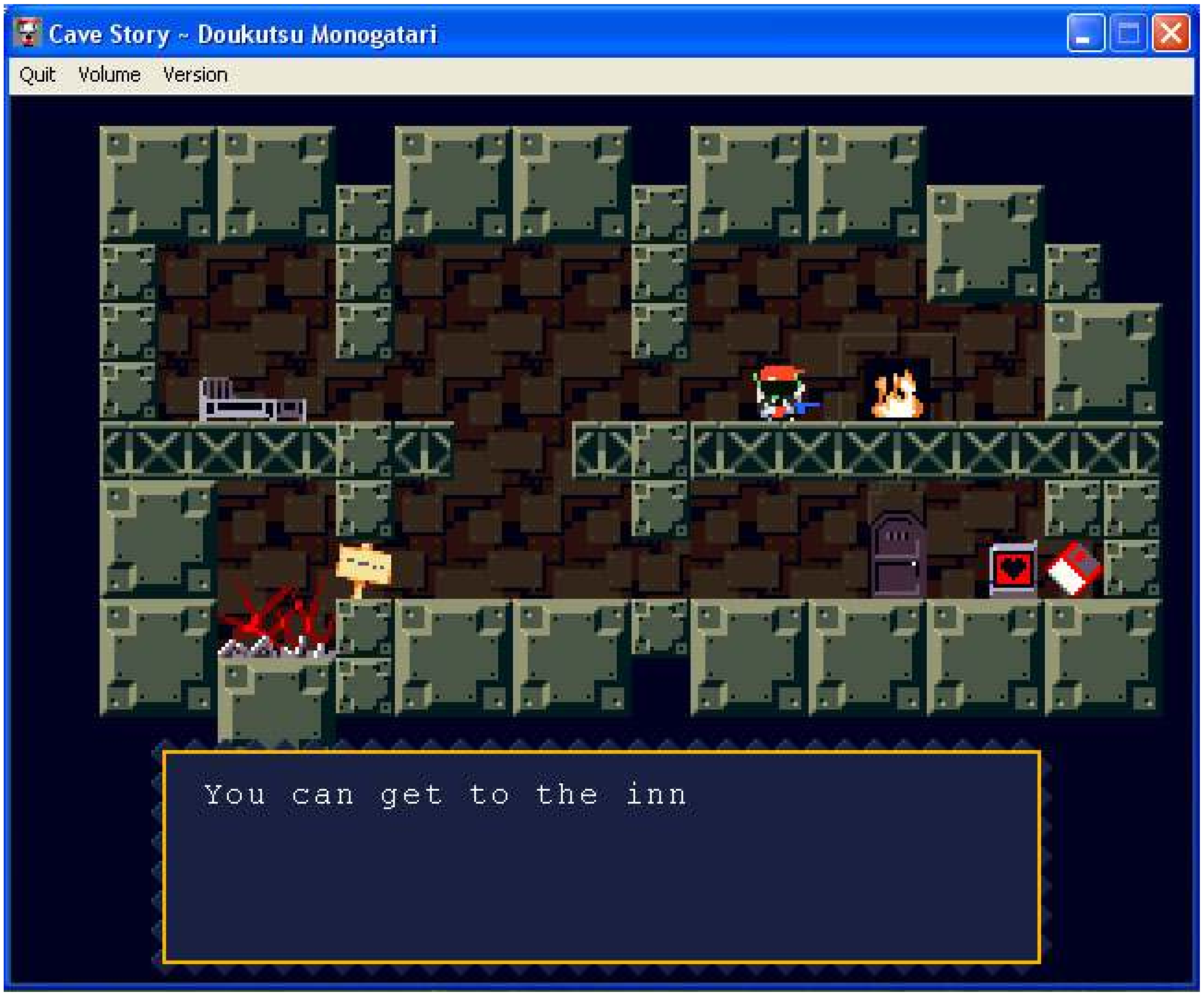
Adam's In-Game Character Dialogue Tutorial

Adam "mrman" Gledhill, Feb 27th 2007 (MMF2 Build R243)

This tutorial shows you one way to create a dialogue system in MMF. There are lot's of ways it can be done, but this one uses ini files. You should already have some working knowledge of "eventing" and the MMF interface.

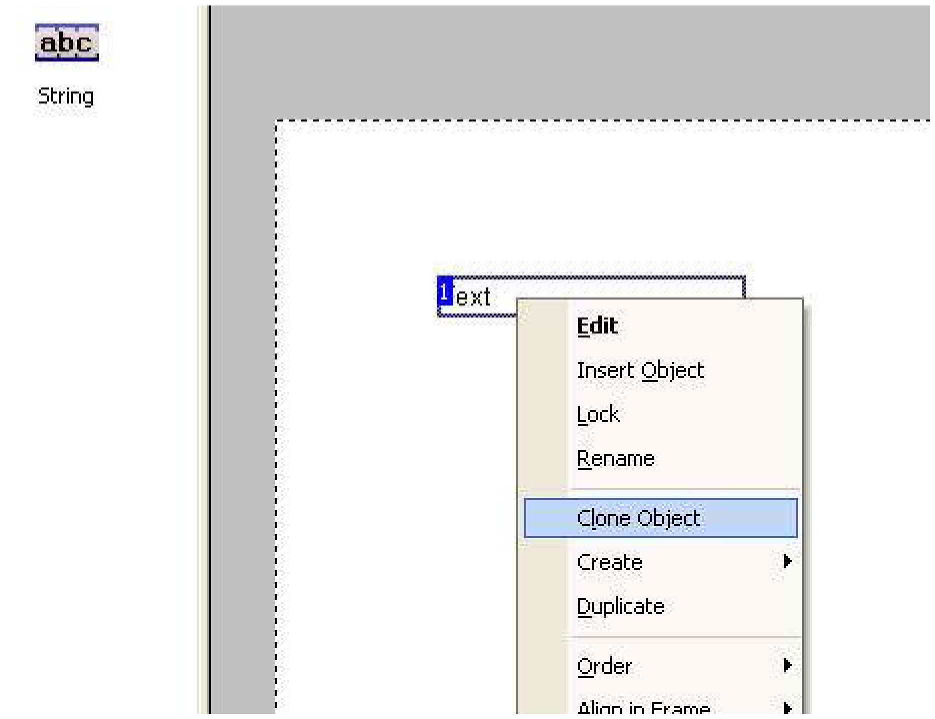
Part 1. Setting up strings and creating the classic type effect!

The first thing we'll do is recreate that classic RPG dialogue effect of having a characters words appear on screen one letter at a time, as though it's being typed.

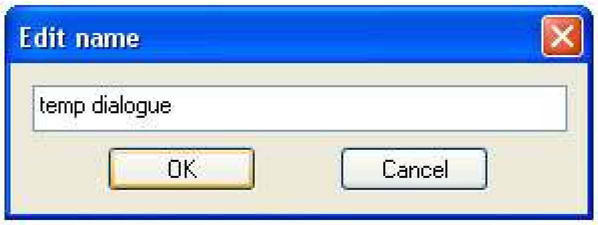


It's very easy to do. All you need is 2 string objects and 1 event.

Let's create the strings now on a new MMF frame.
Just make one, resize it, then clone it to make the other. This will make sure they're the same size.



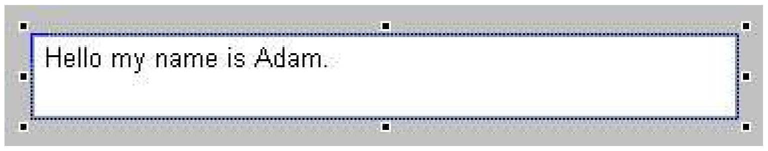
Rename the first string object **"temp dialogue"**, and the second string object **"dialogue"**.



"temp dialogue" will be used to store the whole complete dialogue we want a character to say. This should be moved off the frame so it's not seen when the games being played.

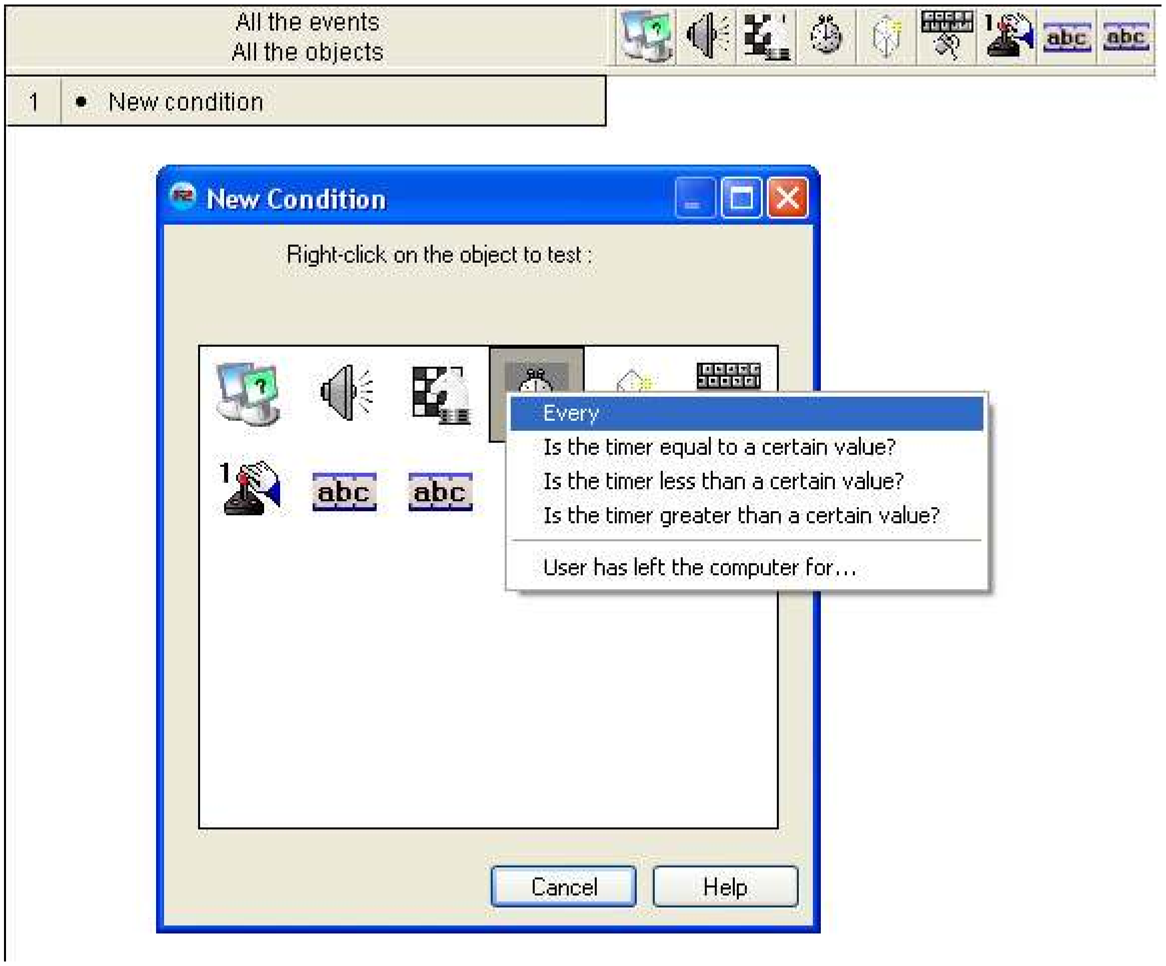
So now we need to type some text into the **"temp dialogue"** string, our actual dialogue. Double click it and type in a sentence, or enter it in the object properties panel. I just wrote:

"Hello my name is Adam".

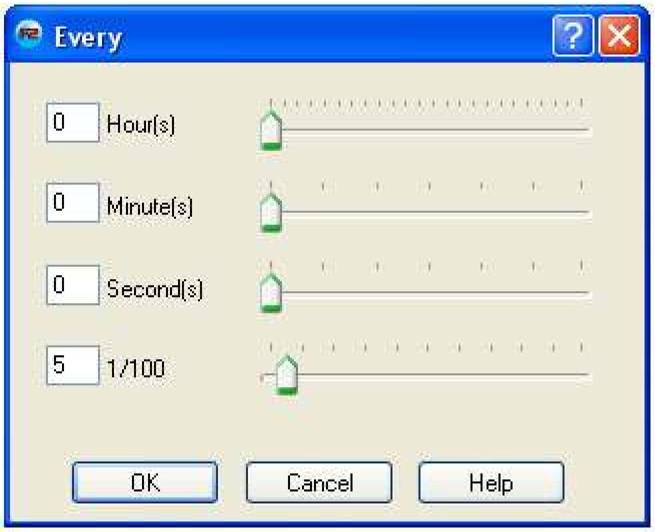

Hello my name is Adam.

Next we'll use an event to copy the dialogue we just wrote in **"temp dialogue"** over to **"dialogue"** one letter at a time.

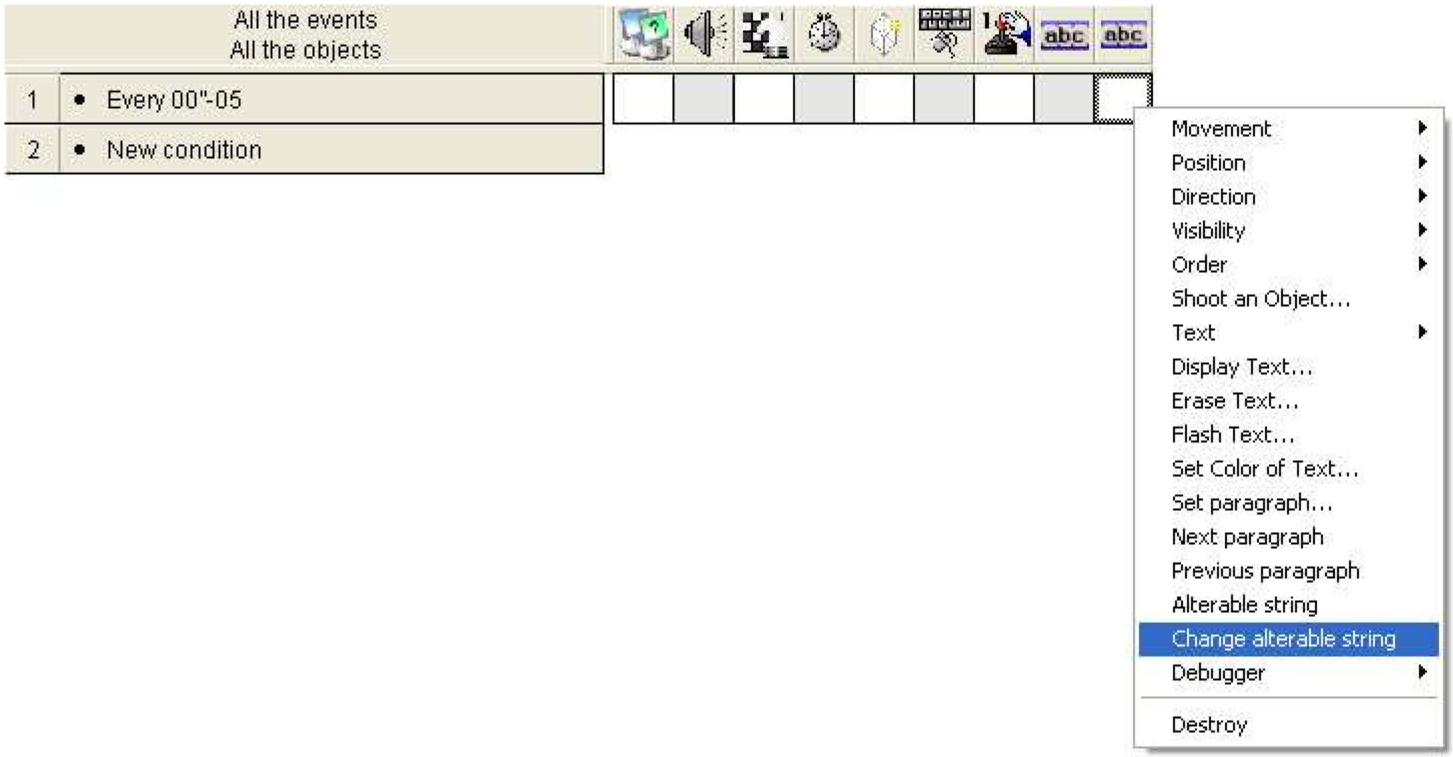
So in the Event Editor click on line 1, right click on the **"timer"** object and select **"Every"** from the menu.


All the events
All the objects
1 • New condition
New Condition
Right-click on the object to test :
Every
Is the timer equal to a certain value?
Is the timer less than a certain value?
Is the timer greater than a certain value?
User has left the computer for...
Cancel
Help

This will control how fast our letter appears on screen. Let's set it to 5 milliseconds.



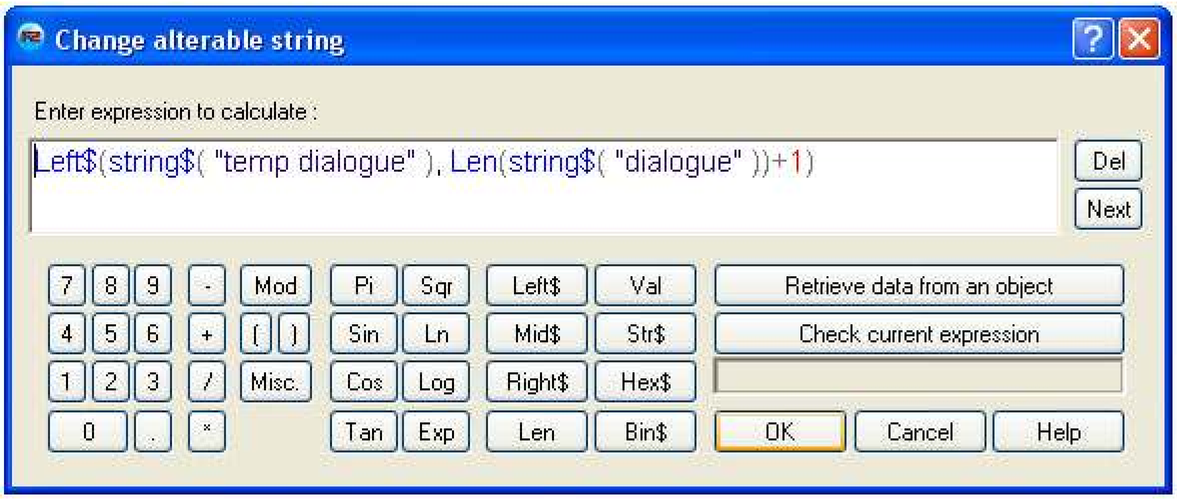
Okay, now right click on the **"dialogue"** string box and select **"Change alt string"** from the menu.



Here's our formula:

Left$(string$("temp dialogue"), Len(string$("dialogue"))+1)

Enter it into the expression editor.


Change alterable string
Enter expression to calculate :
Del
Next
7 8 9 - Mod Pi Sqr Left$ Val Retrieve data from an object
4 5 6 + ( ) Sin Ln Mid$ Str$ Check current expression
1 2 3 / Misc. Cos Log Right$ Hex$
0 . * Tan Exp Len Bin$ OK Cancel Help

```
Left$(string$( "temp dialogue" ), Len(string$( "dialogue" ))+1)
```

What's happening here is that every 5 milliseconds we are choosing what letters to copy over from "temp dialogue" based on how many letters have already been copied over to the "dialogue" string (by getting it's length (Len)), then adding 1 (because we want the next letter), then using this figure to count along the "temp dialogue" string starting from the left (Left$) so we pick the next one.

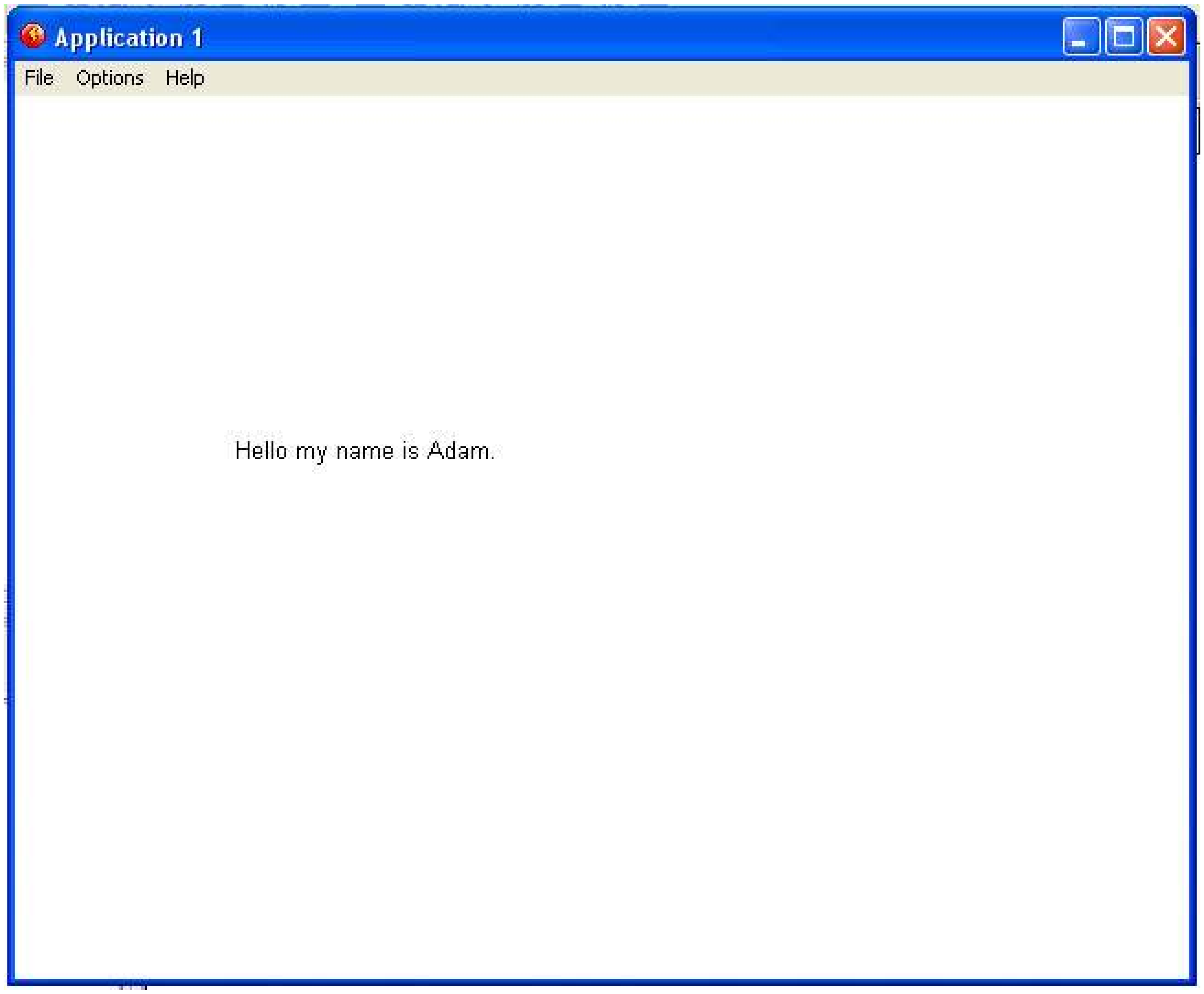
So if there's nothing written in "dialogue" (like when we first run the frame) it's length will be 0 (0 letters)+1=1 so the first letter taken from "temp dialogue" will be "H".
5 milliseconds later it starts again, only now there's a "H" in "dialogue" that we just copied, so that's 1 letter +1=2, so counting along left in "temp dialogue" again the 2 letters are "He". So now "dialogue" "H" is replaced with "He", giving us that classic type effect.

H
He
Hel
Hell
Hello

You get the idea.

Changing the **"every"** event from 5 milliseconds to, let's say 10 milliseconds, will make our little type effect seems slower. Play about with this until you get a speed you like.

Now let's run the frame and see what we get.



Hello my name is Adam.

There is one problem. The **"dialogue"** string already has **"text"** written in it from when it was created. Which means that instead of starting at 0 characters it starts at 4. When we run the frame "Hell" is copied from **"temp dialogue"** all at once, because it gets the length of the string.

To get around this problem add a another event line with the condition **"Start of frame"**, then in the **"dialogue"** string box, right click and select **"Change alt string"** from the menu.
In the expression calculator simply put **""**, which means blank.
At the start of the frame the **"dialogue"** string will now be changed to have no characters.

Cool, let's save our work, stop and admire our nice effect.

Now that we've covered the basics, in part 2 we'll look at how to store the dialogue not in the **"temp dialogue"** string object as we are now, but in a separate .ini file. Read on.

<u>Part 2. Storing dialogue in ini files!</u>

Since this is a dialogue tutorial and not an ini tutorial it would be really helpful if you already understand what ini files are and how they work, but i should probably go over the basics real quick anyway.

An ini file is basically a text file that keeps things organized by arranging all the text into **"groups"**. This special format is good because we can easily access one of these groups and get what text is written inside.

Here's what an ini file looks like.

```
[my group]
my item = my string text
my item : my values
```

Ini's can have as many groups as you like. You can call groups what ever you want so long as they are in- cased in brackets **[ ]**.

Just like ordinary text files, ini files can hold both words (abc, also called strings) and numbers (123, called values).

Each group stores its strings and values in an **"item"**. Again have as many as you like, separate the item name and it's string with an equals sign " **=** " , or if it's a value, with a colon sign " **:** ".

Here's another example of an ini file with some strings and values stored in it.

```
[battle ship]
ship name = Battle Star Galactica
captain = Admiral Adarma
shields : 150
weapon strength : 200
mission = To lead a ragtag fleet beyond the cylons tyranny to find a shining planet known as
earth!

[cylon base ship]
captain = evil Dr boltar!
weapon strength : 500
mission = destroy the galactica!
```

Okay so how are we going to use ini's to store our dialogue?

Like this:

```
[Adam]
say 1 = My name is Adam.
say 2 = Each item represents something I'm saying. It can be long or short, but not too long
because           each item can only hold about 1000 characters (letters/numbers).
say 3 = Ini files themselves are limited to a maximum file size of 64kb.
say 4 = Don't worry though, that's quite big just for storing text.
```

We'll make the group the name of the person who is speaking, and the items will store the different dialogue.

Also you should be aware that ini files are text files so they are not by default secure, this means any one can look at or change the text. But since we are only storing our dialogue it's not like it will help a player cheat.

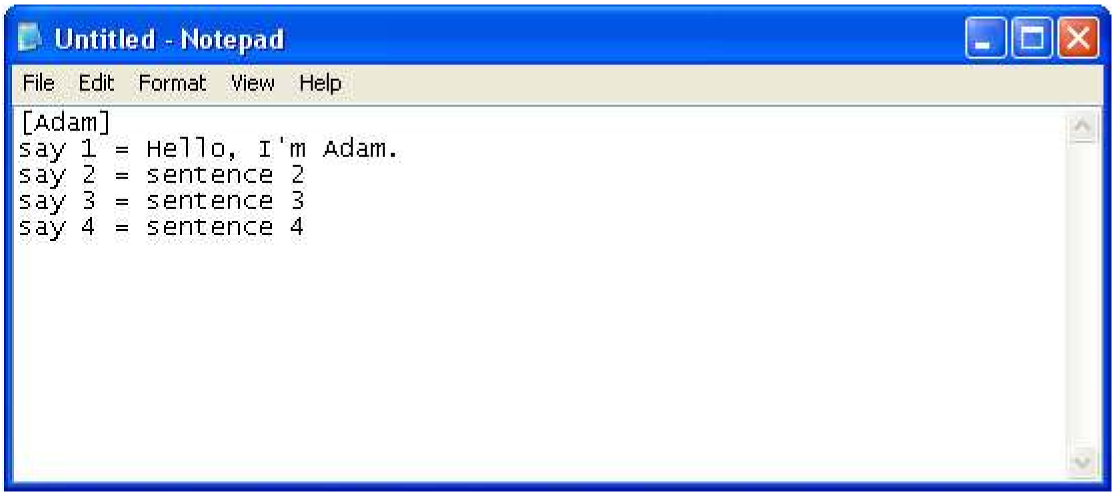
It could also be a good thing in our case. Imagine if you wanted to release your game for a different country. All you would have to do is change the item text to a different language!

Right let's make our ini file.

Open up any text editing program you have. If you are on windows it's best to use Notepad.

Create a new file and type this in it, nothing else.

[Adam]
say 1 = Hello, I'm Adam.
say 2 = sentence 2
say 3 = sentence 3
say 4 = sentence 4

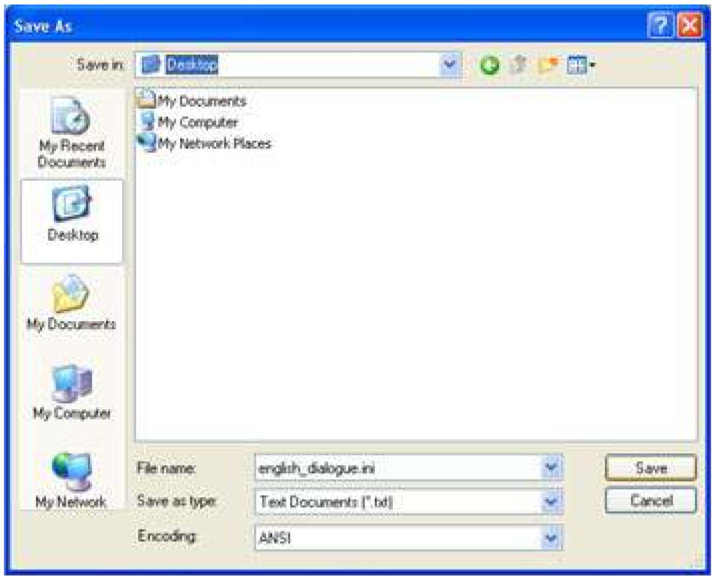


```
[Adam]
say 1 = Hello, I'm Adam.
say 2 = sentence 2
say 3 = sentence 3
say 4 = sentence 4
```

To create the actual "ini file", you need to add the .ini extension when you save the file.

Save it as **english_dialogue.ini**

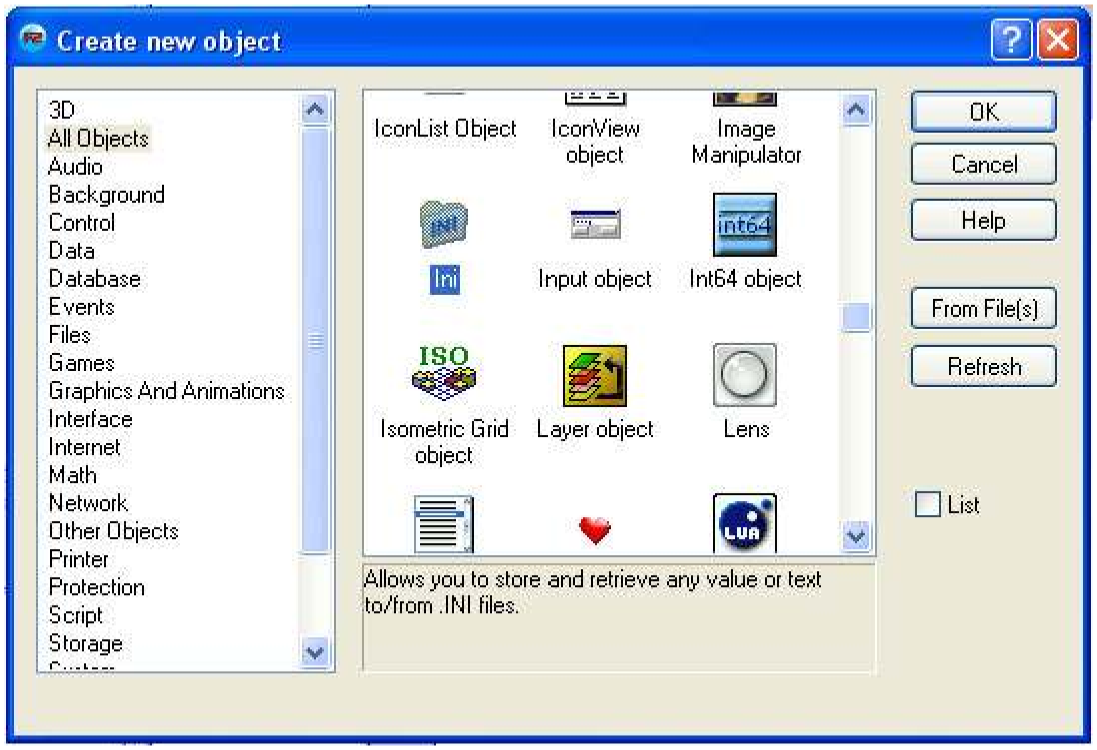
It's also important for now that you save our ini file in the same place / folder that your MMF file is located.
I've saved the MMF file from part 1 of this tutorial to the desktop along with this ini file.



Save in: Desktop
File name: english_dialogue.ini
Save as type: Text Documents (*.txt)
Encoding: ANSI

Right, now we've created our ini file let's open up the MMF file from part 1.

Since we are going to be using an ini file we need to add the ini object to our frame.

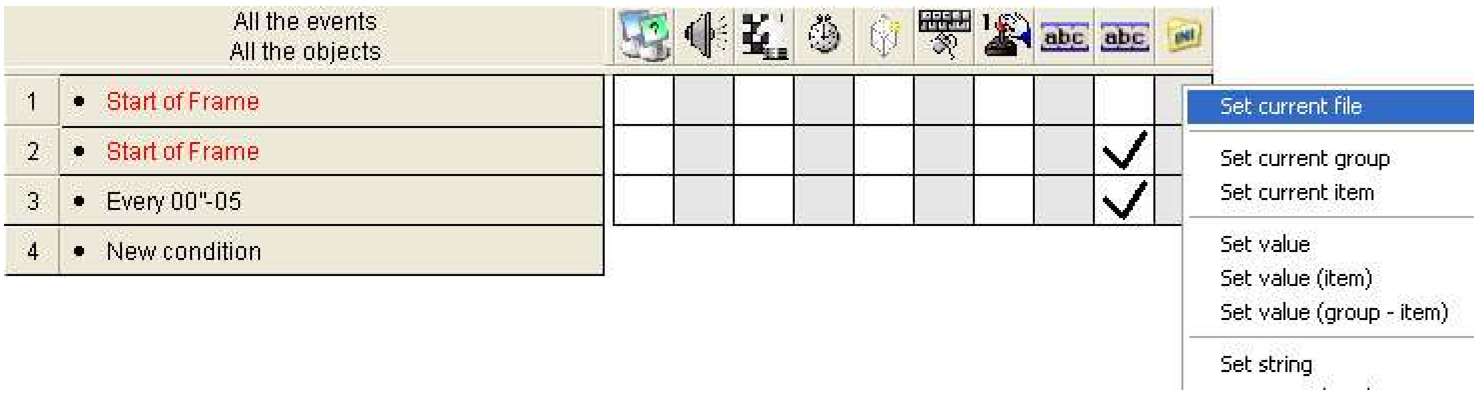


Allows you to store and retrieve any value or text to/from .INI files.

Rather than typing our dialogue straight in the **"temp dialogue"** string object like before we are going to load some text from our ini file into it instead.

But before we can do that MMF needs to know where it is. We'll do that by creating another **"Start of frame"** event which will locates our ini file when the game starts.

So do that, then on that event line, right click the ini box and select **"set current file"**.

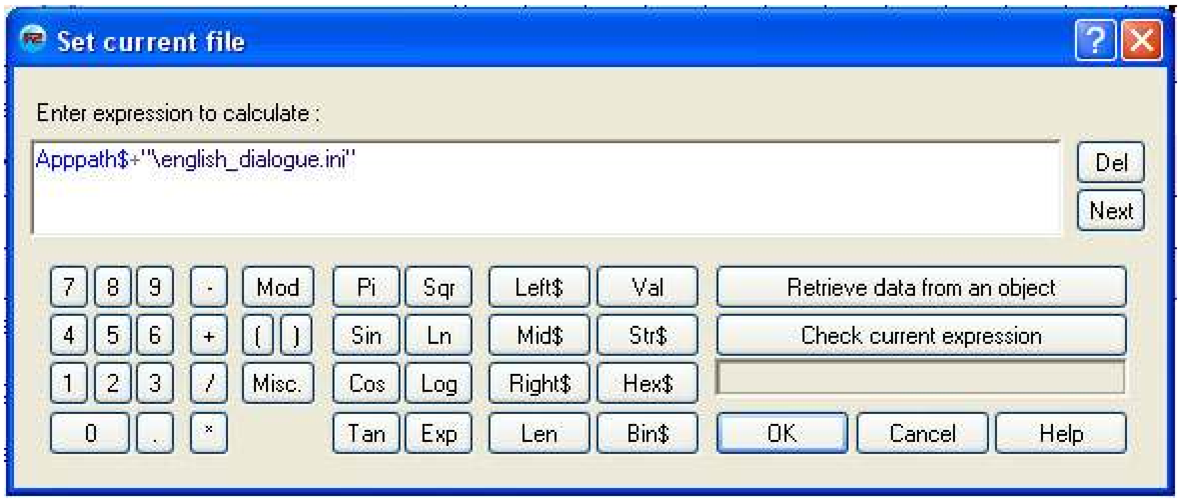


	All the events / All the objects
1	• Start of Frame
2	• Start of Frame
3	• Every 00"-05
4	• New condition

Since the ini file is in the same place as our MMF file, we'll tell MMF to look there.

Apppath$+"\english_dialogue.ini"

Apppath$ is the location, and "\english_dialogue.ini" is our file name.



Set current file

Enter expression to calculate :

Apppath$+"\english_dialogue.ini"

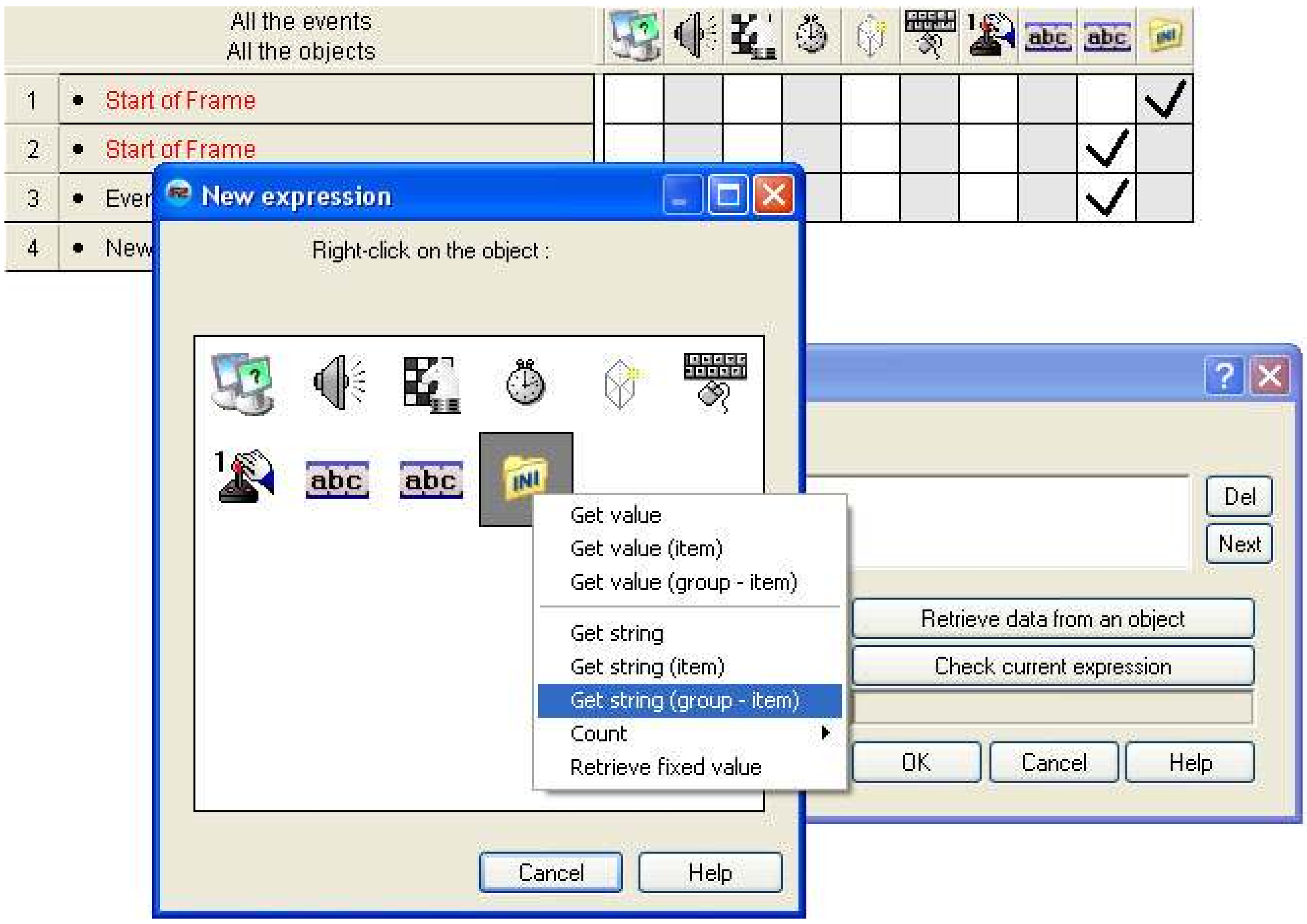
Del
Next
7 8 9 - Mod Pi Sqr Left$ Val
4 5 6 + ( ) Sin Ln Mid$ Str$
1 2 3 / Misc. Cos Log Right$ Hex$
0 . * Tan Exp Len Bin$
Retrieve data from an object
Check current expression
OK
Cancel
Help

MMF knows where our ini is so now we can access it and retrieve some text for our dialogue.

In the same **"start of frame"** event line, right click the **"temp dialogue"** box and select **"change alterable string"**.

In the expression calculator click "retrieve data from object", right click the ini file object and select **"get string (group-item)".**

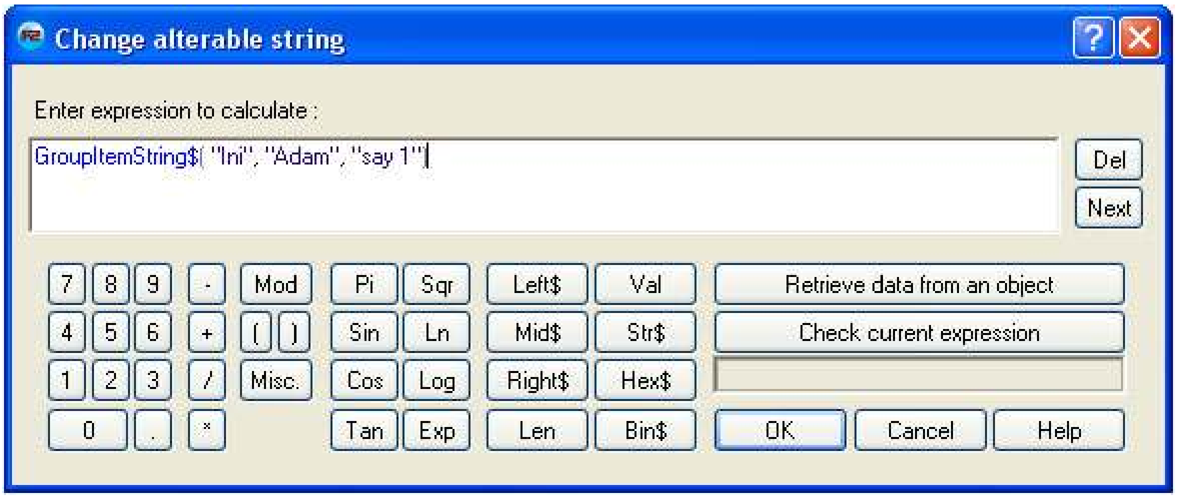
Make sure you select **"get string (group-item)"** and not **"get value (group-item)"** since we are using strings (text) and not values (numbers).



This will appear in the expression calculator:

GroupItemString$("Ini", >Group name<, >Item name<)

Change **>group name<** to **"Adam"** and change **>Item name<** to **"say 1"**. Remember to put them in quote marks.


Change alterable string
Enter expression to calculate :
Del
Next
7 8 9 - Mod Pi Sqr Left$ Val
4 5 6 + ( ) Sin Ln Mid$ Str$
1 2 3 / Misc. Cos Log Right$ Hex$
0 . * Tan Exp Len Bin$
Retrieve data from an object
Check current expression
OK
Cancel
Help

```
GroupItemString$( "Ini", "Adam", "say 1")
```

What we have done is that when the frame starts MMF locates our ini, looks inside it and finds the group **"Adam"**, then finds the item **"say 1"**, then loads the text stored there into our string object **"temp dialogue"**, and we copy that to the other string object **"dialogue"** one letter at a time.

Now run the frame and the dialogue should now read **"Hello, I'm Adam."**, or what ever you wrote in **"say 1"** of the ini file.

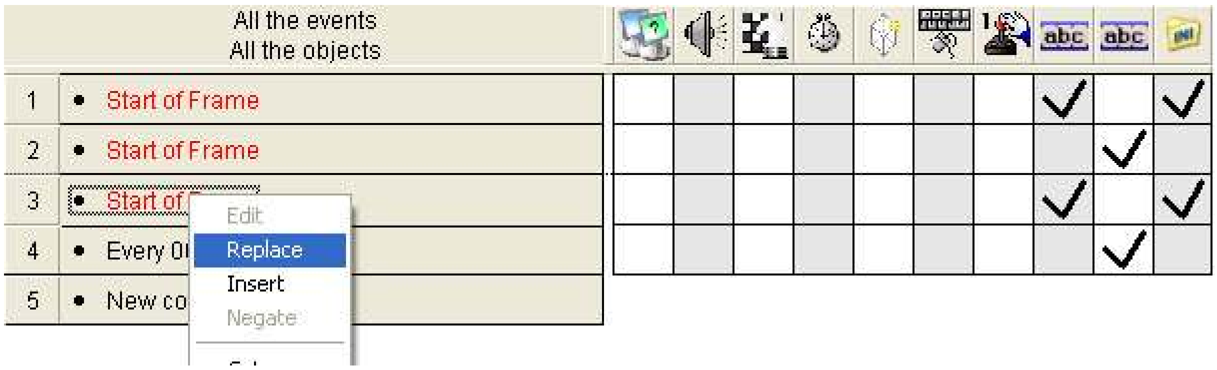
That's cool but we have more than 1 item in our group, we have **"say 1"**, **"say 2"**, **"say 3"** etc. Read on in part 3.

Part 3. Choosing dialogue at random

Okay then lets make it so that instead of just getting **"say 1"** all the time, when ever we press the **space bar** it will choose an item to say at random.

Using the same MMF file, copy and paste the last event we added. It should still have the **"temp dialogue"** and **"ini"** actions ticked from before.

Right click the **"start of frame"** condition and select **"replace"**.



Right click the **"mouse pointer and keyboard"** object from the **"new condition"** window that

appeared.
Select **"upon pressing a key"** and press the **space bar**.

Now right click the **"temp dialogue"** box (it should already have an action in it we made earlier), and click **"edit"**.

In the expression calculator change this:

GroupItemString$("Ini", "Adam", "say 1")

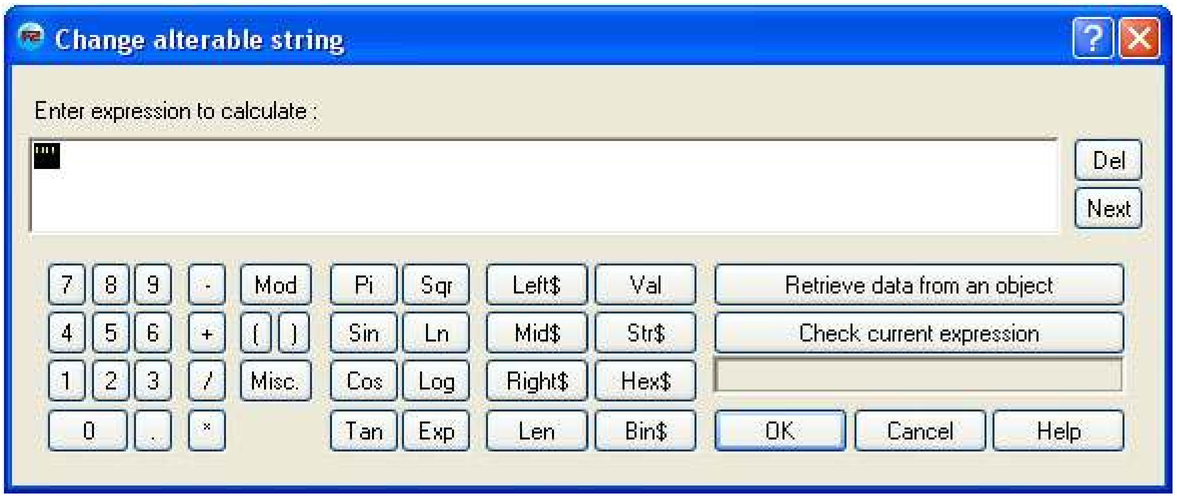
to this:

GroupItemString$("Ini", "Adam", "say "+Str$(Random(4)+1))

What we have done is replaced the **"1"** in **"say 1"** with **Random(4)** and we are adding 1 to what ever number is randomly picked because we don't want 0 to be picked. We have 4 items in our ini so we put random **(4)** (mmf could pick 0, 1, 2, or 3, but we are always adding 1).
the number is then converted into a string (Str$()) because we are making it part of **"say "**.

Before we can test it we need to make sure that our string object **"dialogue"** is cleared before any new text is copied over from **"temp dialogue"** otherwise our type effect wont work properly as explained at the end of part 1.

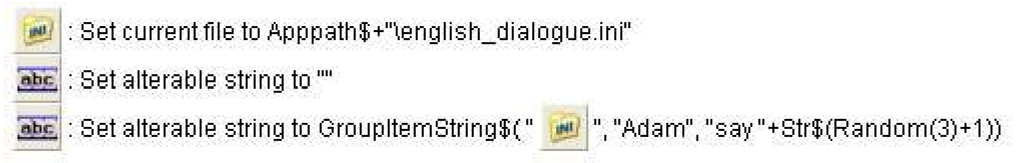
So, in the same event line, right click the **"dialogue"** action box and select **"change alt string"**.


Change alterable string
Enter expression to calculate :
""
Del
Next
7 8 9 - Mod Pi Sqr Left$ Val
4 5 6 + ( ) Sin Ln Mid$ Str$
1 2 3 / Misc. Cos Log Right$ Hex$
0 . * Tan Exp Len Bin$
Retrieve data from an object
Check current expression
OK Cancel Help

You don't need to write anything in the expression calculator because we want it to appear blank again, so just click ok.

We now need to make sure that all the action in that event are in the right order, we need to clear the **"dialogue"** string object before any new text is put in.

It should look like this:
(***Edit - in the image below it should say random(4) not 3, i made a mistake when I captured the picture, doh! But the order of actions is still correct***).



```
: Set current file to Apppath$+"\english_dialogue.ini"
: Set alterable string to ""
: Set alterable string to GroupItemString$( " ", "Adam", "say "+Str$(Random(3)+1))
```

Okay, let's test it. Run the frame and press the space bar. It should pick an item to say at random. Nice!
Save the MMF file.

Part 4. Adding another character.

Let's try adding another character.

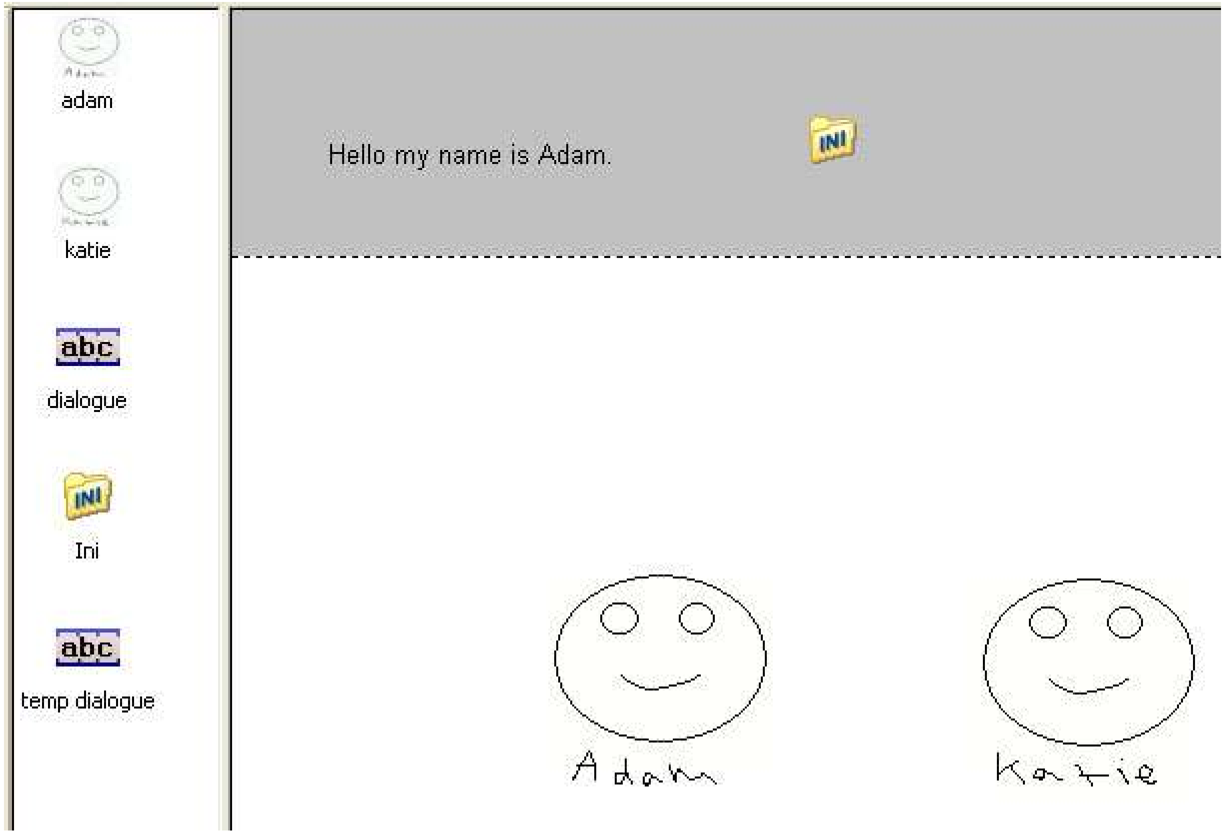
Open up your ini file in your text editor again and make a new group. Don't delete the **[Adam]** group.

[Katie]
say 1 = Hi, i'm katie!
say 2 = cows go moo!
say 3 = horses go naa!
say 4 = sheep go baa!
say 5 = Have a nice day.

Save the ini file again and open up our MMF file where we left off.

We are going to create a couple of **"Active"** objects which we will use to represent our 2 game characters, Adam and Katie. So add 2 active objects to the frame and rename them accordingly.

I've resized mine to make them bigger and made them look like faces, and written their names underneath.



Okay, what we are going to do is make it so that we can change between different character dialogue (groups in our ini file) depending on which character (active object) we clicked on.

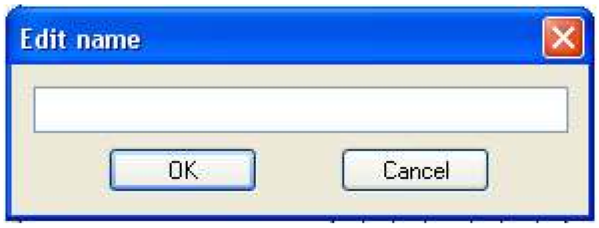
So if we want to hear what Adam has to say we can click on Adam, if we want to listen to Katie we can click on Katie. We can press space bar as before to get a random response from them.

The first thing we need to do is store each of their names in one of their **"Alterable Strings"**. This will be used to pick the [group] we want from our ini when we click on them.

So click on the Adam active object first, then in the **"properties"** panel, click on the **"Values"** tab.

Click on the **"New"** button. Make sure you are making a new **"Alterable String"** and not an **"Alterable Value"**.

Text saying "Alterable String A" will appear. Double click on it to rename it.



We'll call it **"character name",** since it will hold the name of our character.
Click ok, now double click in the empty space to the right of **"character name",** and type in **"Adam".**

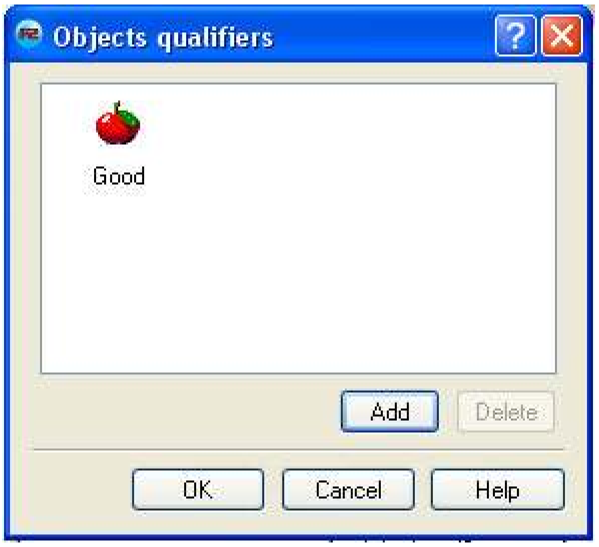
Make sure it is spelled the exact same way as you spelled it in the group of the ini file.

Now do the same thing with the "Katie" active object. Only type **"Katie"** instead of Adam.

Now we need to group these 2 actives together in a **"Qualifier"**. A qualifier allows you to control all the objects grouped in them through 1 set of events rather than having to repeat the events for each object.

Select both active objects, Adam and Katie, click on the **"events"** tab of the properties window. Click in the empty space to the right of the text "**qualifier(s)**" and click the button **"Edit".**

Click the **"add"** button and select a qualifier object. I chose **"Good"**, which looks like an apple, but it can be any.



Now that both objects belong to the qualifier **"Good"**, there's one last step we have to do before we can move on to editing our events.

Since String objects and ini objects don't have any alt values or alterable strings of their own, we still need a place to store what character name we have picked (what character active object we have clicked on).

We could store this in a **"Global"** alterable string, or create another string object, or even create a new group and write it to the ini file we've been using. But since I want to continue to build on what we have done in this part of the tutorial in the next part of the tutorial we will instead take a different approach.

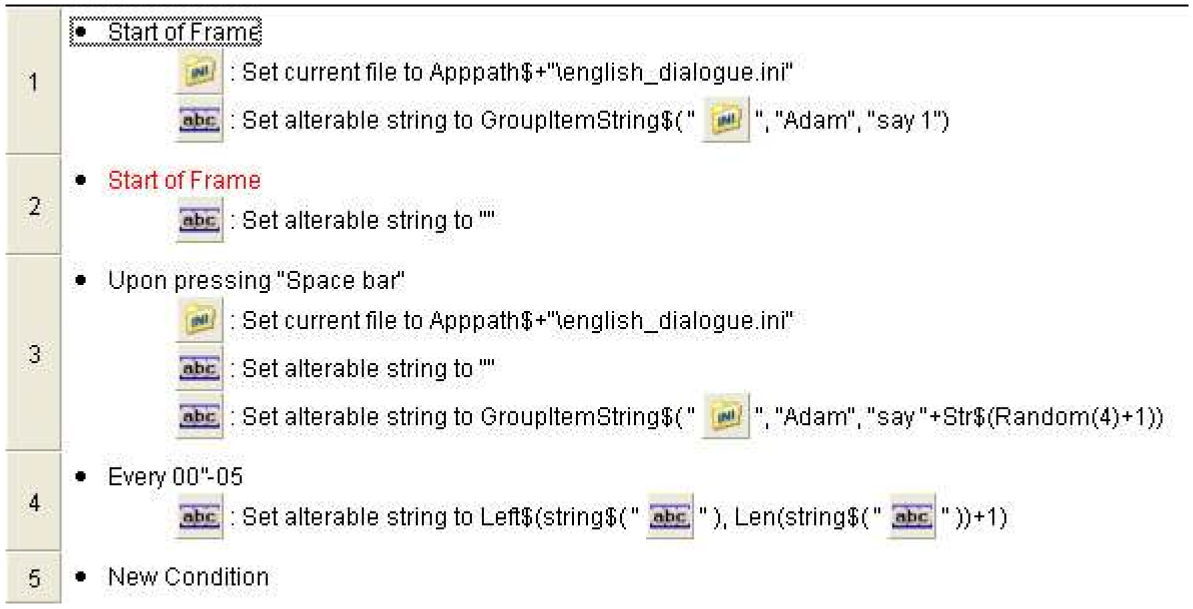
We will create a new active object with the one specific purpose of storing our alt string. In the next chapter we will also use it to store some other things too.

So create a new object on our frame, rename it **"local values"** and position it off the frame, because we don't need to see it when the game runs.

Like we did with the Adam and Katie active objects, in the object properties panel add a new alterable string to it. Call it **"selected group"**, and leave it blank.

What we are going to do is when we click on the Adam or Katie active object we will get their name that's stored in their alterable string **"character name"**, then set **"local values"** alterable string **"selected group"** to the same thing. This will then be used to select what group we want in side our ini file, [Adam] or [Katie].

We're now ready to make it happen with events.
This is what our events should look like at the moment:

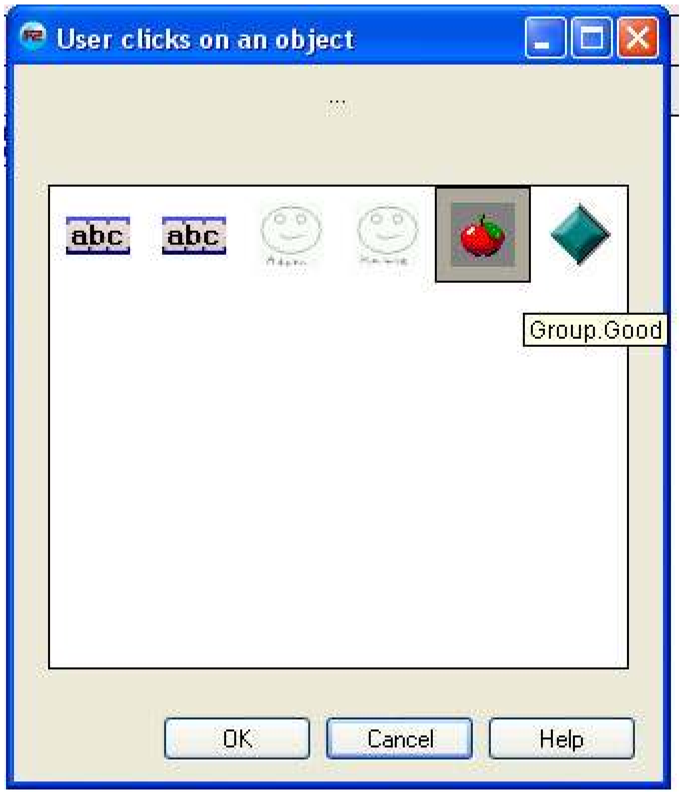


```
1  • Start of Frame
        : Set current file to Appath$+"\english_dialogue.ini"
        : Set alterable string to GroupItemString$(" ", "Adam", "say 1")
2  • Start of Frame
        : Set alterable string to ""
3  • Upon pressing "Space bar"
        : Set current file to Appath$+"\english_dialogue.ini"
        : Set alterable string to ""
        : Set alterable string to GroupItemString$(" ", "Adam", "say "+Str$(Random(4)+1))
4  • Every 00"-05
        : Set alterable string to Left$(string$(" "), Len(string$(" "))+1)
5  • New Condition
```

Insert a new event line in between 2 and 3, that's before our **upon pressing "space bar"** event.

Select the **"mouse pointer and keyboard"** object, select mouse, **"user clicks on an object"**, set it to **"left button"** and **"single click"**.

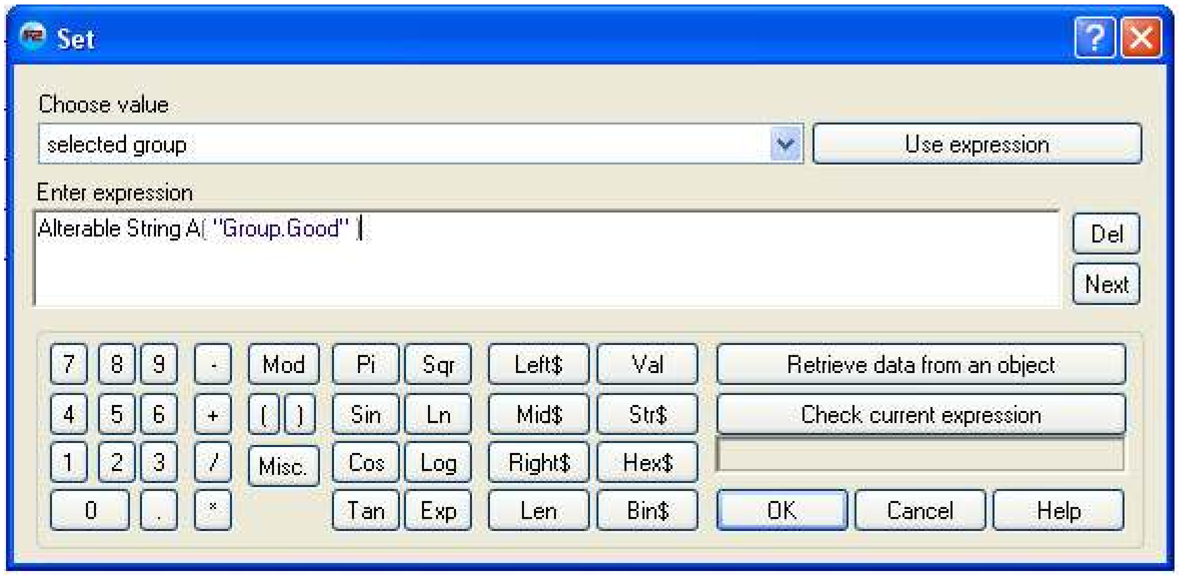
Rather than selecting the Adam or Katie active objects we will select our qualifier that we grouped them into earlier. I chose the **"Good"** qualifier which is an apple icon.



Let's add some actions to that event. Remember we want it so that when we click on Adam or Katie, **"local value"** active's alterable string **"selected group"** is set to Adam or Katie's alterable string value **"character name"**, which will either be Adam or Katie depending on who we click on.

So right click on **"local values"** action box, choose **"alterable string"** and **"set"**. Make sure **"selected group"** is our value. Now we'll enter our expression in the box below.

In the expression calculator click **"retrieve data from an object"**, right click **"good.group"** (apple icon), and select **"values"**, **"alterable strings"** and **"retrieve alterable string A"**. Remember that alterable string A is actually the alterable string we renamed **"character value"**, but there's a bug in MMF that doesn't show what an alterable value has been renamed to if you select it from within a qualifier. But don't worry about it.


Set
Choose value
Use expression
Enter expression
Del
Next
7 8 9 - Mod Pi Sqr Left$ Val
4 5 6 + ( ) Sin Ln Mid$ Str$
1 2 3 / Misc. Cos Log Right$ Hex$
0 . * Tan Exp Len Bin$
Retrieve data from an object
Check current expression
OK
Cancel
Help

selected group

Alterable String A("Group.Good")

Now we need to change the **"temp dialogue"** action in the event line **"upon pressing space bar"**

So right click the box and select "edit".

We need to change the expression that we wrote in before:

GroupItemString$("Ini", "Adam", "say "+Str$(Random(4)+1))

so that it doesn't select "Adam" all the time, but selects what ever is in **"local values"** alterable string **"selected group"**, instead.

So delete **"Adam"** from the expression and click **"retrieve data from an object"**. right click **"local values"** active object icon, **"values"**, **"alterable strings"**, **"retrieve selected group"** (which is what we named "alterable string A".

The new expression should now read:

GroupItemString$("Ini", selected group("local values"), "say "+Str$(Random(4)+1))

We can now run the game and test it out.
Adams dialogue should appear on screen straight away. Click on either Adam or Katie then press **space bar** to get a random response. Nice!

You might of noticed a problem however. When the game first starts, if you just press the space bar before clicking Adam or Katie then no text will appear. That's because at the start of the frame **"local values"** alterable string **"selected group"** (which is what is used to determine which group is picked in the ini file) is blank.

We can easily solve the problem by either adding **"Adam"** to **"selected group"** in the properties panel, or we can do it through events. In the 1st event line with **"Start of frame"**, add the action to **"local values"**, set alterable string **"selected group"** to **"Adam"**.

Okay we are done with this part. Save your work, we will continue in part 5.

<u>Part 5. Random speech with more than 1 character.</u>

You may have noticed that while the Adam group of our ini file has 4 items (**"Say 1"** to **"say 4"**), the Katie group has 5 items. Maybe because girls always have something to say? haha.

At the moment if you run the game and click Katie and press space bar, her item 5 **"Have a nice day"** will never be chosen. This is because we are choosing the group items based on a random value of 4 in the expression editor simply because at the time Adam had no friends and 4 was the amount of items he had in the ini file.

```
GroupItemString$( "Ini", "Adam", "say "+Str$(Random(4)+1))
```

You might have lots of characters in your game and some might have 1 line, some might like giving speeches and have 20 lines etc.

What we need is a way to tell how many items each group has. Now there's lots of ways this can be done.
We've already solved a similar problem when we choose our group name by having the characters name stored in their alterable string "character name", and when it was selected copied into "local values" "Selected group" alterable string:

```
GroupItemString$( "Ini", selected group( "local values" ), "say "+Str$(Random(4)+1))
```

We could do a similar thing now for the items and store how many items each game character has in 1 of their alterable values and put something like this:

```
GroupItemString$( "Ini", selected group( "local values" ),"say "+Str$(Random(number of lines( "local values" ))+1)
```

But there's another way i wanted to show you, which will pave the way for the next chapter.

You don't want to use up all a characters alterable values on keeping track of how many lines of dialogue they have to say. At the moment there is only 1 Adam group but what if Adam has lots to say through out the course of the game on multiple levels and at multiple encounters?

Our ini could look like this:

```
[Adam encounter 1]
say 1=blah blah blah.
say 2=blah blah blah.

[Adam encounter 2]
say 1=blah blah blah.

[Adam encounter 3]
say 1=blah blah blah.
say 2=blah blah blah.
say 3=blah blah blah.
```
In which case we would need to use 3 alterable values to keep track of many items in each group.

So forget that. The way we will solve this problem is to have each group have an extra item, not for dialogue but for storing how many items is in that group.

Open the ini file again and add this item to each group.

"number of lines="

Then **4** for Adam or **5** for Katie.

It should look like this:

[Adam]
say 1 = Hello, I'm Adam.
say 2 = sentence 2
say 3 = sentence 3
say 4 = sentence 4
number of items = 4

[Katie]
say 1 = Hi, i'm katie!
say 2 = cows go moo!
say 3 = horses go naa!
say 4 = sheep go baa!
say 5 = Have a nice day.
number of items = 5

Remember to save it.

We don't need to worry about Adam or Katie actually saying "4" or "5" because remember only items starting with "say" will be chosen.

Okay, back in our MMF file, find the event line **"upon pressing space bar"** and we want to edit the **"temp dialogue"** action. Right click the box and select **"edit"**.

Now in the expression calculator we already have:

GroupItemString$("Ini", selected group("local values"), "say "+Str$(Random(4)+1))

delete this part of the expression:

Str$(Random(4)+1)

click the **"retrieve data from object"** button. Right click the ini object icon, and select **"get string (group -item)"**.

The expression should now look like this:

GroupItemString$("Ini", selected group("local values"), "say "+GroupItemString$("Ini", >Group name<, >Item name<))

Delete "**>Group name<**" and type in "selected group("local values")".

Delete "**>Item name<**" and type in **"number of items"** exactly as you wrote it in the ini file, don't forget the quotation marks because its a string.

The expression should now look like this:

GroupItemString$("Ini", selected group("local values"), "say "+GroupItemString$("Ini", selected group("local values"), "number of item"))

This next bit is a bit tricky to explain and is probably the hardest part of this tutorial to get your head around.

If we run the game again now and click the **"Adam"** and press the **space bar** it will get the number 4 that we have stored in the item **"number of items"** in our ini file, so it will always select item "say 4" from the group Adam. Selecting Katie will always give us item "say 5". That's what our expression is currently saying. To make it pick a random item again we need to put the **random()** function back into the expression. This function only works with values and our **"number of items"** is a string so we need to first covert

GroupItemString$("Ini", selected group("local values"), "number of item")

to a value using the **Val()** function. So add **Val(** open bracket and close the bracket at the end of the expression.

It should look like this:

GroupItemString$("Ini", selected group("local values"), "say "+(Val(GroupItemString$("Ini", selected group("local values"), "number of items")))

Now we can add the random function just before where we put Val. Close the bracket at the end of the expression again.

Our expression should now look like this:

GroupItemString$("Ini", selected group("local values"), "say "+Random(Val(GroupItemString$("Ini" , selected group("local values"), "number of items"))))

But because we are originally working with strings (we want to get what number **"say"** item to pick), we now need to convert all that back into strings again using, you guessed it, the **Str$()** function. So add that just before our Random function, close the bracket at the end of the expression.

Our expression should now read:

GroupItemString$("Ini", selected group("local values"), "say "+Str$(Random(Val(GroupItemString$ ("Ini", selected group("local values"), "number of items")))))

What a mouth full!

One last thing to do. Remember that when MMF uses the Random function whatever number you choose starts at 0, so Random(4) would be 0,1,2 or 3. Our "say" items start at 1 not 0. We could edit the items in the ini file or we could just add + 1 to the random function so it can never pick 0. Let's do that now.

3 brackets from the end of the expression (remember the string function wraps the random function which wraps the val function which wraps our expression), type in +1.

Our final expression should now read:

```
GroupItemString$( "Ini", selected group( "local values" ), "say "+Str$(Random(Val(GroupItemString$
( "Ini", selected group( "local values" ), "number of items")))+1))
```

hallelujah!

Run the frame and play about with it.
You can now add more items to the ini groups, just make sure that the item **"number of items"** always matches how many items you have.

Let's save our work for the next chapter.

Part 6. Sequential and random dialogue.

So we now have multiple characters each with their own dialogue, and each able to have their own number of lines. That's great but what if we want to make it so that our characters don't just say lines randomly?

Some times a character might be telling us what our mission is in the game, or our next objective for example, so we need a way so that when the space bar is pressed it says each **"say"** item in turn, one after the other.

"say 1" press space bar "say 2" press space bar "say 3" etc.

Since we might still want some characters in our game to say things randomly, the first thing we need to do is find a way to identify which [groups] should have their items read randomly and which [groups] should have their items read sequentially.

We'll keep Katie saying things at random, but we'll change Adam so that he says his items in sequence.

So, open your ini file again and add the following new item to each group:

talk=

For the Adam group put: **sequentially**

For the Katie group put: **randomly**

Be aware that MMF has protected words which you can't use. The word "random" is protected because of the random() function so we will use the word "randomly" instead.

Our ini file should now look like this:

```
[Adam]
say 1 = Hello, I'm Adam.
say 2 = sentence 2
say 3 = sentence 3
say 4 = sentence 4
number of items=4
```

talk = sequentially

[Katie]
say 1 = Hi, i'm katie!
say 2 = cows go moo!
say 3 = horses go naa!
say 4 = sheep go baa!
say 5 = Have a nice day.
number of items=5
talk = randomly

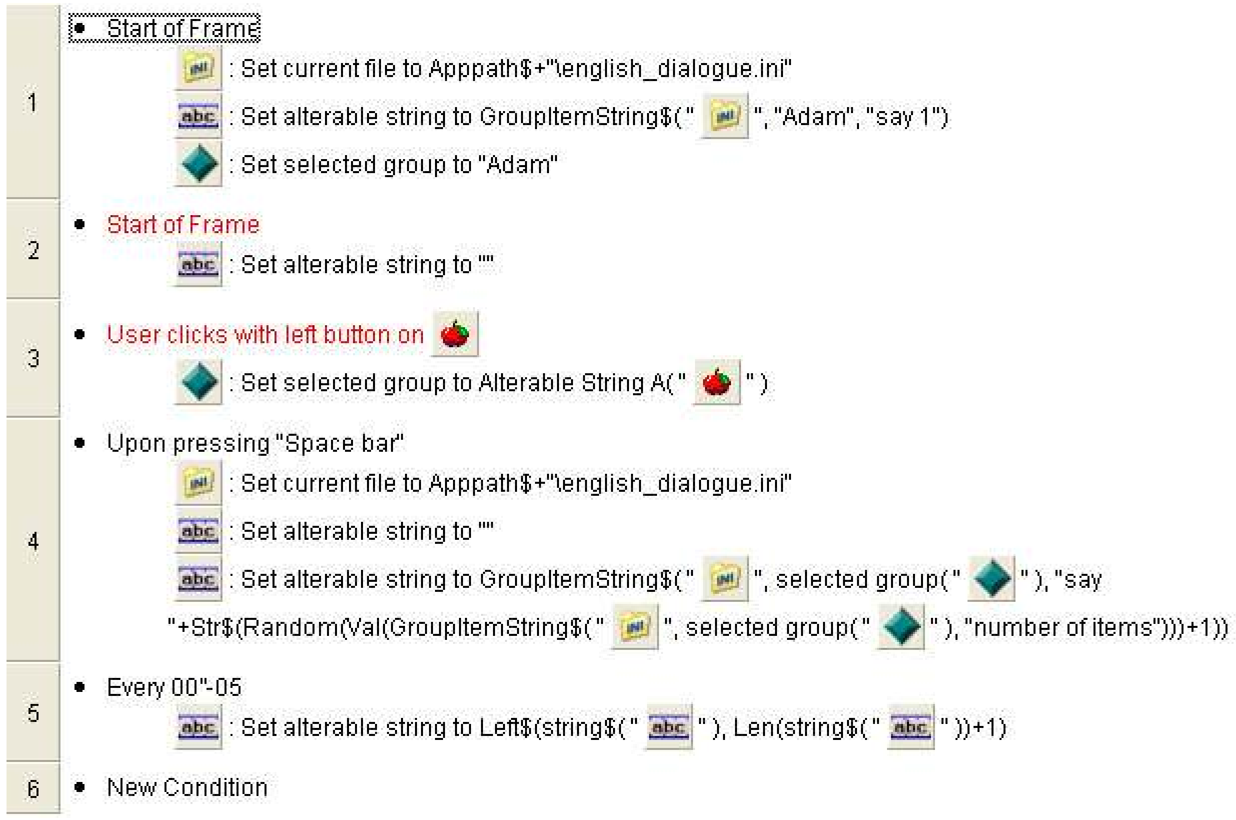
Let's get back to the MMF file we've been working with.

At the moment MMF isnt concerned with our new items. It still thinks both characters what to speak randomly.

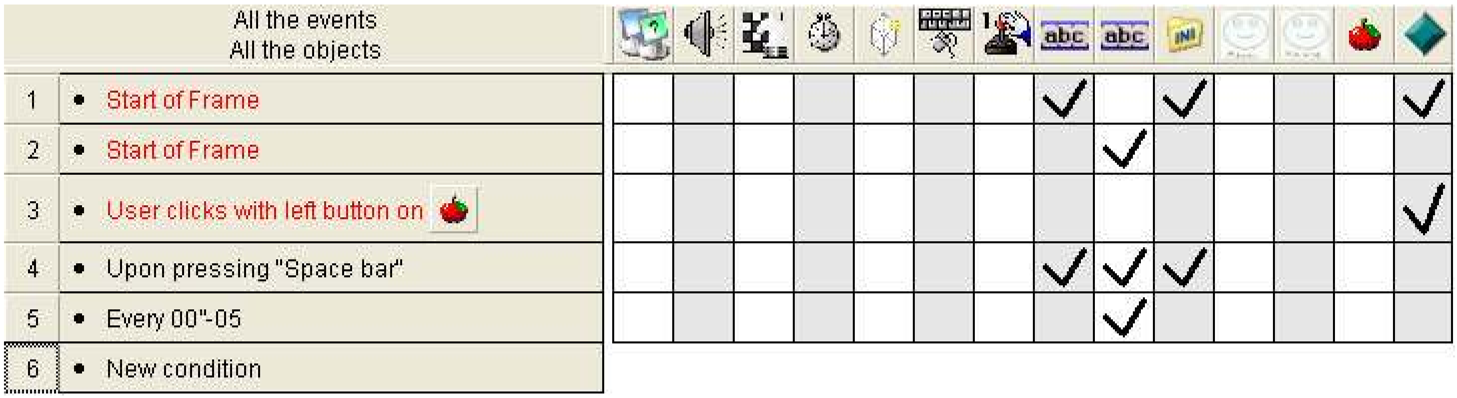
We need our game to check how a selected character, Adam or Katie, want to **"talk"**, which is where our new **"talk"** item comes in.

To do this we need to change the way in which our game works slightly.

So far our events should look like this, from the last part of the tutorial:



```
1  • Start of Frame
           : Set current file to Apppath$+"\english_dialogue.ini"
           : Set alterable string to GroupItemString$(" ", "Adam", "say 1")
           : Set selected group to "Adam"
2  • Start of Frame
           : Set alterable string to ""
3  • User clicks with left button on
           : Set selected group to Alterable String A(" ")
4  • Upon pressing "Space bar"
           : Set current file to Apppath$+"\english_dialogue.ini"
           : Set alterable string to ""
           : Set alterable string to GroupItemString$(" ", selected group(" "), "say
          "+Str$(Random(Val(GroupItemString$(" ", selected group(" "), "number of items"))+1))
5  • Every 00"-05
           : Set alterable string to Left$(string$(" "), Len(string$(" "))+1)
6  • New Condition
```



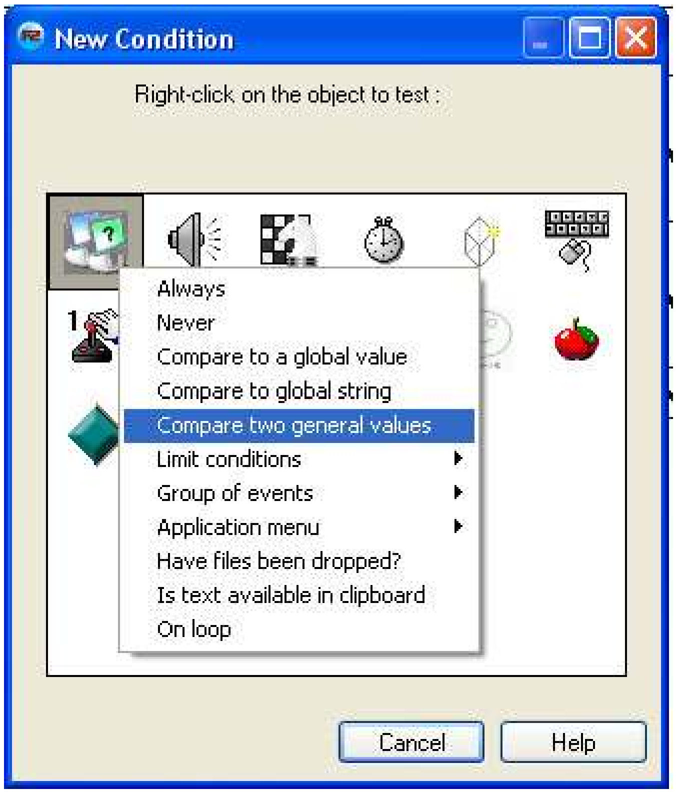
All the events All the objects															
1	• Start of Frame								✓		✓				✓
2	• Start of Frame									✓					
3	• User clicks with left button on														✓
4	• Upon pressing "Space bar"								✓	✓	✓				
5	• Every 00"-05									✓					
6	• New condition														

Right now we have just 1 event for when the **space bar** is pressed, we'll add to this a condition that compares what's written in the ini item **"talk"** with the word **"randomly"**. If they match then the actions of that event will be carried out, if they don't nothing will happen.

Let's do that now then.

Right click on the condition **"upon pressing space bar"** on line **4**, and select **"insert"**.

Right click the **"special"** object icon and select **"compare two general values"**.



In the expression calculator we want to compare what's in our ini item **"talk"** to the word **"randomly"**.

So click in the top box and click the **"retrieve data from an object"** button. Right click the ini icon, and select **"get string (group-item)".**

This expression will appear:

GroupItemString$("Ini", >Group name<, >Item name<)
Okay, highlight **>Group name<** and click the button **"retrieve data from an object"** button again.

Right click the active object **"local values"**, select **"values"**, **"alterable strings"** and then **"retrieve selected group"**.

Remember the alterable string **"selected group"** that we have in the active object **"local values"** is where we store which character we've clicked on, their name, either Adam or Katie.

Now our expression should read:

GroupItemString$("Ini", selected group("local values"), >Item name<)

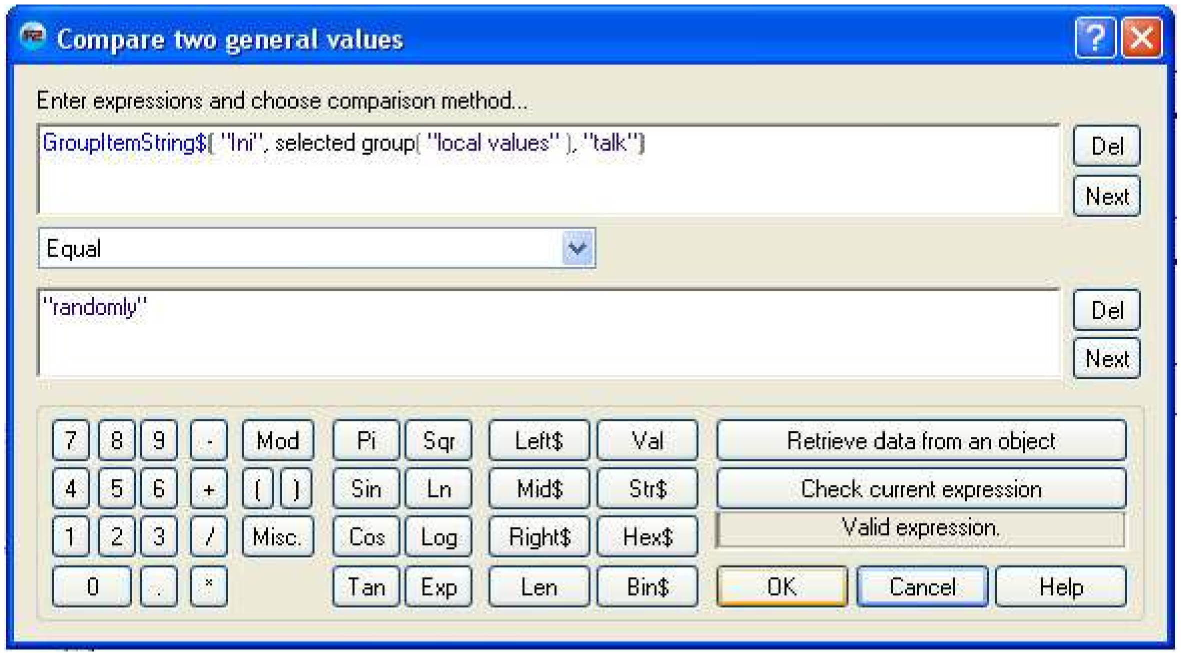
Now change the **>item name<** to **"talk"**, just as you spelled it in the ini file.

In the bottom box of the expression calculator type in:

"randomly"

exactly as you spelled it in the ini file. Remember to include the quote marks because its a string.

Make sure the comparison in between the 2 box's is set to **"equal"**.


Compare two general values

Enter expressions and choose comparison method...

GroupItemString$("Ini", selected group("local values"), "talk")

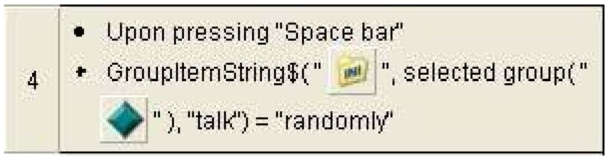
Equal

"randomly"

Valid expression.

Ok now our condition for that event line should read:

"upon pressing space bar" and "GroupItemString$("Ini", selected group("local values"), "talk") = "randomly"



```
4  • Upon pressing "Space bar"
   • GroupItemString$( "   ", selected group( "
        " ), "talk") = "randomly"
```

Now if we run the game you will find that only katie can speak because only her item "talk" says "randomly".

So now we need an event for those groups who's **"talk"** items say **"sequentially"**, like Adam. **Copy** event line 4, actions and all, and paste it in between line 4 and 5.

Now we have 2 identical events. Right click the condition **"GroupItemString$("Ini", selected group("local values"), "talk") = "randomly"** in line 5 that we just pasted, and edit the expression so in the bottom box of the comparison it now reads **"sequentially"** rather than "randomly". Make sure it is spelled like it is in the ini file.

Okay now what's happening is that every time the space bar is pressed and our selected group item "talk" reads **"randomly"** it will carry out those actions in that event. If however when the space bar is pressed and it instead reads **"sequentially"** it will carry out those actions instead.

But at the moment both events actions are the same because we just copied and pasted the whole event and have so far only changed a condition. Our "sequentially" event on line 5 still has actions that cause items to be selected at random. We need to change this.

But there's a problem. Because we want to select the **"say"** items from 1 to how many there are in the group, in order, we need a way to keep track of what "say" item we selected last so we can select the next one.

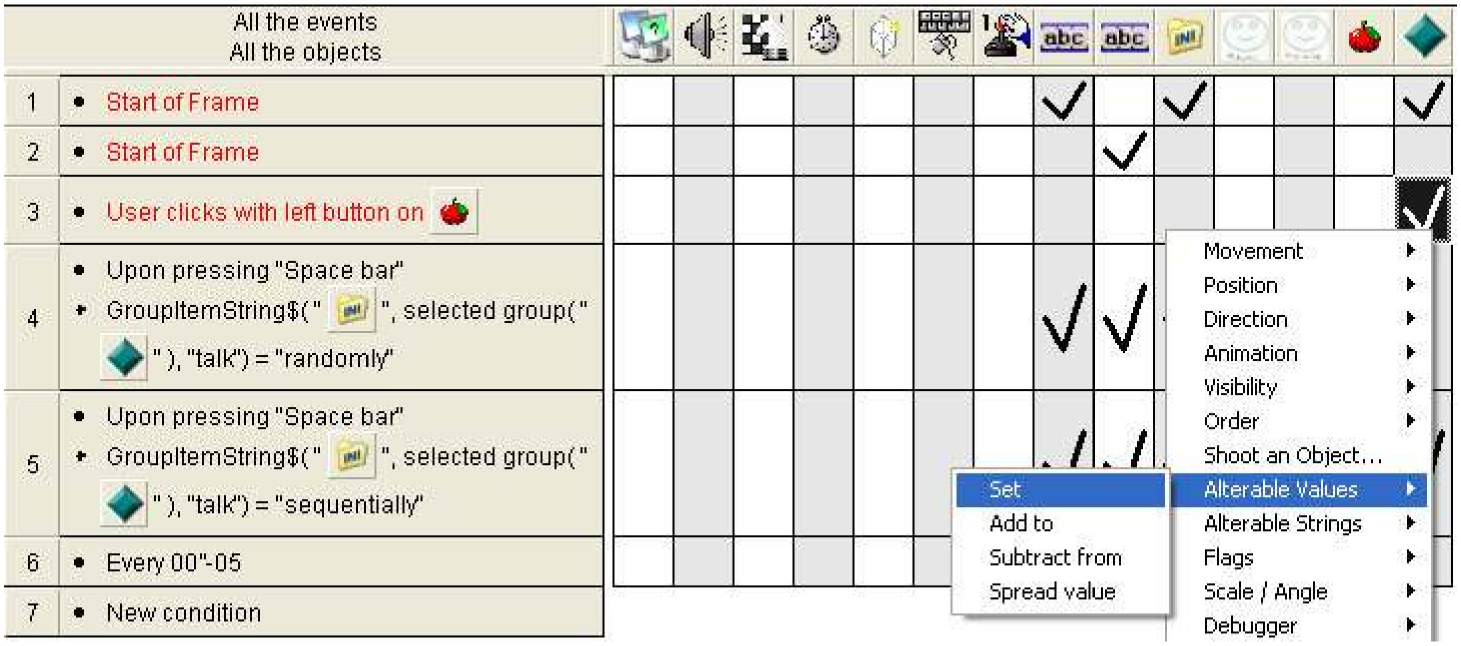
Again there are lots of different way we could accomplish this.

What we will do though is use the active object **"local values"** again, and we will store what item to pick next in an alterable value. Remember we are already using this active object to store what group we want in one of its alterable strings, that we call **"selected group"**.

So now then, in the frame editor select the **"local values"** object and in the properties window, under the **"values"** tab, create a new **"alterable value"**. Rename it to **"selected item"**. We need to use an alterable value in this case and not an alterable string because we will be performing some very basic math's with what's stored in side, so it needs to be a value (number) and not a string (text).

We are going to have to do some minor editing of our events now and change a few actions to make our improved system work correctly.

Back in the event editor, find the event with the condition **"user clicks with left button on group.apple",** or whatever qualifier you chose. Right click the **"local values"** object action box, select **"alterable values"** and **"set"**.

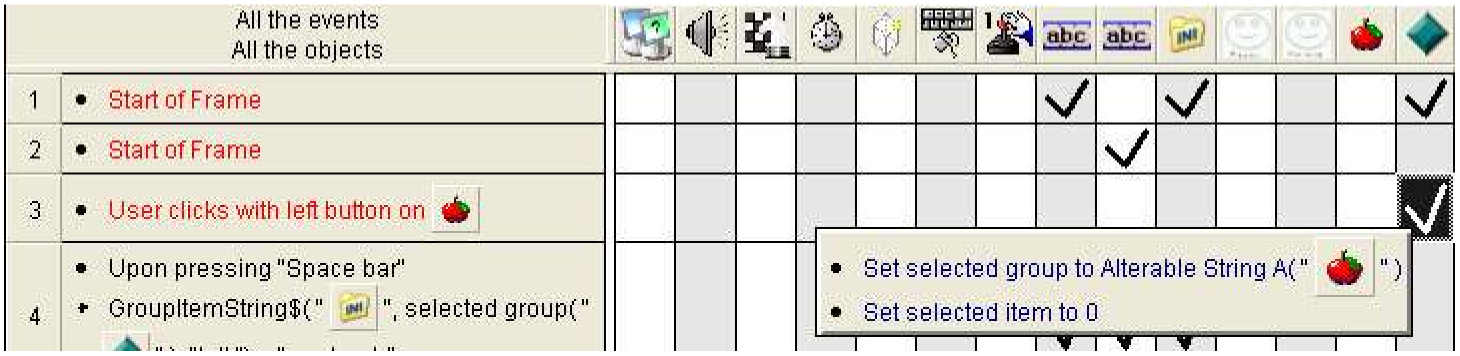


All the events
All the objects

1 • Start of Frame
2 • Start of Frame
3 • User clicks with left button on
4 • Upon pressing "Space bar"
• GroupItemString$(" ", selected group(" "), "talk") = "randomly"
5 • Upon pressing "Space bar"
• GroupItemString$(" ", selected group(" "), "talk") = "sequentially"
6 • Every 00"-05
7 • New condition

Now in the expression editor, make sure the alterable value **"selected item"** (the one we just created) is selected in the top menu box, then in the box below enter 0.

The action should now look like this, which also has the other action we entered earlier.



All the events
All the objects

1 • Start of Frame
2 • Start of Frame
3 • User clicks with left button on
4 • Upon pressing "Space bar"
• GroupItemString$(" ", selected group("

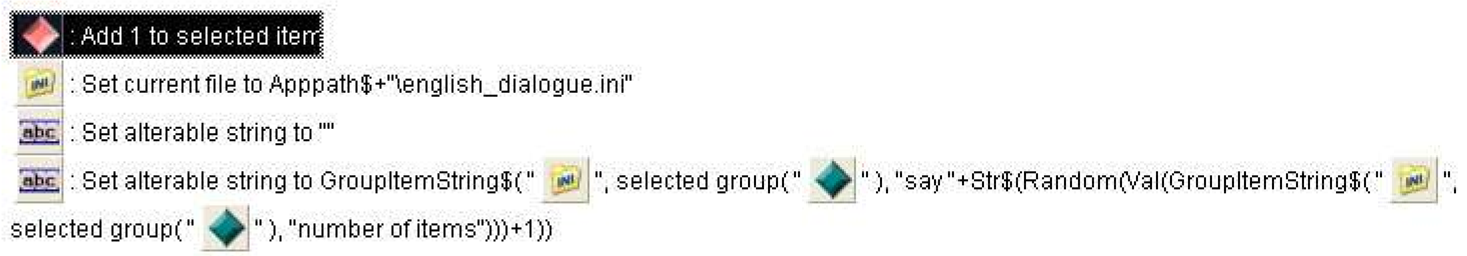
• Set selected group to Alterable String A(" ")
• Set selected item to 0

Now what will happen during the game is that every time we click on a character the value stored in **"selected item"** (which we will eventually use to select our ini items with) will be set back to 0.

Okay so now the value will reset every time that character is selected. What we also need now is to make that value increase by 1 every time the character says a new bit of dialogue. Since we already make our characters pick a new item to say every time the space bar is pressed, we will also use this condition to add 1 to our **"selected item"** alterable value.

So now, on event line 5, which has the conditions **"upon pressing space bar"** and the **"sequentially"** comparison (because the other condition we want to keep random, remember we are only wanting groups in the ini file that have **"talk"** set to **"sequentially"** to say their **"say"** items in order), Right click the **"local values"** object action box again, and again select **"alterable values"**, only this time instead of choosing **"set",** we want to select **"Add to"**. Change the 0 to a 1.

Now in the "event list editor" we need to make sure that action is at the top of the list, because want to add 1 to the value before we select our item. So rearrange the order of the actions.



: Add 1 to selected item
: Set current file to Apppath$+"\english_dialogue.ini"
: Set alterable string to ""
: Set alterable string to GroupItemString$(" ", selected group(" "), "say "+Str$(Random(Val(GroupItemString$(" ",
selected group(" "), "number of items"))+1))

Working with he same event, it's time to remove the random from the actions.

Right click **"temp dialogue"** action box and click **"edit"**.

In the expression calculator which currently looks like this:

GroupItemString$("Ini", selected group("local values"), "say "+Str$(Random(Val(GroupItemString$
("Ini", selected group("local values"), "number of items")))+1))

delete the bit after "say" +, that's this bit:

Str$(Random(Val(GroupItemString$("Ini", selected group("local values"), "number of items")))+1))

so we are left with:

GroupItemString$("Ini", selected group("local values"), "say "+

After the **+** sign click the **"retrieve data from an object"** button, right click the **"local values"** object icon and select **"values"** then **"values A-M"** and **"retrieve selected item"**.

The expression should now look like:

GroupItemString$("Ini", selected group("local values"), "say "+ selected item("local values")

Because our alterable value is a value and we are working with strings again we need to convert **"selected item"** in to a string with the string function. Add it just after the **+** sign with its brackets wrapping **"selected item("local values").**

Like this:

GroupItemString$("Ini", selected group("local values"), "say "+ Str$**(**selected item("local values")**)**

Good. Now we just need to add 1 extra bracket to the end because we are wrapped inside the Groupitemstring$() already.

Click ok. Now lets run our game and try it out!

Click on Adam and press space and watch him say his dialogue from sentence 1 and up, in order.

Click on Katie and press space and watch her continue to say her dialogue randomly.

We're almost there, but you have probably noticed a couple of problems.

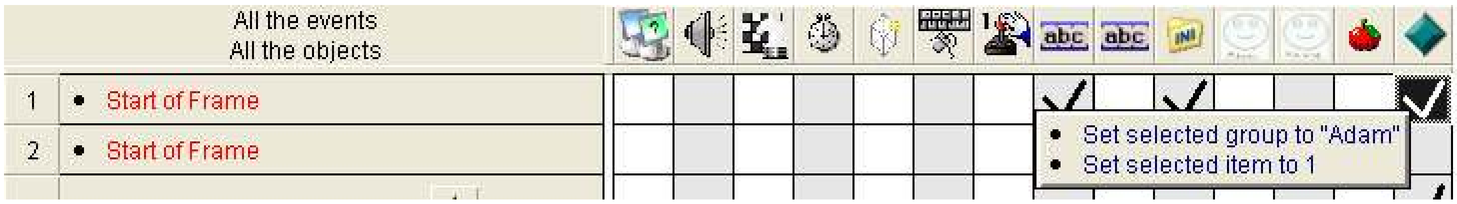
First when you select Adam he says the first dialogue **"Hello, I'm Adam"** twice in a row before saying "sentence 2", "sentence 3" etc.

The second thing that's wrong is that after Adam runs out of dialogue he says nothing.

These are both minor problems which we will now fix.

The first problem of Adam repeating himself is caused by the event on line 1 with the condition **"start of frame"**. It's actions mean that at the start of the frame, we find the location of our ini file. Set the **"temp dialogue"** string to ini group Adam and item **"say 1"**, but what we haven't done is set our **"selected item"** value. This means by default it starts at **0** so when Adam already says his first dialogue line, and space is pressed for the next line which is now set to **1**, it is get the item **"say " +** what ever value is in **"selected item"**, which is **1** again.

So all we have to do to fix the problem is either stop Adam from speaking when the game first starts, or as we will now do, set the **"selected item"** value to **1** when the frame starts. Easy!
On line 1 which has the condition **"start of frame"**, add the action to the **"local values"** object **"Set selected item to 1"**.



- Set selected group to "Adam"
- Set selected item to 1

The second problem of Adam running out of dialogue is caused when the value **"selected item"**, which increases every time we press space bar, becomes greater than the number of items in the Adam group.

What we need to do is add a limit so that the value **"selected item"** can not exceed the number of items in the selected group. This is easy to do because of the item we already added in an earlier part of the tutorial. The **"number of items"** item.

Find the event with the conditions **"upon pressing space bar"** and the comparison to = **"sequentially"**, it should be line 5. Right click on the condition and click "insert".

Right click the **"local values"** icon and select **"alterable values"** and **"compare to one of the alterable values"**.

Choose value **"Selected item"**.

Choose comparison method **"Lower"**.

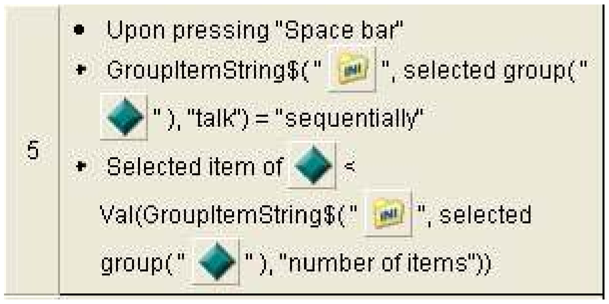
Enter the expression box and click the **"retrieve data from object"** button. Right click the ini object and **"get string(group -item)"**. Now change **>group name<** to the alterable string **"group name"** in the **"local values"** object. Change the **>item name<** to **"number of items"**. Spell it like it is in the ini file, don't forget the quote marks because its a string.

Because we are comparing a value to a string, we need to wrap that expression in the value function.

It should read:

Val(GroupItemString$("Ini", selected group("local values"), "number of items"))

The condition in that event should now look like this:



```
5  • Upon pressing "Space bar"
   • GroupItemString$(" ", selected group("
     " ), "talk") = "sequentially"
   • Selected item of  <
     Val(GroupItemString$(" ", selected
     group(" " ), "number of items"))
```

Now only if **"selected item"** is lower than the item **"number of items"** stored in our ini, will it be increased by 1 when the space bar is pressed.

Run the game and test it out.

Try changing Katies "talk" item from "randomly" to "sequential".

Try adding more "say" items Adam and katie, just remember to make their "number of items" item reflect how many "say" items they have.

Okay, thats about it.

Final words

Whilst you can't just take what we've done in this tutorial and stick it straight into a game, I do hope that you instead have an idea and understanding of the principles and concepts of how this kind of dialogue system works, and that it inspires you to create your own.

It really is quite a good way to do dialogue since once you have the initial eventing in MMF done, it's a really easy process to add new dialogue or change dialogue that a game character can say.

Unfortunately one other thing i didn't get around to covering was **multiple groups** for each character, and have them choose which one to use depending on specific conditions. For example, imagine in a game you go up to a character and he speaks to you. His dialogue might come from a group which looks something like this:

```
[character 1]
say 1 = Hello i'm Bob.
say 2 = I've lost my keys, please help me find them and i'll give you a reward.
number of lines = 2
talk = sequentially
```

So you go off and find his keys and bring it back to him, at which point the character would access another group

```
[same character 2]
say 1 = Hey it's you again.
say 2 = Have you found my keys?
```

```
number of lines = 2
talk = sequentially
```

At which point if you had he would access another group

```
[same character 3]
say 1 = Hey you found them!
say 2 = Heres your reward.
number of lines = 2
talk = sequentially
```

or if you had not yet found the keys he would access another group, and give you a random response:

```
[same character 4]
say 1 = Found my keys yet?
say 2 = I really need my keys?
say 3 = You can't have your reward until you find my keys!
number of lines = 3
talk = randomly
```

You get the idea. But i think with what we have done in this tutorial it should not be too difficult for you to figure it out.

Hope you have fun!

Thanks, Bye!

Adam.