the Enhancing FEEL

Weather effects:

Snow & Wind

Koobare
marchewkowy@gmail.com

Hi there, all!

Welcome to another one of Koobare's tiny tutorials, teaching you how to effectively and efficiently use the best multimedia authoring tool ever - Multimedia Fusion 2 by Clickteam! The main purpose of this tutorial is to help you create a stunning snow & wind effects for your games, using the Sinewave movement from the Gwerdy Movement Pack.

As it was said in the first "Enhancing the Feel" tutorial (the one about creating a rain effect), this series isn't really about the "create a full game from scratch" approach. Nope, if you're looking for something like that – download the "*Glob Wars*", "*Smelly Claw*", "*Fusion Player*" or "*Catch the fruit*" (by Andos) tutorials from Clickteam's website. This series is about a completely different thing – about focusing on creating a few particular game elements that will be ready to implement into your own games and applications, helping you not only to enhance the overall gameplay experience, but also teaching you a few tricks that might come in handy somewhere in your game development process. Remember: this is all about tips & tricks for you to use in your own game projects, so don't be shy and just use them!

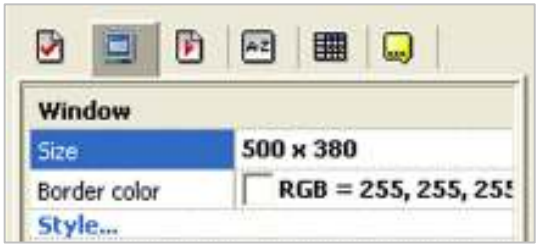
As it was just said – all we want to do here is to enhance the "feel" of our projects – by making them seem more real and "alive". To do so, we're going to introduce two tiny effects, that can not only add a little life to your game's world, but can also be quite eye-pleasing and add a little spice to your game's graphical area. To achieve this, we're going to use event groups, buttons, counters, a bunch of active objects and – last but not least – the ***Gwerdy Movement Pack*** add-on, that can be found in the "*Download Center*" corner of Clickteam's website.

Before you continue, be sure to obtain the *Gwerdy Movement Pack*!

- You can download the pack directly from Clickteam's website, here:
  http://www.clickteam.com/eng/downloadcenter.php?i=187
- If you don't want to type in the whole address given above – just browse through the "*Download Center*" area of Clickteam.com.
- If something's wrong, or Clickteam's website is currently having a short vacation, seek out the pack on Gwerdy's site:
  http://www.gwerdy.com/products/mmf2_movements/index.php
  Be sure to send Gwerdy a big "thanks for the pack!" e-mail, if you wish!

Part I: Setting up the application.

Before we even start… Please note that the "*Weather effects: Snow & Wind*" tutorial uses the same application settings as the previous tutorial from "*Enhancing the Feel*" series – "*Weather effects: It's raining!*". In other words: if you've been playing around with the "*It's raining!*" tutorial, you already have the required application prepared – just open the file, add a new frame and go for Part II of this tutorial, skipping Part I. If you haven't done the "*It's Raining*" tutorial – or if you enjoy a bit of repetition from time to time – let's create our new base-app!



OK, let's do it, then. Open Multimedia Fusion 2, **create a new application** and save it onto your hard drive (hint: turning the "autobackup" option ON in MMF2's preferences and – additionally – saving some backups on your own can sometimes be a good idea – we all know that Windows isn't exactly bug-proof, don't we?). Go to your application's **Properties Toolbar** (if it didn't open up by itself, right click on your application's name in the Workspace Toolbar and select "*Properties*" from the drop-down menu), and select the **Window** tab (second from the left, the one with the little computer screen). Set the window size to **500x380**. A standard 640x480 would work here well too, but – as I mentioned before, in one of the earlier tutorials – I'm sometimes struck by this strange urge to work on window sizes different than the usual 640x480 or 800x600… So, let's do it my way, shall we? After changing our window sizes MMF2 should ask you whether you'd like to modify the size of your frames as well – click "yes" (if – by some strange, twisted accident – MMF2 won't ask you this question - just make our first frame's size identical to the size of our window: **500x380**). Later on we're going to create a second frame (each effect will be shown in a separate frame), but let's not think about this now. When you're done - continue to part two of this tutorial.

Part II: Creating buttons and counters. Setting up the objects for the Wind effect.

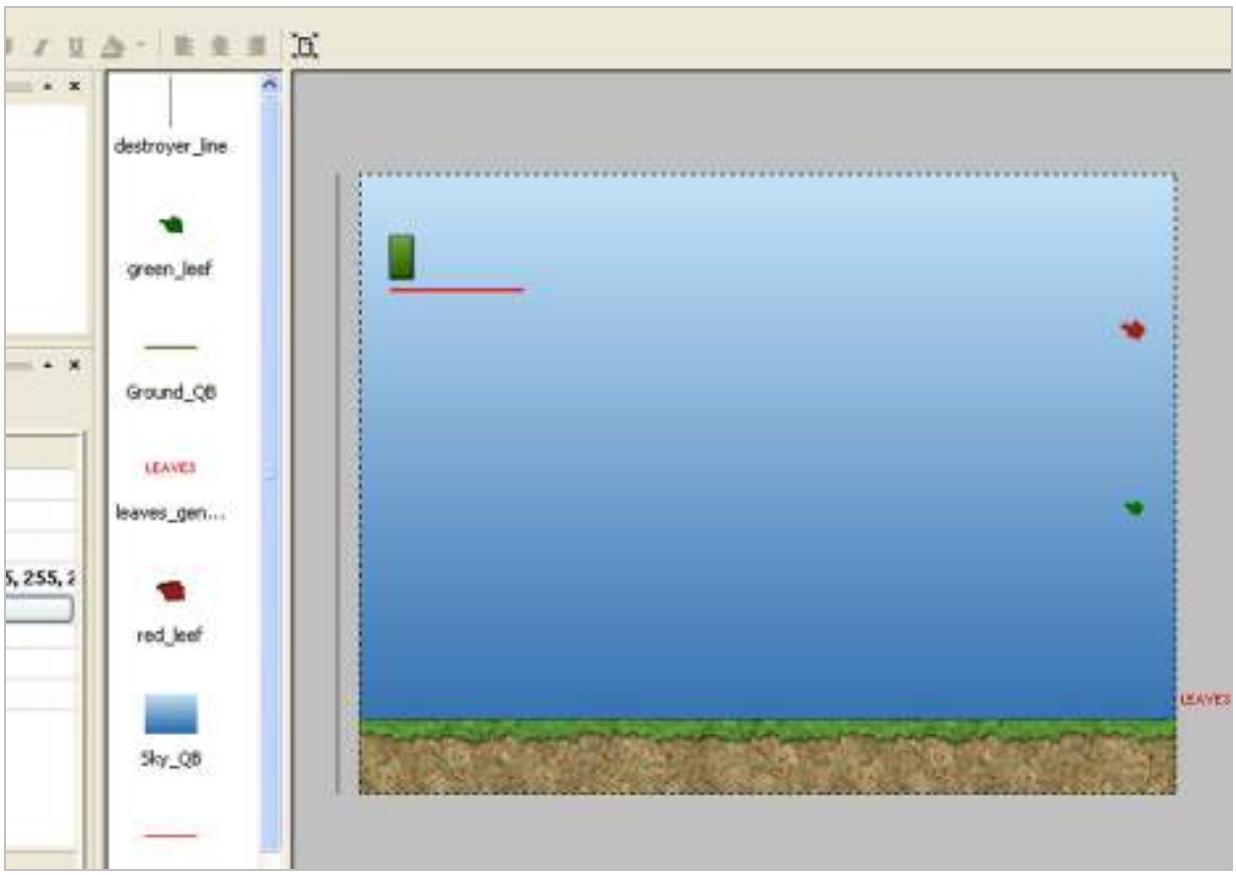
Once we have what we needed – a new application with a 500x380 frame – it's time to import and/or create all the needed objects. Let's do this now. Firstly, make sure that you have the "*snap to grid*" option turned off (check the "*view*" menu and search for the Snap to Grid button). Secondly, find the "**weather2library.mfa**" file (it was zipped into the same archive as this .PDF tutorial) and open it. Enter the first frame from the library file. Select all the objects there

(hint: press **CTRL+A** or find the *Select All* command in the *Edit* menu), copy them into your application (fastest method? Use the **CTRL+C** and **CTRL+V** keyboard shortcuts, you could also drag&drop objects between apps in the *Workspace Toolbar* area) and place them in a way, that will make the "Sky_QB" quick backdrop perfectly fit the frame (if the "snap to grid" option is turned off, there shouldn't be any problems with this). Remember that if you experience any problems with positioning, you can always use the cursor keys to move an object by one pixel (just be sure to select it first).

All TGF2 users: be advised!

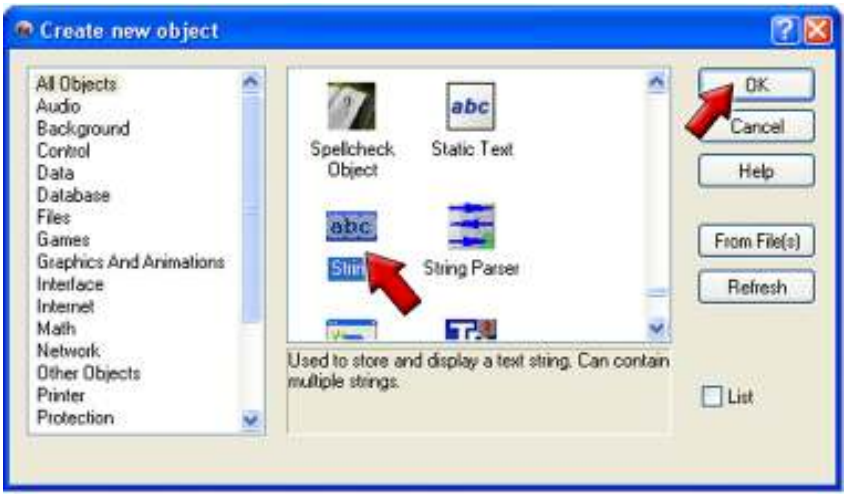
- Please note that some of the objects created for this tutorial use **alpha channels**, a feature that is unavailable in The Games Factory 2. TGF2 users should use basic library objects or create their own graphics instead.

Once you've copied the library objects, our frame should look exactly like this (if it doesn't… well, then perhaps one of us has done something wrong):



Time to see what objects do we have here. We've got two quick backdrops (one for the sky and one for the ground – same way as in the "*It's raining!*" tutorial), two active objects portraying leaves (the "red_leaf" and "green_leaf" objects), one active object that will act as our "leaf-creator" object ("*leaves_generator*" – it will help us create leaves, that are used to indicate the wind's strength), two counters ("*wind_overall*" and "*wind_current*" – the first one has the "animation" display type set up, the second one is a typical horizontal bar counter), and yet another active object – this one being really thin – vividly named "*destroyer_line*" (this object will help us to control the number of leaves in the game, by destroying those that will be flying right into it – having too many active objects at once on the screen can make some of the older computers choke and die). And… That's about it. At least when it comes to importing objects. We still have to create a couple more on our own, though. Let's do it now.

Let's start with creating a simple **String object**. In case you didn't know – a String object stores and displays text strings, in a single-formatted fashion (e.g. you can't underline single words of the given text, you can only reformat the whole string at once – to have a greater control over your text's formatting, use one of the other texts objects, such as the Formatted Text or the Rich Edit objects). Double-click on a blank spot in your Frame editor's main work area, or select the *Insert* > *New Object* command from the top menu. The "**Create New Object**" dialog window will appear (take a look at the image below for a bit of visual aid).



Now, let's find our *String object* from the objects list. Push the "S" button on your keyboard to scroll down to objects beginning on the letter "S". Then, search the list until you'll find the String object. Select it and click the "OK" button.

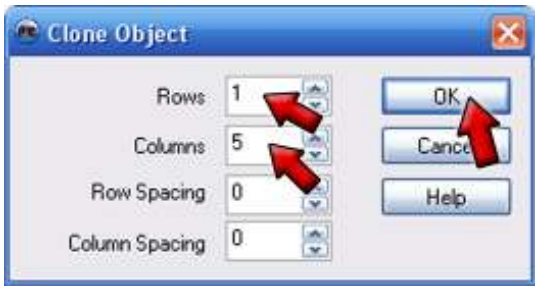
When you'll click somewhere on your unoccupied workspace, a new String object will be created. Let's set it up a bit, shall we? Click on the String object and go to the Properties toolbar (if you've closed that toolbar, just right click on your String object and select the "*Properties*" command). Go to the first tab ("*Settings*") And change Paragraph 1 to "**Wind Power:***"* (just click on that paragraph's text and type "*Wind Power:*" in). Once that's done, go for the "*Display options*" tab (second tab from the left, the one with the monitor screen as it's

icon) and set the **Anti-aliasing** option ON. This will make our text a bit smoother. Now, let's voyage further on, click on the third tab from the left (the "Size / Position" tab) – we'll set up this String object's coordinates on the screen. Just go to the "X" line and input **17**. Then, let's edit the "Y" item, and set it to **18**. Our String object should now spring onto it's position. Last thing before we move on: go to the first tab from the right (the "About" tab) and change the string's name to "**wind_power_string**".

OK, we've just prepared our String object. Before we move on to the coding part, we'll still have to create a couple of buttons. Seven to be exact…Let's do that know. Once again, open the "Create a new object" dialog by right-clicking on an empty space in your Frame editor and selecting the "*Insert object*" command from the drop-down menu. Select the **Button object** from the object's list, press the OK button and click on an empty space in your Frame editor. Go to the *Properties toolbar* and open the "*Settings*" tab (first one to the left). Now, change the button's text to "**Random**" (if you need visual guidance, check the image to the left) and check the "**Disabled at start**" on. When that's done, move on to the "*About*" tab (first from the right) and change this button's name to "**random_button**". Now, go to the "*Size / Position*" tab (second from the left) and change the button's **X Position** to **320**, **Y Position** to **13**, **Width** to **64** and **Height** to **21**. The button should reposition itself.

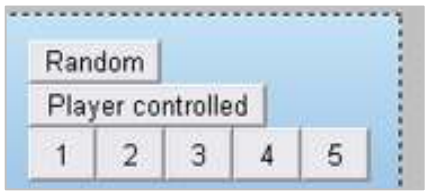
OK, having the first button created – let's have a go for the second one. **Create a new button**, input "**Player controlled**" as it's text and make sure that the "*Disabled at start*" option is **OFF**. Yep, this button should be enabled when our frame starts, so don't get confused here. Go to the "*Size / Position*" tab (you should know where to find it by now, right?) and change this button's **X Position** to **320**, **Y Position** to **34**, **Width** to **115** and **Height** to **21**. Last thing to do here: open the "*About*" tab and change our second button's name to "**pcontrolled_button**".

Two down, five to go. Well, that seems quite a lot, doesn't it? But don't worry – we're not going to create them all in the same way as the above ones – setting their position, width and height one by one. Nope, we'll just create one more similarly and then – voila! – we'll experience the magic of cloning. **Create a new button** once again, set it's text to "**1**" (yep, just the digit "one"). Change this button's **X Position** to **320**, **Y Position** to **55**, **Width** to **33** and **Height** to **26**, then change it's name to "**strength_button 1**".



It's now time for some cloning! Don't worry, you don't have to own a degree in genetics, nor your own sheep, to use MMF2's cloning system. It's all a matter of a few clicks here and there, totally Dolly-free. So, what's cloning all about? It's as simple as it gets – all you need to know about it, is that it creates the given number of new objects identical to the one you're. To clone our little button, right-click on it and select the "**Clone Object**" command from the drop-down menu. The "*Clone Object*" dialog should appear. Set the **number of rows** to **1**, set the **number of columns** to **5**, and make sure that both the *row spacing* and *column spacing* are set to **0** pixels (if you need visual guidance, take a look at the image to the upper right and follow the red arrows). Click "OK" when you're done.

OK, so we now have five identical buttons standing in a row. Right-click on the second button, counting from the left (and, by the way, notice that it's name is "strength_button 2" - the cloning system automatically renames freshly created clones). Select the "**Edit**" command (unless you've been messing around with MMF2's preferences, you can access the "*Edit*" window by double-clicking on the given object). Set this button's text to "**2**" (just the digit "two") and click the OK button. After that's done, move on to the next button and repeat the procedure, this time changing the text to "**3**". Next button should be set to – of course – "**4**" and I guess that I don't really have to say what should you input as the fifth button's text.



We now have seven buttons on the gamefield, waiting in the upper-right corner for someone to script their actions and conditions (take a look at the nice little image to the right). Emmm, sorry Koobare, but…– some of you could ask – what exactly do we need those buttons for? The answer is quite simple: **we need them to control our wind effect**. To explain this in detail: our wind effect will have five strength-states ("1" will stand for just a little gush of wind, "5" for almost a hurricane) and two control modes. When the whole application is set to the "player controlled" mode, the user will have to select one of the strength-states manually. In other words: our user will become the lord of the weather (and if you're not into that mythological climate, think of Storm from the "X-Men" series)! When the "random" mode is switched on – player will loose his abilities, and the whole wind system will be controlled by randomly generated numbers. Seems simple, right? A bit even TOO simple! Something fishy around here… An evil plot, a diabolical intrigue perhaps? Nope, it's really just THAT simple.

Since all our buttons are set up correctly, we can continue. Time to play with all those features that the **Gwerdy Movement Pack** has in stock for us! Select the **red_leaf** object, go to the *Properties toolbar* and open the **Movement** tab (third from the left). Click on the movement selection list (the one with the *Static* movement currently selected) and select the **Sinewave movement** (note: the *Sinewave movement* will be available ONLY if you have previously installed the *Gwerdy Movement Pack*!). Now, let's mess a bit with those settings there. Change the **Final X** value to **-40** and the **Final Y** to **230**. Set the **Amplitude** to **20**. Leave the *Speed*, *Angular Velocity* and *Starting Angle* settings unbothered. When you're done – move on!

Having the **red_leaf** set up, let's not forget about the **green_leaf** object. Select it now, set off for the *Properties toolbar*, select the *Movement* tab and change the movement type to *Sinewave movement*. Just like with the *red_leaf*, a couple of lines before. Now, change the **Final X** value to **-40**, **Final Y** to **170**, set the **Amplitude** to **25** and **Angular Velocity** to **45**. You can click on the "*Try movement*" button if you wish, but that's not necessary – we'll soon see all of this moving and floating around, once the programming part is complete.

Only one thing to do here before we go onto *Part III*… Remember that **leaves_generator** object mentioned earlier? The one that was supposed to help us create leaves on runtime? Sure you do. All we need to do here, is to give that *leaves_generator* thingy a nice **Path movement**. And that's gonna' be pretty simple. Select the "**leaves_generator**" object, go to it's properties, open the *Movement* tab... Open the drop-down movement list and select the *Path movement*. Now, click on the "*Edit*" button. The *Path Movement toolbar* should appear:

Click on the first button to the left (the one with a single line linking a box and an arrow) selecting the basic "*New line*" tool. Click at the **503, 10** coordinates of your frame (if you'll click a few pixels lower or higher, don't worry about it) to create a path running from the current

position of the *leaves_generator* object (that should *x=504*, *y=322* if I'm not mistaken) to the point found at *x=503, y=0*. Push the "*Loop the movement*" and "*Reverse at end*" buttons (those are the fourth and fifth buttons counting from the left) to receive a nice looping path movement. Click OK to close the *Path Movement toolbar* and return to your *Frame Editor*.

When that's done, it's time to move on to the next part! Will you manage to complete this tutorial in one piece? Will Koobare (*that's me, folks!*) lend you a helping hand in a time of struggle? Will we find the answers to all the riddles that were found in the first eight pages of this epic story? Who was the murderer and why wasn't the gardener involved? How come Hurley doesn't loose any weight and what exactly did Vader mean when he told Luke that he was his father? And where the heck is Waldo?! All this, and much more, right after these commercials…

OK, let's just skip it, throw away all those bad jokes and just head on to *Part III*, mmm-key? No advertisements, really. Nothing to see here. Just move on, people, move on...

Part III: It's time for some coding!

Koobare's MMF-to-paper coding system

Open the **Event Editor**. You know what's it all about, right? Sure you do. If, by any chance, you're not too familiar with these here surroundings – better get back to your MMF2 user manual or download the "Interface Guide" that can be found on Clickteam's website.

I guess that some of you have already met my MMF2-to-paper event-recording system, that helped us a lot with my earlier tutorials. If not – read on, it's pretty simple to learn and really quite useful. If you know what it's all about, just skip this introduction and head on to the scripting. Anyway, here's the deal:

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

All the conditions are marked in red color, while actions are written in lovely blue. Object names are always put in **[**square brackets**]**. The final condition/action is always in *Italic*. If we'll have a multi-condition event, then we'll have:

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*
THEN (Action 3): [Object for the action 3] > Action group 3 > *Action 3*

If you'll have to input anything by keyboard (for example: a value to set the counter to, or a text that is going to be displayed with the alterable string option – or other things that you use the *Expression Editor* for) it will be indicated by coloring the text in green and using < angle brackets >, like in this example (note that sometimes the given text will be set *Italic* for easier detection – it doesn't really mean anything):

< Set the Global Value A to 32 >

Additional comments, info and instructions will be put in << double angle brackets >>, using a different color:

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

There's not much philosophy in it, you just have to go step-by-step through all the events and keep one eye on your Event Editor, and the second one on this tutorial. Seems pretty simple, right? Let's start coding, then!

Coding the wind

Create all the events listed here, one after one:

1) Firstly, let's start with the "**Always**" event… Usually I start with the "*Start of Frame*" condition, but this time there's really no need for it and we won't even use it in our whole gamescript. Anyway, our "*Always*" event will make sure that the speed of our leaves (both the *red_leaf* and the *green_leaf* objects) is always set to the **wind_current** counter's value. All of this is quite simple, really.

IF: [Special Object] > *Always*

THEN: [red_leaf] > Movement > *Set speed*

< input: *value("wind_current")*2+5* >

<< Additional way to do this: click on the "*Retrieve data from object*" button, right-click on the "*wind_current*" counter, select "*current value*" and then input **2* and*+5* >>

THEN: [green_leaf] > Movement > *Set speed*

< input: *value("wind_current")*2+15* >

<< Additional way to do this: click on the "*Retrieve data from object*" button, right-click on the "*wind_current*" counter, select "*current value*" and then input **2* and*+15* >>

<< Please note: the number you're adding to the *above* equation is *fifteen*, not *five* >>

2) Our second event will help us to create smooth speed transitions between the current wind-speed and the speed that the player (or – when in the "random" mode – the computer) wants to set… Every 20/100 of a second the current wind speed adjusts itself by 1 to match the speed that has been set:

IF: [The Timer Object] > *Every*

<< Set the timer to *20/100* of a second>>

IF: [wind_current] > *Compare the counter to a value*

<< Check if this counter is *Lower* than the current value of the *wind_overall* counter >>

<< To set such a comparison, select *Lower* from the comparison method drop-down list, and then either input *value("wind_overall")* in the comparison edit box, or click on the "*Retrieve data from object*" button, right-click on the "*wind_overall*" counter and select "*current value*" from the list >>

THEN: [wind_current] > *Add to counter*

< input: *1* >

3) Here's the same event, but this time adjusting the current wind speed if it is higher than the one set by the player or by the random mode:

IF: [The Timer Object] > *Every*

<< Set the timer to *20/100* of a second >>

IF: [wind_current] > *Compare the counter to a value*

<< Check if this counter is *Greater* than the current value of the *wind_overall* counter >>

<< To set such a comparison, select *Greater* from the comparison method drop-down list, and then either input *value("wind_overall")* in the comparison edit box, or click on the "*Retrieve data from object*" button, right-click on the "*wind_overall*" counter and select "*current value*" from the list >>

THEN: [wind_current] > *Subtract from counter*

< input: *1* >

4) It's now time to create some an event that specifies what would happen if one of the **red_leaf** objects bounced right into the **destroyer_line** object:

IF: [red_leaf] > Collisions > *Another object* > [destroyer_line]
THEN: [red_leaf] > *Destroy*

5) The same event as above, but this time for the **green_leaf** object:

IF: [green_leaf] > Collisions > *Another object* > [destroyer_line]
THEN: [green_leaf] > *Destroy*

Additional coding information

- I've decided to skip this in the "Snow & Wind" tutorial, but there's an easy way to combine the 4th event and 5th one into a single line: just add a chosen Qualifier to both the *red_leaf* and *green_leaf* objects, and then use their group as the basis for the collision event. This can be considered as optimizing the code a bit.
- You can find more info on how to use Qualifiers in the "Fusion Player" tutorial.

6) And here's yet another event… As it was said before, the snow effect will end up in the second frame, whereas the wind effect will remain in frame numbero uno. This will event will enable us to jump to frame 2:

IF: [Keyboard & Mouse Object] > The Keyboard > *Upon pressing a key*
THEN: [Storyboard Controls] > *Next frame*

Oh, and one more thing: if you haven't done that yet, **save the app**, and do it now!

7) Moving on… Create a new **event group** (right click on the empty event-line and select *Insert > A group of events*), name it **"Wind is at random mode"** and make sure that the **"Active when the frame starts"** option is **ON**. While we're at it, create another two groups below: "**Wind is player controlled**" (make it's "*Active when frame starts*" option OFF) and "**Creating leaves**" (it should be *active* when the frame starts). When you're done, create the given event inside the first group ("*Wind is at random mode*"):

IF: [Special Object] > Group of events > *On group activation*

THEN: [Special Object] > Group of events > *Deactivate*

<< Select the "*Wind is player controlled*" group >>

THEN: [pcontrolled_button] > *Enable*

THEN: [strength_button_1] > *Disable*

THEN: [strength_button_2] > *Disable*

THEN: [strength_button_3] > *Disable*

THEN: [strength_button_4] > *Disable*

THEN: [strength_button_5] > *Disable*

THEN: [random_button] > *Disable*

8) Another event that should be created inside the "*Wind is at random mode*" group… This one will set the *wind_overall* counter to a random number between 0 and 49 – this random number is generated every 5 seconds:

IF: [The Timer Object] > *Every*

<< Set the timer to *5* seconds >>

THEN: [wind_overall] > *Set counter*

< input: *Random(50)* >

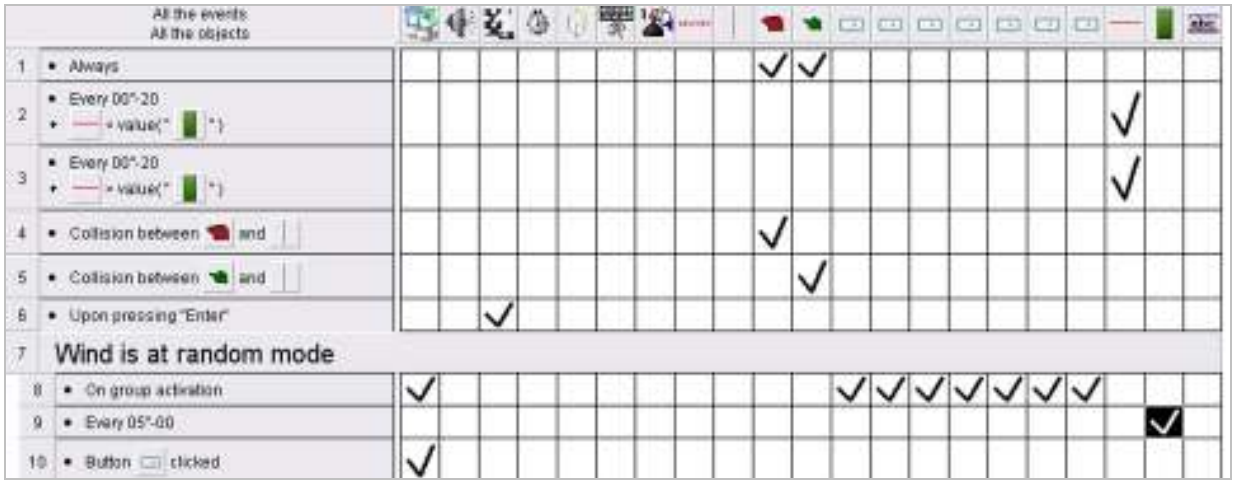
9) A short one, enabling the user to switch onto the "*Player controlled*" mode (create it inside "*Wind is at random mode*" group):

IF: [pcontrolled_button] > *Button clicked?*

THEN: [Special Object] > Group of events > *Activate*

<< Select the "*Wind is player controlled*" group >>

Here, let's take a look at what we've got by now (note that *it doesn't have to look identical*):



10) Now it's time to move on to the next group, the "*Wind is player controlled*" one. Create this event inside the previously mentioned group:

IF: [Special Object] > Group of events > *On group activation*
THEN: [Special Object] > Group of events > *Deactivate*
<< Select the "*Wind is at random mode*" group >>
THEN: [pcontrolled_button] > *Disable*
THEN: [strength_button 1] > *Enable*
THEN: [strength_button 2] > *Enable*
THEN: [strength_button 3] > *Enable*
THEN: [strength_button 4] > *Enable*
THEN: [strength_button 5] > *Enable*
THEN: [random_button] > *Enable*

11) And here comes a whole series of simple events, very similar to each other – these events will control what happens when the user clicks on a specific button in order to change the strength of the wind (all should be created inside the "*Wind is player controlled*" group):

IF: [strength_button 1] > *Button clicked?*
THEN: [wind_overall] > *Set counter*
< input: *10* >

IF: [strength_button 2] > *Button clicked?*
THEN: [wind_overall] > *Set counter*
< input: *20* >

IF: [strength_button 3] > *Button clicked?*
THEN: [wind_overall] > *Set counter*
< input: *30* >

IF: [strength_button 4] > *Button clicked?*
THEN: [wind_overall] > *Set counter*
< input: *40* >

IF: [strength_button 5] > *Button clicked?*
THEN: [wind_overall] > *Set counter*
< input: *50* >

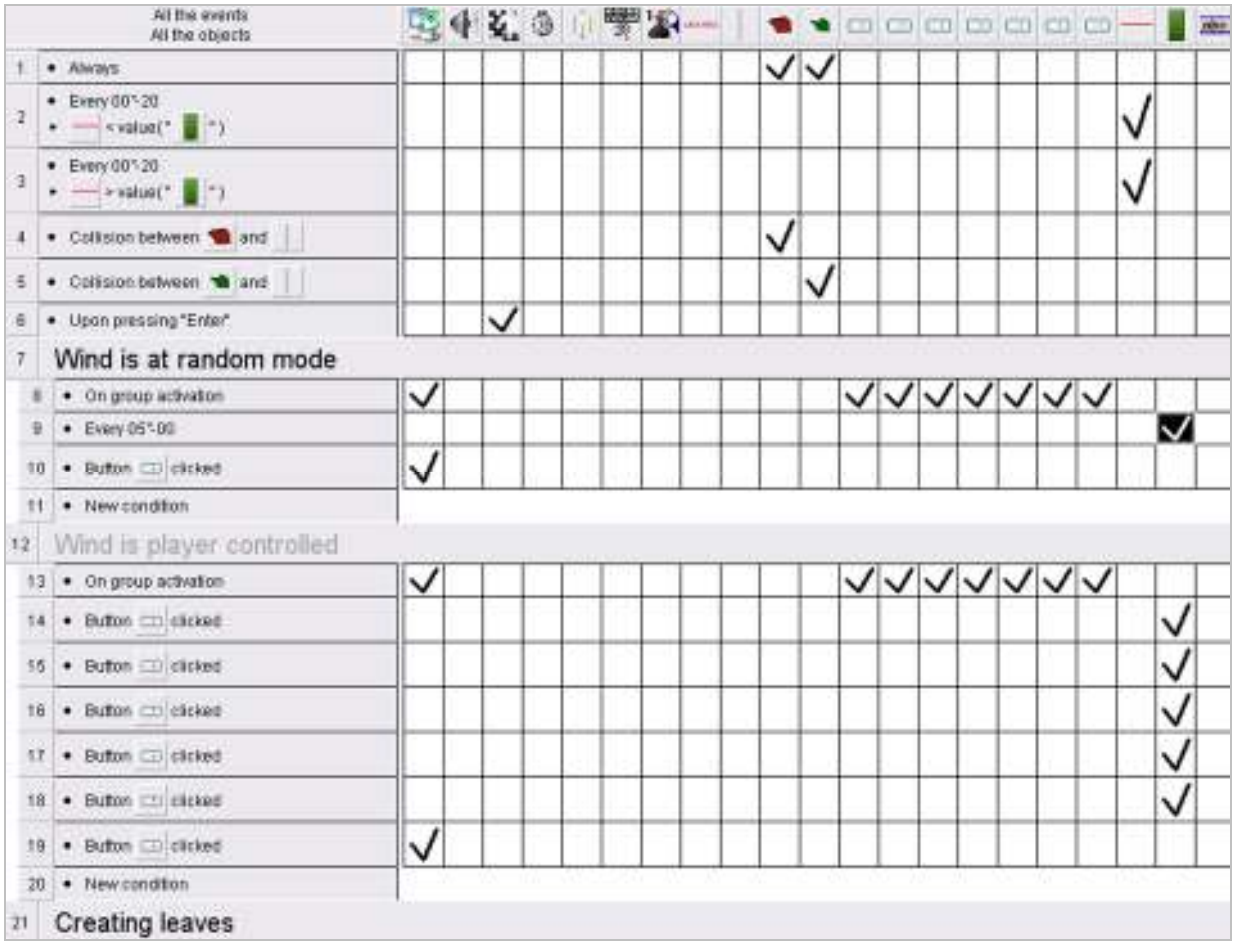
12) And yet another button-related event…Enables the player to change to random mode:

IF: [random_button] > *Button clicked?*

THEN: [Special Object] > Group of events > *Activate*

<< Select the "*Wind is at random mode*" group >>

OK, once again, let's take a look at what we've created by now… Just for orientation:



13) When all those button-clicking events are done, we can leave the "*Wind is player controlled*" group and head to the next one – "*Creating leaves*". This group will contain a bunch of counter-related conditions, spawning our leaves on every *X* amount of time (note that the timer settings can be changed – if you wish, you can input your own time settings):

IF: [The Timer Object] > *Every*

<< Set the timer to *04.03* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *red_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

IF: [The Timer Object] > *Every*

<< Set the timer to *05.31* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *red_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

IF: [The Timer Object] > *Every*

<< Set the timer to *07.01* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *red_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

IF: [The Timer Object] > *Every*

<< Set the timer to *02.65* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *green_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

IF: [The Timer Object] > *Every*

<< Set the timer to *04.12* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *green_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

IF: [The Timer Object] > *Every*

<< Set the timer to *08.45* seconds >>

THEN: [Create New Objects] > *Create Object*

<< Select the *green_leaf* object >>

<< Set the coordinates to x=0, y=0 relatively to the "leaves_generator" object >>

Additional idea: *Autumn comes to town!*

- Would you like to have a bit more leaves flying around? Just change the timer's settings in the events above to 01.87, 02.97, 04.69, 01.06, 02.26 and 04.67 – and then observe! Wow, that's a lot of leaves, ain't it?
- Please, be aware that setting the timer to the numbers given above can make your game work a bit slower on some older computers, or even bring it to a total halt!

OK, when that's done, we can go for the next even… What? What do you mean there's no more left? Oh? Well… Hooray then! That means we're done! We have now created a fully functional wind effect to use in your games and apps… Save your workfile and test the application – it really works! And, I must say, works quite well! Congratulations, lad! The first section of this tutorial is over! It's now time for the second one… And this one will be a bit more challenging for ya'!

Part IV: The challenge

So… You've just finished the first section, haven't you? Sure you did. Wasn't it just a walk in the park? Quite easy? Too easy, perhaps? Like taking candy from a baby? Sure it was. But that's going to change right now, and when I say "change", I mean "change radically". No more step-by-step instructions. No more "visual aids" and helpful screenshots all around you. No, *my dear padawan*, it's time to test your own skills, to take advantage of what you have learned! It's time to show your quality, to reveal the hardcore clicker that's trapped inside you, struggling to get out! Yes, you've got me right – **I want YOU to create the snow effect, without step-by-step guidance!**

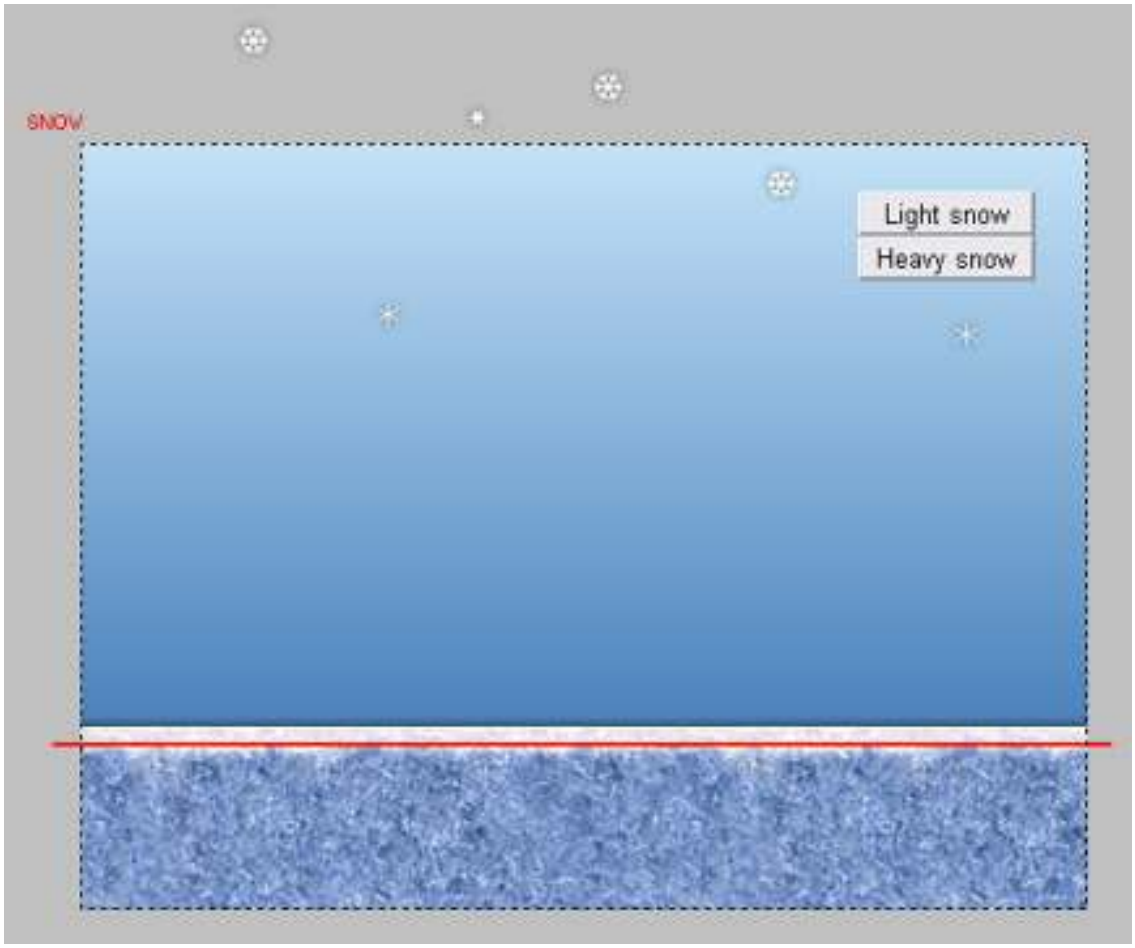
Are you frightened? Yes? Well, not nearly frightened enough! This will be a bumpy road, but I'm pretty sure that you'll manage to get to the finish line. And think of the benefits… There's only one way to learn MMF2's usage faster than by following a tutorial… And that one way is by creating a small project by yourself, with only a couple of tips & tricks shown to you before the start! You've already learned what you need – it's now just a matter of using your skills!

If, by some accident, ya' shall not succeed… If you fail… Well, then nothing's bad gonna' happen, really. If you're stuck with a problem that you cannot solve, just contact me by e-mail and I'll try to help you (you can find my e-mail address at the end of this document). Anyways… Let's not waste any more time, shall we? Prepare yourself! Take a deep breath before the plunge, gather your strength and let's get going!

Preparing the frame, importing the objects

First of all, we need to prepare our frame and import all the needed objects. Create a new frame, then find the "**weather2library.mfa**" file (you know where to look, right?), open it and go into it's second frame (it is named – how convenient! – "*It's snowing!*"). Select all the objects there, copy them into your application and place them in a way, that will make the "*Sky_QB*" quick backdrop perfectly fit the frame – just like you did a couple of pages ago. Got it? Great. Now, create two buttons – a "*Light snow*" button and a "*Heavy Snow*" button. And… You already have all the needed objects around you, buddy!

Here's a little screenshot showing all the objects that you should have by now… And believe me – this is the last screenshot that you'll get in this tutorial, so just enjoy it.



Why are there so many snowflake objects? – you could wonder. Well, that's just because I'd like this effect to look a bit random. Each and every one of those little snowflake objects has a differently programmed movement (they all use the Sinewave movement – but it's not using the same settings for all the objects) – and thus this really can look like a genuine snowfall.

What you need to do…

Your basic objective is to create a stunning snow effect, controllable by two buttons – if the first button is pushed, the snow is set to "light mode", and thus the snowfall is lighter. When you push the "heavy snow" button – you're going to unleash a real blizzard! You should use your knowledge, gained throughout this tutorial, to create this effect with the use of your "*snow_generator*" object, 6 different *snowflake* objects, a "*destroyer_line*" object and previously mentioned two buttons.

What you need to know…

First of all – using event groups is going to be essential. You could also use an Alterable Value or a Counter to control the frequency of the snowfall, but using groups will not only speed up the creation process, but will also make your project better organized and optimized. Also, note that all the *snowflake* objects have been added to a qualifier group "*Bonus*" – this info can prove quite useful, believe me.

Second of all – all the techniques you need to know were already used on the pages of this tutorial. Don't be fooled by the fact that in the *wind effect* frame the *generator* was moving up-and-down, whereas your current one is going to move horizontally – this is still the same thing, and if you'll remember that, you'll succeed in no time. All the snowflakes have already been set up for your convenience (they already have both their movements and fade-out animations selected), but – if you wish – you can reset their settings and set them up yourself, just to experiment a bit with Gwerdy's *Sinewave movement*.

Try to remember that creating a very large number of alpha-channeled objects can really make your production unplayable on older computers. Keep that in mind when selecting the time settings for your snow-creating events – sometimes just a fraction of a second can really make a difference. And don't forget about the "*destroyer_line*" object!

You can find a completed and fully functional snow-effect project within the same archive that this tutorial was in. Just open the **tricks_snow&wind.mfa** file and take a peak at it's second frame. Also, if you're a bit lost, but still not lost enough to mail me for help, you can take a look at the "*Enhancing the feel: It's raining!*" tutorial, where you can find a couple of hints that can come in handy.

So… are you up for a challenge? Sure you are! Good luck, then! You'll do this in no time!

And that's the way the cookie crumbles, folks!

Hope that you found all of this – both the main tutorial and the "homework" part – pretty educative and fun. There's a lot more to MMF2 than just the features presented in this tutorial – and we'll soon venture on yet another expedition to the wonderful world of multimedia-fusioning! And in the meantime – if you haven't done it yet – check all the other tutorials on Clickteam's website! And remember: practice makes perfect!

Thanks for your time and see you again soon!

Cheers!

Koobare
marchewkowy@gmail.com

If you have any questions, suggestions or just need help –
*mail me at **marchewkowy@gmail.com***

Once again: thanks to Gwerdy!

Gwerdy, thank you for your movement pack! It really simplifies the way of using sinewave-based movements in MMF2, and it's a great addition to the tools that an average clicker – like me – has in his hands! Thanks again and keep them comin'! ;)