Interactive Fiction and Adventure Creator

A multi-platform Adventure and Fiction creation system to enable anyone to create rich, interactive stories that capture the imagination and show your creativity

"It's your world. Make it do something."

Runs on 64 bit versions of :-

Windows
Mac
Linux

Localized in:-

English
French
Spanish

PART 1 - A General Introduction

What is IFAC?

IFAC is an adventure and interactive fiction creation system that applies advenced technologies to the creation of old school text adventure games and digital interactive fiction, "choose your own adventure" stories to enable a rapid and easily debuggable process.
If you have a story within you that you have been dying to tell then IFAC will allow you to easily write it, edit it and share it with others.
If you have some puzzles that need an original game world setting, you can realize that game world and place the puzzles in it with IFAC.
Best of all, IFAC is a club, a community where you can exchange ideas and finished games or get help and discuss anything related to creating stories with IFAC. Our website is https://ifac.club/community.

What is Adventure?

Adventure (or text adventures) is a computer moderated world with locations, characters and objects that can be interacted with (in old school games) by two word verb noun commands. It started with a game named "Adventure", also known as Colossal Cave by Crowther and Woods and is often described as a type of role playing game like D&D with the program as the Dungeon Master.
IFAC and its own internal language allow the creation of all the location and objects with in the game as well as how it responds to certain instructions given by the player. Puzzles based on game state, timing and other conditions are easy to implement to provide a challenge to the player that they have to solve.

What is Interactive Fiction?

Interactive Fiction is essentially a book where you get to make choices about how the story will progress. IFAC allows you to enter all the different scenarios as locations and specify how the choices connect to each other. Interactive fiction has a more narrative style as you are reading a story and at the end of each segment the reader is presented with a list of choices of what to do next which result in the story diverging. A good interactive fiction story can be read multiple times to discover multiple endings.
You can instruct the reader to implement a combat system if required but this is outside of the program and is not covered by IFAC as the variety of systems that could be needed cannot be known in advence.

How does IFAC work?

IFAC creates and edits a database of the game you are creating which stores the locations, objects and game instructions which define the game. Locations and objects are simple storage but the instructions and conditions under which they are activated are defined in a small but powerful domain specific language that IFAC uses internally. When conditions and instructions are combined correctly this releases a powerful way of defining how the game reacts to the player.
An explanation of the different tabs in IFAC and the input fields on them can be found in PART 3, but you will go to many of them in creating the sample adventure in PART 2.
All of the IFAC keywords are defined and explained in PART 4 of this manual but the best way to see the magic they can make is to start using them.

With that in mind, the preliminaries ore OVER!

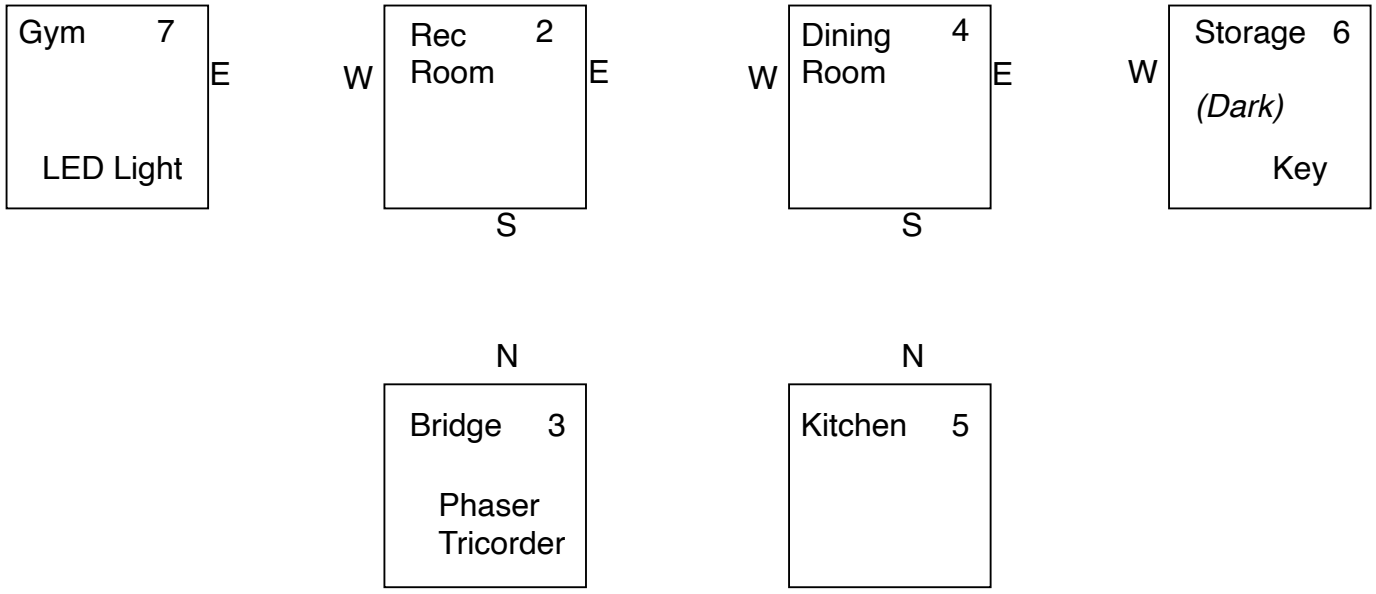
Let's make our very first adventure. Although it is only a small world of a few locations ond objects it will demonstrate much of the power of IFAC and how simply that power is achieved. Bring on PART 2 ...

PART 2 - Setting Up Your First Adventure

All of the sections from here need to be completed in order, as the later sections rely upon the commands and data entered in the earlier sections. However, as the program saves your data after each entry you can close the program and the database will be in the same state when you return to the program to carry on from where you left off.

When writing an adventure or piece of interactive fiction it helps enormously to plan. For writing fiction you may have a plot outline and character profiles. For an adventure you will usually begin with a map. Graph paper can be helpful for this but is not essential. Quite often I will just do a rough map on a page with boxes for locations, lines between locations to show connections and directions between locations and , finally, names of objects that can be found at each location, if any.

It's not as complicated as it sounds. Here's the first few locations of our first adventure drawn out:-



For each box the name of the location is at top left, the location number (or ID) at top right and any notes in the middle of the box, such as that the Storage is a dark location.
The lines show the connections and direction of travel between each location. Traditionally, as in all mapmaking, north is towards the top of the page and the sides of the squares follow the compass points. This is instinctive for most people so I would encourage you to stick to it.
For Up and Down directions you can use a line from somewhere unusued on the box and clearly mark it as U or D. Many people like to use dotted lines for these directions to show they are not in the normal compass but you are free to do what is clearest for you. If you choose to make your map publicly available later just make sure you clearly state your intentions with the notations.

Our adventure will take place on an abandoned spaceship. So, with that, let's enter our locations.

The Location Texts

The location IDs are VERY important so make sure you plan them out carefully and enter the location data in the order of the location IDs. You will see later how important this can be, but for now just ensure you get the right text with the right ID. The editor assigns ID automatically in ascending sequence so, once again, forward planning is key to giving you the maximum flexibility later.

Location #1 has special significance to the game as it is used for the things you are carrying (the Inventory) and so Location #2 is the start location for your game. More on this when we add objects later.

Go to the Location tab which is the one normally open by default when IFAC starts up. The entry form has two fields - the Name, which also acts as the short description, and the Detail which is the long description, which is output the first time a player enters a Location and when the player requests it by entering the command LOOK.

Enter the following for each location and submit the form. You will see the data entered appear in the table on the right of the pane after each form is submitted.

Location 2
Name - Rec Room
Detail - This is the Recreation Room. The place where the crew relaxes and socializes. There are several tables and chairs here but the place is eerily quiet. The Dining Room is to the East. The Gym is off to the West and to the South is the Bridge.

Location 3
Name - Bridge
Detail - You are standing on the bridge of the ship. Normally the nerve center of activity on the ship it now looks strangely desolate, as no one is here and all the flight stations are dark and lifeless. It would appear the bridge has no power as only the emergency lighting casts a low glow over the scene. A small pedestal in the center has three prongs as if to hold something. To the North is the Rec Room.

Location 4
Name - Dining Room
Detail - The dining room appears to have been hastily abandoned. Dried and rotted food sits on several plates remaining on the tables. The Kitchen is to the South and a Storage Room is East of here.

Location 5
Name - Kitchen
Detail - A spotless shiny kitchen. The chef was clearly a neat freak. North of here is the Dining Room.

Location 6
Name - Storage
Detail - A room full of shelving that carries many different kinds of things. Food, blankets and many other things of varying usefulness. The only exit leads to the West.

Location 7
Name - Gym
Detail - The small gym has six different exercise machines. Despite the dust there is still a lingering smell of sweat here. A door to the East leads to the Rec Room.

Once all those have been entered you can, if you'll pardon the pun, move on to...

The Movement Entries

Click on the Movements tab which is just to the right of the Locations to see the Movements panel. The first thing you should notice is that there are already entries in the table on the right side of the panel. This is because when you created the locations in the previous step the program automatically creates a blank movement entry for each one at the same time.

These movement entries tell our program how the locations are interconnected. So let's enter those connections for Location 2. Find the movement entry with LocID 2 and double click on it. It should normally be the first record in the table. Note that there is the ID of the movement entry itself and then the LocID, short for Location ID, of the Location which the movements apply to. Most often starting from a clean database these will be the same number but if they do get out of sync it is the LocID you should pay more attention to.

After you have double clicked the entry you will see the values in the form on the left of the panel display new values - Location #2, Movement ID 2 and Loc Name Rec Room. Always check these to make sure you are editing the right movement entry for the right location.

Click on the arrow of the dropdown combo box for E (East) and select 4 which is the number of the location to which the player will travel if he enters E while in the Rec Room. Refer back to our map and see that Location 4 is the Dining Room. Notice that the numbers in the dropdown are limited to those of the locations we have already created. When we create more locations then more numbers (IDs) will appear here but for now it is 2 through 7.

To complete the other connections for the Rec Room change the West dropdown to 7 and the South dropdown to 3 and click the Submit button. You should see the entry in the table on the right update and the form will clear back to default values.

Double click the row for LocID 3 and edit the North value to 2 so that going North from the Bridge gets us back to the Rec Room.

By examining the map try and edit the remaining rooms to have the movement entries they need to connect them together. As a reference when you have finished the entries that should have been created for all Locations is listed below:-

Location 2 - E 4, S 3, W 7
Location 3 - N 2
Location 4 - E 6, S 5, W 2
Location 5 - N 4
Location 6 - W 4
Location 7 - E 2

When you are done the table in the Movement panel should look like the one below:-

ID ▲	LocID	N	NE	E	SE	S	SW	W	NW	Up	Down
2	2	–	–	4	–	3	–	7	–	–	–
3	3	2	–	–	–	–	–	–	–	–	–
4	4	–	–	6	–	5	–	2	–	–	–
5	5	4	–	–	–	–	–	–	–	–	–
6	6	–	–	–	–	–	–	4	–	–	–
7	7	–	–	2	–	–	–	–	–	–	–

Everything OK? Great job! Now it is time to test the work we have done so far both to see that everything is as expected and get the satisfaction that we already have a world that we can move around in which is a MAJOR step.

Testing the Adventure So Far

It's time to experience your game from the players perspective for the first time. Our objective here is to ensure we can move to all the locations and end up at the location we expect when we issue the movement commands (N, S, E, W in this case to keep it simple for us).

Testing your game after entering certain types of data is essential to keep things under control. If you don't test as you go and just wait until the end, you will find things that are wrong that have knock on effects and you increase your chances of just being buried in the sheer number of problems to solve. Testing as you go breaks this task down into bite size chunks that are more manageable and you also learn from mistakes early so that you can do things better later on in the game.

Click on the Test Adventure tab to see that panel. The majority of the panel is taken up by the output area with a small area at the bottom for us to issue our commands to the game. Click on the Start button and you will see the description of the Rec Room appear in the output area. The game always starts at Location 2 and that is the Rec Room in our adventure.

In the black command entry box near the bottom type in W (lower or uppercase does not matter but I will always use uppercase in these docs) and hit the Enter key. The output window echoes your command and then describes the Gym location, as it should. To get back to the Rec Room type in E and hit Enter again. If you are now back in the Rec Room you have tested that connection successfully. Notice that when you enter the Rec Room for the second time it only outputs the Name or short description of the location. If you want to see the full desription again simply enter the command LOOK or more quickly just L.

Referring to the map check all the movements between locations and then return yourself to the Bridge.

If everything does not work as planned then take a look at the graphic of the Movement Table and make sure that yours looks the same. Edit any entries you need to by double clicking on the row in the table and changing the values in the form on the left and submitting them. If you made any changes then test again until it works as expected. Learning to correct errors in entry is an essential skill for any programmer.

Now get yourself back on the Bridge. Let's try a few things to see some of the built in things the game interpreter does for you.

Type the following commands and check the result is as expected:-
W or WEST - "You cannot go that way!"
GET PHASER - "That is not here!"
L or LOOK - You will see the full description of the Bridge
I or INV or INVENTORY - "You are not carrying anything."

If everything is now working as we want, Congratulations! We have a working basic abandoned spaceship world just waiting for us to add more interesting things to it. Talking of "things" let's move on to...

Items

Items are anything in your adventure that can be manipulated and by manipulated we usually mean picked up (GET), dropped (DROP), worn (WEAR) and used in some way to be determined by you.

Hopefully it will come as no shock to you that Items are entered and edited in the Items tab, so click on that now to see the now familiar layout of an entry/edit form on the left and a table displaying the Items on the right. Our map earlier detailed the Items that we need for our initial Locations but there are more details to the entry form than what is contained in the annotations on the map, so here's the full details below.

Before you enter the Items take a quick look at the Vocabulary tab which has some basic words needed for every adventure and the Event tab where the table should just have six entries which manage the light. Notice there are two entries each for GET and DROP of the light. That is because it is really two separate Items (one lit and one unlit) and you can see that the entries for turning the light on and off actually swap the items with the Instruction SWAP.

The first two items are already present and so you should double click to edit the Name and Description to match what we have here and also the Start Location for Item 2. From Item 3 onwards you will need to create new Items and try to create them in order so that the IDs we use later will match up. Don't panic if they don't - just change the IDs we have given to the right ones for the item you created when making later edits.

Item 1
Name: Lit LED Light
Description: A bright LED light with a self-regenerating battery

Item 2
Name: LED Light
Start Location: 7
Description: Looks like it would be very bright.

Item 3
Name: Phaser
Special Pickup: checked
Keyword: Phaser
Start Location: 3
Description: A standard issue phaser set to Stun. It aslo seems to be very, very hot.

Item 4
Name: Tricorder
Keyword: Tricorder
Start Location: 3
Description: A Tricorder. If you know how it works you are better than I.

Item 5
Name: DiLithium Crystal
Keyword: Crystal
Start Location: 6
Description: Powers lots of stuff

Item 6
Name: Oven Glove
Keyword: Glove
Start Location: 5
Description: Protects against hot things

These are basic objects with nothing fancy about them so far. Item 2 (and its alter ego Item 1) are different because Item 1 is always a source of light. Item 2 is the light as it is first found, turned off, and it has Special Pickup checked so we can take special actions on it to make it light up. Additionally, Item 3, the phaser, is also a Special Pickup and has had two entries created in Event (go check). Special Pickup means we want more control over this Item and that is why IFAC creates entries in Event for you. The use of this will become clearer as we go along but, to apply Conditions and Instructions to its use we have to mark it as a Special Pickup. The detailed meaning of this setting will be explained later as we provide more functionality.

So now (if you didn't do it yet) go back to the Item tab and enter the Items listed above in the order above.

Once you are done your Item table should look like the one below:-

ID ▲	Name	Keyword	Description	SP	Wear	Start	Dis...
1	Lit LED Light	LIGHTON	A bright LED light with a self-regenerating battery	false	false	0	false
2	Light	LIGHT	Looks like it would be very bright.	true	false	7	false
3	Phaser	PHASER	A standard issue phaser set to Stun. It is also very hot for some reason.	true	false	3	false
4	Tricorder	TRICORDER	A Tricorder. If you know how it works you are better than I.	false	false	3	false
5	DiLithium Crystal	CRYSTAL	Powers lots of stuff	false	false	6	false
6	Oven Glove	GLOVE	Protects against hot things	false	false	5	false

If you have that correct then let's quickly move to the Vocabulary tab and if you take a look at the table there you will see that it has more entries than it had before because there is now a Vocabulary Word for every Item we entered. It is, in fact, a copy of the contents of the Keyword field that we entered in the Item form. These are automaticlaly created for you to save a lot of time and potential errors expanding the game's vocabulary. Also to make it clear these Vocabulary words are for Items, the Item ID is displayed in the right hand column.

Try and double click one of the new Vocabulary words for an Item. You will see an alert stating that you cannot edit it. You can't edit it here but only on the Item tab. More on that in the next paragraph.

One important thing IFAC will do is if you decide that your light source is actually a torch not a light then you can change it in the Item tab and IFAC will keep all the Vocabulary and Event entries in sync automatically. Change the Keyword on our Item 2 to TORCH. Check the Vocabulary tab and you will see the entry for light has changed to torch, but even more importantly go to the Event tab and see that all the entries there have changed to use torch keeping everything in sync. This is a very powerful feature of IFAC which can save large amounts of time if you start editing and changing your adventure, as well as prevent errors.

If you remember on the Event tab there were also entries there. These were initialized by IFAC but if you create an Item that is Special Pickup it will also creates entries in Event for you too.

We're using an LED Light though, so go back to the Item tab and change it back to LIGHT so that the rest of tese instructions will be consistent.

Now that we have seen all the extra work IFAC does for us when creating Items we can move on to...

Testing the Adventure (Again)

We have made major additions so we should test again. You should move through all the locations like last time, but this time you should test GETting, DROPing and EXAMining (or EXAMINE) all of the items you find at their locations. All the commands in Caps are followed by the Item Keyword. For now, pick them up and drop them in the same location. You doin't have to use caps - I use them here to highlight them to you. Try using the I (or IINV or INVENTORY) command periodically to make sure you actually pick up and drop the Items. Try to EXAMINE them when you are in the room with them, when you are holding them (in your inventory) and try and EXAM something that is in another room to see what happens.

Take particular note of what happens (or doesn't happen) with our Special Pickup Item - the phaser whcih starts on the Bridge.

Done Testing? OK, so you should have been able to pick up and drop everything except the phaser. All that Item manipulation of GETting, DROPping and EXAMining was provided for you by IFAC without any extra work on your part.

So what's happening with the phaser? It's a Special Pickup. What that means in practice is that IFAC does not provide automatic pickup and drop for that Item because you want to do something more complicated with it. Remember the entries in the Event tab that we had you look at briefly before? Those will implement the GETting and DROPing so click on the Event tab now and double click the DROP PHASER row in the table. The entry will fill in the form on the left for us to edit it. In the Instructions box enter DROP 3 and submit the form. Edit the GET PHASER row by double clicking it and enter GET 3 in the Instructions box. The number here is the ID of the Item, check the ID of the Phaser in the Item tab and use that ID.

Test the game again and just concentrate on checking out the phaser and that you can now pick it up and drop it.

You may ask why did we bother. We could have just let IFAC handle it, but now we have more power over what happens with the phaser. This power can be used to put Conditions on how and when it can be manipulated. Let's apply some conditions under which it can be picked up. If you are not on the Event tab go back to it and double click the GET PHASER row to edit it. In the Conditions section enter this text CARRIED 6 and submit the form. This means there is condition that has to be met before the Instructions are executed. In this case we must be carrying Item 6, which is the Oven Glove - this will protect our avatar from the heat of the phaser.

Test the game again to see what happens when you try to GET PHASER while carrying or not carrying the Oven Glove.

So you can pick up the phaser with the glove now but things still aren't quite right. If you try and pick up the phaser without the glove the program says you cannot. We need to tell the player why their action did not work. for that we will need a Message - so click on the Messages tab. In the form add this text "The phaser is far too hot to touch with your bare hands." and submit the form. That message will appear in the table with an ID of 3. Remeber the ID as we will need it in a minute.

Go back to the Event tab. We need a new Event for when the player tries to GET PHASER without carrying the Glove. Enter this in the form:-

Word1: GET
Word2: PHASER
Conditions: PRESENT 3
NOTCARR 6
Instructions: MSG 3

This tells IFAC that if the player enters GET PHASER and Item 6 is NOTCARRied (not in theplayers inventory) to print out Message #3.

Test the adventure again picking up and dropping the phaser with and without carrying the glove and check that all the output matches what we expect.

It works! BUT the more logical scenario would be for the player to have to WEAR the glove in order to not be burned. So let's fix that. Go to the Item tab and double click the Glove to edit it, check the Wearable checkbox and submit the changes. Check the table to ensure Wear is now true for the Glove. Click on the Event tab and edit the entries for GET PHASER. In the first entry (with ID 7) change the Conditions to be WORN 6. In the second entry (with ID 9) change the NOTCARR 6 to be NOTWORN 6. Now we need to add two more entries for the actual WEARing and REMOVEing of the glove.

Word1: Wear, Word2: Glove, Conditions: CARRIED 6, Instructions: WEAR 6

Word1: Remove, Word2: Glove, Conditions: WORN 6, Instructions: REMOVE 6

Test the adventure again and this time verify that the Glove must be worn for the player to be able to pick up the phaser.

We've covered a lot of ground here and introduced some real complexity into how we can limit the use of Items by applying Conditions in the Event Table and give the player informative messages. This type of logical process is what will enable you to design puzzles and specify their solution in your game.

Congratulations! You've taken a huge leap forward in harnessing the power of IFAC. We're going to introduce another couple of Instructions to handle the dark Storage room and turning the light on and off and then we will describe the Event table Conditions and Instructions in detail.

Darkness

Making locations dark is one of the simplest ways to complicate things for the player. This is especially true if a light source is initially difficult to obtain or has a limited life (new batteries after 20 turns for example). In a dark Location the player cannot see the Location description nor any Items that are at that Location. As a result of that the player can't GET anything or EXAMINE anything either.

So let's set about making our Storage room dark.

There is a boolean (true or false) System Flag that controls whether it is currently dark. We can set or clear this Flag with the Instructions MAKEDARK AND MAKENOTDARK. (It's not makelight because that would just cause confusion with the actual light Item). So we know the Instructions we need but we can only issue them in the Event entries. Consequently we need to shift the movement in and out of Storage from the Movement section to the Event section.

To disable the Movement in and out of Storage we need to remove the entries in the Movement table for the Dining Room EAST movement and the Storage WEST movement. Click on the Movement tab and double click those entries in the table to edit them and set E to NONE for LocID 4 and W to NONE for LocID 6. If you tested the game now you would not be able to move into the Storage room.

To replace that movement and apply the darkness we will add two entries to the Event table. Click on the Event tab and enter these two new entries:-

Word1: E		Word1: W	
Word2: *		Word2: *	
Conditions:	AT 4	Conditions:	AT 6
Instructions:	MAKEDARK	Instructions:	MAKENOTDARK
	GOTO 6		GOTO 4

Test the game again and see that when you now enter Storage it says it is dark and when you leave to the Dining Room it is light there. Note that we are in effect making the whole game dark when we do this which is why we turn it back to not dark when we leave the location - it gives the appearance of just the Storage being dark. Why would we do this and not associate the darkness with the Location? Because if you want to do a power cut scenario the darkness would be for many locations at the same time, and this way we only have to set the darkness once.

Item Swapping

Item swapping is very useful for creating Items that have two mutually exclusive states. Almost always a lit and unlit light source but applies to an open or closed safe/cupboard or a long sword and a broken long sword.

We shall explain with the almost universal one - the light source. We already have two LED Light Items, our

job as game designer is to ensure that, like Highlander, there can be only one... seen by the player at any given time. More simply, they swap places when the light is turned on or off.

If you go to the Voacblulary tab you will see that we have already got the words we need as part of IFACs default setup - LIGHT, ON and OFF.
Then got to the Messages tab and check out the messages provided by IFAC for turning the light on and turning it off. The ones we provide are very simple and you can feel free to alter them to be more atmospheric as you wish. The Message with ID 1 is for turning the light on and one for turning off the light is ID 2. Use whatever text you want for the messages that suit your setting, just keep them to the same meaning.

Now let's look at the Event tab which is where all the hard work gets done.

As we kinda mentioned before the light is, in fact, two Items. A light that is on (Item #1) and a light that is off (Item #2). We need to make them appear as one item to the player and that Item is one that he turns on and off. So we use the same words GET LIGHT on two different Event entries to get the different verisons of the light depending on which one is in existence at the time. We ensure only one of these is executed by having a Condition on each one that checks for the right Item being PRESENT.

Similarly for DROP LIGHT it is CARRIED 1 with DROP 1 and OK for the first entry and CARRIED 2 and DROP 2 and OK for the second one.

Finally to turn the light on or off we swap the two items over. We check which one we currently have first so that we can output the appropriate message but the swap is the same. The SWAP command exchanges the Locations of the two Items that it references, irrespective of where they are. In this case one will be a Location or Inventory (which is a type of Location) and one will be zero or null (which means it does not currently exist in the game visible to the player).

If you wanted to do some teleporting with other items you could teleport them across a pair of teleport pads with this too. Assuming one Item was on a pad and the other was on a pad in a different room.

So that is how the light works. Test your adventure again turn the light on and off when it is in your inventory or when it is visible at the location. The commands are LIGHT ON and LIGHT OFF.

Once you verify that is working you can test going into the Storage with the light on and off again. Turn the light on and off while you are in there and LOOK with the light in both states. Try GETting and DROPing the DiLithium Crystal in the light and the dark also so that you can see the effect of having the light on GETting Items and EXAMinig them also. It doesn't matter that you know the crystal is there, you can't see it so you can't pick it up.

The Status Table

The Status Table works in a very similar way to the Event Table except where the Event Table operates on the players input and just on those entries that match the input, the Status Table examines every entry it has on every turn and checks the Conditions to see whether to implement the Instructions. The Status Entries are examined in order by Section (up to 255) and within each Section there is a Priority (up to 255) in order to provide very fine grained control over the order that Status Events process and should be enough for even the largest of games.

The Status table entries run regardless of player input after the players actions have been processed.

Let's use this power to ensure that the phaser cannot be held without wearing the glove. Currently you can pick up the phaser by wearing the glove but then remove the glove and still have the phaser. This really does not make sense if the phaser is so hot, so we will make it that removing the glove make the player drop the phaser because it is hot and provide a message to tell the player what happened.

Firstly, we need a message. Go to the Message tab and add a new message "Ouch! You remove the glove but now the hot phaser touches your bare hand so you immediately drop it to the ground". Make a note of the ID of this Message - it should probably be 4 if you have been following along.

Now go to the Status tab. As we are only going to have one entry you can leave the Section and Priority values at their default for the first one. In Conditions enter CARRIED 3 NOTWORN 6 and in Instructions MSG 4 DROP 3 (each command on a sparate line). If the message had a different ID use it here.

Once we have that saved test the game again and pick up the phaser with the glove, then remove the glove and the phaser should drop automatically. Notice that this happens no matter where you are as we did not specifiy any particular location in the Conditions, so now the player is forced to wear the glove at all times if he wants to carry the phaser.

Let's do one more thing using Status, System Flags/Counters and a Pickup Disabled Item (which can't be picked up, ever). Make a new Item named Big Red Button, check Pickup Disabled, keyword Button, starting at Location 4, any description you want. Make two new Messages - "Now you've done it. Ship destructs in three turns" and "Boom! Now don't press it again!". Make a new Vocabulary entry for PRESS. To allow the player to press the button we need a new Event.

Word1: PRESS
Word2: BUTTON
Conditions: Present 7
Instructions: Msg 5 (or the ID of the now you've done it message)
Syslet 4 4
and now a Status entry:-
Section: 2
Priority: 128
Conditions: Syseq 4 1
Instructions: Msg 6 (or the ID of the Boom message)

Test the adventure again and press the button. After you have pressed it take a look at the Flags tab and you will see that Turns Decrement 1 is set to 3. Each time you enter a command now that flag will decrease by one. It already got decremented once after you pressed the button which is why we set it to four. When it equals one (Syseq 4 1) the Boom message is output. There are 12 System Flag Counters and 50 User Flag Counters and their state can always be checked here. The System ones are values that are altered by the system at various times. They are named to describe those times but a detailed explanation will be given in PART 3 of the manual.

In the last few sections you have learned some very powerful stuff while only using a few types of commands, but you have learned how to combine them in ways that produce logical game behavior and puzzles. We will detail all the commands you have at your disposal later but for it is time to...

Finish the Adventure

I'm not sure we ever said how we would finish the adventure. Well, we've made it simple. Remember the pedestal on the Bridge. If we have the DiLithium Crystal when we enter, the pedestal will detect it and will power up the Bridge and we will have won the game!

We need a game winning message to congratulate the player first. Go to the Message tab and create one of your own choosing or use ours, which is "As you enter the Bridge the pedestal and the DilIthium Crystal both begin to hum and glow and in a second power returns to all the screens and consoles. You have succeeded in repairing the ship. Congratulations!"

This is very simply achieved with one Status Entry. We probably want this one to run before anything else so set Section to 1 and Priority to 1. Here are the Conditions and Instructions:-

Conditions:	AT 3	
	PRESENT 5	
Instructions:	MSG 6	(or the ID of the win message just entered if it is different)
	SYSPLUS 12 100	
	SCORE	
	TURNS	
	END	

It's time for the final and full test of our adventure. Run through the whole game and finish in the Bridge.

CONGRATULATIONS TO YOU! Your first adventure game is complete. It's time for you to learn all the conditions and commands IFAC has to offer so that you can make your own masterpiece of interactive fiction or adventure game.
Remeber to share your journey on https://ifac.club/community where we and the community are always available to help and encourage you.

PART 3 - The IFAC Interface

IFAC splits its interface into many tabs that each provide for the input and display of certain types of data used in the game. These tabs can be freely selected at any time, even when playing or testing your world so they assist in debugging. Also, typos or other errors can be fixed on the fly, which is a very convenient and powerful feature that speeds development.
Changing Staus entries while playing can be tricky and should be done with care. If an entry relies on other entires for a chain of actions you may have to start playing from the beginning to get an accurate test.

Location Tab

This is where you enter and edit your locations. In adventures, locations are physical locations or what the player "sees". In fiction, locations are the segments or scenes of the story.
Most of the tabs follow the format of the location tab with an entry/edit form on the left and a table for the current data on the right.
The Name is a tag for the location which is also the short description in adventure games (displayed when the player has been here before). It is usually very brief - just 3 or 4 words max.
The Detail is the full or long description, displayed the first time a player visits the location in an adventure or the text that is always displayed for a fiction story.
To add a new location just fill out the form and click Submit. To edit an existing location double click the entry in the table and the form will populate with that entry's data for you to edit and Submit. That mechanic is also replicated on subsequent tabs and we will not repeat those details for the sake of brevity.

Movement Tab

This is where you edit the movement entries for a location. Movement entries are created when a location is created so you will always be editing existing records here by double clicking the table entry. The dropdowns only allow you to select an existing location number (ID) and by doing so you tell IFAC where any of the compass directions together with up/down move the player to. In Interactive Fiction mode the player should be asked for selections of 1, 2, 3, 4 which correspond to N, E, S, W respectively to get to the next scene.

Items Tab

Where you enter and edit the items. (We're following an obvious pattern here but while the tab might be obvious some of the fields here are not and need some explaining).
Name is the name of the object and is also it's description when it appears in the game. Keep it short and sweet.
Pickup DISABLED - All items are made automatically GETable by default by using GET [item keyword]. This option disables that functionality and the item cannot be picked up even though it displays as an object in the game. This is useful for objects that will be interacted with but you don't want them being moved all over the place by the player.
Special Pickup - Selecting this tells IFAC that you do not want to use the default GET in the interpreter but instead you will provide entries in the Event Table that will detail the conditions and instructions for picking up this item. Used when a condition, such as possessing another item, is required to GET this one.
Keyword - is the word that the interpreter will use to identify this object when analysing player input. One word only.
Wearable should be selected if this item can be worn by the player.
Start Location is the place where the object starts the game. NONE means it does not exist yet in the game world. 1 is the Inventory and would mean the player begins the game carrying this object. All opther selections are the ID of the location for the object to be at the start of the game. Only currently existing locations are present in the list.

Vocabulary Tab

In this tab you enter the words that the interpreter will recognize. When you start with a new database (game) you will notice that there are many items pre-populated in the table. These are the common commands that every game needs in order to operate and they are created so you don't have to do them.
Another thing to note is that when you create an Item the keyword of the Item is entered in the Vocabulary automatically. This is another time saving feature and it ensures that the auto GET works.
To enter a new word ensure that NEW is displayed at the ID field (if not, then clear the form with the Clear Form button) and type in your word. You have to assign a synonym ID so if this is a new word that is NOT a synonym of an existing word then the easiest thing is to click the Next Synonym ID button to fill this field and then click to Submit to create it.
If you are creating a word that is a synonym for an already existing word then set the Synonym ID to the same value as that existing word. So if "cheese" and "cheddar" are used for the same thing in your game set their synonym IDs to be the same.
You cannot edit the pre-populated (system) words at all, nor can you edit the Item words on this tab. To edit Item keywords do it on the Item tab - it will automatically be updated here to save you having to change it in two places or make the error of only changing it in one of the tabs. Another time saving feature that also protects against mistakes.

Messages Tab

Messages are used to give the player extra info about what they did or about game events that may have been triggered. One of the simplest tabs - type your message and save it or double click to edit and Submit to update.

Flags Tab

This tab is useful for debugging your game if things are not going as you planned. Nothing here can be changed but you can see the value of all 50 of the user Flags in the table and the values of the System Flags and Counters on the left. It is particulary useful to check the values of the System Counters that decrement (subtract one from their value) either each turn or under other conditions. These decrement counters are used most often to give the player a limited amount of time to accomplish something before something else (usually bad!) happens. You have five turns to fire the retro rockets or have four turns in the dark before you are eaten by a grue (go look it up on Wikipedia for a quick history lesson)

Settings Tab

Sundry settings.
CYOAM mode uses 1,2,3,4 choices and full descriptions always - use it for Interactive Fiction.
Font and Windows size - allows changing to a larger font and bigger window which is useful on some systems or if you just find the standard too small.
The bottom section is the adventure database file selector. I STRONGLY recommend you keep your files in the same IFAC folder the application creates. If you decide not to then please keep a list of where all your files are as if the settings file gets corrupted for any reason you will need to be able to find them again for the application.

When you change settings here and Save them, IFAC will warn you it has to close. You will ned to relaunch IFAC for the settings to apply becasue these settings have to be applied on application startup.

System Messages Tab

The game interpreter has certain messages that it uses to report to the player whether actions such as GETting an item have been successful, if it does not understand what was typed or the player tries to look at something in the dark.
On this tab you can edit those messages to better suit the game world you are creating.You may want to change them to a more ropbotic tone for a high tech spaceship based game, for example.
As always, double click to edit and Submit to save changes.
As an extra feature you can change all the messages to any of the three supported interface languages by one button click.

Credits Tab

Acknowledgements to folks who have helped along the way and some things we just wanted to say.

Event Tab

This tab is one of the most important in the application because it is here that you determine the verb and noun combinations that your game recognizes and what it will do when the player issues these commands (Player Events). This is one half of the games brain that reacts to the player, changes the world accordingly and then lets him know what happened as a result.
Word 1 - Usually the verb. One word only.
Word 2 - Usually the noun. One word only.
Conditions - a list of IFAC language condition keywords and their parameters. One per line. ALL of the conditions must evaluate to True in order for the Instructions to be executed. IFAC language Conditions are listed in PART 4.
Instructions - a list of IFAC language instruction keywords and their parameters. One per line. Each instruction will be executed in order from the first to last. IFAC language Instructions are listed in PART 4.

When there are several entries with the same verb/noun pair they will be evaluated in ID order. It is important when you have several entries like that to get them in the right order because only the first one that passes the Conditions will have its Instructions executed. It is highly recommended that you remove any possibility of ambiguity by making the conditions for each entry mutually exclusive of the others so that by your entered logic only one could be executed anyway.

Saving and editing are done as in the other tabs.

Status Tab

This tab is the other half of your games brain. These entries respond to the current state of the game and are processed after the player input and Event entry has been processed.
The order of entries in this table is CRITICAL especially if you want some changes to fire other changes in a sort of domino effect. To that end Status entries have a Section and a Priority within that section. They are processed from lowest number to highest and both have values from 1 to 255 so in combination there are 65000+ possibilities.
As an example an entry with Section 1 Priority 250 will be processed before an entry with Section 2 Priority 1 because any entry in Section 1 will be processed before any entry in Section 2 and subsequent sections.
EVERY entry in the Status table is processed on every turn. Each entry's Conditions are evaluated and if they are true the Instrcutions for that entry are executed. Whether the Instrcutions are executed or not the game engine moves on to the next entry and evaluates its Conditions and so on until it gets to the end of list.

Conditions and Instructions keywords and parameters are detailed in PART 4 and their entry in this form is exactly the same as for the Event Tab..

Test Adventure Tab

This is where you can test (or play) your game world. Simply click the Start button to play as a normal user would or click the Debug button to get some extra messages about which Event got triggered in response to your input and other helpful messages that give you more insight into how the game is working behind the scenes.
Type your player input in the box near the bottom and the large game window displays everything else including echoing your commands.
Enter quit to end the game session.

PART 4 - IFAC Language Reference

For all Conditions and Instructions they may need to be used with zero, one or two integer parameters which are noted in their definitions by Int.

Condition Keywords

AT Int
Checks the players current location ID to see if equals the parameter and return TRUE if it does.

NOTAT Int
Checks the players current location ID and if it is not equal to the parameter it returns TRUE.

ATGT Int
Check the players current location ID and if it is greater than the paramter returns TRUE.

ATLT Int
Check the players current location ID and if it is less than the parameter return TRUE.

PRESENT Int
Returns TRUE if the Item with ID of the parameters is at the players location or in inventory.

ABSENT Int
Returns TRUE if the Item with ID of the parameter is not at the players location or in inventory.

WORN Int
Returns TRUE if the Item with ID of the parameter is currently being worn by the player.

NOTWORN Int
Returns TRUE if the Item with ID of the parameter is not currently being worn by the player.

CARRIED Int
Returns TRUE if the Item with the ID of the parameter is in the players inventory.

NOTCARR Int
Returns TRUE of the Item with the ID of the parameter is not in the players inventory.

CHANCE Int
Returns TRUE if the value of the parameter is greater than a random number generated by the computer. In practice this equates to a percentage chance. So if you want something to have a 25% chance of returning TRUE then you would use CHANCE 25 as your condiiton.

ZERO Int
Returns TRUE if user flag number denoted by the parameter has the value zero.

NOTZERO Int
Returns TRUE if the user flag number denoted by the parameter has a value greater than zero.

ISDARK
Returns TRUE if the Dark flag is currently set.

NOTDARK
Returns TRUE if the Dark flag is currently not set (or cleared)

EQ Int Int
Returns TRUE if the value of the user counter defined by the first parameter is equal to the value of the second parameter.

GT Int Int
Returns TRUE if the value of the user counter defined by the first parameter is greater than the value of the second parameter.

LT Int Int
Returns TRUE if the value of the user counter defined by the first parameter is less than the value of the second parameter.

SYSZERO Int
Retruns TRUE if the system flag number denoted by the parameter has the value zero.

SYSNOTZERO Int
Returns TRUE if the system flag number denoted by the parameter has a value greater than zero.

SYSEQ Int Int
Returns TRUE if the value of the system Flag defined by the first parameter is equal to the value of the second parameter.

SYSGT Int Int
Returns TRUE if the value of the system Flag defined by the first parameter is greater than the value of the second parameter.

SYSLT Int Int
Returns TRUE if the value of the system Flag defined by the first parameter is less than the value of the second parameter.

Instruction Keywords

INVEN
Outputs the contents of the players Inventory.

DESC
Outputs the full detailed description of the current location.

SCORE
Outputs the players current score which is stored in System Flag #12.

TURNS
Outputs the number of turns the player has made in the current game.

GOTO Int
Immediately changes the player's location to the location with the ID of the parameter.

MSG Int
Outputs the user message with the ID of the parameter.

GET Int
Attempts to get (pick up) the item with the ID of the parameter.
NOTE: This executes exactly the same way as if the player had issued a GET command to the interpreter. If this is the implementation of a Special Pickup that is fine but if it is not then to avoid unexpected messages it is normal to have a **PRESENT** condition for the same object ID in the Condtion block whereever you plan on using the **GET** command.

DROP Int
Attempts to drop the item with the ID of the parameter.
NOTE: As above this executes in the same way as if the player had issued a DROP command to the interpreter. Unless implemeting a Special Pickup then usually match this up with a **CARRIED** condition.

WEAR Int
Attempts to wear the item with the ID of the parameter.

REMOVE Int
Attempts to remove the item with the Id of the parameter and leave it in the inventory.

DESTROY Int
Sets the location of the item with ID of the parameter to the null location, effectively removing it from the game world.

CREATE Int
Sets the location of the item with ID of the parameter to the current location, effectively moving it into the game world.
NOTE: Unless the player LOOKs after this has happened he will not see that the item is there so you will most usually output a **MSG** to inform the player that this item just popped into existence.

SWAP Int Int
Swaps the locations of the two objects identified by the IDs passed as parameters. This is most obviously used for objects that have multiple states such as the lamp (lit and unlit) but can be used to swap items over that are on two teleport pads that are in different locations.

MAKEDARK
Sets the Dark flag.
NOTE: Once you do this EVERYWHERE is dark. It's implemented this way so you can have a power failure in your world if you wish. Usually though you just want it to be dark in specific locations. You do this by setting the Dark flag with this instruction when the player enters the location.

MAKENOTDARK
Clears the Dark flag.
NOTE: Relating to the previous command you need to use this when the player exits the dark location. This requires using the Event entries for movement. An example is in the small adventure that you should have worked through earlier in the manual.

SET Int
Selects the user flag with ID identifed by the parameter and changes its Flag value to TRUE and Counter value to its max value (currently 10,000 as I write this).

CLEAR Int
The user flag with ID identified by the parameter has Flag set to FALSE and Counter to zero.

LET Int Int
Assigns the value of the second parameter to the user counter identified by the first paramter.

PLUS Int Int
Increases the value of the user counter identified by the first parameter by the amount of the second parameter.

MINUS Int Int
Decreases the value of the user counter identified by the first parameter by the amount of the second parameter.

SYSLET Int Int
Assigns the value of the second parameter to the system counter identified by the first paramter.

SYSPLUS Int Int
Increases the value of the system counter identified by the first parameter by the amount of the second parameter.

SYSMINUS Int Int
Decreases the value of the system counter identified by the first parameter by the amount of the second parameter.

SYSCLEAR Int
Clears to zero the system counter identified by the parameter.

Appendix A - IFAC Player

There are three commands specific to the IFAC Player and all relate to save games.

All the text you use for game names is filtered so stick to upper and lower case letters only please to avoid unexpected results.

SAVE [savedgamename] - saves the current game state

LOAD [savedgamename] - loads a previously saved game state

and GAMELIST which will list the names of the games you have saved.

Export is available from the Settings tab and will export to:-

{User home}/IFAC/GameName

and the players for all platforms will be under there in their own folders.