

the Enhancing **FEEL**

Weather effects:
it's raining!



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

Koobare
marchewkowy@gmail.com

Hi there, all!

Welcome to another one of Koobare's little tutorials, teaching you how to effectively use the best multimedia authoring tool ever - [Multimedia Fusion 2](#) by Clickteam! This tiny tutorial will be a bit different from the ones that I've introduced earlier, as you won't experience the usual "create a full game from scratch" approach here. Nope, this time we'll focus 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 gameplay experience for each player that takes a bite out of your creativity, but also teaching you a few tricks that might come in handy somewhere in your game development process. If you'll find this tutorial useful, we'll perhaps continue with this series later on, supplying you with even more useful tips & tricks for you to use in your own game projects.

The main purpose of this tutorial series is to teach you how to easily make your game more interesting to players – how to enhance your game's "feel" – usually by making your project's world seem more real and "alive". We're going to do this by creating interesting rain, snow and wind weather effects, a night & day system and other eye-catching elements that not only look nice, but also are fully controllable on runtime, allowing the game to smoothly change whether the player will encounter just a few snowflakes or will be struck by a horrific, freezing blizzard. The particular tutorial that you're having in front of you right now will cover just the first out of a whole bunch of topics: with the help of this here tutorial, you're going to learn how to create a truly working rain effect, with it's angle, density and speed being controllable on runtime!

To do this, we're going to use event groups, counters, button objects and a bunch of active objects. You'll see: creating anything with MMF2, even if it seems quite complicated, is always easier that you can even imagine!

To sum this all up – here are some features of the project we're going to complete:

- We're going to create a really good-lookin' rain effect. Memorize as much of this tutorial as you can, since we'll use the knowledge gathered here to create snow and wind weather effects in the next tutorial from this series, coming out soon enough.
- Our rain effect will use the MMF2's built-in Bouncing ball movement and will either rain straight, from top to bottom, or at a slight angle, with both the angle and speed of the rain being fully controllable on runtime (we'll use buttons to control this).
- A few additional, bonus game ideas will be introduced here and there, allowing you to go further with your creativity. I suggest that you run thru the whole tutorial completely

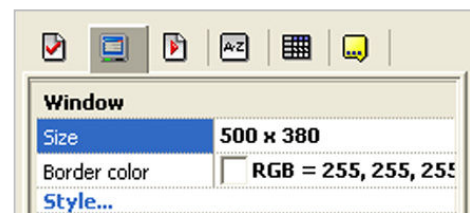
ignoring them at first, and then try to play with them a bit. Remember: practice makes perfect, and without practicing newly learnt stuff on your own, you'll soon forget them!

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

Part I: Setting up the application.

OK, let's have some fun! Open Multimedia Fusion 2, **create a new application** and save it onto your hard drive (remember that it's always a good idea to stash up some copies here and there – I always save my projects to a few different files and even make a solid cd-copy once a week). Go to your application's **properties window** (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). Set the window size to **500x380**. This isn't really necessary, as a standard 640x480 would suit us well too, but I sometimes have this strange urge to work on windows different than the usual 640x480 or 800x600 sizes... And since I'm the captain of this boat (Ha ha! I always wanted to say this! Welcome aboard *USS Tutorial*, trooper!), we'll do it my way.




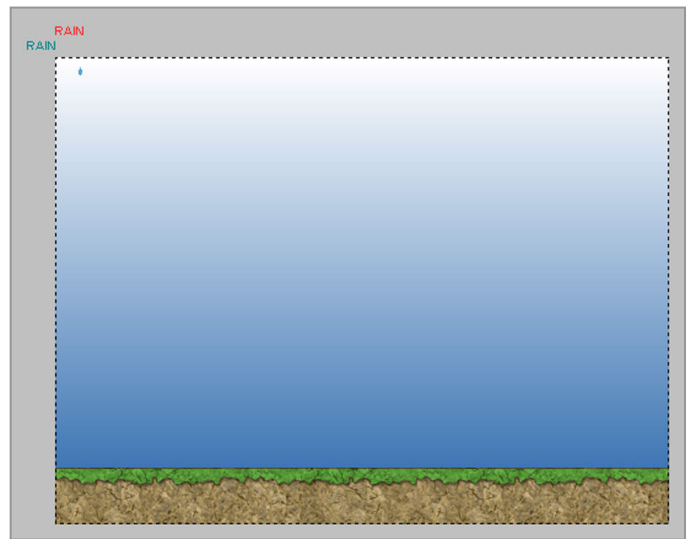
When you're done with that, create our first frame (make it's size identical to the size of our window: **500x380**) and continue to part two of this tutorial.

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

We all know what we want to do now – we want to create a working and nice looking rain effect, right? The only problem is that we don't exactly have anything to work with right now, do

we? Don't be troubled, though. You can find all the needed objects in the same pack this little tutorial was in.

Let's import (or create) all the needed objects. First of all, 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 "**rainlibrary.mfa**" file, which was packed into the same archive as this .PDF tutorial, and open it (load it into MMF2). Got it? Superb. Open the first (and only) frame from the library file. Select all the objects there (hint: press **CTRL+A**), copy them into your application (note: some of the objects 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) and place them in a way, that will make the "Sky_QB" quick backdrop perfectly fit the frame (if you remembered to check the "snap to grid" option off, there shouldn't be any problems with this). If you have any problems with this, just take a look at the visual aid to the right – this is how your frame workspace should look like. Got it? Let's move on, then!

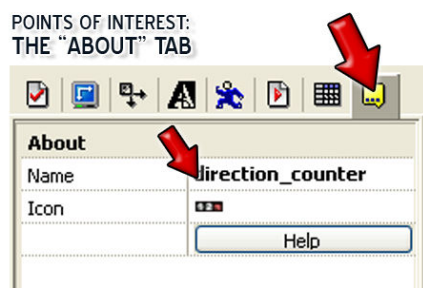
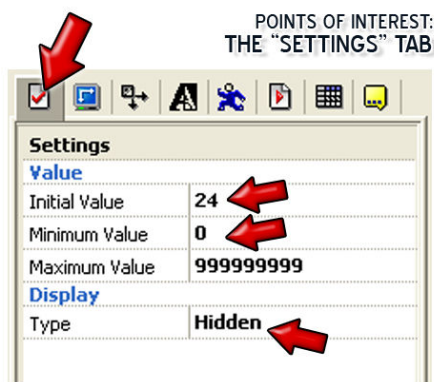


Now, take a look at the imported objects. We've got two quick backdrops here (one for the sky and one for the ground), a "**raindrop**" active object and two actives that will act as our rain generators (named – mysteriously! - "**rain_generator_1**" and "**rain_generator_2**"). The basic idea is to make the rain generators create raindrops that will fall from the sky and vanish after hitting the ground. Doesn't sound that complicated, right? Well, that's because it isn't complicated. We'll add a twist or two to this idea (as it was said at the beginning of this tutorial, both the angle and speed of the rain will be fully controllable on runtime), but after we'll start coding it, you'll notice that it is all as simple as it gets. That's just the way MMF2 functions: enormous power is just a few clicks away!

OK, now, let's create all the extra objects that will come in handy – that would be seven buttons and two counters, to be exact. What do we need them for? Both the buttons and counters will help us control our rain's properties, making it possible to easily change them on runtime. In other words: if you'll want the rain to pour down a bit faster, you'll just have to press the "*faster speed*" button, and it will adjust the speed counter respectively. Of course, this method of

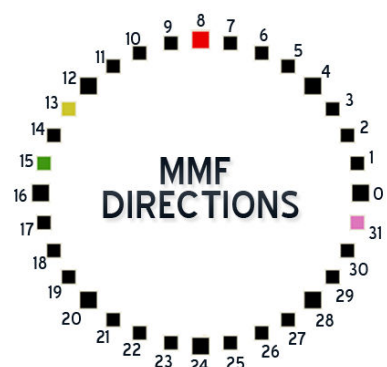
controlling the rain is not meant to be implemented into your games, dear Clicker – this is for tutorial purposes only, just to show you that the rain’s properties can be easily changed on runtime! It’s best to avoid handing over the weather control to the player – he should feel like he’s a part of a living world, not it’s creator (unless, of course, you’re creating a clone of Bullfrog’s *Populous* game), so keep in mind that having the weather changing randomly can be quite an interesting idea! Just like in real life: you never know when it’s going to rain!

Having the introduction and a bit of theory behind us, let’s create those counters and buttons, shall we? **Right-click** on an empty space in your **Frame editor** and select the **“Insert Object”** command from the drop-down menu. When the **“Create new object”** dialog window will appear, scroll down to find the **Counter** object from the object’s list (to make it a bit easier: remember, that if you’ll press the **“C”** key on your keyboard, the list will automatically scroll to the next object beginning with the letter **“c”**), select it, click the **“OK”** button and then click somewhere in the frame to create a new counter object. Voila! We have just created our first counter. Let’s set



it up properly, shall we? Firstly, let’s make it’s display type **“Hidden”** – to do this, open the object’s properties screen (if it didn’t open automatically after creating the counter, right click on the object and select **“Properties”** from the drop-down menu), open up the first tab to the left (the **“Settings”** tab) and select **“Hidden”** from the **“Display type”** list. Set it’s **Initial value** to 24 and **Minimum value** to 0. After that’s done, go to the **“About”** properties tab (the first one from right, the one with the yellow speech bubble icon) and change the counter’s name to **“direction_counter”** (type it in after clicking on the **“Name”** field – use exactly this name, as it is the one that is used in this tutorial). If you need some help – take a look at the visual aid to the left and follow the **“points of interest”**, indicated by the red arrows.

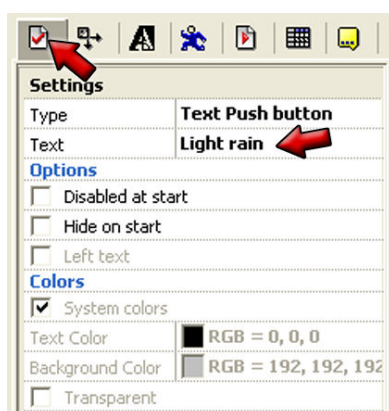
OK, so we’ve got the direction counter set up. Would you like to know what it does? It basically **“tells”** the game at which direction should the raindrops **“fall”** after creation. If it’s **Initial Value** will be set to 24, then – until the counter’s value will be modified on runtime – it will **“tell”** the rain to fall right down. Just take a peak at the image to the right – it shows you all



32 MMF directions in which your objects can move. If you'd like to make an object move at the direction of the red square (precisely up), just select "8" as that object's initial direction. If you'd rather like the object to move in the direction represented by the green square – select "15". The yellow one – "13", the pink one – "31", and so on. Pretty simple, right? Take a look at the illustration above once again – knowing it can be really handy when creating a game with MMF, so try to memorize the way the diagram works and where the direction count starts – "0" is always to the right, "16" is always to the left.

Now, since the **"direction_counter"** is done, let's move on and create the **second counter** – the one that will be responsible for "telling" the game at what speed the raindrops should fall down... Right-click somewhere in your **Frame editor** (try not to click on another object!) and select the **"Insert Object"** command. As soon as the **"Create new object"** dialog window appears, scroll down and select the *Counter* object from the object's list, click the "OK" button and then click somewhere in the frame to create a new counter object. And thus we have another counter in our game/app! Using the tips that I've gave you when we created the first counter, make the second counter's display type **"Hidden"** as well (open up the **"Settings"** tab and select **"Hidden"** from the **"Display type"** list) set it's **Initial value** to **35** and **Minimum value** to **0**. After that's done, go to the **"About"** properties tab and change the counter's name to **"speed_counter"**. Got it? Great! Our counters have been created then, it's time to have a go for the buttons.

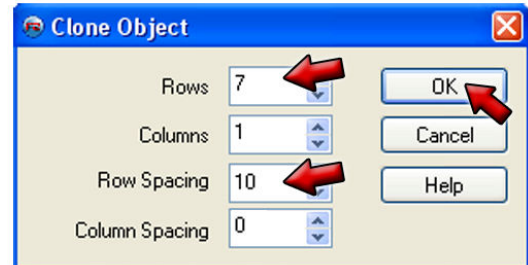
Create a new **Button** object (right-click on an empty space in your Frame editor, select the **"Insert object"** command, select the *Button* object from the object's list, press the OK button and click on an empty space in your Frame editor), enter it's Properties workspace (if it didn't open by itself, just right-click on the object and select **"Properties"** from the menu) and select the



"Settings" tab (it's the first one to the left, it usually opens up by default). Now, change the button's text to **"Light rain"** (just click on the "Empty" text and input **"Light rain"** – if you need visual guidance, check the image to the left). When that's done, move on to the **"About"** tab (first from the left) and change this button's name to **"light_button"**. The last thing to do here: go to the **"Size / Position"** tab (second from the left) and change the button's **X Position** to **384**, **Y Position** to **32**, **Width** to **88** and **Height** to **22**.

Once the first button is created, we'll create 6 more, as there are going to be 7 buttons in general. This can be achieved either by creating them all with the **"Insert Object"** command, or

– which will make it easier for us – by using the “**Clone Object**” one. Cloning is pretty simple – all you need to know about it, is that it creates the specified number of new objects identical to the one you’re cloning – and yet, they are all different objects, so you can control all of them separately: changing something in one of them doesn’t change them all. To clone our button, right-click on it and select the “**Clone Object**” command from the drop-list. A new dialog window should appear. Set the **number of rows** to **7** and the **Row Spacing** to **10** pixels (if you need visual guidance, take a look at the image to the right). Click “OK”.



The screenshot to the right shows what you should have there by now: seven identical buttons in a column, separated by 10 pixels of space. Right-click on the second one of them (counting from the top, it should be currently named “*light_button 2*”), and select the “**Edit**” command (unless you have changed that in MMF’s preferences, you can access the “*Edit*” window by double-clicking on an object). Set this button’s text to “*Heavy Rain*” and click the OK button. After that’s done, right-click on that object again, but this time select the “**Rename**” command and rename this button to “**heavy_button**”. Note that both the “*Rename*” and “*Edit*” commands change properties of the object that can be also changed via the **Properties toolbar** (in the “*Settings*” and “*About*” tabs, to be exact).



Two down, five to go. Using the knowledge that you’ve just gained, set up all the remaining buttons by changing their text and renaming them. Here’s how you should do this (the names below the buttons are what you should rename the buttons to):

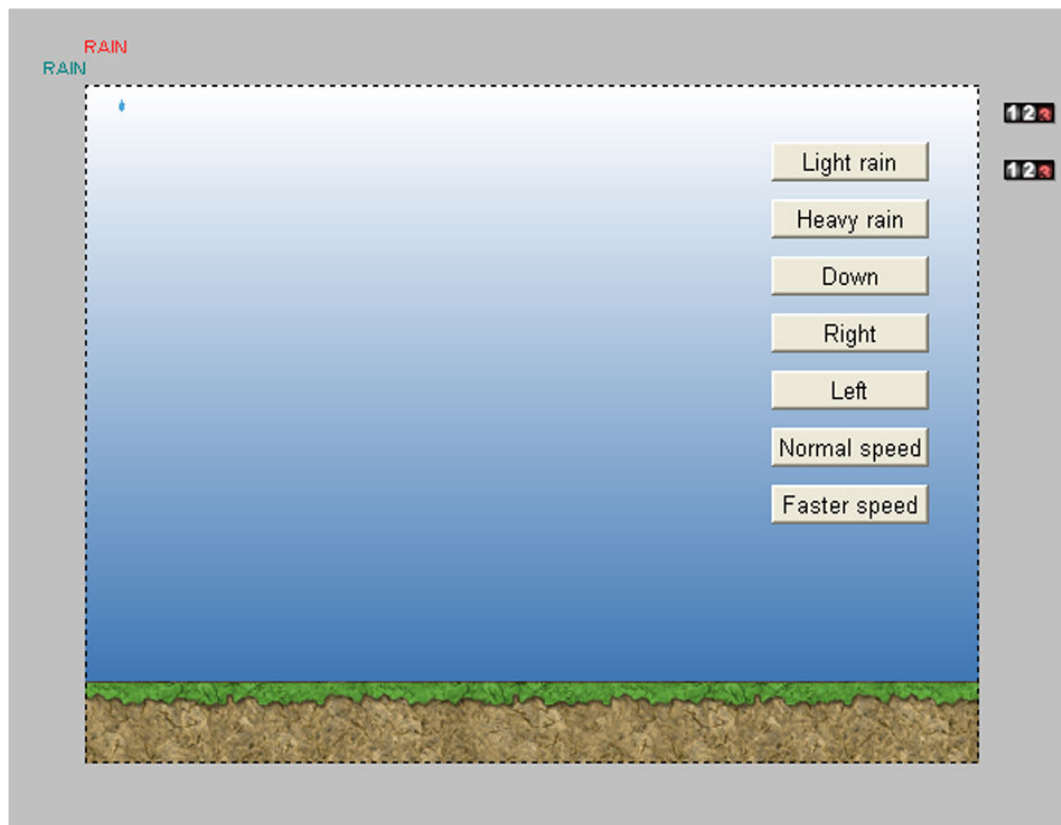


To make it out as words: you should take the third button (counting from the top), set it’s text to “*Down*” and rename it to “*down_button*”. Then, you should take the fourth button from the top, set it’s text to “*Right*” and rename it to “*right_button*”. When that’s done, select another one... And so on, and so on.

You get the picture, right?

When all your buttons are set up correctly, we can continue. If you haven't done that before, you may consider dragging those two counters out of your frame (it's not necessary, but I personally prefer to keep any hidden counters out of the playframe – I usually create them a few millimeters out of the frame's border, just to keep everything well organized).

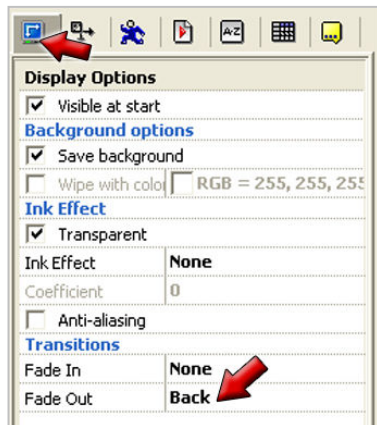
You can take a look at how the frame should look like, here:



OK, we're almost at the coding part. The last thing we have to do before getting there is setting up preferences of all the previously imported objects. It may not be as interesting and fun as coding, but it's a thing that you just have to do to keep everything working properly.

Setting up the “raindrop” object

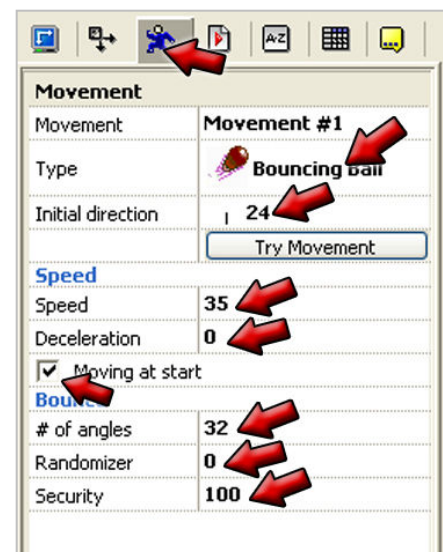
Firstly, select the “**raindrop**” object (it's that little blue thingy – if you'll have any problems with finding it on the frame, just use the objects list, left of your main workspace) and open it's Properties screen (if it won't open by itself, right-click on the object and select “*Properties*” from the drop-down menu). Now, in the “*Display*” options tab, set the **Fade Out** transition to **Back**, at



0.11 second and the **Open (Scrolling)** setting - this isn't really necessary, but will add a nice "splash" effect when a raindrop will touch the ground, and thus be destroyed. If you require visual guidance, check the nifty little image to the left (the same procedure as before: look for the red arrows to point you in the interesting direction).

Using animated transition effects (*fade ins* and *fade outs*) can sometimes make a great visual difference, making your game look smoother, nicer and generally better – and it's a great way to save up time too, since you don't have to draw all those fadings frame by frame, you just pick the fade you're interested in, apply some settings, and – Voila! – Its ready, operational and really nice-looking. The only disadvantage of using transitions is that they can sometimes make your application work a bit slower on older computers – but usually, unless you're working on a medieval PC with bats and spiders living in its interior, the problem shouldn't even exist.

Once you're done with the transitions, go to the **Movement** settings tab (third from the left, the one with the running little man). Select the **Bouncing Ball** movement from the movement type list – don't worry, we won't create a "bouncing rain" of any kind, we're using this movement only because it's pretty easy to configure and you really can do some interesting stuff with this. After selecting the movement type, set the **Initial direction** of the movement to **24 (down)**. Set the movement's **Speed** to **35** (this will be the initial speed of our raindrops, we'll modify this on runtime), **Deceleration** to **0** (we don't want the rain to slow down over time, do we?), set the **Number of angles** to **32**, **Randomizer** to **0** and **Security** to **100**. Make sure that the "**Moving at start**" option is **ON**. If you need any visual guidance, check the image above and to the right.



Setting up the "rain_generator_1" object

When we're done with the "raindrop" object, it's time to set up the rain generators – and all we need to set up when it comes to them, is setting the path movement to the first rain generator. Select "**rain_generator_1**", go to it's properties, select the **Movement** tab (third from the left).

Open the drop-down list and select the “*Path*” movement. Now, click on the “*Edit*” button. The *Path Movement toolbar* should appear:



Using the first button to the left (the one with a single line linking a box and an arrow) select the “*New line*” tool. Click at the **520, -10** coordinates of your frame (they don’t have to be the same) – or click somewhere near those coordinates - to create a path similar to the one shown on this screenshot:



Now, select the first node of the movement (click on the first of the two rectangles that have been created and linked with a path), set the speed – using the speed slider – to 100, the maximum speed value. Do the same thing for the second rectangle too, this time setting the speed to 75. 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.

Got it? Great! We’re done with mangling with the settings, then! It’s time to move on to the fun part – using the Event Editor to script all those events that will make this little app come alive!

Part III: It’s time for some coding!

Koobare’s MMF-to-paper coding system

Open the **Event Editor**. Take a look around – I hope that what you see here is quite recognizable to you. If not – it would be best to get familiar with the MMF2’s users manual or one of my earlier tutorials, “Smelly Claw”. It isn’t really necessary, since all this is pretty self-

explanatory, but - if it's your first time ever with Multimedia Fusion 2 – a quick look at the manual won't hurt you, y'know?

Anyways, whether you know a thing or two about the Event Editor, or not, let me introduce you to my MMF2-to-paper event-recording system, that helped us a lot with the “Glob Wars” and “Fusion Player” tutorials:

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 fancy 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 recognition):

< 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 rain

Create all the events listed here, one after one:

1) Firstly, let's start with the **"Start of frame"** event... This little thingy – as soon as the frame starts - disables the buttons that should be disabled at the beginning of the frame and sets the counters to correct values (this isn't really that necessary, since we set those counters correctly in their properties – you remember that, right? Well, I just like to be certain everything's gonna' be fine, by making sure that the counters will have exactly the value I need... Just in case I've forgot something. Nonetheless, you can skip the last two actions of this event, if you wish):

IF: **[Storyboard Controls] > Start of frame**
THEN: **[down_button] > Disable**
THEN: **[normal_button] > Disable**
THEN: **[direction_counter] > Set Counter**
 < input: 24 >
THEN: **[speed_counter] > Set Counter**
 < input: 35 >

2) Secondly, let's go for the **"Always"** event... This event will make sure that the speed of the raindrop is always set to the **speed_counter's** value, and that the direction of the raindrop is always set to the value of the **direction_counter**. All of this is pretty simple, but still it remains a basis for the rest of our program to work correctly.

IF: **[Special Object] > Always**
THEN: **[raindrop] > Direction > Select direction**
 << Press the "1+1" button to use a calculation >>
 << Click the "retrieve data from object" button >>
 << Select [direction_counter] > Current value >>
THEN: **[raindrop] > Movement > Set speed**
 << Click the "retrieve data from object" button >>
 << Select [speed_counter] > Current value >>

3) And here's our third event, destroying the raindrop when it touches the ground:

IF: [raindrop] > Collisions > Backdrop
THEN: [raindrop] > Destroy

Now, create a new **group of events** (right click on an event number and select *Insert > A group of events*) – name it “**Speed & direction controls**”. Events from sections 4 & 5 (below) should be inserted into this group.

4) Here you have a series of three events (you should insert all of them in the “*Speed & direction controls*” event group), controlling the angle at which the rain falls. If you’ll click on the **down_button**, you’ll set the **direction_counter** to 24, deactivate the *down_button* (so that it cannot be clicked twice – this is purely aesthetical, as I like to disable all buttons that can’t change anything at the moment) and activate the two remaining buttons that control the raindrops’ falling angle. Similar actions are initiated when the player clicks on either the **right_button** or **left_button**, changing the rain’s falling angle to “slightly right” or “slightly left” (directions 25 and 23).

IF: [down_button] > Button clicked?
THEN: [right_button] > Enable
THEN: [down_button] > Disable
THEN: [left_button] > Enable
THEN: [direction_counter] > Set Counter
 < input: 24 >

IF: [right_button] > Button clicked?
THEN: [right_button] > Disable
THEN: [down_button] > Enable
THEN: [left_button] > Enable
THEN: [direction_counter] > Set Counter
 < input: 25 >

IF: [left_button] > Button clicked?
THEN: [right_button] > Enable
THEN: [down_button] > Enable
THEN: [left_button] > Disable
THEN: [direction_counter] > Set Counter
 < input: 23 >

5) And yet another two events (remember to put them inside the “*Speed & direction controls*” group) – this time controlling the speed of the falling rain:

IF: [normal_button] > *Button clicked?*
THEN: [fast_button] > *Enable*
THEN: [normal_button] > *Disable*
THEN: [speed_counter] > *Set Counter*
 < input: 35 >

IF: [fast_button] > *Button clicked?*
THEN: [fast_button] > *Disable*
THEN: [normal_button] > *Enable*
THEN: [speed_counter] > *Set Counter*
 < input: 65 >

It's now time to create two more **groups of events** – name them “**Light rain**” and “**Heavy rain**”. They will be activated and deactivated by clicking on the specified buttons, changing the density of the falling rain. Events from sections 6, 7 & 8 (you can find them all below) should be inserted into the group “*Light rain*”, whereas events from sections 9, 10 & 11 should be inserted into the group “*Heavy rain*”.

6) Create this event inside the “*Light rain*” group. This will make sure that when the “*Light rain*” group is activated, the “*Heavy rain*” one stays inactive, plus that the buttons behave correctly.

IF: [Special Object] > *Group of events > On group activation*
THEN: [light_button] > *Disable*
THEN: [heavy_button] > *Enable*
THEN: [Special Object] > *Group of events > Deactivate*
 << *Select the Heavy rain group* >>

7) OK, here come the events that will control the creation of raindrops when the “*Light rain*” group is active! There are quite a lot of them, although if we would make either some more or some less – nothing would go wrong. Why are there so many of them and why do they have such strange numbers selected? Because this makes them appear more random. Yep – take a look at the rain outside your window, it's not raining by a mathematical expression, it's raining randomly – and that's what we're trying to achieve here (that's also the reason why we have two rain generators – one moving via a path movement, the second one “jumping” around the frame via a random “set X position” event, as shown below). Anyway, telling long story short: you can play a little with these here numbers, change them as you like, it – unless you change them too much – won't affect the overall “feel” of what we're doing here:

IF: [The Timer Object] > Every

<< Set the timer to 01.17 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=0, y=0 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 02.54 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=-13, y=0 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 02.89 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=-2, y=9 relatively to the “rain_generator_2” object >>

THEN: [rain_generator_2] > Position > Set X coordinate

< input: Random(500) >

The event above will not only create a new raindrop at the (-2, 9) coordinates, relatively to the **rain_generator_2** object, but will also – what’s worth mentioning - move the *rain_generator_2* to a random X position (MMF2 will generate a number from **0** to **499** and transport the *rain_generator_2* onto that X position, while that object’s Y position will remain unchanged).

IF: [The Timer Object] > Every

<< Set the timer to 03.21 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=18, y=-3 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 03.87 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=17, y=1 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 04.01 seconds >>

THEN: [Create New Objects] > Create Object

```

    << Select the "raindrop" object >>
    << Set the coordinates to x=4, y=5 relatively to the "rain_generator_1" object >>
IF: [The Timer Object] > Every
    << Set the timer to 04.33 seconds >>
THEN: [Create New Objects] > Create Object
    << Select the "raindrop" object >>
    << Set the coordinates to x=0, y=2 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every
    << Set the timer to 05.11 seconds >>
THEN: [Create New Objects] > Create Object
    << Select the "raindrop" object >>
    << Set the coordinates to x=1, y=3 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every
    << Set the timer to 05.44 seconds >>
THEN: [Create New Objects] > Create Object
    << Select the "raindrop" object >>
    << Set the coordinates to x=-9, y=-1 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every
    << Set the timer to 06.11 seconds >>
THEN: [Create New Objects] > Create Object
    << Select the "raindrop" object >>
    << Set the coordinates to x=-7, y=-3 relatively to the "rain_generator_1" object >>

```

8) Here's the last event that you should insert into the "*Light rain*" group:

```

IF: [heavy_button] > Button clicked?
THEN: [Special Object] > Group of events > Activate
    << Select the Heavy rain group >>

```

OK, that's it when it comes to the "*Light rain*" group – if you wish, you can save your work and take a look at how it works at the current stage of development. When you're done testing – let's move on! And try to remember that all the events listed below should be created inside the "*Heavy rain*" group, not the "*Light rain*" one!

9) Create a new event (notice it's similarity to one of the events that you can find above):

```

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

```

THEN: [heavy_button] > *Disable*
THEN: [light_button] > *Enable*
THEN: [Special Object] > Group of events > *Deactivate*
 << Select the *Light rain* group >>

10) Its raindrop creation time again! But this time the raindrops are going to be created a bit faster, showing the player that it's seriously raining down there, in our game's world:

IF: [The Timer Object] > *Every*
 << Set the timer to 00.42 seconds >>
THEN: [Create New Objects] > *Create Object*
 << Select the "raindrop" object >>
 << Set the coordinates to x=-2, y=9 relatively to the "rain_generator_2" object >>
THEN: [rain_generator_2] > Position > *Set X coordinate*
 < input: *Random(500)* >

IF: [The Timer Object] > *Every*
 << Set the timer to 01.12 seconds >>
THEN: [Create New Objects] > *Create Object*
 << Select the "raindrop" object >>
 << Set the coordinates to x=12, y=6 relatively to the "rain_generator_2" object >>

IF: [The Timer Object] > *Every*
 << Set the timer to 00.35 seconds >>
THEN: [Create New Objects] > *Create Object*
 << Select the "raindrop" object >>
 << Set the coordinates to x=0, y=0 relatively to the "rain_generator_1" object >>

Additional game idea: *it's raining like madness, I tell ya'!*

- Would you like to change that rainy weather into a truly apocalyptic hurricane? Just change the timer's settings in the three events above to 00.02, 00.09 and 00.05, then – while testing the app – click on the "Heavy rain" button. Wow, a flood seems imminent!
- 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!

IF: [The Timer Object] > Every

<< Set the timer to 00.71 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=-13, y=0 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every

<< Set the timer to 01.55 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=22, y=2 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every

<< Set the timer to 01.87 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=-1, y=2 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every

<< Set the timer to 02.28 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=18, y=-3 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every

<< Set the timer to 02.53 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=5, y=3 relatively to the "rain_generator_2" object >>

IF: [The Timer Object] > Every

<< Set the timer to 02.71 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=17, y=1 relatively to the "rain_generator_1" object >>

IF: [The Timer Object] > Every

<< Set the timer to 03.11 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the "raindrop" object >>

<< Set the coordinates to x=4, y=5 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 03.65 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=0, y=2 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 03.86 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=-9, y=-1 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 04.15 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=1, y=3 relatively to the “rain_generator_1” object >>

IF: [The Timer Object] > Every

<< Set the timer to 05.12 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the “raindrop” object >>

<< Set the coordinates to x=-7, y=-3 relatively to the “rain_generator_1” object >>

11) Here's the last event that you should insert into the “Heavy rain” group and the last event we're going to create, as well. Yep, it's only this and we're done here!

IF: [light_button] > Button clicked?

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

<< Select the Light rain group >>

And that's all, folks!

We've just created a professional, good-lookin' rain effect that you can use in your games or applications any time! And yes, you can use those little raindrop graphics too – they're not of the highest quality, but can come in handy if you don't want to waste any time on drawing some simple drops of water.

Perhaps you should try to do this tutorial all over again, but this time without this little tutorial to guide you... Maybe even try to expand this project's concept a bit? Here's an additional idea that you might find interesting...

Additional game idea: *toxic poison from the skies*

- Here's an idea that can give one of your games an additional twist: what if the world of your game got contaminated and there's *poisonous rain* pouring outside? Every drop that falls on the player brings him one step towards mutation and a big, red "*game over*" sign. Player's objective? While he's outside, he's got to run from one cover to another (those covers could be slowly eaten away by the contaminated rain) and fight off some hideous mutants in the same time.
- A simple trick that can get this idea going: every time a drop of green rain touches the Player (these two objects collide), add 1 to the *Alterable Value A* of the Player object. When the *Alterable Value A* of the Player object reaches 100 – destroy the Player, show the "*game over*" sign. Every time the Player collides with a bottle of water or something like that (and thus is able to wash the poison of his skin) – subtract 50 from the *Alterable Value A* of the Player object. If the player finds a shower – set the *Alterable Value A* of the Player to 0.

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*

Coming up next!

Hope that you enjoyed this episode of “*Enhancing the Feel*” tutorial series, because more tutorials are already on their way! Here you can take a peak at the *wind & snow tutorial*, that will be soon available to download from Clickteam’s Learning Resources center!

