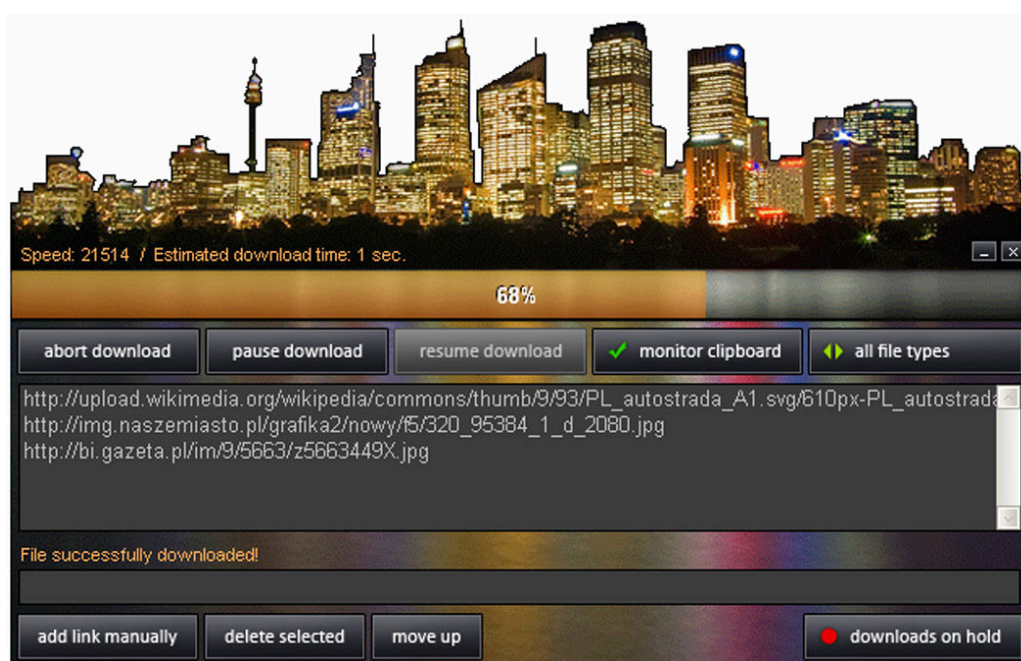


BiGciTY DOWNLOADER



You may not use this tutorial for any other purpose than learning, working or having fun... In other words: You can use this PDF tutorial for anything you'd like, as long as it doesn't involve both a hammer and a squirrel.

Welcome to another one of **Koobare's little tutorials**, teaching you – as always – how to effectively and efficiently use the best multimedia authoring tool ever – [Multimedia Fusion 2](#) by Clickteam! This tutorial is meant for intermediates, people who already spent some time with MMF2 and experimented with various types of games and applications. Don't jump into this one if you've just started your journey through the fascinating world of Fusion – you can hurt your fingers and break your toes. If you need a bit of advice on where to start, take a look at this simple lesson guide (note that this is just a suggestion, only a slight hint on where to go):

MMF2 Interface: Interface Guide + Image Editor Guide	<i>First time with MMF2</i>
Basics: Smelly Claw tutorial	<i>Beginners</i>
Game tutorials: Glob Wars, Risky Waters, You've Got Spacemail!	<i>Beginners</i>
Game tutorials: Mystery of Paris, Castle Defender, Space Corsair	<i>Beginner-intermediates</i>
App tutorials: Byrd: Free Text Editor	<i>Intermediates</i>
You are here → BiGciTY Downloader	<i>Intermediates</i>
Next: Fusion Player	<i>Intermediates</i>

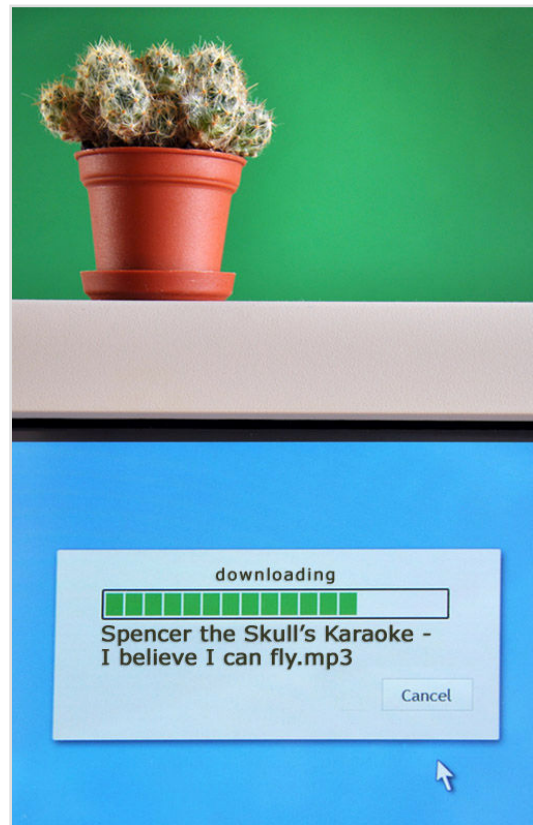
In this tutorial we'll create a simple, yet awfully useful, **file downloader**, a nifty little application which will enable us to easily create download lists (either by adding file links manually, or by monitoring the clipboard for specific file types) and then to effortlessly get all the listed files to our hard drives. By doing this, we'll learn how to use the Download, String Parser and Window Control objects, as well as the built-in MMF2 clipboard expressions and mechanisms. This will be quite an educative ride, so be sure to pay attention.

Download in progress...

As you may have heard, the internet is quite a spacious thing – and, as it seems, it's full of files, documents of different types, shapes and sizes, just like the ocean is full of fish, jellies, and singing mammals. And whether it's a funny picture of a squirrel just chilling in the sun, a text file full of nerdy jokes your cousin sent ya', or a zipped archive containing the latest version of your favorite freeware game – I'm pretty sure you'd like to download it to your hard drive, just to take a look at it later on, when you'll have a bit more time, when you're not at school/work/not writing a term paper or whatever...



Downloading files is usually as simple as it gets – just click on a link and start downloading with your favorite website browser (no matter whether its Opera, Firefox, IE, Maxthon or some other gizmo you're sporting), we've all done it before, gabrazillions of times. But what if... Bear with me here, what if you'd like to create a nice list of files that you want to download later on, because doing that right now would interrupt your plans to conquer the world? Well then, you can always use a downloader, a smart little program that enables you to create lists of links that you can download at the time of your choosing. There are many free ones out there, but today we're gonna' create one more: BiGciTY Downloader, a small little app that will not only teach you a lot but also can prove to be real handy in everyday work.



You know what we're aiming at, you know where we want to get at the end, now it's time to show you how to get there. Let's not waste anymore time here! Move on!

- 📄 **If you have any problems with this tutorial, or notice that there are some mistakes present, please, contact me and I'll do my best to help you and replace all the errors with correct information.**

Contact me at: marchewkowy@gmail.com

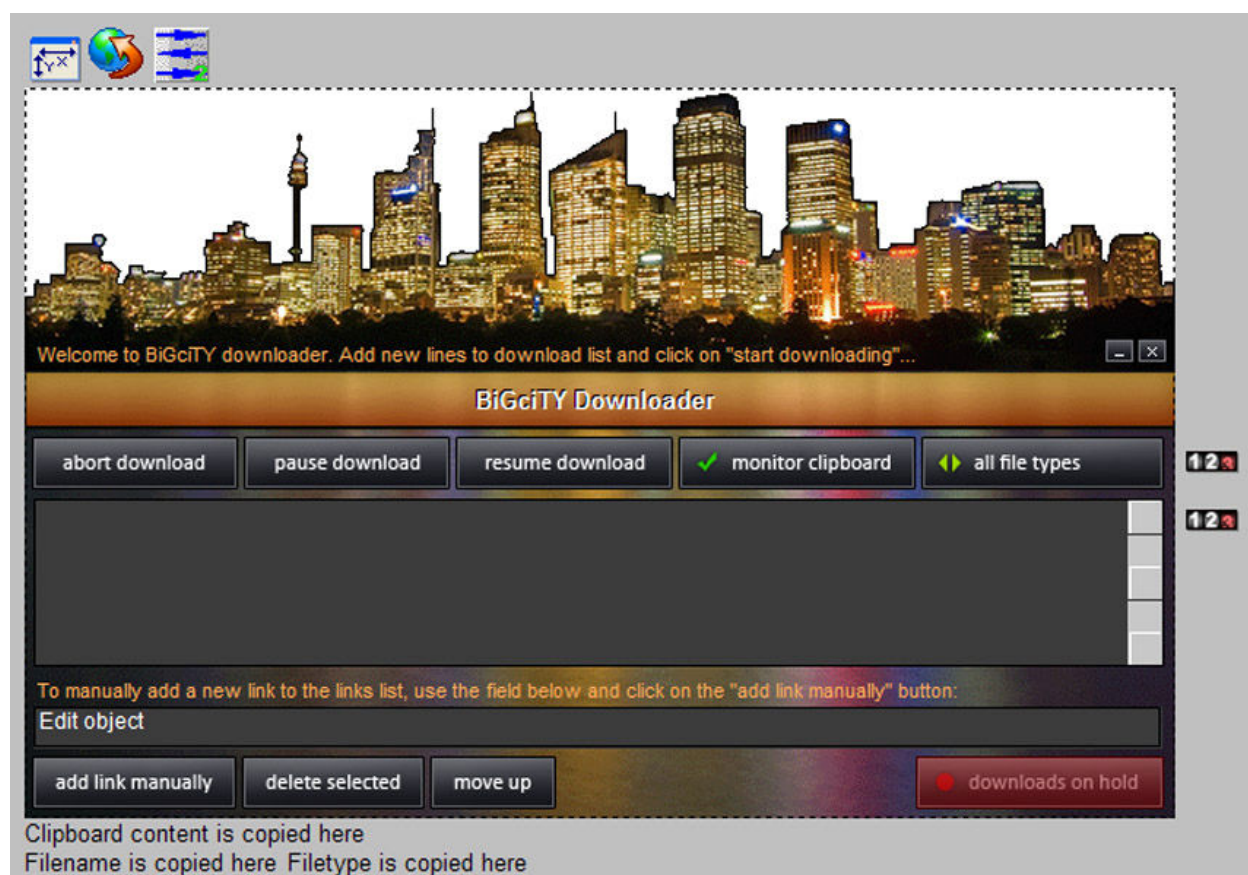
Alternative e-mail: koobare@panzerflakes.com

- 📄 **Note: I've been receiving several reports that not all e-mails get to me for some reason. Seems that some of them (quite a lot) end up in my spambox or are blocked out by the server.** I dunno why this is happening, so if you're experiencing any difficulties with delivering me a message or haven't received a reply in quite some time, please, send me another e-mail at marchewkowy@wp.pl, making sure that its title begins with "To Koobare:". I'll do my best to check both these e-mails regularly.

☰ If you're interested in all the other stuff I do, not just my tutorials, check out this nifty little site: www.panzerflakes.com – there's some free stuff there, wallpapers and royalty-free resources for your games, plus you can hire the wackiest loony amongst clicking mercenaries (yup, that would be me) to do your graphical bidding. Argh!

Part I: This one should be quick...

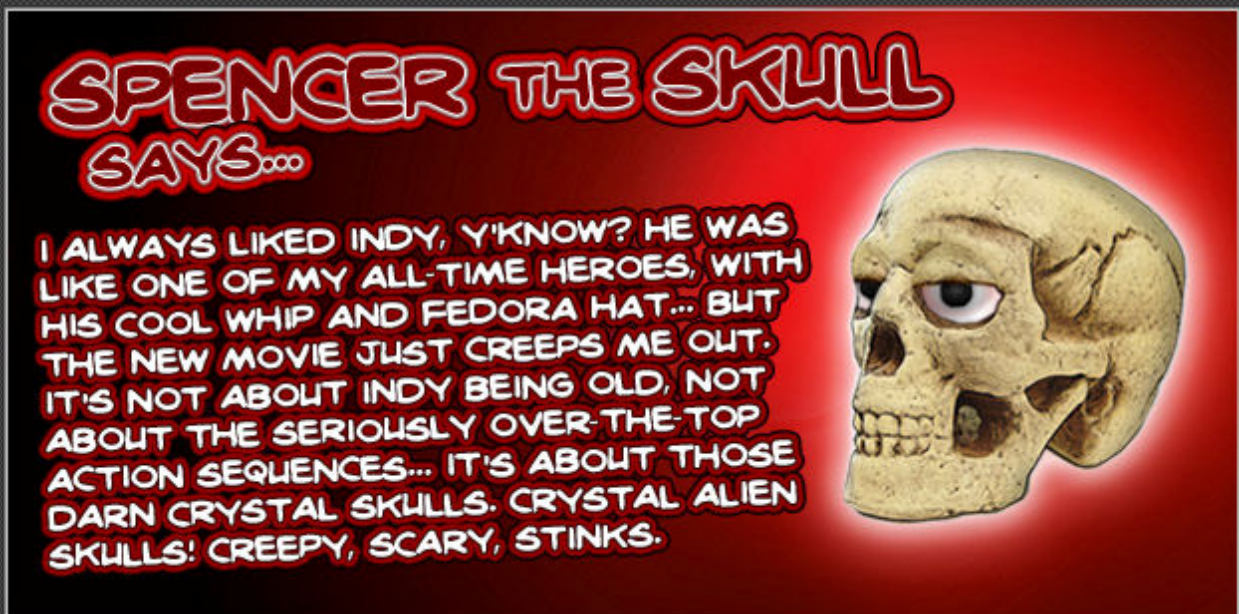
The title of this here part says it all: this one will be quick. As I said a bit earlier – this tutorial isn't meant for novices, so there won't be any beginner-level object creation here, we'll just open up a file that I geared up a bit earlier and move on. So... Let's do that now. Open the **"BiGciTY-starter.mfa"** file in MMF2, and take a look at what we've got there...



Looks kinda' nice, doesn't it? A lot of buttons, some text here and there, a bunch of counters and other stuff... Let's examine them, one at a time:

Object's name:	So... What is it?
button.abort	<p><u><i>A button for aborting your download.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>A button set to Bitmap Push type ("Normal" and "Pressed" states are present). When pushed, it aborts the current download – it's really as simple as that.</i></p>
button.close	<p><u><i>A button for closing our application.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>We're using the Window Shape object to change the shape of our window and because of that there is no system "close" button anywhere out there – this button supplies the user with the option to close down the whole app.</i></p>
button.minimize	<p><u><i>A button for minimizing our application.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>It enables our user to minimize the application, get it out of the way of his usual day-to-day work (as was said before – the usual system buttons are unavailable, since we're using the Window Shape object in this app)..</i></p>
button.pause	<p><u><i>Another button, this time for pausing the download.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>This button enables you to pause the current download – the downloading process is being paused and can be resumed later on with another button.</i></p>
button.resume	<p><u><i>A button for resuming our paused downloads.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>This Bitmap Push button enables you to easily resume the download you've paused earlier on.</i></p>
counter.bar	<p><u><i>A horizontal bar counter, displaying the download's progress.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Well, we should enable our users to keep track of how much of their files have been completely downloaded yet, right? This is what this counter is all about – it displays the percentage of the file that has currently found its way onto our hard drive.</i></p> <p>SETTINGS</p> <p><i>Initial Value is set to 0, Minimum value is set to 0, Maximum value set to 100. Display type set to Horizontal bar, counting from left, with a Vertical Gradient fill type (Color 1 = 250, 181, 37; Color 2 = 236, 77, 0).</i></p>

counter.clipboard.filetypes	<p><u><i>A hidden counter that helps us with the app's mechanics.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>This small hidden counter is pretty darn useful, it helps us control the whole app by defining the filetype that can be found in the clipboard... You'll see what it's for later on, when we'll get to the coding part of this here tutorial.</i></p>
counter.selected.filetypes	<p><u><i>Another hidden counter, similar to the one above.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>It's function is similar to the counter we've just looked at, so stay tuned to learn about what it does – we'll get to that as soon as we get to the programming part of this tut.</i></p>



Download Object	<p><u><i>The object that made all of this possible...</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Well, this is essentially the backbone of our whole application – this is the neat object that enables us to download files from the web and save them at our hard drive. We'll be using this one quite a lot once we're at the programming part.</i></p>
downloads.hold	<p><u><i>A button that isn't really a button.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>And here's a nice little active object used by me as a clickable button – you can double click on it to open the Animation editor and check out how it looks inside... It isn't too impressive, though, so feel free to just skip this and move on. When clicked, this button resumes or pauses all downloads, enabling us to create a download list first and start downloading a bit later on.</i></p>

Edit Box	<p><u><i>An Edit Box that enables you to input links manually.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>The sentence above says it all – this is just a simple Edit Box that can be used to add links manually to our download list – just input the full link here (“http” included) and press the “add link manually” button.</i></p>
helper.clipboard.content	<p><u><i>A string object that helps us out with our app’s mechanics.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>To tell you the truth, we could have just skipped this object and worked the whole thing around it, but – heck – I’m not here to make things more difficult on your... or me, on that matter. So we’ll be using this little object as a helper of sorts, which will enable us to keep the code as simple as possible.</i></p>
helper.filename	<p><u><i>Another string object that acts as a “helper” to our mechanics.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>You’ll see what I used these little guys for later on, as for now all you gotta’ know is that this thingie helps us simplify things by keeping the name of the file being downloaded inside.</i></p>
helper.filetype.in.clipboard	<p><u><i>Yet another string “helper”.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Once again, you’ll see what’s this for soon enough – all I can say now is that this one keeps the filetype extension of the file that’s currently being downloaded (or rather that is being prepped for download after being found in the clipboard).</i></p>
linklist.add	<p><u><i>A button for adding links manually.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Just press this button to copy the contents of the “Edit Box” object into the downloads list.</i></p>
linklist.delete	<p><u><i>Another button for controlling our linklist.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Press this button to delete the selected link from our downloads list – it’s as simple as that.</i></p>
linklist.up	<p><u><i>Move the selected link one position up in the list.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Wanna’ download some links faster than the others? Just select them and press this little button a couple of times to get them to the top of the downloads list.</i></p>

List	<p><u><i>A list object containing all our download links.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>This object is pretty crucial for our application – this is the famous “linklist” (also called known as the “download list”) we’ve spoken of before – it contains a list of files that are about to be downloaded by our app. We’ll be back to this object as soon as we hit the programming part of this tut.</i></p>
monitor.clipboard	<p><u><i>Another active object acting as a button.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>When pressed, it either enables or disables the clipboard monitoring option of our application – when it’s enabled, the app searches through the clipboard for the specified filetypes and adds them to the linklist if it finds them.</i></p>
monitor.filetypes	<p><u><i>Active object that is used to cycle through monitoring options.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>This active object is used to determine which filetypes should be copied from the clipboard into our downloads list – it can be set to “all file types”, “images & videos”, “document files” and “rar & zip archives”. It also has a “disabled” frame, with the grayed-out “no monitoring” text right on top of it, which is displayed when the user shuts off the clipboard monitoring function of our app.</i></p>
percentage.1 + percentage.2	<p><u><i>Two strings that display the download percentage.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Just another way of telling the user exactly how far is he with downloading that desired file of his.</i></p>
red.space	<p><u><i>A small semi-transparent active object.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>I’ve placed it over the “downloads.hold” active-button, just to achieve a nice “clicking” graphical effect that I’ve incorporated onto all the other buttons.</i></p>
String Parser	<p><u><i>A very useful object, used for... well, parsing strings.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>To tell our little app which of the strings copied into the clipboard are actually html links, pointing to a specific type of file.</i></p> <p><i>Basically it works like this: when you specify a delimiter of some sorts, the String Parser will divide a single text string into several strings, using the delimiter as a “borderline” that tells the object where to cut. For example: parsing a string saying “omgxxftw” using an “xx” delimiter, will give you two separate strings in result: “omg” and “ftw”.</i></p>

text.messages	<p><u>Just a string object, used for displaying program messages.</u></p> <p>WHY IT'S HERE?</p> <p>Well, this basically just tells the user about a file being successfully downloaded, but it is also used as a “tutorial string”, displaying some info about how to manually add links to the list.</p>
text.upper	<p><u>Another string object, similar to the one above.</u></p> <p>WHY IT'S HERE?</p> <p>It says “Welcome to BiGciTY downloader. Add new lines to download list and click on >>start downloading<<” on start, and then goes on to displaying some useful info, such as estimated download time and download speed.</p>
Window Control	<p><u>An object that enables us to manipulate the window of our app.</u></p> <p>WHY IT'S HERE?</p> <p>This object was a bit more useful in the first version of this app, which I scraped and built from the start after some thoughts – right now, it is only used to minimize our application, a simple, yet important task.</p> <p>WHAT CAN YOU DO WITH IT?</p> <p>Although in this tutorial we use only the simplest and most trivial of the actions that this object supplies us with, you should always remember that this object can help you a lot in your own projects, every time you need to change the size of your window, set its position, attach it to desktop or restore its focus. Check out this objects collection of actions, conditions and expressions – there are lots of ways you can use the Window Control in your games and applications.</p>
Window Shape	<p><u>An object that enables us to manipulate our app's window shape.</u></p> <p>WHY IT'S HERE?</p> <p>As you've probably already guessed by looking at the screenshot on the first page of this here tutorial, BiGciTY Downloader is one of those apps that use their own unique window shape, instead of relying on the usual quadrangle-shaped one. Thanks to this object I was able to import a nice cityscape image and use it as our window shape (just check this object's preferences).</p>

Please note: some of the objects use alpha channels, a feature that is unavailable in Games Factory 2 – If you're a TGF2 user, you might need to use your own objects instead (I would also advise you to upgrade to MMF2).

Part II: Coding.

As it was written in the ancient scripts, this tutorial is pretty straightforward, so we're not gonna' waste any more time on pointless details and other chit-chat. Open up the **Event Editor**. You all know my event-scripting system, dontcha'? If you need me to refresh your memory, take a look at this:

Koobare's MMF-to-paper coding system

IF (Condition): [Object for the condition] >> Condition group >> Condition
THEN (Action): [Object for the action] >> Action group >> Action

Conditions are red, actions are written in blue. Object names are put in [square brackets]. If we'll have a multi-condition event, then it'll be like this:

IF (Condition 1): [Object for condition 1] >> Condition group 1 >> Condition 1
IF (Condition 2): [Object for condition 2] >> Condition group 2 >> Condition 2
THEN (Action): [Object for the action] >> Action group >> Action

Whereas a multi-action event looks like this:

IF (Condition): [Object for condition] >> Condition group >> Condition
THEN (Action 1): [Object for the action 1] >> Action group 1 >> Action 1
THEN (Action 2): [Object for the action 2] >> Action group 2 >> Action 2

Any additional comments, instructions and info (including everything you have to input by keyboard) will be put in << double angle brackets >>, like this:

<< Select any wave sound from the MMF2's sound library >>

From time to time I'll also use this style to throw in some extra tips and tricks about MMF2 and more advanced coding techniques. All you have to do is to go step-by-step through all the listed events and keep one eye on your Event Editor, and the second one on this tutorial...

Come on, let's get this show on the road!

1) Firstly, let's start off with the event I usually create at the very beginning of each tutorial's programming part – the "Start of frame" one... This one will be as simple as they get, as all it does is clearing the clipboard:

IF: [Storyboard Controls] >> Start of frame

THEN: [Special Object] >> Clipboard >> Clear clipboard

2) The second one will be a bit longer, but still easy to follow and pretty much self-explanatory. All we do here is make sure that the strings display correct information and that all of our buttons are currently displaying the right animation:

IF: [Special Object] >> Always

THEN: [percentage.2] >> Change alterable string

<< input: Str\$(CurrentPercent("Download object", 1))+ "%" >>

THEN: [percentage.1] >> Change alterable string

<< input: string\$("percentage.2") >>

THEN: [monitor.clipboard] >> Alterable value >> Set

<< Set Clicked to 0 >>














THEN: [monitor.filetypes] >> Alterable value >> Set

<< Set Clicked to 0 >>

THEN: [downloads.hold] >> Alterable value >> Set

<< Set Clicked to 0 >>

Got it? Great! Now it's time to create all the groups we're gonna' need in this frame... Take a look at this here screenshot to learn what has to be created where...

All the events All the objects		            												
1	• Start of Frame													
2	• Always			✓	✓						✓	✓	✓	
3	Window controls													
4	Download controls													
5	Downloading													
6	Hold and resume downloads													
7	Monitor clipboard													
8	Linklist controls													

3) ...and let's create those groups right now. Create 6 primary event groups and name them respectively: **Window controls**, **Download controls**, **Downloading**, **Hold and resume downloads**, **Monitor clipboard** and **Linklist controls**. Make sure that they are all set to active when the frame starts. Once that's done, create two additional groups inside the **Monitor clipboard** group – name them **Controls** and **Files in clipboard**. Got it? Then let's move on...

4) Open up the **Window controls** group and create these two events inside, one after the other... These two thingies basically control the *minimize* and *close* functions of our app:

IF: [button.minimize] >> Button clicked?

THEN: [Window Control] >> Resize >> Minimize Window

Here's the second event... Note that we don't have to destroy all these objects before shutting down our app – but it can be useful if you want to create a nice fade out transition effect for this app (since buttons and some system objects won't fade into black, they'll just stay solid):

IF: [button.close] >> Button clicked?

THEN: [button.close] >> Destroy

THEN: [button.minimize] >> Destroy

THEN: [linklist.add] >> Destroy

THEN: [button.abort] >> Destroy

THEN: [button.pause] >> Destroy

THEN: [button.resume] >> Destroy

THEN: [Edit Box] >> Destroy

THEN: [List] >> Destroy

THEN: [monitor.clipboard] >> Destroy

THEN: [monitor.filetypes] >> Destroy

THEN: [downloads.hold] >> Destroy

THEN: [text.messages] >> Destroy

THEN: [text.upper] >> Destroy

THEN: [percentage.2] >> Destroy

THEN: [percentage.1] >> Destroy

THEN: [red.space] >> Destroy

THEN: [linklist.delete] >> Destroy

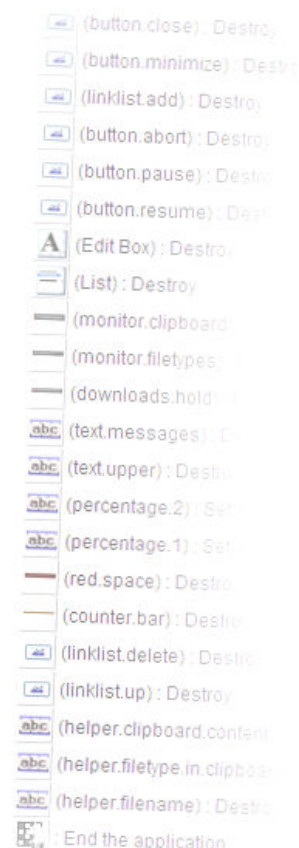
THEN: [linklist.up] >> Destroy

THEN: [helper.clipboard.content] >> Destroy



THEN: [helper.filetype.in.clipboard] >> Destroy

THEN: [helper.filename] >> Destroy

THEN: [Storyboard Controls] >> End the application



And here's what we've got so far:

All the events All the objects																			
1	• Start of Frame	✓																	
2	• Always																		
3	Window controls																		
4	• Button  (button.minimize) clicked																✓		
5	• Button  (button.close) clicked			✓						✓	✓	✓	✓	✓	✓	✓	✓		
6	• New condition																		
7	Download controls																		
8	Downloading																		
9	Hold and resume downloads																		
10	Monitor clipboard																		
11	Controls																		
12	Files in clipboard																		
13	• New condition																		
14	Linklist controls																		
15	• New condition																		

5) Now, move on to the next event group – the one named **Download controls**. Create this event inside, to enable our users to control the download's progress:

IF: [Download Object] >> Download in progress?

<< input 1 to select slot 1 >>

THEN: [counter.bar] >> Set Counter

<< input: CurrentPercent("Download object", 1) >>

THEN: [text.upper] >> Change alterable string

<< input: "Speed: "+Str\$(CurrentSpeed("Download object", 1))+ " / Estimated download time: "+Str\$(TotalTime("Download object", 1))+ " sec." >>

6) Once that's done, create these three events inside the same group... These will enable the user to pause, resume and abort the download:




IF: [button.pause] >> Button clicked?

THEN: [Download Object] >> Pause download

<< input 1 to select slot 1 >>

THEN: [button.pause] >> Disable

THEN: [button.resume] >> Enable

6	• Button  (button.pause) clicked												✓	✓				✓	
7	• Button  (button.resume) clicked												✓	✓				✓	
8	• Button  (button.abort) clicked																	✓	


```

IF: [button.resume] >> Button clicked?
THEN: [Download Object] >> Resume download
<< input 1 to select slot 1 >>
THEN: [button.resume] >> Disable
THEN: [button.pause] >> Enable

```

```

IF: [button.abort] >> Button clicked?
THEN: [Download Object] >> Abort download
<< input 1 to select slot 1 >>

```

7) Now, close up the **Download controls** group and enter the next one – the one with the oh, so intricate name: **Downloading**. The events we’re going to create inside this group are a bit more tricky and complex than those we’ve made before, in the previous two event groups, but – no worries – when you’ll create them one by one, it’ll all make perfect sense.

```

IF: [Special Object] >> Compare to a global value
<< Choose value Hold downloads >>
<< compare if it equals 0 >>
[NEGATE] IF: [Download Object] >> Download in progress?
<< input 1 to select slot 1 >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
THEN: [String Parser] >> Set source string
<< input: List Line Text$( "List", 1) >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Add delimiter
<< input: "/" >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Delete delimiter
<< input: ." >>
THEN: [Download Object] >> Download file
<< input List Line Text$( "List", 1) as the URL >>
<< Set the destination file via expression editor to listLast$( "String Parser" ) >>
<< input 1 to select slot 1 >>
THEN: [List] >> Delete line
<< input: 1 >>

```

So... How does it work? Look at it closely: the app checks whether the downloading process is on hold or not – if not (and nothing is downloaded at the moment), it splits the file’s URL with the help of the String Parser (using the “/” delimiter) to get the file’s name (for example: *duck.jpg* or *panzerflakes-com-rules.zip*), adds a new download task from the list, deletes the link from the list and saves the file to the correctly named file. Pretty simple, right?

8) And here's the next event, which goes into the same event group (**Downloading**). This one will download the next file from the list once the first one is finished downloading:

```
IF: [Download Object] >> Download successfully completed?
<< input 1 to select slot 1 >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
[NEGATE] IF: [Download Object] >> Download in progress?
<< input 1 to select slot 1 >>
IF: [Special Object] >> Compare to a global value
<< Choose value Hold downloads >>
<< compare if it equals 0 >>
IF: [Special Object] >> Compare two general values
<< check if List Nb Lines( "List" ) is greater or equals 1 >>
THEN: [Download Object] >> Download file
<< input List Line Text$( "List", 1) as the URL >>
<< Set the destination file via expression editor to listLast$( "String Parser" ) >>
<< input 1 to select slot 1 >>
THEN: [List] >> Delete line
<< input: 1 >>
THEN: [String Parser] >> Set source string
<< input: List Line Text$( "List", 1) >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Add delimiter
<< input: "/" >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Delete delimiter
<< input: "," >>
```

9) Another event that should be created inside the **Downloading** group. This one let's the user know that a file has been downloaded, both with a sound and a text message:

```
IF: [Download Object] >> Download successfully completed?
<< input 1 to select slot 1 >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
THEN: [Sound Object] >> Samples >> Play sample
<< Choose some kind of a short clicking sound from MMF2's libraries >>
THEN: [text.messages] >> Change alterable string
<< input: "File successfully downloaded!" >>
```

10) Once that's done, close up the current group and move on to the next one: **Hold and resume downloads**. Create this event inside:

IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*
 << select: left button, single click >>
 << choose the [downloads.hold] object >>
 IF: [Special Object] >> *Compare to a global value*
 << check if *Hold downloads equals 1* >>
 IF: [downloads.hold] >> *Alterable Values >> Compare to one of the alterable values*
 << check if *Clicked equals 0* >>
 IF: [Special Object] >> *Compare two general values*
 << check if expression *List Nb Lines("List")* is greater than *0* >>
 THEN: [Special Object] >> *Change a global value >> Set*
 << Choose value *Hold downloads* >>
 << input: *0* >>
 THEN: [Download Object] >> *Resume download*
 << input *1* to select slot 1 >>
 THEN: [downloads.hold] >> *Animation >> Change >> Animation Sequence*
 << select *Stopped* >>
 THEN: [button.pause] >> *Enable*
 THEN: [button.resume] >> *Disable*
 THEN: [downloads.hold] >> *Alterable value >> Set*
 << Set *Clicked to 1* >>

Here's a small screenshot of this event (note that I use custom colors and "check" icons):



11) Got it? Then it's time for the next one, just below the previous one (inside the same group):

IF: [Keyboard & Mouse Object] >> The Mouse >> *User clicks on an object*
 << select: left button, single click >>
 << choose the [downloads.hold] object >>
 IF: [Special Object] >> *Compare to a global value*
 << check if *Hold downloads equals 0* >>
 IF: [downloads.hold] >> *Alterable Values >> Compare to one of the alterable values*
 << check if *Clicked equals 0* >>
 THEN: [Special Object] >> *Change a global value >> Set*
 << Choose value *Hold downloads* >>
 << input: *1* >>
 THEN: [Download Object] >> *Pause download*
 << input *1* to select slot 1 >>

```
THEN: [downloads.hold] >> Animation >> Change >> Animation Sequence
<< select Resume downloads >>
THEN: [button.pause] >> Disable
THEN: [button.resume] >> Enable
THEN: [downloads.hold] >> Alterable value >> Set
<< Set Clicked to 1 >>
```

12) Here's a shorter event for ya', be sure to put it in the same group as the two previous ones (the **Hold and resume downloads** group):

```
IF: [Special Object] >> Compare two general values
<< check if expression List Nb Lines( "List" ) equals 0 >>
THEN: [Special Object] >> Change a global value >> Set
<< Choose value Hold downloads >>
<< input: 1 >>
THEN: [downloads.hold] >> Animation >> Change >> Animation Sequence
<< select Resume downloads >>
```

13) Here we go with another one... Once again: create this inside the same group...

```
IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: left button, single click >>
<< choose the [downloads.hold] object >>
THEN: [red.space] >> Visibility >> Make Object Reappear
THEN: [red.space] >> Animation >> Change >> Animation Sequence
<< select Blink >>
```

14) And here's the last event from the **Hold and resume downloads** group:

IF: [red.space]>> Animation >> *Has an animation finished?*
 << select *Blink* >>
 THEN: [red.space] >> Visibility >> *Make Object Invisible*

[illegible]

So... Four groups done, two to go (even though one of them is quite massive, divided into two separate subgroups). Don't waste any time, let's move on!

15) Open up the **Controls** subgroup, located inside the **Monitor Clipboard** group and create all of these events inside:

```
IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: left button, single click >>
<< choose the [monitor.clipboard] object >>
IF: [Special Object] >> Compare to a global value
<< check if Monitor clipboard equals 0 >>
IF: [monitor.clipboard] >> Alterable Values >> Compare to one of the alterable values
<< check if Clicked equals 0 >>
THEN: [monitor.clipboard] >> Alterable value >> Set
<< Set Clicked to 1 >>
THEN: [monitor.clipboard] >> Animation >> Change >> Animation Sequence
<< select Stopped >>
THEN: [Special Object] >> Change a global value >> Set
<< Choose value Monitor clipboard >>
<< input: 1 >>
THEN: [counter.selected.filetypes] >> Set Counter
<< input: 1 >>
THEN: [Special Object] >> Group of events>> Activate
<< Select the Files in clipboard group >>
```

This one is pretty similar to the one above... But keep track of all those differences:

```
IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: left button, single click >>
<< choose the [monitor.clipboard] object >>
IF: [Special Object] >> Compare to a global value
<< check if Monitor clipboard equals 1 >>
IF: [monitor.clipboard] >> Alterable Values >> Compare to one of the alterable values
<< check if Clicked equals 0 >>
THEN: [monitor.clipboard] >> Alterable value >> Set
<< Set Clicked to 1 >>
THEN: [monitor.clipboard] >> Animation >> Change >> Animation Sequence
<< select Don't monitor >>
THEN: [Special Object] >> Change a global value >> Set
<< Choose value Monitor clipboard >>
<< input: 0 >>
THEN: [monitor.filetypes] >> Animation >> Change >> Animation Sequence
<< select No monitoring >>
```


THEN: [counter.selected.filetypes] >> Set Counter
<< input: 0 >>
THEN: [Special Object] >> Group of events>> Deactivate
<< Select the *Files in clipboard* group >>

Time for the third event in this group:

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: left button, single click >>
<< choose the [monitor.filetypes] object >>
IF: [Special Object] >> Compare to a global value
<< check if *Monitor clipboard equals 1* >>
IF: [counter.selected.filetypes] >> Compare the counter to a value
<< check if it's *Lower or equal 4* >>
IF: [monitor.clipboard] >> Alterable Values >> Compare to one of the alterable values
<< check if *Clicked equals 0* >>
THEN: [Special Object] >> Change a global value >> Set
<< Choose value *Monitor clipboard* >>
<< input: 1 >>
THEN: [monitor.filetypes] >> Alterable value >> Set
<< Set *Clicked to 1* >>
THEN: [counter.selected.filetypes] >> Add to Counter
<< input: 1 >>

And here's the fourth one...

IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks on an object
<< select: left button, single click >>
<< choose the [monitor.filetypes] object >>
IF: [Special Object] >> Compare to a global value
<< check if *Monitor clipboard equals 1* >>
IF: [counter.selected.filetypes] >> Compare the counter to a value
<< check if it's *Equal 5* >>
IF: [monitor.clipboard] >> Alterable Values >> Compare to one of the alterable values
<< check if *Clicked equals 0* >>
THEN: [Special Object] >> Change a global value >> Set
<< Choose value *Monitor clipboard* >>
<< input: 1 >>
THEN: [monitor.filetypes] >> Alterable value >> Set
<< Set *Clicked to 1* >>

THEN: [counter.selected.filetypes] >> *Set Counter*
<< input: 1 >>

Here are four more... But don't worry, these are pretty short and straightforward:

IF: [counter.selected.filetypes] >> *Compare the counter to a value*
<< check if it's *Equal 1* >>
IF: [Special Object] >> *Limit conditions* >> *Only one action when event loops*
THEN: [monitor.filetypes] >> *Animation* >> *Change* >> *Animation Sequence*
<< select *Stopped* >>

IF: [counter.selected.filetypes] >> *Compare the counter to a value*
<< check if it's *Equal 2* >>
IF: [Special Object] >> *Limit conditions* >> *Only one action when event loops*
THEN: [monitor.filetypes] >> *Animation* >> *Change* >> *Animation Sequence*
<< select *Images & Videos* >>

IF: [counter.selected.filetypes] >> *Compare the counter to a value*
<< check if it's *Equal 3* >>
IF: [Special Object] >> *Limit conditions* >> *Only one action when event loops*
THEN: [monitor.filetypes] >> *Animation* >> *Change* >> *Animation Sequence*
<< select *Documents* >>

IF: [counter.selected.filetypes] >> *Compare the counter to a value*
<< check if it's *Equal 4* >>
IF: [Special Object] >> *Limit conditions* >> *Only one action when event loops*
THEN: [monitor.filetypes] >> *Animation* >> *Change* >> *Animation Sequence*
<< select *Archives* >>

16) We're almost at the end, so keep your trousers on and don't run away. Close up the *Controls* subgroup and open up the next one: **Files in clipboard**. Create these event inside:

IF: [Special Object] >> *Compare to a global value*
<< check if *Monitor clipboard equals 1* >>
THEN: [String Parser] >> *Set source string*
<< input: *ClipText\$* >>
THEN: [String Parser] >> *List tokenizing* >> *Delimiters* >> *Add delimiter*
<< input: *"/"* >>
THEN: [String Parser] >> *List tokenizing* >> *Delimiters* >> *Delete delimiter*
<< input: *"."* >>

```

THEN: [helper.filename] >> Change alterable string
<< input: listLast$( "String Parser" ) >>
THEN: [String Parser] >> Set source string
<< input: string$( "helper.filename" ) >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Add delimiter
<< input:  "." >>
THEN: [String Parser] >> List tokenizing >> Delimiters >> Delete delimiter
<< input:  "/" >>
THEN: [helper.filetype.in.clipboard] >> Change alterable string
<< input: listLast$( "String Parser" ) >>
THEN: [helper.clipboard.content] >> Change alterable string
<< input: ClipText$ >>

IF: [Special Object] >> Always
THEN: [counter.clipboard.filetypes] >> Set Counter
<< input: 0 >>

```

17) Below you'll find even more events that should be placed inside the same group (and subgroup) – these thingies help us to determine the type of file that is currently stored inside our clipboard. Note that I've used the filtered OR operator between some of these conditions to simplify our code a bit. You can add OR operator between condition lines by right-clicking on them and choosing the operator from the drop-down list. Anyways, create these events inside **Files in clipboard**, one under the other:

```

IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "jpg" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "jpeg" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "gif" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "bmp" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "png" >>
THEN: [counter.clipboard.filetypes] >> Set Counter
<< input: 1 >>

```

```

IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "avi" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "mpg" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "mpeg" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "wmv" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "flv" >>
THEN: [counter.clipboard.filetypes] >> Set Counter
<< input: 2 >>

```

And here are the most popular office documents extensions...

```

IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "doc" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "txt" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "rtf" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "pdf" >>
[OR] (filtered)
IF: [Special Object] >> Compare two general values
<< check if expression string$( "helper.filetype.in.clipboard" ) equals "docx" >>
THEN: [counter.clipboard.filetypes] >> Set Counter
<< input: 3 >>

```

So, when the "counter.clipboard.filetypes" counter is set to a value of 1, it means that our "helper.filetype.in.clipboard" object contains a link to an image. Value of 2 means that it's linking to a video file, whereas a value of 3 indicates that the given link points toward a document of some sorts, either a doc, txt, rtf, pdf or docx file. Oh, and we shouldn't forget about archives:

IF: [Special Object] >> Compare two general values

<< check if expression `string$("helper.filetype.in.clipboard")` equals `"rar"` >>

[OR] (filtered)

IF: [Special Object] >> Compare two general values

<< check if expression `string$("helper.filetype.in.clipboard")` equals `"zip"` >>

THEN: [counter.clipboard.filetypes] >> Set Counter

<< input: 4 >>

Note that I've used the filtered OR operator, not the "logical" one.

The screenshot shows a logic editor interface. On the left, there are two event rows labeled 14 and 15. Row 14 has a condition: `String$("abc" (helper.filetype.in.clipboard)") = "doc"` OR `String$("abc" (helper.filetype.in.clipboard)") = "txt"` OR `String$("abc" (helper.filetype.in.clipboard)") = "rtf"` OR `String$("abc" (helper.filetype.in.clipboard)") = "pdf"` OR `String$("abc" (helper.filetype.in.clipboard)") = "docx"`. Row 15 has a condition: `String$("abc" (helper.filetype.in.clipboard)") = "rar"` OR `String$("abc" (helper.filetype.in.clipboard)") = "zip"`. To the right of these conditions is a table with 20 columns. The columns are grouped into four sets of five. The first set of five columns is green, the second is grey, the third is green, and the fourth is grey. In the first green column of the third green group, there is a green checkmark icon. In the first green column of the fourth grey group, there is a green checkmark icon.

18) These four events should be created in the same subgroup as the ones above (inside the Files in clipboard group):

IF: [counter.selected.filetypes] >> Compare the counter to a value

<< check if it's Equal 1 >>

IF: [counter.clipboard.filetypes] >> Compare the counter to a value

<< check if it's Greater or equal 1 >>

IF: [Special Object] >> Compare to a global string

<< Choose string Clipboard history >>

<< check if it is Different than `string$("helper.clipboard.content")` >>

IF: [Special Object] >> Compare to a global value

<< check if Monitor clipboard equals 1 >>

THEN: [List] >> Add a line

<< input: `string$("helper.clipboard.content")` >>

THEN: [Special Object] >> Set global string

<< Select the "Clipboard history" Global String >>

<< input: `string$("helper.clipboard.content")` >>

IF: [counter.selected.filetypes] >> Compare the counter to a value

<< check if it's Equal 2 >>

Here's a quick look at how these two events look in my MMF2:

Remember that they don't have to look identical in yours: graphical nuances aside, it's not guaranteed that your object columns are set in the same sequence as they are in my editor.

Page 24/27

```

IF: [counter.selected.filetypes] >> Compare the counter to a value
<< check if it's Equal 4 >>
IF: [counter.clipboard.filetypes] >> Compare the counter to a value
<< check if it's Greater or equal 4 >>
IF: [Special Object] >> Compare to a global string
<< Choose string Clipboard history >>
<< check if it is Different than string$( "helper.clipboard.content" ) >>
IF: [Special Object] >> Compare to a global value
<< check if Monitor clipboard equals 1 >>
THEN: [List] >> Add a line
<< input: string$( "helper.clipboard.content" ) >>
THEN: [Special Object] >> Set global string
<< Select the "Clipboard history" Global String >>
<< input: string$( "helper.clipboard.content" ) >>

```

19) Close the **Files in clipboard** subgroup, close up **Monitor clipboard**, and move on to the **Linklist controls** events group. Three events to go...

```

IF: [linklist.add] >> Button clicked?
IF: [Special Object] >> Compare two general values
<< check if expression Left$(Edittext$( "Edit Box" ), 7) equals "http://" >>
THEN: [List] >> Add a line
<< input: Edittext$( "Edit Box" ) >>
THEN: [Edit Box] >> Editing >> Set text
<< input: "" - yup, that means we want to set it to no text >>

IF: [linklist.delete] >> Button clicked?
THEN: [List] >> Delete line
<< input: List Select( "List" ) >>

IF: [linklist.up] >> Button clicked?
IF: [Special Object] >> Compare two general values
<< check if expression List Nb Lines( "List" ) is greater than 1 >>
IF: [Special Object] >> Compare two general values
<< check if expression List Select( "List" ) is greater than 1 >>
THEN: [Special Object] >> Set global string
<< Select the "Swapped line" Global String >>
<< input: List Line Text$( "List", 1 ) >>
THEN: [List] >> Delete line
<< input: 1 >>

```

```
THEN: [List] >> Insert a line  
<< insert after line 1 >>  
<< input: List Select$( "List" ) >>  
THEN: [List] >> Delete line  
<< input: List Select( "List" ) >>  
THEN: [List] >> Add a line  
<< input: Swapped line >>
```

Aaaaand... That's it! You've done it! Congratulations!

And just like that... We're done! You've just created your own file downloader, ready to download your share of images, videos, documents and archives of the internet! There's still much to do, if you'd like to polish this app into something really, really useful – you can make the whole app a bit more automated, you can add new features, new filetype filters and some functional keyboard shortcuts – but as far as this tut is concerned, we're at the finish line! Play a bit with this app to learn new tricks on your own... And see ya' later, in another one of my tutorials. Drop me a word or two if you like 'em!

Thanks for your time and see you again soon!

Cheers!



*If you have any questions, suggestions or just need help –
mail me at marchewkow@gmail.com
oh, and remember to check out
www.panzerflakes.com*

Something I can help you with? I'm for hire! If you need any graphics, animations, illustrations, anything I can help you with – just contact me and we'll see what we can do about it.

You have been reading...

BiGciTY DOWNLOADER

brought to you by
Koobare

Clickteam's funkiest
~~mercenary~~
nerd



Created for Multimedia Fusion 2 & Multimedia Fusion 2: Developer

Always be sure to have your MMF2 up-to-date!