

THE GHOST HUNTER

A MULTIMEDIA FUSION 2 TUTORIAL



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.

Koobare

marchewkowy@gmail.com

Welcome to yet 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 beginner intermediates, people who have done by now their share of novice-level tutorials (you should read the “Interface Guide”, “Glob Wars”, “Smelly Claw” and “Risky Waters” tutorials before doing this one). Take a look at this simple lesson guide to help you decide which tutorials should be played with first (note that this is just a suggestion, and you may want to skip a tut or two, or just do them in a different sequence, based on your individual preferences):

MMF2 Interface: Interface Guide + Image Editor Guide	<i>First time with MMF2</i>
Basics: Smelly Claw tutorial	<i>Beginners</i>
Game tutorials: Glob Wars and/or Risky Waters	<i>Beginners</i>
Game tutorials: Castle Defender and/or Space Corsair	<i>Beginner-intermediates</i>
You are here → The Ghost Hunter	<i>Beginner-intermediates</i>
Next: You've Got Spacemail!	<i>Beginner-intermediates</i>
Next: Save & Load tutorial	<i>Intermediates</i>
Next: The BiGciTY Downloader	<i>Intermediates</i>

In this tutorial we will create a nice-looking “point and shoot” type of game, in which the player shoots down incoming ghosts, armed with nothing more than his trusty shotgun. The main idea behind this game is pretty basic: there you are, scrolling through the playfield, waiting for those nasty ghouls to sniff you out and fly towards you, making them vulnerable to fire. You gotta’ aim with your mouse and shoot them down (you pull the trigger by left-clicking with your mouse, right-clicking reloads the gun), before they get a bite out of your neck. The problem is that these ghosts just keep coming and coming, up until you shoot exactly one hundred of them, which – for some unknown, arcane reason (most probably due to a poor scenario) – makes them suddenly less violent and they just fly away to Brazil, or something, to get a nice ghost-tan. Anyways, you know the drill: just point, click, and let your exorcizing shotgun shells do the rest.

“If there's somethin' strange, in your neighborhood...”

Bobby Joe MacBobson is just one of those people, who always find themselves in the wrong place, at the wrong time. At the age of fifteen he managed to accidentally discover a gory cult of the Bloodboilers, devilish monks that – as it so often happens – decided to build their own temple in his basement. Although it all ended in a bloodbath, with SWAT teams storming the

labyrinth of pickle jars and canned preserves, Bobby managed to escape from under the ritual hatchet, surprisingly unharmed. Some could say that he's a lucky guy, even if his luck often shines in the most bizarre and gloomy situations...

Three years after the developments with the Bloodboilers, Bobby had the misfortune to be abducted by beady-eyed, hippie-haired aliens from the Zulrax Optimum galaxy – which later accidentally crashed their flying saucer into Mount Rushmore, knocking out all of Teddy Roosevelt's teeth (and everyone who know history, knows that no one should cross Teddy). Once more, Bobby was the sole survivor of these events, as it seems that the abductees' cells were the only place the aliens have decided to put airbags, for some reason. Anyways, it wasn't long before he wandered on into trouble once more – this time discovering a lizardmen conspiracy on the East Coast. Two years after that – the famous Big Foot massacre, then the Magneto wars, then zombies, then the meteorite, then the Reapers... No matter what and where, if something weird was happening, Bobby was there, in the middle of it all.



At this point of our story, you could think that the universe gave Bobby a brake – he had to wrestle Big Foot, after all, and that was one heck of an excruciating ride... Nah. It happened

again. Bobby was just minding his own business, hunting some deer or something, when a pack of ghosts appeared from nowhere, booing and wooing their way around the forest, searching for living souls, to scare, behead and devour.

Ghostbusters 101

Time for a short revision of what we know about our project, just a quick check-up before moving on to the "know thy objects" part of this tut. First of all: all the player has to do is to point his mouse (or rather the crosshair) at an incoming ghost and click his left mouse button to shoot. Bobby wields a single-barreled shotguns with a shell capacity of three, so after shooting three times, he'll have to reload, using the right mouse button (actually, he can reload anytime in the game, even if he still has some rounds left inside the barrel). One more thing – these are shotgun shells, not some wacky proton energy beams, so don't expect the ghosts to fall dead after just one shot... Some will do exactly that (even if I'm not entirely sure that a ghost can be

anymore “dead” than it currently is...), others will not, as their constitution (as we, RPG nerds, like to call it) is generated randomly when they are introduced to the playing field and then stored in one of their Alterable Values. Be careful not to give them a chance to come too close, though, since they will attack once they are closed enough, once you’re within their reach (actually, it’s a simple system using MMF2’s in-built scaling, but we’ll get to that a bit later). Hmm, anything more? Guess not. Move on forward, to the first part of this tut!

- ☐ **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

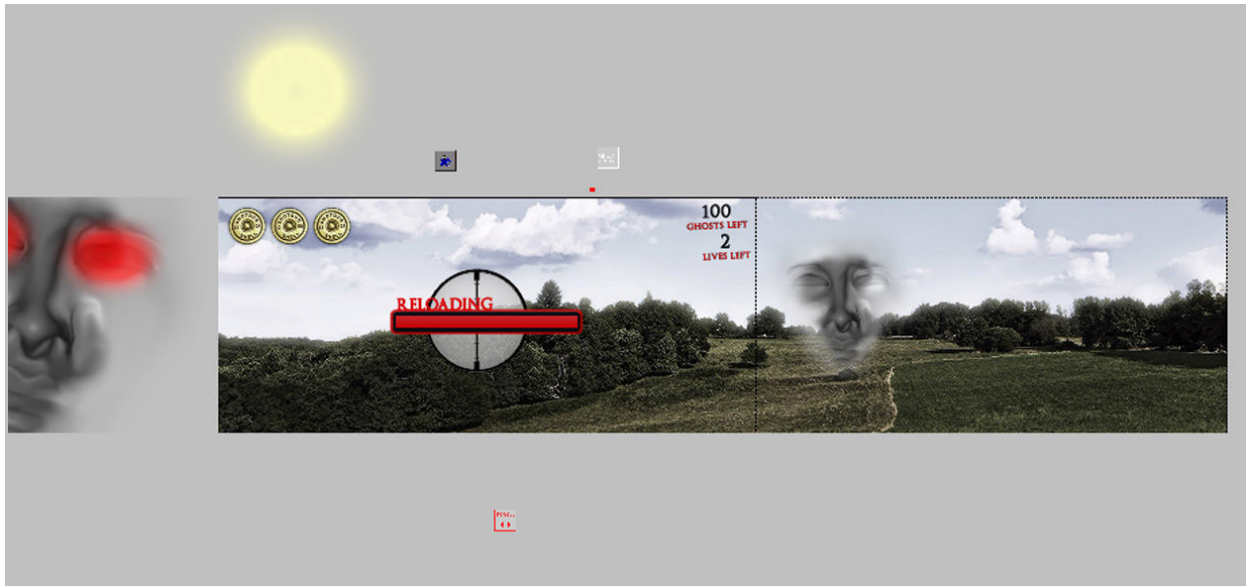
Alternative e-mail: koobare@panzerflakes.com

- ☐ **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. Yaay!

Part I: Taking a look at our application

Ready or not, here we go... Open up your copy of Multimedia Fusion 2, and load the starter file for this tutorial, named **TheGhostHunter-starter.mfa** and most probably located in the same directory as this here PDF file (since they were packed into the same archive). We’re going to work on a previously created basis, an application that I’ve prepared earlier for our mutual benefit, so to speak. Thanks to this neat little trick you won’t have to set up everything

yourself – just watch, take notes if you have to and, well, you know, learn. Here, just to get you started, a nice screenshot of my Frame Editor, with all the objects scattered around in what seems to be unorganized, uncontrollable chaos (but, actually, it pretty organized). We'll get to this in a moment, but it's here to give you a taste of things to come:



First things first: if you won't mind taking a look at our application's preferences, you'll see that I've set the Window Size to **800x350**, to make it appear somewhat "panoramic", definitely bigger horizontally than vertically. What's more to see here: there are **no Global Values** in this app – a rare occasion in my tutorials, since I usually like to use them wherever applicable – as all the game data is stored either in counters or Alterable Values.

Now, open up the first and only frame in our application and check out its preferences... It's size is set to **1500x350**, it has a nice **Fade-in** (2 sec 14") and a **Fade-out** (3 sec 84") frame transitions (both set to "Fade" animation), and has a bunch of objects scattered all around – objects that we're going to check out in a sec.

Interested in one game type in particular? Would like to learn about something that hasn't been covered in any of the released tutorials yet? Got an idea that could interest other tutorial-readers? Just drop me an e-mail!

Contact me at: marchewkowy@gmail.com

or write to the auxiliary address: koobare@panzerflakes.com

Part II: Know thy objects

Since you've already opened our frame and glanced around, let's take a more thorough examination of what's laying around. Time to check out all those little details... Below you can find an alphabetical list of all the objects, with their short description, their purpose and some characteristic properties all written down...

Please note: most of the objects use alpha channels, a feature that is unavailable in Games Factory 2 (TGF2 users should use basic library objects or create their own graphics instead – I generally would advise you to upgrade to MMF2 as soon as possible).

Here we go... See you on the other side, brotha':

Ammo	<p><u><i>Lives Object used as a counter for our player's ammo.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Well, we need some kind of a way to check how many shells does the player have... Sure, we could have just used a counter, but the Lives Object makes it a bit easier on us when it comes to displaying a sequence of an image, side-to-side.</i></p> <p><i>If this object's value (actually, it's a player's value, but let's not get into details here) is greater than 0, left-clicking anywhere within the play area will shoot the rifle. If it's lower or equal 0, all we get is a "click" sound, indicating that there are no more shells inside the barrel, and the player has to reload to be able to shoot once again. Anyways, all you have to remember about this object is that the "Lives of Player 1" properties is in fact our player's shotgun ammo.</i></p>
Background Image	<p><u><i>The background for our game.</i></u></p> <p>WHY IT'S HERE?</p> <p><i>Its purpose is pretty basic... and pretty obvious. This is the image that serves as a background for our game – nothing more, nothing less.</i></p> <p><i>Having said that, never underestimate the importance of a proper, visually attractive backdrop image – this is basically the main thing the player sees throughout the whole level, so it should be at least interesting, if not nice-looking. Sure, our players should be concentrating on their aim and those horrible flying ghouls, but don't fool yourself: the background image is an important part of the whole game, so be sure that it has a proper contrast, isn't too dark, and... well, just isn't ugly.</i></p>

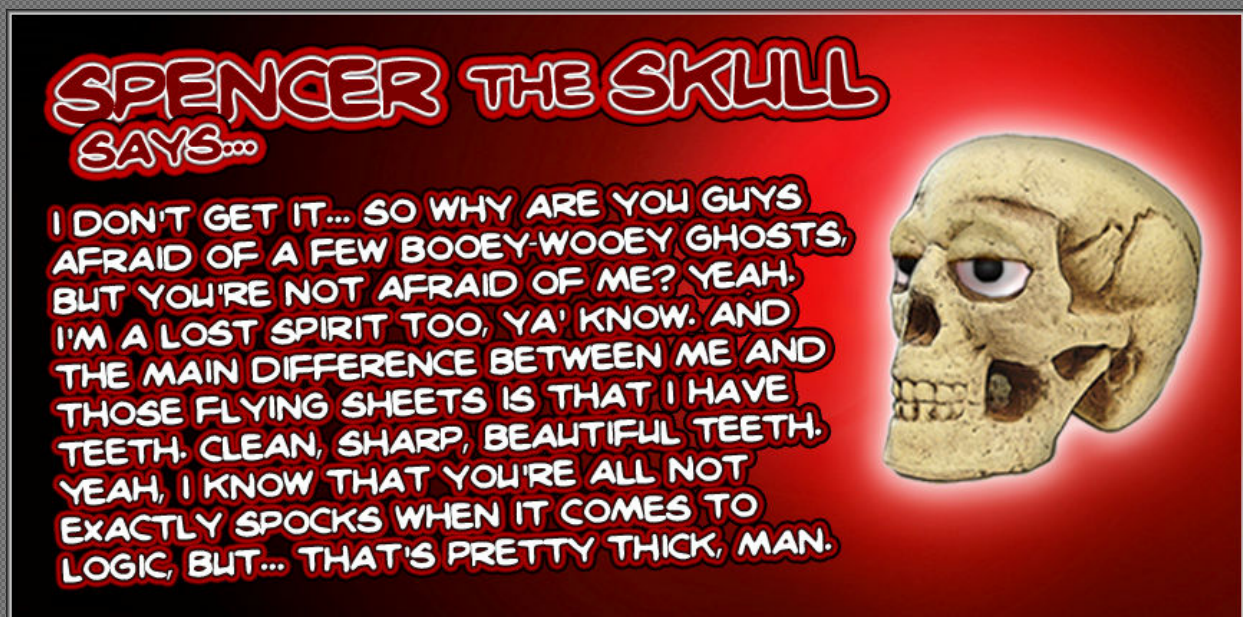
Clickteam Movement Controller	<p><u>Helps us control movement of our ghastly ghosts,</u></p> <p>WHY IT'S HERE?</p> <p>Most of Clickteam's in-built movements can be easily controlled from the Event Editor itself, by choosing certain actions, conditions and expressions from the "Movement" sub-menu of an Active object... But "most" doesn't exactly mean "all" now, does it? At this time some movements (including the Circular Movement, which we use for our ghosts) need to be accessed by an additional object, the Clickteam Movement Controller, which provides you with all the options you might need during your design process. In this application we will use this object for one thing only: to change the centre of our Circular Movement on runtime.</p>
Counter - Ghosts	<p><u>A counter that shows us how many ghosts do we have to shoot before the end of the game.</u></p> <p>PROPERTIES</p> <p>It is set as a Numbers counter, with a fixed number of digits (set to 3). Its Initial and Maximum Values are both set to 100, its Minimum Value is set to 0. Also, note that it is displayed as an active (the "display as background" option is off) and doesn't follow the frame.</p> <p>WHY IT'S HERE?</p> <p>'Cause we need an objective for the player. And since this is a pretty straight-forward and simple game, the easiest objective here would be to destroy each and every ghost that haunts these here forest – and I just declared that there are in fact exactly 100 ghosts floating around.</p> <p>Note that as soon as all 100 ghosts are destroyed, the game jumps to the next frame – which we won't construct in this tutorial, but you may want to do it by yourself. Just copy this frame, add a new background, change some settings and preferences (for example, give more lives to the ghosts or change their number) and – voila! – you'll have frame number 2!</p>

„Are you the keymaster?"
„Not that I know of."
(She slams the door in his face)
(Venkman knocks again. The door opens)
„Are you the keymaster?"
„Yes, actually yes. Actually, I'm a friend
of his, he asked me to meet him here."
- from which movie does this scene come from?

<p>Counter - Lives</p>	<p><u><i>A counter that shows us how many lives does the player still have.</i></u></p> <p>PROPERTIES</p> <p><i>It is set as a Numbers counter. Its Initial and Maximum Values are both set to 2, its Minimum Value is set to 0. It is displayed as an active (the “display as background” option is off) and doesn’t follow the frame.</i></p> <p>WHY IT’S HERE?</p> <p><i>Because the player should know how many lives in the game does he have – and, boy, surely he doesn’t have too many of them. Two lives, to be exact, which means that once Bobby feels that second ghost’s bite on his neck, it’s already too late.</i></p>
<p>Crosshair</p>	<p><u><i>An active object that follows the mouse.</i></u></p> <p>ALTERABLE VALUES</p> <p><i>This object has a single Alterable Value set from the beginning: it’s named Reloading, it is set to 0 and it helps define whether or not is the player currently in the process of reloading his shotgun.</i></p> <p>HOW DOES IT WORK?</p> <p><i>On every loop the Crosshair object is positioned at the X,Y coordinates of our mouse cursor (which is hidden, by the way), and... that’s it. It’s just here to give us a nice “hey, I’m really looking through a rifle’s scope!” feel, alongside with the Lens object. Oh, and it also hosts the aforementioned “Reloading” Alterable Value, that is crucial for our game.</i></p>
<p>Ghost</p>	<p><u><i>The enemy – one of the player’s unearthly antagonists.</i></u></p> <p>ALTERABLE VALUES</p> <p><i>Each “Ghost” active is created with two predefined Alterable Values: Health (which represents the Ghost’s energy and is set randomly to a number between 1 and 4 at the time of his creation) and Countdown (which is set to 25 and is basically used as a countdown timer, defining when exactly should the enemy attack the player).</i></p> <p>MOVEMENT</p> <p><i>Circular Movement, with Angular Velocity of 200, Radius of 55, Starting Angle of 0, Spiral Velocity of 0, Minimum Spiral Radius of 4 and Maximum Spiral Radius of 510.</i></p> <p>HOW DOES IT WORK?</p> <p><i>If there are less than 3 Ghosts at the same time in the playing zone, our game spawns a ghost at a random location (every 3 seconds, or – if there are no more foes in the area – immediately), setting his Health to Random(3)+1 (which means that our enemy has a Health value of 1, 2, 3 or 4), setting his Countdown to 25 and then releasing him upon the player, in a scale of 0.1 (since he is “in the distance” and has to swirl towards the player, gaining 1% of his scale with each passing 3/10 of a second).</i></p>

	<p>On each passing second 1 is subtracted from the Countdown Alterable Value, and when it reaches 0, the Ghost attacks (the only way around this is to shoot the Ghost before it reaches you, with each of your shots subtracting 1 from your foe's Health). As soon as the specter makes his blow, he disappears, turning into a cloud of stink (it has a short Fade-out animation, set to "Fade" at 0.90 of a second).</p>
--	---

<p>Ghost attack</p>	<p><u>Active object used as a special effects gimmick.</u></p> <p>WHY IT'S HERE?</p> <p>It's a simpler object than it's "big brother", mentioned above. This object appears for a few seconds every time our player gets hit by one of the Ghosts – as it is basically a big, scaled-up version of the Ghost's head, with colored-up, red eyes and an open mouth, it's just a great way to tell the player that he's just been attacked, without spraying pixel blood all across the screen. Used solely as a graphical gimmick, it has no other functionalities and is destroyed once it has played through its single-frame animation. Oh, and it's got a short fade-out transition too!</p>
----------------------------	--



<p>Ghost creator</p>	<p><u>A small active object which helps us with our game.</u></p> <p>WHY IT'S HERE?</p> <p>Because we needed an object to "spit out" these Ghosts from time to time, in an organized and simplified manor. This object jumps from position to position every now and then (exactly: every time the conditions are met to create a new ghost), creating those darn ghouls on the way. Note that it chooses a position of $\text{Random}(1320)+120$ for its X coordinate (120-1440 pixels) and $\text{Random}(120)+120$ for its Y coordinates (which means that its Y is from the range of 120-240 pixels).</p>
-----------------------------	---

<p>Gun blast</p>	<p><u>Another special effects gimmick – this time a “gun blast” effect.</u></p> <p>WHY IT’S HERE?</p> <p>Because if there’s no flash after pulling the trigger, it means you’re shooting duds, and not very realistic looking either. This object is created at the center of the crosshair every time our player shoots his shotgun, and is also quickly disposed of, once it’s two-frame animation ends.</p>
<p>Lens</p>	<p><u>The Lens object – a nice-looking, crucial part of our scope.</u></p> <p>WHY IT’S HERE?</p> <p>Because we want our shotgun’s scope to look just a tad realistic, with the scope’s lenses distorting the view a bit, don’t we? This object is always positioned at the mouse cursor’s coordinates, hidden behind our crosshair, so it looks like the crosshair has been painted on top of the lens. Basically I’m using the same default settings here as I used in “The Mystery of Paris”, so don’t get over-excited about this one. Yet, somehow I find this default lens really satisfactory when it comes to all scopes and magnifying glasses, so let’s just leave it as it is and go to the next object...</p>
<p>Reloading counter</p>	<p><u>A horizontal bar counter, used to visualize how much time does it take to reload the player’s shotgun.</u></p> <p>PROPERTIES</p> <p>It is set as a horizontal bar counter, counting from left, with a vertical gradient fill type (set to red). Its Initial and Minimum Values are both set to 0, its Maximum Value is set to 150.</p> <p>WHY IT’S HERE?</p> <p>Well, because we really need to give the player some tips on how much time will it take him to fully reload his gun.</p> <p>HOW DOES IT WORK?</p> <p>Every time the player hits his right mouse button, the reloading sequence is initiated – basically, the game just adds 1 to this counter on every loop, up until when it reaches the value of 150, which means that the shotgun has been loaded and can shoot again. Note that this counter isn’t displayed as background and neither does it follow the frame.</p>
<p>Reloading frame</p>	<p><u>Just a nice graphical frame on which the reloading counter is placed.</u></p> <p>WHY IT’S HERE?</p> <p>Active object that we place behind the reloading counter. Well, to be honest, it doesn’t really have to be in our game, but somehow I think that it adds a nice touch to our interface, just improves the overall look of our app. It doesn’t follow the frame and it’s “Visible at start” option is turned off and its always brought to front, just behind the reloading counter.</p>

Scrolling helper	<p><u>One of those object's, that are being used to control the game.</u></p> <p>WHY IT'S HERE?</p> <p><i>This object basically points to the place in which the "camera" of our game should be centered upon. It's position is determined with a simple set of events that you will create later on, in the programming part of this tut.</i></p>
Shooting mark	<p><u>Another invisible "helper", your aim's detector.</u></p> <p>WHY IT'S HERE?</p> <p><i>A small Active object that stays invisible for the whole game and helps us determine whether the player shot the ghost or not. It is always placed at the X,Y coordinates of the mouse cursor.</i></p>
Text - Ghosts left	<p><u>A "Ghosts left" text.</u></p> <p>WHY IT'S HERE?</p> <p><i>Actually, it's a small, simple Active object, which just happens to have "Ghosts left" written all over it. It doesn't follow the frame and is placed right under the "Ghosts left" counter.</i></p>
Text - Lives left	<p><u>A "Lives left" text.</u></p> <p>WHY IT'S HERE?</p> <p><i>The last object in our frame, exactly the same as the one above, but with a different text written on it: this time it's "Lives left".</i></p>

Part III: Time for a bit of programming.

It's time for my favorite part of each and every tutorial! Save your project (always remember to save it from time to time, and turn the autosave option on in MMF2's preferences – it's a life saver!) and then open the **Event Editor**. If you're new to my tutorials (or at least would like a little reminder), let me introduce you to the event-recording system that I use. If you know it already – just skip this frame below and quickly move on to the coding part:

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

Seems simple, right? Well, that's just because IT IS simple. All the conditions are marked in red, 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 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...

Let's dive into the MMF2 scripting!

1) Let's start off with the conventional "**Start of frame**" event, which I usually create at the very beginning of the events list. This event – triggered when someone starts our game – will hide the original Windows mouse pointer (while the mouse is above the game's frame), will play our chosen background music (looping it, of course) and then will rescale the one "Ghost" object that is already within the frame to a smaller size (pretending that it's in the distance):

IF: [Storyboard Controls] > *Start of frame*
THEN: [The Mouse Pointer and Keyboard] >> *Hide Windows Mouse Pointer*

THEN: [Sound Object] > Samples > *Play and loop sample*
<< Choose the “trip_zone” music loop, loop it 0 times, which means infinitely >>
<< If you can’t find this sample on your MMF2’s Bonus Disc, use another music >>
THEN: [Ghost] >> Scale / Angle >> *Set scale*
<< set scale to 0.1 >>
<< set quality to 1 >>

And with just a few clicks we’ve got our first event up and ready!

2) Time for our second event, this one will be a bit longer and just a tad more complicated... It is based on an “always” condition, which means – as you already know – that each and every action listed here is carried through on each and every loop of our app (note that the “bring to front”-events sequence is important):

IF: [Special Object] >> *Always*
THEN: [Crosshair] >> Position >> *Set X Coordinate*
<< Type in: *XMouse* >>
THEN: [Crosshair] >> Position >> *Set Y Coordinate*
<< Type in: *YMouse* >>
THEN: [Lens] >> Position >> *Set X Coordinate*
<< Type in: *XMouse* >>
THEN: [Lens] >> Position >> *Set Y Coordinate*
<< Type in: *YMouse* >>
THEN: [Storyboard Controls] >> Scrolling >> *Center window position in frame*
<< center position to 0,0 relative from the [Scrolling helper] >>
THEN: [Shooting mark] >> Position >> *Set X Coordinate*
<< Type in: *XMouse* >>
THEN: [Shooting mark] >> Position >> *Set Y Coordinate*
<< Type in: *YMouse* >>
THEN: [Lens] >> Order >> *Bring to front*
THEN: [Crosshair] >> Order >> *Bring to front*
THEN: [Ammo] >> Order >> *Bring to front*
THEN: [Counter – Lives] >> Order >> *Bring to front*
THEN: [Counter – Ghosts] >> Order >> *Bring to front*
THEN: [Text – Ghosts left] >> Order >> *Bring to front*
THEN: [Text – Lives left] >> Order >> *Bring to front*
THEN: [Reloading frame] >> Order >> *Bring to front*
THEN: [Reloading counter] >> Order >> *Bring to front*
THEN: [Shooting mark] >> Order >> *Bring to front*

3) Now, before we march onto the next event, let's set up all the **Event Groups** that we'll need in our event list... Using groups can be very, very useful and is basically a good idea if your app has more than just a few events... It keeps your event list tidy, it helps you find the right event when debugging and – thanks to the option to switch whole groups of code off with a single event – it enables you to optimize your code (which is pretty important in bigger, more complicated games and apps). Anyways, create five main groups beneath the second event and rename them to **Game conditions**, **Scrolling**, **Shooting**, **Reloading** and **Ghosts**. Now, open up the **Ghosts** group and create these three sub-groups inside: **Ghost creation**, **Ghost behavior** and **Ghost attack**. Close this group, make sure that all groups are active since the start of the frame, and then open up **Game conditions**, since this is the group we'll be using in a second. Now, once that's done, let's move to the next event...

4) Create these two events inside the **Game conditions** group:

IF: [Counter – Lives] >> *Compare the counter to a value*

<< *compare whether it is Lower or equal 0* >>

THEN: [Lens] >> *Destroy*

THEN: [Ammo] >> *Destroy*

THEN: [Reloading counter] >> *Destroy*

THEN: [Reloading frame] >> *Destroy*

THEN: [Text – Ghosts left] >> *Destroy*

THEN: [Text – Lives left] >> *Destroy*

THEN: [Counter – Lives] >> *Destroy*

THEN: [Counter – Ghosts] >> *Destroy*

THEN: [Gun blast] >> *Destroy*

THEN: [Crosshair] >> *Destroy*

THEN: [Storyboard Controls] >> *Restart the application*

So... Why exactly do we need to destroy all those objects before restarting the app? It's not really necessary, I just thought that it would look nicer if – during the fade out of our game, once the player has no more lives – all that the player could see before the restart would be the background, some ghosts in the distance and a big attacking ghastly face, with no interface whatsoever, no counters, ammo or crosshairs... Now, here's the second event that should go inside the same group:

IF: [Counter – Ghosts] >> *Compare the counter to a value*

<< *compare whether it is Lower or equal 0* >>

THEN: [Storyboard Controls] >> *Next frame*

Here's what we have right now (notice that it doesn't have to look identical):

[illegible]

5) Now, close the **Game conditions** group and open up the next one – the one named **Scrolling**. Create this two events inside to set up our scrolling routine:

IF: [Crosshair] >> Position >> Is getting close to the window edge?

<< Set the distance to 50 pixels >>

IF: [Scrolling helper] >> Position >> Compare X position to a value

<< check if *X position* is *Lower or equal* than 1120 >>

IF: [Scrolling helper] >> Position >> Compare X position to a value

```
<< check if X position is Lower than X( "Crosshair" ) >>
```

THEN: [Scrolling helper] >> Position >> Set X Coordinate

```
<< Type in: X( "Scrolling helper" )+4 >>
```

And here's the second event that goes into **Scrolling**:

IF: [Crosshair] >> Position >> Is getting close to the window edge?

<< Set the distance to 50 pixels >>

IF: [Scrolling helper] >> Position >> Compare X position to a value

```
<< check if X position is Greater or equal than 380 >>
```

IF: [Scrolling helper] >> Position >> Compare X position to a value

```
<< check if X position is Greater than X( "Crosshair" ) >>
```

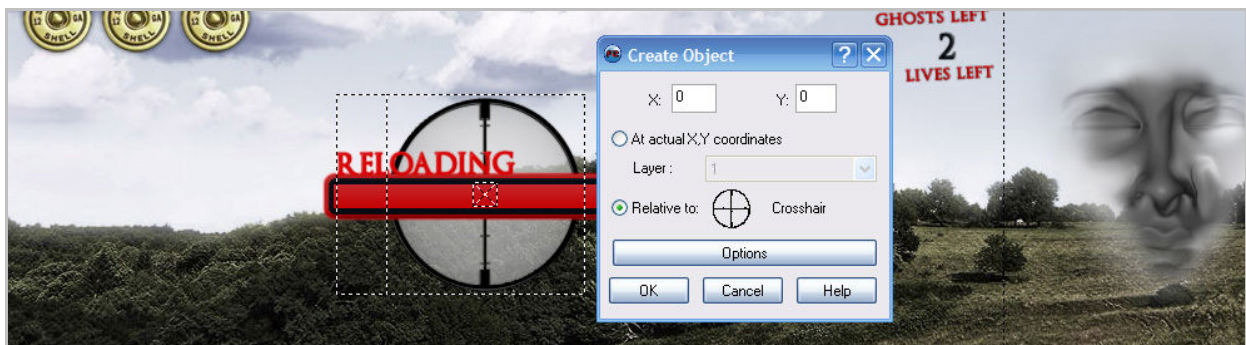
THEN: [Scrolling helper] >> Position >> Set X Coordinate

```
<< Type in: X( "Scrolling helper" )-4 >>
```

[illegible]

6) Leave the **Scrolling** group and enter the next one: **Shooting**. T'is pretty easy to guess what's this one all about, right? Now, let's make sure that ol' Bobby can defend himself... Create this three events inside this group, one after the other:

IF: **[Keyboard & Mouse Object] >> The Mouse >> User clicks**
<< select: left button, single click >>
IF: **[Player 1] >> Compare to player's number of lives**
<< check if they are Greater than 0 >>
IF: **[Crosshair] >> Alterable Values >> Compare to one of the alterable values**
<< check if Reloading is Equal to 0 >>
THEN: **[Sound Object] >> Samples >> Play sample**
<< Choose the "Gun shot1.wav" sound from MMF2's Bonus Disc >>
THEN: **[Player 1] >> Number of lives >> Subtract from number of lives**
<< input: 1 >>
THEN: **[Create New Objects] >> Create Object**
<< Select the [Gun blast] object >>
<< Set the coordinates to X=0, Y=0, relative to [Crosshair] object >>



And here are the two other events from this group... The first one is all about what happens if the player has no longer any ammo loaded in his shotgun – well... there's a click:

IF: **[Keyboard & Mouse Object] >> The Mouse >> User clicks**
<< select: left button, single click >>
IF: **[Player 1] >> Compare to player's number of lives**
<< check if they are Lower or equal 0 >>
THEN: **[Sound Object] >> Samples >> Play sample**
<< Choose the "Switch 3.wav" sound from MMF2's Bonus Disc >>

IF: **[Gun blast] >> Animation >> Has an animation finished?**
<< select Stopped >>
THEN: **[Gun blast] >> Destroy**

Take a look at this screenshot if you require some sort of a visual aid:


[illegible]

7) Time to once again abandon the current group and travel to the next one – this time it's **Reloading**. We'll start off with the most basic event in here:

```
IF: [Keyboard & Mouse Object] >> The Mouse >> User clicks
<< select: right button, single click >>
THEN: [Crosshair] >> Alterable value >> Set
<< Set Reloading to 1 >>
THEN: [Reloading frame] >> Visibility >> Make Object Reappear
THEN: [Reloading Counter] >> Visibility >> Make Object Reappear
```

Another event to create inside the same group – this one is crucial for our reloading process:

```
IF: [Crosshair] >> Alterable Values >> Compare to one of the alterable values
<< check if Reloading is Equal to 1 >>
THEN: [Reloading Counter] >> Add to Counter
<< input: 1 >>
THEN: [Reloading frame] >> Order >> Bring to front
THEN: [Reloading Counter] >> Order >> Bring to front
```













16	Reloading																				
17	• User clicks with right button							✓							✓			✓			
18	• Reloading of  (Crosshair) = 1														✓			✓			

Create these four events underneath, one after the other, inside the **Reloading** group:

IF: [Reloading Counter] >> *Compare the counter to a value*
 << compare whether it is *Equal 10* >>
 IF: [Special Object] >> Limit conditions >> *Only one action when event loops*
 THEN: [Sound Object] >> Samples >> *Play sample*
 << Choose the “Reload 1.wav” sound from MMF2’s Bonus Disc >>

Here's the most important one from this batch of events... This one tells the game what to do when the **Reloading Counter** finally counts to the maximum number (which means that Bobby has finally managed to reload his shotgun):

As you can see here, once the counter equals 150, the game sets the number of Player's 1 lives (which – as you most likely remember – we use as our ammo counter) to 3, makes the reloading frame and counter invisible, and ends the whole reloading sequence.

19	<ul style="list-style-type: none"> •  (Reloading Counter) = 10 • Only one action when event loops 																		
20	<ul style="list-style-type: none"> •  (Reloading Counter) = 70 • Only one action when event loops 																		
21	<ul style="list-style-type: none"> •  (Reloading Counter) = 110 • Only one action when event loops 																		
22	<ul style="list-style-type: none"> •  (Reloading Counter) = 150 • Only one action when event loops 																		

8) Now, open up the next group – **Ghosts** – and then the first subgroup there, which should be **Ghost creation**. These two events should be created inside that subgroup:

```
IF: [The Timer Object] >> Every
<< Set the timer to 3 seconds >>
IF: [Ghost] >> Pick or count >> Compare to the number of objects
<< Lower than 3 >>
THEN: [Ghost creator] >> Position >> Set X Coordinate
<< Type in: Random(1320)+120 >>
THEN: [Ghost creator] >> Position >> Set Y Coordinate
<< Type in: Random(120)+120 >>
THEN: [Create New Objects] >> Create Object
<< Select the [Ghost] object >>
<< Set the coordinates to X=0, Y=0, relative to [Ghost creator] object >>
THEN: [Ghost] >> Scale / Angle >> Set scale
<< set scale to 0.1 >>
<< set quality to 0 >>
THEN: [Ghost] >> Alterable value >> Set
<< Set Health to Random(3)+1 >>
THEN: [Ghost] >> Alterable value >> Set
<< Set Countdown to 25 >>
THEN: [Ghost] >> Order >> Bring to back
THEN: [Sound Object] >> Samples >> Play sample
<< Choose the "BUBBLE3.wav" sound from MMF2's Bonus Disc >>
THEN: [Clickteam Movement Controller] >> Set object
<< Select the Ghost object >>
THEN: [Clickteam Movement Controller] >> Circular movement >> Set centre X
<< input: X( "Ghost creator" ) >>
THEN: [Clickteam Movement Controller] >> Circular movement >> Set centre Y
<< input: Y( "Ghost creator" ) >>
```

And here's the second one, almost identical (differs only in conditions):

```
IF: [Ghost] >> Pick or count >> Compare to the number of objects
<< Equals 0 >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
THEN: [Ghost creator] >> Position >> Set X Coordinate
<< Type in: Random(1320)+120 >>
THEN: [Ghost creator] >> Position >> Set Y Coordinate
<< Type in: Random(120)+120 >>
```


IF: [Player 1] >> Compare to player's number of lives
 << check if they are Greater than 0 >>
 IF: [Shooting mark] >> Collisions >> Overlapping another object >> [Ghost]
 IF: [Crosshair] >> Alterable Values >> Compare to one of the alterable values
 << check if Reloading is Equal to 0 >>
 THEN: [Sound Object] >> Samples >> Play sample
 << Choose the "SNARL2.wav" sound from MMF2's Bonus Disc >>
 THEN: [Ghost] >> Alterable Values >> Subtract from
 << Subtract 1 from Health >>

...and this one takes care of what happens when one of the ghosts is out of health:

IF: [Ghost] >> Alterable Values >> Compare to one of the alterable values
 << check if Health is Lower or equal 0 >>
 IF: [Special Object] >> Limit conditions >> Only one action when event loops
 THEN: [Sound Object] >> Samples >> Play sample
 << Choose the "BIGBEAST.wav" sound from MMF2's Bonus Disc >>
 THEN: [Counter – Ghosts] >> Subtract from Counter
 << input: 1 >>
 THEN: [Ghost] >> Destroy

And that's it, as far as this subgroup is concerned. Wanna' take a look at my Event Editor?

29	Ghost behavior																			
30	<ul style="list-style-type: none"> • Every 00"-30 • User clicks with left button • Number of lives of (Player 1) > 0 																			
31	<ul style="list-style-type: none"> • (Shooting mark) is overlapping (Ghost) • Reloading of (Crosshair) = 0 																			
32	<ul style="list-style-type: none"> • Health of (Ghost) <= 0 • Only one action when event loops 																			

10) We're almost at the end here, four events to go. Now, close this subgroup and open up the next one (and last one), the one named **Ghost attack**. Create these events inside:

IF: [The Timer Object] >> Every
 << Set the timer to 1 second >>
 THEN: [Ghost] >> Alterable Values >> Subtract from
 << Subtract 1 from Countdown >>

Now, let's get those ghosts a tad more aggressive, shall we?

IF: [Ghost] >> Alterable Values >> Compare to one of the alterable values
 << check if *Countdown* is *Lower or equal 0* >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
THEN: [Ghost] >> Position >> Select position
 << choose a position outside the frame, for example: X at -249, Y at 131 >>
THEN: [Ghost] >> Destroy
THEN: [Create New Objects] >> Create Object
 << Select the [Ghost attack] object >>
 << Set the coordinates to X=-1, Y=-303, relative to [Scrolling helper] object >>
THEN: [Ghost attack] >> Order >> Bring to front
THEN: [Counter – Lives] >> Subtract from Counter
 << input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
 << Choose the “ROAR7.wav” sound from MMF2’s Bonus Disc >>

This here event is almost identical, it serves as a simple “failsafe”, so that we can be sure that each and every ghost has a chance to attack when his Countdown Alterable Value goes all the way down to zero:

IF: [Ghost] >> Pick or count >> Pick one at random
IF: [Ghost] >> Alterable Values >> Compare to one of the alterable values
 << check if *Countdown* is *Lower or equal 0* >>
IF: [Special Object] >> Limit conditions >> Only one action when event loops
THEN: [Ghost] >> Position >> Select position
 << choose a position outside the frame, for example: X at -249, Y at 131 >>
THEN: [Ghost] >> Destroy
THEN: [Create New Objects] >> Create Object
 << Select the [Ghost attack] object >>
 << Set the coordinates to X=-1, Y=-303, relative to [Scrolling helper] object >>
THEN: [Ghost attack] >> Order >> Bring to front
THEN: [Counter – Lives] >> Subtract from Counter
 << input: 1 >>
THEN: [Sound Object] >> Samples >> Play sample
 << Choose the “ROAR7.wav” sound from MMF2’s Bonus Disc >>

And here comes our last event... Come on, let’s not waste any time here:

IF: [Ghost attack] >> Animation >> Has an animation finished?
 << select *Stopped* >>
THEN: [Ghost attack] >> Destroy

You have been reading...

THE GHOST HUNTER

brought to you by
Koobare

Clickteam's funkiest
~~mercenary~~
Gray Warden



Created for Multimedia Fusion 2 & Multimedia Fusion 2: Developer

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