

# SPACE CORSAIR



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**  
koobare@koobare.com

**Welcome** to another one of Koobare's little tutorials, teaching you how to effectively and efficiently use the best multimedia authoring tool ever – [Multimedia Fusion 2](#) by Clickteam! In this tutorial we'll learn some cool new stuff about what MMF2 can do – and we'll, of course, learn that by practice, creating a groovy space shooter with tons of action, killer droids, lasers, green explosions and – last but not least – a simple *bullet-time* mode.

"Wait a second! Has he just said bullet-time?", asked Cobra Commander astonished beyond measure. "That's what I've heard!", quickly replied Destro, his silver head shining like a giant light bulb. "The man is obviously lunatic! Doesn't he know that only games with gargantuan budgets can employ such things as bullet-time and time manipulation? Hah!". They both chuckled and returned to their plans for worldwide domination.

Well... Dear Destro, dearest Cobra Commander, never underestimate the power of Multimedia Fusion 2! Not only shall we create a bullet-time effect (if you don't know what it is – think *Matrix*, *Max Payne*, *Hard Boiled*, *Blade*...), but we'll do it in just a few Events! Furthermore, I'll toss in a hip all-graphics-inverted look and – voila! – we'll have a professional movie-like special effect with hardly any work involved. How's that even possible? Thank Clickteam and their brilliant creation, 'cause MMF2 makes even the hardest coding tasks easy to accomplish. Heck, it's as easy as finding a Stormtrooper on the Death Star, and I guess that's saying something.

Anyways, let's take a deeper look at what we're going to create here: *Space Corsair* is going to be a simple *shoot 'em up* game, in which the player controls a single battle-hardened starship (which is named Space Corsair, by the way – yeah, I guess I went quite cheap with the title, but I've done that before with the *Smelly Claw* tutorial so don't act surprised). Space Corsair's captain, Jacob Keyes the Second (Jacob Keyes of *Halo* fame, I salute you!), has got a pretty big problem at his hands – he just went out of hyperspeed and found himself and his crew surrounded by hundreds of spinning killerbots, ready to either tear his ship apart with their lasers or to ram into it's hull kamikaze-style. The point is – whether Keyes likes it or not, he now has to shoot his way out of massive trouble.

Oh, there's also a catch, almost forgot about that – our starship ain't that new, you see, it has seen better days, and because of that it's laser cannon is... well, let's call it "less effective than you might expect". It takes two head-on blasts to damage any of the droids out of commission, and their complete annihilation is actually out of your reach, no matter how much shots you'll

fire towards on of them... Yeah, that's right – we can damage the droids so that they cannot use their cannons at us, but we still have to maneuver ourselves out of a collision course with that blasted pile of metal and wires, it doesn't just disappear into thin vacuum.

So, you now officially know the story, cadet. It's time to set up our game and make sure that everything plays out just the way we want it to. No more chit-chatting, no more G.I.Joe references (at least for a while), let's go and teach those spacedroids a lesson!

And may the stars guide us!

- ☰ 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](mailto:marchewkowy@gmail.com)

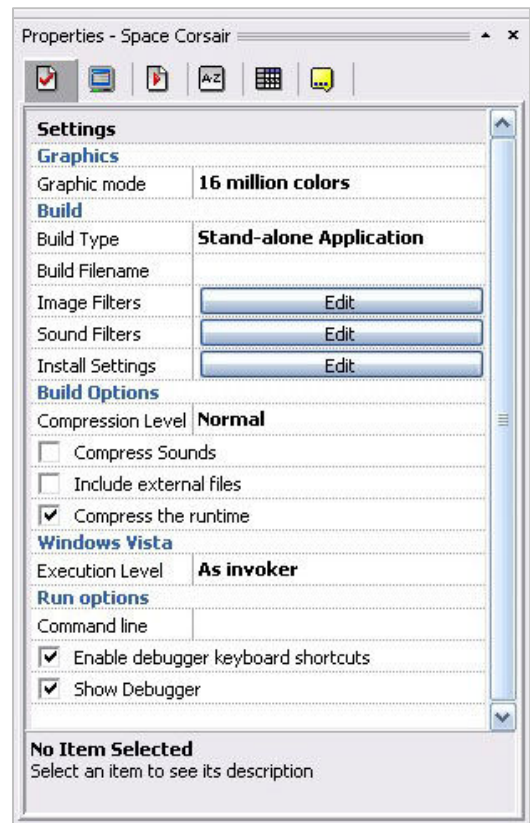
- ☰ **Note:** I've been receiving some reports that not all e-mails get to me for some reason. Seems that some of them (quite a lot) end up in my spambox or are blocked out by the server. I dunno why this is happening, so if you're experiencing any difficulties with delivering me a message or haven't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



## Part I: Setting up the application.

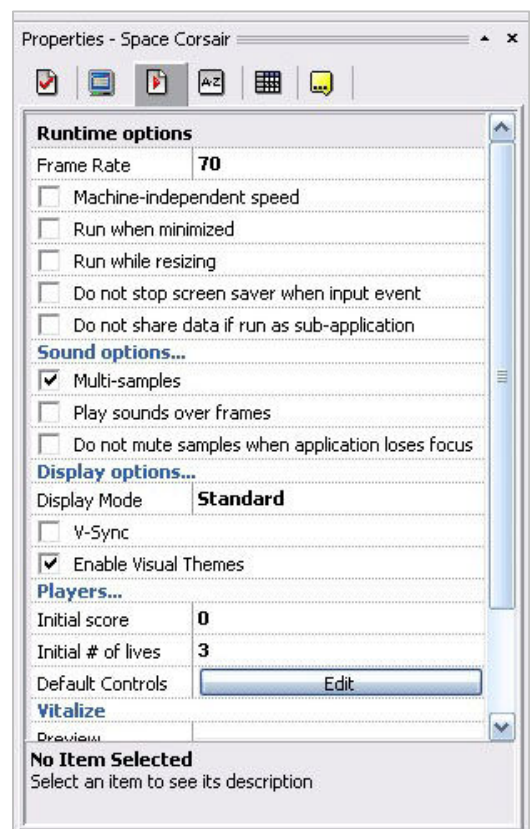
Let's make this magic happen already! Firstly, open up Multimedia Fusion 2, **create a new application** and be sure to save it onto your hard drive (I'm gonna' repeat myself one more time here: it's a good thing to have the "Autobackup" option turned on – be sure to check your "Preferences" window).

Now, once we have our new app ready & waiting, go to your application's **Properties window** (it should open up automatically – if it doesn't, right click on your application's name in the workspace toolbar and choose "Properties" from the drop-down menu). Leave everything as is in the first tab (*Settings*) – it should look just like this little screenshot in the top right corner of this page.



Let's move on. Select the **Window** tab (second from the left, the one with the little computer screen). Set the window size to **800x600**. MMF2 will ask you whether you'd like to modify all the frames that have the same size as the application's window – click on "Yes" (it will resize the one and only frame that's in our application). Before we move on – double check if the "Change Resolution Mode" option is **off**. We want our game to be windowed, not full screen.

Once that's done, let's move on to the **Runtime Options** (it's shown to the right if you need some visual aid). Make sure that the *Multi-samples* option is **on** (our game will play multiple sound effects at one time, so that's pretty important if we don't want any sound glitches).



Then, last but not least (actually, this is THE most important setting for this game!), set the **Frame Rate** to **70**. Setting a higher *Frame Rate* (also known as the *FPS* Count – *Frames per second*, not *First Person Shooter* in this case) can change quite a lot of things, from the overall speed of the game to the animation speed of each and every Active object on the playfield.

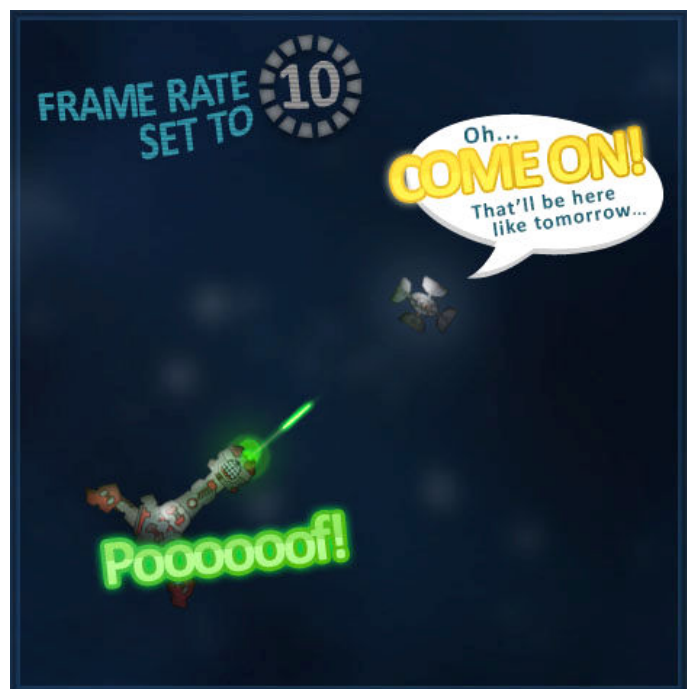
Setting a higher Frame Rate makes your game run faster and look smoother (animations take less time to loop), but has a bit of a negative impact on your game's system requirements – if it has lots and lots of heavily animated objects on screen, with lots of calculations being made in real-time, a higher Frame Rate can make your game a bit jumpy and rough-looking on some older computers (although that's not a common case). Setting the Frame Rate to a lower value makes yooouurrr gaameee ffaassstt aaasss aaa tuurttllleee – set it too low and you'll have time to make yourself some tea while your laser beam crawls through space in search of a droid, moving so sluggish that you won't be sure if it's even really traveling any distance.

You can choose any Frame Rate from 1 to 1000, but from my experience it's better to stick to values between 30 and 100 for the best effect – **50 is considered to be the standard, generic FPS value**. I usually stick to 50 FPS with most of my creations, but you can experiment on your own to check which Frame Rate would suit your game...

Anyways, we've just set our Frame Rate to 70 and let's keep it that way, shall we? We'll poke this subject a bit more once we get to the *Bullet time*

section of this tutorial, as for now just remember that changing the FPS can be pretty useful sometimes – especially that you can change it in the Event editor, during runtime.

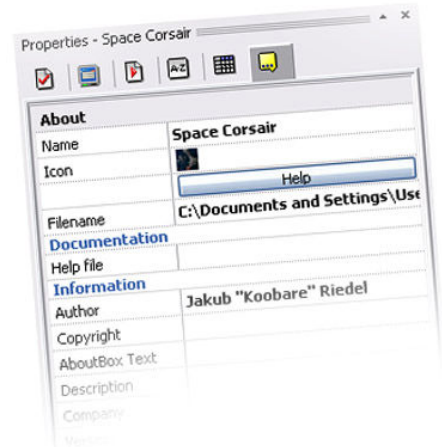
OK, let's get back to our *Properties* window. Open up the **Values** tab (fourth from the left) – we're going to set up all the global values we'll need to create this game the way it's meant to be. Don't be worried, there's not many of them. Create three values – name them "**Gamespeed**", "**DestroyedDroids**" and "**TimeOut**" respectively, from top to bottom (so that "Gamespeed" is the first one on our list) and make sure they're all set to zero.





Got it? Great! By now, we've made all the important changes in the main properties of our game, although there are still some less important thingies to do before we wander off to the next part of this tutorial.

Firstly, let's change our application's name to "*Space Corsair*" (right click on your app's name in the Workspace Toolbar and choose "Rename" from the drop-down menu). Secondly, make sure that our frame has a size of 800x600 pixels and rename it to "tutorial". Renaming all this stuff is purely cosmetic of course, but I'd like to keep your creation as similar to mine as possible – just to be sure that we won't have any problems when there'll be a need for troubleshooting or error correction.



If you wish, you can also change the icons for our application (open up the **About** tab, select the "Icon" field and click on the "Edit" button), but that's not really necessary. Try to remember about this when creating your own games though – it's good to have your own custom-made icons for a project that you've spent some time on. Anyways, let's move on to the next part of this tutorial, there's still plenty to do.

## Part II: Let's get these objects on stage!

---

It's now time to import all the objects we want to have in our little game. Normally you'd have to create Backdrop or Active objects inside the Frame editor and either fill them with your drawings using MMF's inbuilt Image Editor or import their "insides" from exterior image files (prepared in a different program, like Photoshop, Paintbrush or GIMP). We'll skip this "prepare from scratch" approach in this tutorial, if you'd like to know how it's done step by step, either check out one of the earlier tutorials or the Image



Editor Guide – they're all available at Clickteam's website of course. Anyways, what we're gonna' do here is open up an *.mfa* file I've prepared earlier and just copy + paste all the objects that are waiting inside. Let's do that now.

Open up the "**spacelibrary.mfa**" file, which was packed into the same archive as this .PDF tutorial. Select all the objects (the easiest way to do so would be by pressing CTRL+A on your keyboard), copy and then paste them into the first (and only) frame of our game, and... Voila! We've got all the objects we'll need in one place!





**Please note: some 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).**













### Part III: Know thy objects.

---

OK, so we've just imported all the needed objects into the frame. It's now time to learn something about them – why do we need them, what are they for, little snippets of info like that.

Let's do this in an organized manner, alphabetically:

Icon:	Object:	So... What is it?
	<b>enemy.creator</b>	<i>Enemy creator, active object that will be used to spawn enemies at it's current position.</i>
	<b>green.sparks</b>	<i>Green flashing sparks that will be "shooting" from a droid when it has been destroyed.</i>
	<b>health.counter</b>	<i>A simple horizontal bar counter. Acts as the health/shields counter for the Space Corsair.</i>
	<b>laser.boom</b>	<i>A big green explosion – an animation that is always created when you're blowing something up (i.e. a droid).</i>

Icon:	Object:	So... What is it?
	<b>laser.enemy</b>	<i>Enemy's projectile – a red laser beam. Sure the enemy's gonna shoot back from time to time, didn't ya' know that?</i>
	<b>laser.field</b>	<i>Energy field displayed when green laser beams are emitted from Space Corsair's guns and when they hit their targets.</i>
	<b>laser.player</b>	<i>Space Corsair's projectile – a green laser beam. Takes two of them to destroy a single spacedroid.</i>
	<b>points.counter</b>	<i>It's here to count points and uses a cool "glowing" blue font. What else to add?</i>
	<b>ripper.droid</b>	<i>Ripper spacedroid, rotator-hunter class. Designed to make your life a bit harder. In other words: the enemy.</i>
	<b>space.corsair</b>	<i>Space Corsair, player's starship. Still a fine piece of metal, but she ain't as young and lively as she once was...</i>
	<b>speed.counter</b>	<i>Another counter, we'll use it to support our game's mechanics, Bullet Time in particular.</i>
	<b>speeding.star.1</b>	<i>A semi-transparent object that'll help us create the illusion that Space Corsair is moving through space really fast.</i>
	<b>speeding.star.2</b>	<i>As above, but a bit thinner. Active object that'll help us imitate fast interstellar movement.</i>
	<b>starfield</b>	<i>An Active object that will act as our background picture – note that we're using an Active object, not a Backdrop.</i>
	<b>stars.creator.1</b>	<i>Active object that'll be used to shoot "speeding star" objects, moves along a predefined path.</i>
	<b>stars.creator.2</b>	<i>As above – an object used to spawn "speeding star" objects, but this time, instead of moving along a path, it appears at a random X position, between 0 and 800.</i>

And that's it – sixteen objects. Three counters (one set up as a horizontal bar, one number-based, one hidden), thirteen actives, no backdrops (we'll see why the "starfield" object is an active instead of a Backdrop later on)...



## Part IV: Know thy goal.

---



What's the first thing a general must know before planning out a battle? No, not his army's status, not his tactical situation, not even the fact that he's sitting on a horse wearing only pajamas – the first thing that he should know, understand and grind in his brain is the main objective, the purpose, the thing that he's aiming for – the über-important GOAL. In other words – to fight, to try to achieve the victory he must know what “victory” means, what is the aim of the upcoming struggle.

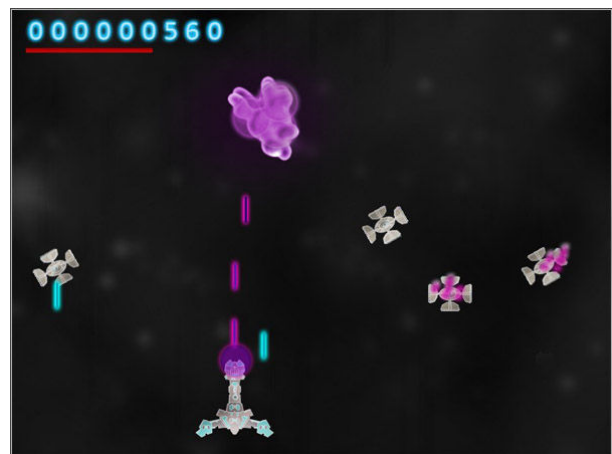
Come to think of it – do we really know what our GOAL is? We now know the pawns that will be used in our game, but do we really know the rules? Let's sum up what we know about all of this, about the thingie we're trying to assemble here... “Space Corsair”, basically, is a space shooter in which the player controls an armed starship, there are some weird spacedroids as enemies and there'll be a Bullet Time mode somewhere in the middle of all of this. Hmm, seems that our info is a bit rough on the edges. Time for an imaginary playthrough.

Someone double clicks on the icon, the game starts. Here you are – controlling the Space Corsair with keyboard arrows (left and right only – we can't move up or down), the ship seems to be speeding through space, stars pass you by with incredible velocity... Suddenly, they attack. They start falling from the top of the screen, spinning like if they were invented by a madman. At first there's only one, but after just a second they start swarming towards you. You move quickly to the left, to get one of the attacking spacedroids right in front of your ship, then you hastily press space. A green laser beam shoots from the cannon installed at your ship's bow, it rushes towards one of the mechanical opponents and hits him with a gnashing sound. A green flash pours all over the vicious machine and... Nothing happens. “Darn droid”, you think, “it's got a reinforced hull! One blast is not enough!”. Quickly, you press space again, releasing another shot of condensed energy. It hits the machine, blasts through it's hull and obliterates it's main CPU. The explosion was not enough to shred the droid to pieces, but it left it's mechanical body drifting through space.

And that's just the beginning – you're going to shoot down hundreds of spacedroids, maneuvering around hollow metallic corpses, trying to avoid both collisions and laser beams aimed at your starship... It may sound easy, but there's a great chance you'll be over your head! Be aware that colliding with both destroyed and active opponents will inflict damage to your ship's shields and armor, although smashing into a "dead" droid is slightly less harmful. Thankfully, Space Corsair is equipped with an extra layer of duranium metal around the main protective shielding, so she should be able to take quite a few bumps & laser hits before loosing hull integrity. What's also worth mentioning is that you can remove a floating spacedroid corpse out of your way – shooting at it will slightly alter it's trajectory, enabling you to slip sideways without taking a hit.

For every shot delivered to an enemy you'll receive points – a single point if your lasers were smashing into an empty tin can, devastated by your earlier blasts, 15 points if you shot a droid that's still active and ready to get you. Your mission is to gather... one million points. "Two bucks and a dime say it won't happen", said Destro to Cobra Commander. "No way anyone's that stubborn. Or foolish." Well... Seems to me you can be right this time, shinyhead. But I'm not gonna' join your bet. Anyone willing to show Destro that reaching one million points is possible? Just don't forget to eat and sleep, take turns with your friends, *wolverines*!

Anyways, let's get to the good part: the aforementioned **Bullet Time**. Once you've destroyed 20 droids, leaving their hull crushed by your lasers, with sparkles shooting all around, something happens. Background music suddenly changes, you hear a strange phasing sound and you abruptly find yourself in a game that looks a tad different than it looked a few seconds before – everything's moving slower,



spacedroids are rotating at a snail's pace, your lasers aren't green anymore, they're purple, and the same goes for explosions... "Nah, this looks nice but it really isn't that cool", you think to yourself. Until you press space. "Leaping fiery lizards of doom!", you howl at your screen, "I can now shoot twice as fast, when compared to those metallic muppets! I'm a messenger of destruction, ready to knock your antennas out of the sky, now bring it on!".

After 20 seconds of this Bullet Time mode your back to your normal speed. And you suddenly realize that scoring one million points ain't gonna' be easy.

## Part V: Know thy properties.

---

After this short fantasy playthrough, let's get back to our Frame Editor. We've got all our objects thrown into the frame, but we have yet to set their properties and reposition them around.

















Firstly, select the "**space.corsair**" object, go to its Properties window and open up the **Movement** tab (the one with the little blue man running). Change the Type of movement to **Eight Directions**, set its **Directions** to **left and right** (0 and 16), set the **Initial direction** to **none** and make sure that **Speed**, **Deceleration** and **Acceleration** values are all set to **50**. If you need any visual aid – take a look at the screenshot to the left. Got it? Great! That means we've just prepared our Space Corsair for its sharp maneuvers around those spinning spacedroids. Now, open up the **Size / Position** tab and change our starship's **X position to 400** and its **Y position to 470**. The object will now spring into the right position.

Let's select the second object, the "**ripper.droid**" one. It has its movement already set up, but let's take a look at how it's done. Open up the **Movement** tab, notice that the type of movement is set to **Path** (one of the simplest movements in MMF2) and press the "Edit" button. The path is just a straight line from top to bottom, without any extra twists and turns, so our enemies will just slide from top to the bottom of the screen. Our spacedroid also has a **second movement** set up – choose it up from the Movements list – a **Bouncing Ball** movement, that will be activated once the droid is destroyed. Once you're done examining this, open up the **Size / Position** tab and set the droid's position to **X=400, Y=30**. Last thing to do here before moving on to the next object: go to the **Values** tab (the one third from right), create a new **Alterable Value** and rename it to "**Damage**". Got it? Great!

You can now examine all the remaining objects, just to understand how they work and how they are set up in this game. Double click at the "**points.counter**" to open up the set of images

that acts as it's numbers – those are just some simple numbers with an additional “glow” effect I've added in an exterior graphics program and imported into MMF2. Check out the “**health.counter**” object to notice that it's Initial and Maximum values are both set to 120, while the Minimum value is set to 0 – if you'd like the Space Corsair to have a heavier hull armor, just change the Maximum and Initial values to 250 (if you want it to be weaker – just put some lower numbers in there). One more thing worth noticing – the “**green.sparks**” object has a Fade out transition set to a 0,11 second fade effect.

Once you're done exploring, let's get all these objects to their proper positions. Use this table to input all those X and Y position values for all the remaining objects – if there's an “[ out ]” text instead of a coordinate, just place your object somewhere outside of the frame (although not too far away, anything within the range of 150 pixels should do good):

Object:	X position:	Y position:	Object:	X position:	Y position:
 health.counter	20	55	 points.counter	285	50
 enemy.creator	40	-70	 starfield	0	0
 stars.creator.1	10	-45	 stars.creator.2	788	-40
 laser.enemy	[ out ]	[ out ]	 laser.player	[ out ]	[ out ]
 laser.boom	[ out ]	[ out ]	 laser.field	[ out ]	[ out ]
 speed.counter	[ out ]	[ out ]	 green.sparks	[ out ]	[ out ]
 speeding.star.1	[ out ]	[ out ]	 speeding.star.2	[ out ]	[ out ]

Once your done with all this repositioning – let's move on. We're finally heading for the programming part of this tutorial, so you may as well start cheering already. Yaay!

## Part VI: Programmer's delight.

---

It's finally time to get to the fun part of this here tutorial – the programming! Save your project (always remember to save it from time to time!) and open up the **Event Editor**. If you're new to my tutorials, let me introduce you to my event-recording system. 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**

If you'll have to input anything by keyboard, it will be indicated by coloring the text green and using < angle brackets >, like this:

**< Set the Global Value A to 32 >**

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

<< 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 about MMF2.

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...

### *Making it all work...*

1) Firstly, let's start with the traditional **"Start of frame"** event, which I usually create at the very beginning. This event – triggered when someone opens up our game – will start some nice music in the background and will make sure that our gamespeed counter (the **"speed.counter"** object) is set to 70 (this isn't actually necessary, I'm just a typical double-checker, I like to be sure that everything's gonna' be fine).

IF: [Storyboard Controls] > **Start of frame**

THEN: [Sound Object] > Samples > **Play and loop sample**

<< Select any music from MMF2's library, set the loop to 0 times – a continuous loop.

I chose the **"Spatial Maths"** track, which I think suits this game perfectly >>

THEN: [speed.counter] > **Set Counter**

< input: 70 >

And that's that – we've got our first event ready & waiting. To all you newcomers – hope that this MM2-to-paper scripting system seemed easy enough? Anyways, let's move on.

2) Time for the second event – this one will be based on the **Always** condition. This event will make sure of a few things: firstly – it'll always set the Frame Rate of our game to the value of the gamespeed counter ("speed.counter"). Secondly – it will always set "stars.creator.2" to an X position between 0 and 800, every event cycle. This means that "stars.creator.2" will be traveling very rapidly between positions, making our stars in the background appear more randomly. Finally – this event will make sure that the points counter is always displayed on top



of other things (for example, if a new spacedroid is created it usually is displayed on top of all the previously created objects, so this event will help to override that). Here's how it's all done:

**IF: [Special Object] > Always**

**THEN: [Storyboard Controls] > Set Frame Rate**

< input: `value( "speed.counter" )` or click on the *Retrieve data from an object* button, select the [speed.counter], choose *Current Value* from the right-click menu >

**THEN:** [points.counter] > Order > *Bring to front*

**THEN:** [stars.creator.2] > Position > Set X position

< input: *Random(800)* >

Got it? Great! Here's what we should have by now... Note that it doesn't have to look identical!

[illegible]

**3)** It's always good to use groups in your creations. Groups make it a lot easier to organize all those events, which can be pretty helpful and help preventing common mistakes. So, let's create ourselves a new group. Right click on the box that has number 3 written on it, then select *Insert > A group of events*. Name the group **"Which Mode"** and make sure that the option **"Active when frame starts"** is turned **on**. After that, create two more groups – **"Slow Mode"** ("Active when frame starts" should be **off** with this one) and **"Normal Mode"** ("Active when frame starts" option should be once again **on**). Once that's done, let's get back to the "Which Mode" group, open it up and create this event inside:

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

<< Choose value *DestroyedDroids* >>

< compare if it is *Equal* to 20 >

**THEN:** [Sound Object] > Samples > *Stop a specific sample*

<< Select the background music you've selected in our first event >>

**THEN:** [Sound Object] > Samples > *Play and loop sample*

<< Select any music from MMF2's library, set the loop to *0 times* – a continuous loop.

I chose the “Black Hole” track, which I think suits this part of the game perfectly >>

**THEN:** [Sound Object] > Samples > *Stop a specific sample*

<< Select the background music you've just selected – "Black Hole" in my case >>

Huh? We're stopping the sample we've just started playing? How come? Well, we actually want this to look a bit differently than it looks now, so dontcha' worry, we'll sort all of this in a second.

To make this as clear as possible... We have to put out the “stop a specific sample” action for both tracks we’ve selected earlier (“Black Hole” and “Spatial Maths” in my example), just to be sure that they won’t double-play, resulting in dreadful cacophony. So, why didn’t we put both the “stop a specific sample” actions already at the beginning? Because we have to tell MMF2 what samples are we going to play as our music – to stop sample “X” you have to select it from a list, and it won’t be there if there’s no “play sample X” somewhere in your event list. Anyways, it’s all good, the only thing we need to do now is to drag & drop the “Play and loop sample” action to be after the “stop a specific sample” one. Just do this and we’re home:

<< Double click on the “check” symbol ☒ under the Sound object for this event >>

<< Drag & the drop the “Play sample [name of sample] 0 times” to the end of the list >>



Got it? Great! Let’s continue with this event, this ain’t the end yet! As I explained earlier, once the player shoots 20 spacedroids (and that’s the condition for this event, in case you already forgot), our game enters a different mode, the Bullet Time mode (also known as the slow mode), and then it resets the “number of destroyed droids” value:

**THEN: [Special Object] > Change a global value > Set**

<< Choose value *Gamespeed* >>

< input: 1 >

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

<< Select the *Slow mode* group >>

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

<< Select the *Normal mode* group >>

**THEN: [Sound Object] > Samples > Play sample**

<< Select any “phasing” sound from MMF2’s library, I chose the “Beam Down” sample >>

**THEN: [Special Object] > Change a global value > Set**

<< Choose value *DestroyedDroids* >>

< input: 0 >

4) Once that’s done, let’s create another event right below, inside the same group of events. The previous event made sure that player enters the Bullet Time mode every time he shoots down 20 spacedroids. This event will make sure that he goes back to normal mode after 20 seconds spent with the Bullet Time on:

IF: [Special Object] > Compare to a global value  
 << Choose value *TimeOut* >>  
 < compare whether it is *Equal* to 20 >  
 THEN: [Special Object] > Change a global value > Set  
 << Choose value *Gamespeed* >>  
 < input: 2 >  
 THEN: [Special Object] > Group of events > Activate  
 << Select the *Normal mode* group >>  
 THEN: [Special Object] > Group of events > Deactivate  
 << Select the *Slow mode* group >>  
 THEN: [Sound Object] > Samples > Stop a specific sample  
 << Select the background music you've selected in our first event, i.e. "Spatial Maths" >>  
 THEN: [Sound Object] > Samples > Stop a specific sample  
 << Select the second background music you've selected– "Black Hole" in my case >>  
 THEN: [Sound Object] > Samples > Play and loop sample  
 << Select the music you've previously selected in the first event, "Spatial Maths" in my example, then set the loop to 0 times – a continuous loop >>  
 THEN: [Sound Object] > Samples > Play sample  
 << Once again the "phasing" sound – the "Beam Down" one in my example >>  
 THEN: [Special Object] > Change a global value > Set  
 << Choose value *TimeOut* >>  
 < input: 0 >

5) Here's an event that will make a smooth transition between the faster and slower game modes (between the normal mode and the Bullet Time one) – remember that this should be inside the "Which Mode" group:

IF: [Special Object] > Compare to a global value  
 << Choose value *Gamespeed* >>  
 < compare whether it is *Equal* to 1 >  
 IF: [speed.counter] > Compare the counter to a value  
 < compare whether it is *Greater* than 30 >  
 IF: [The Timer Object] > Every  
 << Set the timer to 4/100 of a second >>  
 THEN: [speed.counter] > Subtract from Counter  
 < input: 1 >

6) Here's yet another event to go into the "Which Mode" group – this one is nearly identical to the one above, but it goes the other way round:

IF: [Special Object] > Compare to a global value  
 << Choose value Gamespeed >>  
 < compare whether it is Equal to 2 >  
 IF: [speed.counter] > Compare the counter to a value  
 < compare whether it is Lower than 70 >  
 IF: [The Timer Object] > Every  
 << Set the timer to 4/100 of a second >>  
 THEN: [speed.counter] > Add to Counter  
 < input: 1 >

7) Time to sort out another little thingie – as I’ve said before, while in the Bullet Time mode our game will look a bit different, everything seems to change it’s appearance. We’re going to use the Inverted / Monochrome ink effects to create the desired outcome... This event and the next one go into the “Which Mode” event group:

IF: [Special Object] > Compare to a global value  
 << Choose value Gamespeed >>  
 < compare whether it is Equal to 1 >  
 THEN: [space.corsair] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [laser.player] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [laser.field] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [ripper.droid] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [laser.boom] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [green.sparks] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [enemy.creator] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [laser.enemy] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [speeding.star.2] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [speeding.star.1] > Visibility > Change Ink Effect  
 << Choose Inverted >>  
 THEN: [starfield] > Visibility > Change Ink Effect  
 << Choose Monochrome >>

Remember – the “starfield” object has it’s Ink Effect set to **Monochrome**, not Inverted as all the others! This is pretty important if you want to keep the darker look of the game...

8) Here’s the event that will erase all those ink effects when the game returns to normal mode:

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

<< Choose value *Gamespeed* >>

< compare whether it is *Equal to 2* >

**THEN: [space.corsair] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [laser.player] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [laser.field] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [ripper.droid] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [laser.boom] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [green.sparks] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [enemy.creator] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [laser.enemy] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [speeding.star.2] > Visibility > Change Ink Effect**

<< Choose *None* >>




**THEN: [speeding.star.1] > Visibility > Change Ink Effect**

<< Choose *None* >>

**THEN: [starfield] > Visibility > Change Ink Effect**

<< Choose *None* >>

That finishes up the events from this group. Here’s how it looks in my Event editor:

All the events Some objects hidden																					
1	• Start of Frame	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	• Always	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	Which Mode																				
4	• DestroyedDroids = 20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
5	• Timeout = 20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6	• Gamespeed = 1 •  > 30 • Every 00"-04	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
7	• Gamespeed = 2 •  < 70 • Every 00"-04	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8	• Gamespeed = 1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
9	• Gamespeed = 2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

9) We're done with events from the "Which Mode" group. Open up the "Slow Mode" group now, make sure that it's "active when frame starts" preference is set off (title of this group should be grey rather than black), and create this event inside:

```
IF: [The Timer Object] > Every
<< Set the timer to 1 second >>
THEN: [Special Object] > Change a global value > Add to
<< Choose value TimeOut >>
< input: 1 >
```

As you can see here, this event will add 1 every second to the TimeOut global value so that the Bullet Time effect can end after 20 seconds.

10) Here's another lil' event to create in the "Slow Mode" group – this one controls how frequently new spacedroids will emerge from the top of the screen while the player is in Bullet time mode:

```
IF: [The Timer Object] > Every
<< Set the timer to 1.63 second >>
THEN: [Create New Objects] > Create Object
<< Select the [ripper.droid] object >>
<< Set the coordinates to x=0, y=0 relatively to the [enemy.creator] object >>
```

11) Now it's time for the "Normal Mode" group – close up the "Slow Mode" group and open up the next one. This event is very similar to the one we've just created, and it'll be the only event present in this group (since there's no need to countdown to 20 in normal mode):

```
IF: [The Timer Object] > Every
<< Set the timer to 0.55 second >>
THEN: [Create New Objects] > Create Object
<< Select the [ripper.droid] object >>
<< Set the coordinates to x=0, y=0 relatively to the [enemy.creator] object >>
```

12) Create a new group of events and rename it to "Background". Leave it's "active when frame starts" option on. Now, let's create the illusion that our Space Corsair is rushing through space:

```
IF: [The Timer Object] > Every
<< Set the timer to 0.25 second >>
```



THEN: [stars.creator.1] > *Shoot an Object...*

<< Select the [speeding.star.1] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

THEN: [stars.creator.2] > *Shoot an Object...*

<< Select the [speeding.star.2] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

IF: [The Timer Object] > *Every*

<< Set the timer to 0.45 second >>

THEN: [stars.creator.1] > *Shoot an Object...*

<< Select the [speeding.star.1] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

THEN: [stars.creator.2] > *Shoot an Object...*

<< Select the [speeding.star.2] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

IF: [The Timer Object] > *Every*

<< Set the timer to 0.78 second >>

THEN: [stars.creator.1] > *Shoot an Object...*

<< Select the [speeding.star.1] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

THEN: [stars.creator.2] > *Shoot an Object...*

<< Select the [speeding.star.1] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

IF: [The Timer Object] > *Every*

<< Set the timer to 0.97 second >>

THEN: [stars.creator.1] > *Shoot an Object...*

<< Select the [speeding.star.2] object. Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

THEN: [stars.creator.2] > *Shoot an Object...*

<< Select the [speeding.star.2] object >>

<< Set Speed of object to 100 >>

<< Select "Shoot in selected directions", choose direction 24 (directly down) >>

**13)** Got it? Great! Once you've got those four events all settled down in the "Background" set, it's time to create yet another group... Or even four. First of all, create the **Main Events** group and open it up. Inside (yup, a group inside a group thing) create three more groups: **Droid Control**, **Ship Control** and **Animation**. Thanks to this our events will remain well organized and you'll be able to edit them anytime later with ease. Now, inside the Droid Control group...

```
IF: [laser.player] > Collisions > Another object > [ripper.droid]
IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values
<< check if Damage is lower than 2 >>
THEN: [ripper.droid] > Alterable Values > Add to
<< add 1 to Damage>>
THEN: [Create New Objects] > Create Object
<< Select the [laser.boom] object >>
<< Set the coordinates to x=1, y=12 relatively to the [ripper.droid] object >>
THEN: [laser.player] > Destroy
THEN: [Create New Objects] > Create Object
<< Select the [laser.field] object >>
<< Set the coordinates to x=11, y=13 relatively to the [ripper.droid] object >>
THEN: [Create New Objects] > Create Object
<< Select the [laser.field] object >>
<< Set the coordinates to x=-21, y=-2 relatively to the [ripper.droid] object >>
THEN: [Create New Objects] > Create Object
<< Select the [laser.field] object >>
<< Set the coordinates to x=13, y=-22 relatively to the [ripper.droid] object >>
THEN: [Sound Object] > Samples > Play sample
<< Play some sort of a "crash" sound, eventually a sound of an explosion >>
THEN: [points.counter] > Add to Counter
<< input: 15 >>
```

Looks pretty complicated, right? But don't worry, it isn't! Basically, this event tells what happens when one of our laser shots reaches a spacedroid while he still has some hull armor on him. If you'd take some time to analyze it line by line you'd see that this is pretty much as easy as it can be – add points, create some "boom" effects, play a sample and destroy the player's laser... And that's it! Let's create another event like this...

```
IF: [laser.player] > Collisions > Another object > [ripper.droid]
IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values
<< check if Damage is greater or equal 2 >>
THEN: [laser.player] > Destroy
```

THEN: [Create New Objects] > Create Object  
 << Select the [laser.boom] object >>  
 << Set the coordinates to x=1, y=12 relatively to the [ripper.droid] object >>  
 THEN: [Create New Objects] > Create Object  
 << Select the [laser.field] object >>  
 << Set the coordinates to x=11, y=13 relatively to the [ripper.droid] object >>  
 THEN: [Create New Objects] > Create Object  
 << Select the [laser.field] object >>  
 << Set the coordinates to x=-21, y=-2 relatively to the [ripper.droid] object >>  
 THEN: [Create New Objects] > Create Object  
 << Select the [laser.field] object >>  
 << Set the coordinates to x=13, y=-22 relatively to the [ripper.droid] object >>  
 THEN: [Sound Object] > Samples > Play sample  
 << Play some sort of a “crash” or “boom” sound >>  
 THEN: [points.counter] > Add to Counter  
 << input: 1 >>  
 THEN: [ripper.droid] > Movement > Bounce

Note that this time – because the spacedroid is no longer operational, it’s CPU has been blasted to pieces by earlier laser shots – player only gets a single point instead of fifteen. Plus, the enemy is “bounced”, which will make more sense if you’ll look at the next few events...

14) Let’s continue – all these events should go into the “Droid Control” group, as they control how do spacedroids behave and what happens to them after they get hit:

IF: [laser.player] > Collisions > Another object > [ripper.droid]  
 IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values  
 << check if Damage equals 1 >>  
 THEN: [Special Object] > Change a global value > Add to  
 << Choose value DestroyedDroids >>  
 < input: 1 >

This one counted all the destroyed droids, so that the game knows when to start the Bullet Time mode – it always starts after you’ve destroyed 20 enemies.

IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values  
 << check if Damage is greater or equal 2 >>  
 THEN: [ripper.droid] > Animation > Stop  
 THEN: [ripper.droid] > Movement > Multiple movements > Select movement

```
<< select "Movement #2" >>  
THEN: [ripper.droid] > Direction > Select direction  
<< choose directions: 18, 19, 20, 28, 29, 30 >>  
<< hold your mouse over a direction box to check it's number >>
```

Thanks to the event shown above the droid stops moving along a path when he's shot two times and starts "wiggling" around in space, like if it was greatly damaged and lost it's main processing unit. Choosing directions from 18 to 20 and from 28 to 30 enables it to still move towards the player (or rather towards the bottom of the screen), even if it's movement isn't along a path anymore.

```
IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values  
<< check if Damage is greater or equal 2 >>  
IF: [The Timer Object] > Every  
<< Set the timer to 0.09 second >>  
THEN: [Create New Objects] > Create Object  
<< Select the [green.sparks] object >>  
<< Set the coordinates to x=6, y=4 relatively to the [ripper.droid] object >>
```

The event above is nothing more than a graphical effect – it creates green sparks around every droid you destroy.

```
IF: [ripper.droid] > Pick or count > Pick "ripper droid" at random  
IF: [ripper.droid] > Position > Compare Y position to a value  
<< check if Y position is lower than 400 >>  
IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values  
<< check if Damage is lower than 2 >>  
IF: [The Timer Object] > Every  
<< Set the timer to 1 second >>  
THEN: [ripper.droid] > Shoot an Object...  
<< Select the [laser.enemy] object. Set Speed of object to 100 >>  
<< Select "Shoot in selected directions", choose direction 24 (directly down) >>  
THEN: [Sound Object] > Samples > Play sample  
<< Play some sort of not-to-loud sound >>
```

The event above makes sure that some of the spacedroids actually shoot back...

```
IF: [ripper.droid] > Position > Compare Y position to a value  
<< check if Y position is greater or equal 650 >>  
THEN: [ripper.droid] > Destroy
```

15) OK, we're done with the "Droid Control" group, time to create all the events our starship needs to operate properly – open up the "Ship Control" set. Firstly, let's create an event that enables the player to shoot laser beams:

IF: **[Keyboard & Mouse Object] > The Keyboard > Upon pressing a key**  
<< press **SPACE** on your keyboard >>  
THEN: **[space.corsair] > Shoot an Object...**  
<< Select the **[laser.player]** object >>  
<< Set Speed of object to 100 >>  
<< Select "Shoot in selected directions", choose direction 8 (directly up) >>  
THEN: **[Create New Objects] > Create Object**  
<< Select the **[laser.field]** object >>  
<< Set the coordinates to x=0, y=0 relatively to the **[space.corsair]** object >>  
THEN: **[Sound Object] > Samples > Play sample**  
<< Play some a "laser" or "shooting" sound >>

Here's a little event that'll make sure that player stays on-screen:

IF: **[space.corsair] > Position > Test position of "space.corsair"**  
<< Select "Leaves in the right?" – arrow leaving the frame to the right >>  
<< Select "Leaves in the left?" – arrow leaving the frame to the left >>  
THEN: **[space.corsair] > Movement > Bounce**

These two events control what happens if Space Corsair smashes into a spacedroid. Note that such a collision makes more damage if the droid is still operational:

IF: **[ripper.droid] > Collisions > Another object > [space.corsair]**  
IF: **[ripper.droid] > Alterable Values > Compare to one of the alterable values**  
<< check if **Damage** is lower than 2 >>  
IF: **[Special Object] > Limit conditions > Only one action when event loops**  
THEN: **[Sound Object] > Samples > Play sample**  
<< Play some sort of a "crash" sound >>  
THEN (optional): **[Sound Object] > Samples > Play sample**  
<< Only if you wish to play another sound: select an "explosion" sample >>  
THEN: **[space.corsair] > Animation > Change animation sequence to**  
<< Select "shot" >>  
THEN: **[ripper.droid] > Alterable value > Set**  
<< Set **Damage** to 2 >>  
THEN: **[health.counter] > Subtract from Counter**  
<< input: 5 >>

```

IF: [ripper.droid] > Collisions > Another object > [space.corsair]
IF: [ripper.droid] > Alterable Values > Compare to one of the alterable values
<< check if Damage is greater or equal 2 >>
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [Sound Object] > Samples > Play sample
<< Play some sort of a “crash” sound >>
THEN (optional): [Sound Object] > Samples > Play sample
<< Only if you wish to play another sound: select an “explosion” sample >>
THEN: [space.corsair] > Animation > Change animation sequence to
<< Select “shot” >>
THEN: [health.counter] > Subtract from Counter
<< input: 1 >>

```

Here's an event that controls what happens when you fall into enemy laser beams:

```

IF: [ripper.enemy] > Collisions > Another object > [space.corsair]
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [laser.enemy] > Destroy
THEN: [Sound Object] > Samples > Play sample
<< Play some sort of a “crash” sound >>
THEN (optional): [Sound Object] > Samples > Play sample
<< Only if you wish to play another sound: select an “explosion” sample >>
THEN: [space.corsair] > Animation > Change animation sequence to
<< Select “shot” >>
THEN: [health.counter] > Subtract from Counter
<< input: 5 >>

```

Since all shield regenerators are on, player's shields recharge over time – every 5 seconds a small portion of the health bar is revived:

```

IF: [The Timer Object] > Every
<< Set the timer to 5 seconds >>
THEN: [health.counter] > Add to Counter
<< input: 1 >>

```

It's time to tell the game what should happen when player's health comes down to 0, or when he reaches that envisioned one million points... I'm gonna' put just some simple basic actions here, but you can tweak it up a bit and – for example – build another level that will be opened once player has gathered enough points:



IF: [health.counter] > *Compare counter to value...*  
<< Check whether the counter is *lower or equal 0* >>  
THEN: [Storyboard Controls] > *Restart the application*

IF: [points.counter] > *Compare counter to value...*  
<< Check whether the counter is *greater or equal 1000000* >>  
THEN: [Storyboard Controls] > *End the application*

16) OK, we're almost at the end here! Very little left to do... Close the "Ship Control" group and move to the "Animation" one... There's not much to do here, just create this:

IF: [laser.boom] > *Animation > Has an animation finished?*  
<< select "Stopped" >>  
THEN: [laser.boom] > *Destroy*

IF: [green.sparks] > *Animation > Has an animation finished?*  
<< select "Stopped" >>  
THEN: [green.sparks] > *Destroy*

IF: [laser.field] > *Animation > Has an animation finished?*  
<< select "Stopped" >>  
THEN: [laser.field] > *Destroy*

IF: [space.corsair] > *Animation > Has an animation finished?*  
<< select "shot" >>  
THEN: [space.corsair] > *Animation > Change animation sequence to*  
<< Select "Stopped" >>

**Mission accomplished!** That's it! Finally! We're done here, space pirates! There you have it – a nice-looking space shooter with a cool Bullet Time effect and lots of spacedroids to destroy. You can add as many modifications as you wish, try tweaking the difficulty level a bit, have fun with every aspect of this game!

*Thanks for your time and see you again soon!*

***Cheers!***



If you have any questions, suggestions or just need help –  
mail me at [marchewkow@gmail.com](mailto:marchewkow@gmail.com) or [koobare@koobare.com](mailto:koobare@koobare.com)

You have been reading...

# SPACE CORSAIR



BROUGHT TO YOU BY

## KOOBARE

CLICKTEAM'S funkiEST

~~mercenary~~

*starship captain*

Created for Multimedia Fusion 2 & Multimedia Fusion 2: Developer

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