player has examined it once.]

[Please remove the automatically generated item list from the room description entirely.]

Note: it's a very good idea at this point to check your room description and remove any references to portable items that made sense to include when you were first brainstorming the room, but might or might not exist in the room at any given time, like the water-stained journal.

INITIAL APPEARANCE OF AN OBJECT

(Jenni's Note: When Ryan & I were talking about what constituted "most basic" in terms of teaching people very simple Inform code, he mentioned initial appearance, from which I can only assume that he uses this feature just all the dang time. I have never used initial appearance and had to look it up to see how it worked. It is up to you to decide whether you are a Jenni or a Ryan; both lifestyles are equally valid in my opinion. Ganbarimashou!)

Inform's initial appearance feature lets you define how you want an object to appear in the automatic item list when the player first sees it. For example, if we add the following line to our previous Horrible Bedroom code:

The initial appearance of the water-stained journal is "You think you spot a stained, rippled journal through the slats of the bed."

then we'll get:

The Horrible Bedroom

What wallpaper remains on the wall is a bright turmeric yellow that threatens to rip your eyeballs out of your face in much the same way that it itself has been torn, vis-a-vis violently, with bloodied fingernails. An empty brass bed frame dominates the space; underneath it, you spy a water-damaged journal.

The door back out to the hallway is east.

You think you spot a stained, rippled journal through the slats of the bed.

You can also see a brass bed frame here.

After the player has picked up the journal and put it down again, its initial appearance no longer applies and the item list reverts to:

You can see a water-damaged journal and a brass bed frame here.

PART THREE: ACTIONS AND INTERACTIONS

Don't get me wrong, objects are great and fun and we love them, but the *real* meat of your room lies in how it responds to the actions the player types in.

In order to respond most effectively, you'll need to have a good idea of what actions the player might reasonably attempt -- they *can* type in literally anything, but most of the time their action attempts will fall into one or more of the following categories:

Standard actions: Basic commands such as TAKE, EXAMINE, OPEN; these are built into Inform and *very* commonly used across all parser IF games. They're part of a standard vocabulary which experienced parser IF players have picked up from playing parser IF games and now expect to be able to use most of the time. Most "how to play parser IF" guides will include most, or all, of these.

Clued actions: Commands that you the author -- in an object description or other player-readable text -- have hinted (or flat-out explained) will be possible. For example, you might write:

The description of the red-robed cultist is "He glares at you angrily. 'Don't try to CONFUSE me, outsider.'"

This very strongly suggests that the command CONFUSE CULTIST will do something, or at *least* give a compelling excuse for not doing anything.

Logical actions: Courses of action it would be reasonable to take if the scenario and situation existed in the real world. For example, if the player is told they cannot reach an orb that is on a high shelf, they might try STAND ON CHAIR or HIT ORB WITH BROOM (assuming they have a chair and/or broom).

Generally it's less than possible to anticipate every real-world-sensible action the player might try, but they'll expect at least a graceful "here's why you can't do that" response for the most obvious choices. (Writing a response for some *less* obvious courses of action will make your room seem more polished and impressive!)

STANDARD ACTIONS

This is the list of default in-game actions included by Inform 7 (minus the metagame stuff we *really* don't need to care about, like starting a transcript or requesting the pronoun explanation):

INVENTORY
TAKE [X]
REMOVE [X] FROM [Y]
DROP [X]
PUT [X] ON [Y]
INSERT [X] INTO [Y]
WAIT
GO [DIRECTION]
ENTER [X]
EXIT
GET OFF [X]
LOOK
EXAMINE [X]
READ [X]
LOOK UNDER [X]
SEARCH [X]
CONSULT [X] ABOUT [T]
LOCK [X] WITH [Y]
UNLOCK [X] WITH [Y]
SWITCH ON [X]
SWITCH OFF [Y]
OPEN [X]
CLOSE [X]
WEAR [X]
TAKE OFF [X]
EAT [X]
GIVE [X] TO [NPC]
SHOW [X] TO [NPC]
WAKE [X]
THROW [X] AT [Y]
ATTACK [X]
BREAK [X]
KISS [NPC]
ANSWER [NPC] THAT [T]
TELL [NPC] ABOUT [Y]
ASK [NPC] ABOUT [Y]
ASK [NPC] FOR [Y]
TOUCH [X]
WAVE
PULL [X]
PUSH [X]
TURN [X]
PUSH [X] TO [DIRECT'N]
SQUEEZE [X]
YES
NO
BURN [X]
WAKE UP
THINK
SMELL [X]
LISTEN TO [X]
TASTE [X]
CUT [X]
JUMP
TIE [X] TO [Y]
DRINK [X]
SWING [X]
RUB [X]
SET [X] TO [P]
BUY [X]
CLIMB [X]
SLEEP

It's important to understand that even though Inform *includes* these actions, it doesn't automatically *handle* them; you have to explicitly define the responses you want.

For example, you might add a mirror to your room:

A cracked mirror is on the solid oak desk.

The player can type BREAK MIRROR, and because BREAK is a default command, Inform will not respond "What is this BREAK? What are you talking about?"

It will *also* not respond "Okay, the mirror is now broken, I have changed the name to 'broken mirror shards' and now you can use it to cut stuff," because you haven't specifically told Inform that you want to do that yet.

Instead, it will respond with its default failure message for attacking or breaking:

Violence isn't the answer to this one.

This is clearly suboptimal, especially if your puzzle solution *requires* the player to break the mirror and use the shards to cut stuff. You'll need to explicitly define what you want to happen when the player types BREAK MIRROR.

When I say "explicitly define," I mean "to me and Ryan using square brackets and standard English," because this stuff can get complicated! Example:

[Instead of breaking the mirror, change the name of the mirror to "broken mirror shards" and the description of the mirror to "Oops."

Before the mirror is broken, CUT [X] WITH MIRROR gives the response "The edge is too smooth." and fails to cut the thing. After the mirror is broken, CUT [X] WITH SHARDS gives the response "The sharp edge of the mirror shards slices neatly through the [X]." and successfully cuts the thing if it's cuttable.

After the mirror is broken, picking up the shards for the first time causes the player to drop a random object they're carrying, saying "Ouch! You cut yourself on a shard of glass and drop the [random object]." Picking up the shards after the first time gives the response "You pick up the mirror shards more carefully this time."]

We *highly* advise going through the list of default actions, thinking about which ones players are likely to attempt with various objects in your room, and writing responses for what happens when they try. A well-written failure message makes a great puzzle hint!

NEW ACTIONS

Since this project is going to be *bonkers huge* and a tiny bit nightmarish on the code level, we're encouraging all of our authors to accomplish everything they can using the default actions listed above.

For example, instead of defining SACRIFICE as a brand new action, you can include a sacrificial knife which the player can use to CUT things after they PUT them on a sacrificial altar. Or, instead of TITRATE [X], you can create a titration indicator with a button, and the player can PUT [X] IN INDICATOR, then PUSH TITRATION BUTTON.

Sometimes, however, you absolutely *have* to have a new action and that's just the way it is. Like, maybe you've been assigned to write a room that is Naomi's nightmare, and you've decided she's flashed back to her substitute teaching days and needs to REPRIMAND students. (I mean, she could always THROW things at them instead, but that's a *significantly* different action.)

Here's how you would explain your new action, step-by-step: (Note the square brackets; you just explain what you want to do and we'll handle making it work as code.)

[Polodna-reprimanding is a new action applying to one person.

Understand "REPRIMAND [person]," "SCOLD [person]," and "YELL AT [person]" as polodna-reprimanding while player is in N25.

Carry out polodna-reprimanding:
if the person being reprimanded is a student: say "[name of student]! Knock it off!" or "[name of student]! Quit it!" or "Consarn it, [name of student], not again!"
If the person being reprimanded is not a student, say "'I don't have to take this from you,' [name of person] responds."]

That's kind of a lot, so let's break it down step-by-step:

[Polodna-reprimanding is a new action applying to one person.]

It's important to let us know what the intended object of your new action is. You might have an action like CONSULT [thing] ABOUT [topic], which would apply to one thing and one topic, or an action like JUMP, which has no object and therefore applies to nothing.

Also note that I've called this polodna-reprimanding to avoid potential collisions in case someone else has filled their room with naughty dogs and also wants to implement a REPRIMAND command.

[Understand "REPRIMAND [person]," "SCOLD [person]," and "YELL AT [person]" as polodna-reprimanding while player is in N25.]

This line tells us that if the player types either REPRIMAND [NPC], SCOLD [NPC], or YELL AT [NPC] while in your room, it should perform the action you're about to define as polodna-reprimanding.

[Carry out polodna-reprimanding:
if the person being reprimanded is a student: say "[name of student]! Knock it off!" or "[name of student]! Quit it!" or "Consarn it, [name of student], not again!"
If the person being reprimanded is not a student, say "'I don't have to take this from you,' [name of person] responds."]

These lines explain what you want to have happen when the player tries polodna-reprimanding, given some set of conditions. I've kept them *very* simple for example purposes but you can make them *much* more complicated and interdependent, if you want to.

For example, let's say that earlier in your design document you've (very formally) defined a set of emotions your nightmare students can have:

[A student has a polodna-emotion. The polodna-emotions are polodna-rowdy, polodna-angry, and polodna-riotous. A student is usually polodna-rowdy.]

You can include these emotions when defining your new action:

```
[Carry out polodna-reprimanding:
if the person being reprimanded is a student:
        -- say "[name of student]! Knock it off!" or "[name of student]! Quit it!" or
                "Consarn it, [name of student], not again!";
        -- if the student was polodna-rowdy, they become polodna-angry;
        -- or if the student was polodna-angry, they become polodna-riotous;
        -- or if the student was polodna-riotous:
                -- say "[here is where you write a gory description of Naomi getting
                        murdered by a riotous student]";
                -- wake Naomi up;
                -- equip Naomi with the dream scar;
                -- move Naomi to the next room chronologically.
If the person being reprimanded is not a student, say "'I don't have to take this from you,'
                                                        [name of person] responds."]
```

If you're ever unsure about the correct syntax for explaining what you want to do with your new action, or anything else, *don't worry about it.* Just explain it as well as you can & we will figure it out!

(Also note that you definitely don't have to use a particular tab format or anything, I'm just doing it here for readability. Use whatever format makes you feel able to best convey your ideas!)

*** VERY IMPORTANT NOTE ABOUT UNWINNABLE STATES ***

If you've been reading Ryan's Cragne Manor communiques, you might have noticed that one of the *few* stylistic choices we are dictating for the entire project is that the game not be *cruel,* in gameplay terms.

What "cruel" means here is "allowing the player to get into a situation where they can no longer win the game, and not telling them this has happened until it's way too late to fix it, so they have to reload a save or restart from scratch."

In old-school IF, including Anchorhead, this form of ludic cruelty was fairly standard, and players *expected* the game to be mean and put them in unwinnable states. Modern players *don't like this stuff at all* and *personally I agree with them.*

We *don't* mean that everything in the game should be "safe," because this is cosmic horror we're dealing with. If you're so inclined, you can *definitely* one-move-kill Naomi, end the world, chop limbs off of NPCs -- with one caveat:

Any situation the player gets into where they cannot win the game anymore has to end the game immediately AND be reversible within the UNDO window (eight turns).

This means no bottomless holes the player can throw random items into. No crows stealing things from their pockets and flying away forever. No poisoned hairpins that kill Naomi thirty turns after she picks them up. No flashlights that have one non-rechargeable battery and if it runs out of juice, you're out of luck.

You *can* have a poisoned hairpin that kills Naomi within *five* turns, a crow that can be resummoned with an infinite supply of birdseed, a monster that gives you a three-turn window to find a sword, sharpen that sword, and stab it to death... there's a lot you can do to make the game *dangerous* without actually making it cruel.

*** END VERY IMPORTANT NOTE ABOUT UNWINNABLE STATES ***

PART FOUR: CONVERSATIONS WITH NPCs

Before we get into the meat of this section, I'd like to remind everybody again that having an NPC in your room is *substantially* more work than not having an NPC in your room, so I'd only recommend including one if you really want one and are excited about it.

If you *do* include an NPC in your room, bear in mind that players will expect to be able to talk with it, and the more interactive it is, the more fun they'll have! Conversations can deliver exposition, create atmosphere, provide entertainment, and hint at solutions to puzzles. Not only that, but an NPC might even give the player an item or change the state of the world as a response to a given line of dialogue.

There are several different ways you can model a conversation in parser IF, but two have emerged as the primary contenders:

ASK-TELL

If you look back at the list of default commands in part three, you might notice a few that seem designed for talking to NPCs, especially ASK and TELL. These two verbs (along with ANSWER [NPC] THAT [TOPIC], which I think is rarely used?) are the backbone of the ask-tell conversational system.

Here's how it works from the player's perspective: they type ASK (or TELL) [NPC] ABOUT [AN OBJECT OR TOPIC], and the NPC, hopefully, responds with something that makes sense.

For example, the player might type "ASK VANESSA ABOUT AMULET," and Vanessa might respond "Amulet? What lost amulet? Definitely don't look under the couch for it." Later, after the player has found the amulet, they might TELL VANESSA ABOUT AMULET, and maybe she'll say "...it was under the couch? That's weird," and suddenly have to leave the room.

For purposes of this project, please include your ASK/TELL dialogue as an informal table in square brackets, like so:

[Table of Vanessa's dialogue (Ask/Tell)

TOPIC	**CONDITION**	**LINE**
Amulet	**Amulet not yet found**	**"'Amulet? What lost amulet? Definitely don't look under the couch for it,' Vanessa says."**
Amulet	**Amulet found**	**"Vanessa won't look at you. '...it was under the couch? That's weird,' she says. 'I have to go.'" [VANESSA EXITS THE ROOM]**
The player	**Player has no scars**	**"'You're cute enough,' she says, 'but I like chicks with scars.'"**
The player	**Player has scars>0**	**"She looks you up and down, eyes lingering on the scarred portion of your flesh. 'Yum,' she says. 'Did that hurt?'"]**

If the distinction between ASK and TELL is important for any of your topics, feel free to include an extra column!

MULTIPLE CHOICE

Multiple choice dialogue-tree conversation is popular in many types of video game, not just parser IF, so it's very likely you're already familiar with it. It looks like this:

Suddenly, you find yourself facing down what looks like an ancient spear! "What are you doing here?" a voice demands.

1) I'm looking for the man who killed my father! Have you seen him?
2) Leaving. The thing I'm doing here is definitely leaving. Immediately.
3) Is this Luby's cafeteria? I heard y'all had chocolate icebox pie to die for.
4) Enough talk! Have at you!

Here is the executive decision that Ryan & I have made regarding multiple choice *or* other non-ask-tell conversation systems: if you really want a specific type of conversational system that is not ASK/TELL, let us know, and we'll sort out the particulars!

SHOWING AND GIVING

No matter what conversation system you wind up using, you'll probably want to write responses for the default commands SHOW [X] TO [NPC] and GIVE [X] TO NPC, as they make sense.

For example, it would make sense for an NPC who believes his wife died twenty years ago to react to a current photograph of her in yesterday's newspaper. At the *very* least, an NPC who sent you on a quest to find their glasses should react when shown their glasses. (I feel very strongly about this.)

[Instead of showing Ariadne her glasses: say "Ariadne swipes the glasses from your hand without even thanking you."; take the glasses away from the player and give them to Ariadne]

VERY IMPORTANT: The default Inform 7 behavior for GIVE is that the player attempts to give the NPC something and the NPC does not accept it or acknowledge it in any way.

If you override this for any reason (for example, to give Ariadne her glasses), make sure your NPC *only* accepts objects that have no further use, to avoid the unwinnable states thing I talked about at the end of Part Three.

PART FIVE: IMPLEMENTATION CHECKLIST

ROOM & OBJECTS

-- Have you named your room?

-- Have you described your room?

-- Have you implemented all the objects listed in the room description?

-- Do all of these objects have descriptions themselves?

-- Do all your objects have the properties you want them to have? (portability, etc.)

-- Are the names of your objects specific enough to not conflict with similar objects?

-- Does the room description mention any objects that might not always be in the room?

-- Have you avoided including any bottomless holes or BlendTec blenders the player can lose items in and make the game unwinnable? (Also a BlendTec blender would be a *90s ANACHRONISM*)

-- Have you written a room-specific description of the player?

PUZZLES

-- Does your room contain all the objects you need for your puzzle that are not puzzle rewards from another room?

-- Does your room *not* contain any objects that are *supposed* to be puzzle rewards from another room?

-- Is the player physically able to complete the puzzle as you have it designed? (e.g. the key to unlock the chest is not locked inside the chest, etc.)

-- Have you thought about alternate ways the player might reasonably complete your puzzle, and either implemented them or written sensible explanations of why they don't work?

-- Is the player nudged towards your intended puzzle solution by object descriptions, failure messages, or other text?

NEW ACTIONS

-- Does your new action really *need* to be a new action or would a standard action work fine with a little tweaking?

-- Does your new action have a unique name that is unlikely to conflict with anyone else's new action?

-- Have you defined all the commands for your new action (e.g. STEAL, SWIPE, SNAG)?

-- Is it clear how many & what kind of object your action refers to (e.g. one topic, one person and one visible object, nothing)?

-- Have you explained in as detailed a way as possible what you want your new action to do in various circumstances?

NPCs

-- Do any NPCs in your room have responses for every conversational topic their character would be expected to know and/or care about?

-- Have you written responses for "SHOW [OBJECT] TO [NPC]" using the objects in your room?

-- Have you written a default catchall response for your character having been shown a random thing?

-- Have you avoided situations where the player can make the game unwinnable if they GIVE your NPC an item that the NPC accepts and never gives back?

PART SIX: A SAMPLE DESIGN DOCUMENT

Y83 is a room. The printed name of Y83 is "The Horrible Bedroom". The description of Y83 is "What wallpaper remains on the wall is a bright turmeric yellow that threatens to rip your eyeballs out of your face in much the same way that it itself has been torn, vis-a-vis violently, with bloodied fingernails. An empty brass bed frame dominates the space.[paragraph break]The door back out to the hallway is east."

Before looking in Y83 for the first time, say "For a moment or two, you hear a frantic scratching from inside the wall, then it fades away as quickly as it began."

The description of the player is "[if the player is in Y83]You look down at your arms to recenter yourself, but all you see is reflected yellow light."

Some violent yellow wallpaper is in Y83. The violent yellow wallpaper is scenery. The description of the violent yellow wallpaper is "The wall is covered with a sickly-looking matte paper in a garish yellow hue, torn away in places. The pattern seems to swirl before your gaze, shifting from innocuous clusters of flora into a swelling of many-eyed faces, and the bile rising in your throat signals you to look away."

A garishly papered wall or walls is in Y83. The garishly papered wall or walls is scenery. Instead of examining the garishly papered wall or walls, try examining the violent yellow wallpaper.

An enterable supporter called a brass bed frame is in Y83. The brass bed frame is scenery. The description of the brass bed frame is "Dented and scuffed, but solid and serviceable."

A water-stained journal is in Y83. The description of the water-stained journal is "The cover is faded black cloth, and the pages are yellowing from age and rippled from water damage. A tattered ribbon emerges from its folds like the tongue of a particularly unlucky serpent." The initial appearance of the water-stained journal is "You think you spot a stained, rippled journal through the slats of the bed."

Instead of reading the water-stained journal, say "Several of the pages have been ripped out, most of the surviving pages are blank, and almost all of the non-blank pages are written in an indecipherable scrawl of symbols. On the penultimate page of the journal, however, you spot a line of English: 'DON'T TRY TO SAVE YOURSELF, THE CIRCLE IS COMPLETE.'"

[-- Jenni's Note: In default Inform 7, READ is a synonym for EXAMINE, but for Cragne Manor we are implementing reading as its own action throughout the entire game.]

[The Wallpaper Puzzle is the only puzzle in Y83. Here is how it works:

The wallpaper has three states: mostly polodna-papered, halfway polodna-papered, and barely polodna-papered, in decreasing order of there being paper on the walls. The wallpaper is usually mostly polodna-papered.

The player has two states: polodna-eyes-open and polodna-eyes-closed. The player is usually polodna-eyes-open.

The player also has some secret hidden eyes that they can open and close without them being, like, an actual object (this seems complicated so I'll let you implement it for me, thanks!) When the player opens and closes their eyes, their polodna-eyes-open / polodna-eyes-closed status changes accordingly. The message you get is normally just "You open your eyes." or "You close your eyes."

Polodna-peeling is an action applying to one thing. Understand "PEEL [thing]", "PEEL OFF [thing]", "TEAR [thing]", and "RIP [thing]" as polodna-peeling when in Y83.

```
Carry out polodna-peeling:
    if the thing being peeled is not the wallpaper: say "That's not exactly peelable."
    if it is the wallpaper and the player is polodna-eyes-open: say "You only manage one step towards the hideous wallpaper before the sight of it overwhelms you and you reel back, retching."
    if it is the wallpaper and the player is polodna-eyes-closed and the wallpaper is barely polodna-papered: say "You've torn enough wallpaper for today."
    if it is the wallpaper and the player is polodna-eyes-closed and the wallpaper is more than barely polodna-papered:
        -- say "You grope around in the dimness until you find a peelable edge of wallpaper, then yank it as hard as you can. For just a moment, you imagine you hear a pained gurgling screech from somewhere distant."
        -- the wallpaper becomes one level less polodna-papered
    if the wallpaper just now became barely polodna-papered, the secret portal opens up in the north wall (as described in my prompt!!)
```

```
Descriptions of the wallpaper at various polodna-papered levels:
    mostly: the initial description
    halfway: the initial description but replace "in places" with "in great chunks"
    barely: "Scraps of defeated-looking yellow wallpaper cling to the walls in scattered clusters, retreating into the corners. The north wall now sports a shimmering portal."
```

Here's how I want the puzzle to work: you have to close your eyes to tear the paper off of the wall the first time. Once you've done that a shoggoth appears in the room from offstage. The description of the shoggoth is "It's coming straight for you!" If you don't use the next turn after the shoggoth appears to open your eyes, it eats you and you die (gruesome description below). If you do open your eyes, then no matter what command you enter next turn, your body is taken over momentarily by another consciousness which utters a banishing word, and the shoggoth vanishes. You are then free to tear the rest of the paper off of the wall.

Text you get when the shoggoth eats you: "[this is where you would write a lovingly gruesome description of Naomi getting eaten by a shoggoth, prose prose prose etc.]".

Text you get when you open your eyes and see the shoggoth: "You open your eyes and see a shoggoth."

Text you get when the unfamiliar consciousness takes over your body and utters the banishing word: "[this is where you would write a nice long description of that happening]".

Thanks!
Jenni]