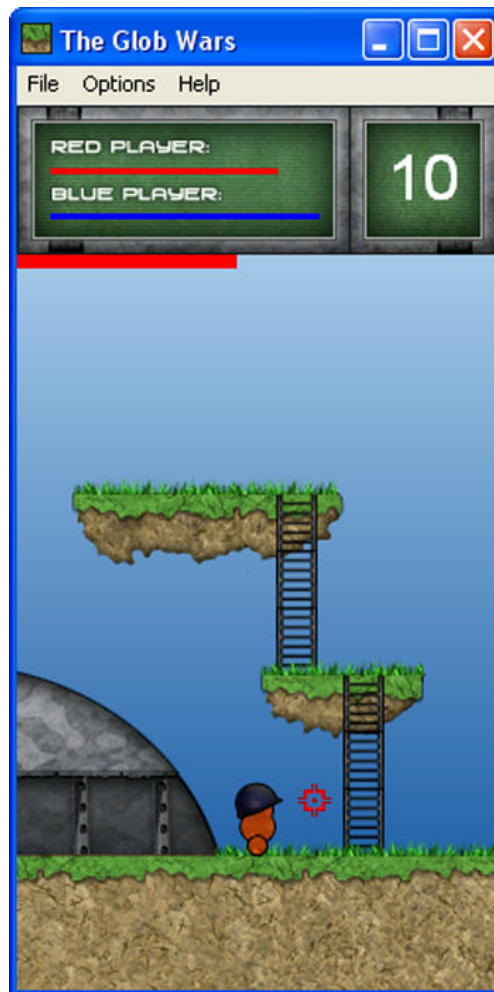


THE GLOB WARS

TUTORIAL



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

Koobare
marchewkowy@gmail.com

Welcome to Koobare's second little tutorial, this time focusing on how to create a "Glob Wars" game using the best multimedia authoring tool ever – [Multimedia Fusion 2](#) by Clickteam! The main purpose of this tutorial is to teach you how to create, organize, activate and deactivate event groups in order to create a turn-based two player combat game, a bit similar to the "Worms" series.

Have you ever played one of the classic "Worms" games? Their core idea is pretty simple really. You control a pack of worms that can use multiple weapons to eliminate another pack of worms in a turn-based and fun-filled game. I'll show you how to create basics for such a game in Multimedia Fusion 2 – instead of a pack, we'll create just a single soldier for each of the teams, instead of multiple weapons – we'll use a grenade. Nonetheless, this tutorial will help you understand the basics of creating a "Worms" clone – and it's up to you whether you'll add more features and content or not.

Here are some basic features of the game we're going to create:

- "Glob Wars" will be a turn-based, two player game, working in a "hot seat" mode (players share the same computer, exchanging places in front of the monitor).
- Using, activating and deactivating event groups will be crucial for our project. We'll use groups to separate players' turns and to give them a few seconds to change their seats (during the "Stand by – Change seats" mode).
- Each turn will be limited to 12 seconds. Players will have to move fast!
- Our game will utilize the built-in MMF2's platform and pinball movements. Platform movement will be used for controlling the Globbs (those little jelly-like creatures with helmets on their heads... err... top), Pinball movement will be used as a movement for the grenades.
- The objective of the game is to eliminate the rival player's Glob by throwing grenades at him. Each grenade that hits the target subtracts 15 points from its health counter.
- Player will decide at what strength the grenade should be thrown, by pressing and holding the "CTRL" key. Grenades will be thrown in the direction chosen by a rotating target-marker, controlled with the "A" and "Z" keys.
- Player's control will be ignored as soon as he throws a grenade and will be regained at the time his next turn starts.
- We'll use a string object to display communicates by using the "alterable string" function. We'll also display the Countdown counter's value in our string object.
- This game will utilize a basic horizontal scrolling system. Current X position of the screen will be based on X positions of either one of the players, or the grenades they have thrown.
- Game will be displayed in a 245x450 px window, using the 65536 colors mode.

Part I: Setting up the application.

OK, let's do this, people! Open Multimedia Fusion 2, **create a new application** and save it onto your hard drive (remember that it's always good to have some copies here and there). Go to the application properties screen (if it didn't open by itself, right click on your app's name in the *Workspace toolbar* and select "*Properties*" from the drop down menu).

Firstly, set the **graphic mode** to 65536 colors. This will make our application work faster on older computers. Remember one thing, though: if you'll set your app to 65536 colors, all graphics will be automatically converted into this mode and you will be unable to revert your changes! In other words: it's usually best to work with the 16 million colors mode and switch to another mode after you're done with creating your whole application and you have a backup copy on your disk or safely burnt onto a CD. Why? Well, perhaps you'll finally decide that 16 million colors would suit you better – and what then? There's no going back when you've selected one of the lower-color modes! Keep that in mind. Anyhow, we're pretty sure that 65536 will suit us well with the "Glob Wars" game (especially that I've already used this mode to save the graphics that we're going to use), so select this option and let's move on.

The second thing on our "to do" list is setting up the **window properties**. Open up the "Window" tab in our application's properties screen (second from the left) and set the size of the game's window to **245x450 pixels**. Contrary to selecting the graphical mode, window options can be reverted at any time, so don't bite your nails before inputting such a weird window size. If you'll decide that this size doesn't suit you at all, you can always return to this window once your game is complete and change the window size to 450x450 or whatever size you'll want. If you wish, you can play a bit with other options - I usually change the **border color** either to black or dark grey, but that's for you to decide.

Now, when that's done, we can leave the application properties window and wander off to the storyboard editor. What are you waiting for, then? Let's go!

Part II: Setting up the frame, importing the objects.

Remember what I've told you earlier about our scrolling? Let me remind you, private: "This game will utilize a basic **horizontal scrolling** system". Got it? That means that our frame will only scroll horizontally, not vertically. Having that in mind, we must create a frame that's equal to the vertical size of the window, but is a lot bigger when it comes to the horizontal size. I'd suggest creating a **2000x450** frame. It should suit our needs perfectly.

Open the frame. Now, it's time to import the objects that we're going to need while creating our game. Open the "**globlibrary**" file (*globlibrary.mfa* – don't open the *globwars.mfa* or the *globelevel.mfa* files

yet!), which was packed into the same archive as this .PDF tutorial. Select all the objects and copy them into your application (Note: most of them use **alpha channels**, a feature that is unavailable to TGF2 users. TGF2 users should use basic library objects instead). Take a look at them – aren't they just gorgeous? You bet they are – it was me who created them after all! :) After you're done with all that admiring, it's time to give those objects some proper movements and set their preferences. Move it along to part III.

Part III: Setting up objects preferences and movements.

Let's play a bit with our objects and their preferences, shall we? Remember that if you're unsure which object is what, you can just hold your cursor over that object and a small tooltip with object's name will appear. You can also use the **Object List** window, the one to the left of your frame (please note that selecting an object from the Object List will select all its occurrences in the currently opened frame!).

- 1) **Select the “RedPlayer” object.** *RedPlayer* and *BluePlayer* are the main characters in our game. RedPlayer is a Glob creature controlled by player one, while BluePlayer is controlled by player two. They both have a set of three animations: standing, walking and hit. These creatures will be controlled by our players with arrow keys (for moving) and the “shift” key (for jumping), using the basic, built-in **Platform movement**.

Right-click on the RedPlayer object. Select “properties” from the drop-down menu. Set the **FADE OUT** transition to **FADE** and set it's timer to **2.60 seconds**. This will make our little red Glob nicely fade into the sky when he'll be destroyed.

Now, go to the “**Movement**” tab (that's the one with the little man running, third from left). Open up the movement type list and select “**Platform movement**”. Set the initial direction to 0 (right). Leave this object under the control of player one. Set the speed to 12, both acceleration and deceleration to 50. You can try your movement, if you wish. Now, set the jump gravity to 30, leave it's strength at 50 – and we're done with setting up the movement!

Go to the “**Values**” tab (fifth from left, third from right). Our characters will need two **Alterable Values** – one for a usage similar to a flag (it's value will be either 0 or 1, to help us determine whether the player has already started powering up his grenade throw or not) and the second one for storing the direction at which this player's target-marker is currently set. We could use different methods of value storage here, like GlobalValues, Counters or – as far as the first Alterable Value goes – Object Flags, but I'd like to introduce you to the wonderful world of Alterable Values.

Alterable Values are something like an **internal counter** of an object, allowing you to store numbers inside the data of your active. This helps a lot when you create a game with lot's of objects, especially that alterable values are NOT shared between different instances of a single object (in other words: if you'll create six soldiers of the same type with the "create object" command, you can store their health counters in each one of them, by using their AV's, instead of having to create multiple Counter Objects and linking them up with multiple events). Alterable Values are also a bit faster than Counters, so using them here and there is usually a good idea.

Now, let's create those AV's. Having the RedPlayer object selected and the Values properties tab opened – click on the "new" button under the "Alterable Values" text. A new Alterable Value will be created. Click on the "new" button again – now you have two AV's, ready to use! You can name them if you want (just double-click on their name), but I don't think that it's necessary.

OK, we're finally done with setting up the RedPlayer object! Let's go for the BluePlayer one.

- 2) **Select the "BluePlayer" object.** Create the same FADE OUT transition, as you did with the "RedPlayer". Go to the "movement" tab and do the same there too, with just a slight difference: set the initial direction to **16** (right), and set the control to **Player 2**. Create two Alterable Values for this object, in the same manner as you did for the RedPlayer one. We're done here.
- 3) **Select the "Grenade" object.** Set it's **FADE OUT** to **BANDS**, and set it's timer to **2.60 seconds**. The player won't actually see the grenade's fade out effect (since we're going to set up an event making this object invisible as soon as it blows – and then create the "Boom" object instead), but it will help us with the horizontal scrolling feature – you'll soon learn why.

Secondly, go to the "**Movement**" tab. We're going to set up two different movements for this object: the main one ("**Pinball movement**" – a basic, built-in gravity-controlled movement), used most of the time, and a secondary one ("**Static**"), selected by the game when the grenade collides with the ground and we don't want it to bounce anymore.

Select the "**Pinball movement**". Set the "Gravity" to 20, "Deceleration" to 40, "Initial speed" to 70 and set the initial direction to the direction number 8 (up). Both the "Initial speed" and "direction number" preferences will be set at runtime, with the help of the events we're going to create later on – we're setting them now just for the sheer pleasure of setting them (and giving you the possibility to see how the grenade works, if you press the "try movement" button).

Now, let's create the second movement. This is quite easy, actually. In the "Movement" tab, just click on the "**Movement #1**" text. A button will appear to the right. Click it. Then, press the

“Create new movement” button in the newly opened dialogue box (it’s to the right). Click OK. A new movement will appear at your list – **“Static”** by default. You’re done here.

The last thing that we want to set up when it comes to the **“Grenade”** object are **Alterable Values**...Well, one **Alterable Value** to be exact. Just create a new **Alterable Value** at the right tab and the **“Grenade”** object is finally ready.

- 4) **Select the “Boom” object.** This object will act as our explosion and will be created every time the grenade goes off. Set it’s **FADE OUT** animation to **ZOOM**, and set it’s timer to **0.56 seconds**.
- 5) **Select the “Targeter” object.** This object will allow the player to choose at which direction he wants to throw the grenade (by cycling through directions with **“A”** and **“Z”** keys). Well, on second thought there’s nothing to change here, so let’s move along.
- 6) **Select the “Interface” object.** This is an object that will act as our interface basis – we’ll put health counters and the countdown timer on top of it. The **“Interface”** object should not scroll along with the rest of the objects – it should always stay in one position. To do so, enter it’s properties, go to the **“RunTime Options”** tab (fourth from the left) and **UNCHECK** the **“Follow the frame”** and **“Destroy object if too far from frame”** preferences. Note that this is an active object – this information will come in handy in a second.
- 7) **Select the “RedHealth” counter.** Note that it is set as a **“Horizontal bar”** with a solid color, starting the count from left, with a minimal value of 0 and both maximal and initial values set to 100. Make sure that the **“Display as background”** (in the **“Display Options”** tab) option is **OFF**. Why is it so important? Because the **“Interface”** object is an active, remember? All actives are always displayed in front of the background objects, so this counter would be behind the **“Interface”** object, and thus would not be visible. Go to the **“RunTime options”** tab (third from the right) and set both the **“Follow the frame”** and **“Destroy object if too far from the frame”** **off**.
- 8) **Select the “BlueHealth” counter.** Configure it in the same way as the **“RedHealth”** counter. Make sure (by right-clicking on the object and entering the **“Order”** sub-menu) that both **“RedHealth”** and **“BlueHealth”** counters are displayed in front of the **“Interface”** object.
- 9) **Select the “ThrowStrength” counter.** Note that it is set as a **“Horizontal bar”** with a solid color, starting the count from left, with a minimal value of 0, initial value of 1 and the maximum value set to 85. This counter will help us determine the strength of the throw, and thus – the initial speed of the thrown grenade. You’ll find out more about this in the later parts of this tutorial.

Make sure that the “Display as background”, “Follow the frame” and “Destroy object if too far from the frame” options are all **OFF**.

- 10) **Select the “Ladder” backdrop object.** Make sure that it’s “obstacle type” (in the “RunTime Options” tab) is set to **“Ladder”**. The default Platform movement will now identify this backdrop as a ladder, and thus enable the player to climb it with the use of the up arrow key.
- 11) **Select the “Grass_QB 1” quick backdrop object.** Make sure that it’s “obstacle type” is set to **“None”**. Make the same for the “Sky_QB” quick backdrop, and the “Bunker_down”, “RedFlag” and “BlueFlag” backdrop objects.
- 12) **Select all the remaining backdrop objects, one by one.** Make sure that all of them have the “obstacle type” set to **“Obstacle”**. You could also switch some of them to the “Platform” mode, but that doesn’t work too well with the Pinball movement, so our grenade’s bounce could be even more glitchy than it will be. :)
- 13) **Create a Counter object.** Set it’s initial and maximum values to 12, minimal value to 0. Set its display type to “hidden” and leave it just a few pixels outside our frame’s borders. Name this counter as “Countdown” – we will use it to determine how many seconds has the player to move before his turn ends.
- 14) **Create another Counter object.** Set it’s initial and minimum values to 0, maximum value to 2. Make it hidden and position it right to the “Countdown” object. Rename it to “WhichPlayer” – we will use this counter to determine who’s turn was it before and who should play now.
- 15) **Select the “Displayer” object.** We will use it to display game texts (such as “get ready!”) and the “Countdown” counter’s value. Make sure that it’s vertical and horizontal alignments are set to “center” (you’ll find those options under the “Text Options” tab, in the properties window).

Ok, that’s it when it come to setting up those objects. It’s time to create a level, which will act as the world shattered with the Globs’ mortal conflict.

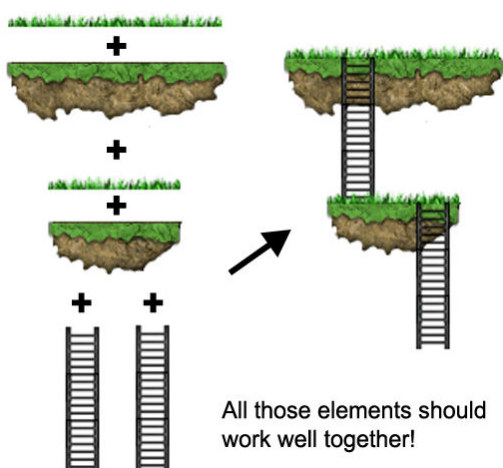
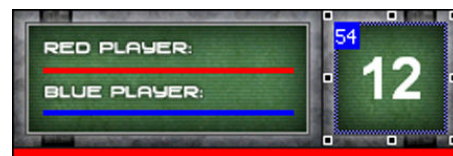
Part IV: Level design.

First of all, let me introduce you to a little trick, a real time-saver and one of the MMF2’s features that save a lot of effort: **moving objects in your frame with 1-pixel precision, using the arrow keys**. It’s really simple – just click on an object, press the up arrow key, and... Wow! It just moved 1 pixel up! Great! Now, select another object and HOLD the left arrow key...Wow again! Isn’t that great? It will

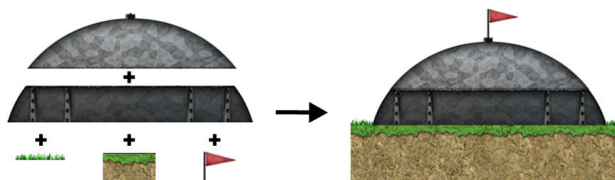
really save you lots of time, especially if you'll remember that you can move couple of objects at the same time with this method (just left-click on the objects you'd like to move while holding the SHIFT key to select them). Or even all of them (just press CTRL+A to select all objects in the frame).

Now, would you like to play a little and create your own level? If not – just open my ready-to-use level from the “**globlevel**” file (search for the **globlevel.mfa** file in the directory/archive that contains this .PDF tutorial) – it has all the properties set up and the level design ready, so all we have to do is to add some action with the event editor. If you decided to load my level – continue to the next part. If not – here are some basic tips for you to use, when creating your own map:

- a) Remember that you can either **duplicate** or **clone** an object (both options are available from the drop-down menu if you'll right-click on an object). **Cloning** creates a totally new object, with the same properties and looks as the original one. **Duplicating** creates a new instance of the same object – if you'll change the settings on one of the duplicates, resize it or add a new ink effect – all this changes will be applied to all of the duplicates, as well as to the original object. You can also duplicate an object by using the **CTRL+C** and **CTRL+V** keys or by dragging & dropping objects from the **object list**.
- b) Remember that you can zoom out and zoom in onto the frame – you'll find zoom options in the **View** menu of MMF2. This can be helpful when you want to see your whole playfield at once.
- c) Remember that you should put the **Interface** part in the top of the frame. Remember about all those vertical bar counters and our **String object**! Take a look at the image to the right if you need help.
- d) Remember to combine various types of backdrop objects. Look at these images for some tips:

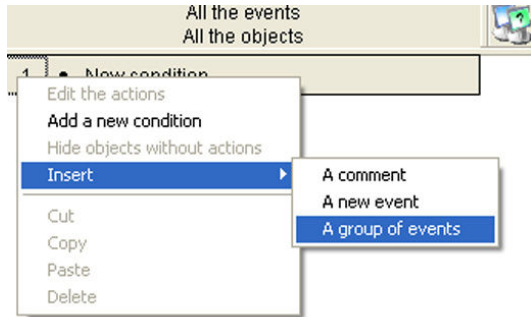


All those elements should work well together!



- e) Keep in mind that the players should have enough space to move around. Adding some levitating platforms connected to the ground via a ladder can be a nice idea too!

When you're done – proceed to the next part.



Before we'll get to coding, allow me to introduce you to the wonderful world of **groups**. Open up your event editor. Right-click on the event number (the box with the number on it, just to the left from the "New condition" text), and choose **Insert > A group of events** from the pop-up menu (take a look on the image to the left if you require visual help). A new dialogue box will appear, asking you for the name of your group – input anything, make sure that the

"Active when frame starts" option is selected and click the "OK" button. Voila! You've just created your first event group! Congratulations! You can now create new events **INSIDE** this group, drag & drop previously created events into it, close it and open it in the Event editor or – what's really important! – You can decide whether you want this group to be currently activated or deactivated, and you can make the decision to do so on runtime! Why is it so important, should you ask? I'll give you this example...

Imagine this: you want to create a game with an object (for example: an alien creature) that behaves **COMPLETELY** different depending on the given circumstances. If the player enters the frame from the left – the alien will greet him, dress up like a clown, give the player a cigar and start a conversation. But if the player enters the frame from the right – the alien will shout something in Venusian, take out his gun, throw a plasma grenade, jump onto his spaceship and run away. Sure, you can code this without groups – you'll just need to check a counter or an Alterable Value on each and every condition... But there's an easier way to do this: just create two event groups and uncheck the "Active when frame starts" checkbox in both of them. Put all the events for the "alien behaving friendly" scenario into group one, in group two – all the "alien is hostile" events. Then, just create two conditions – if the player enters the frame from the left: activate the "Friendly alien" group and leave the "Enemy alien" group deactivated. If the player enters the frame from the right: activate the "Enemy alien" group and leave the first group inactive. Isn't that simple?

Event groups can come very handy, especially when you're creating a bit more advanced applications. Groups are great at simplifying your programming, and in MMF2 you can even create **sub-groups** inside groups, establishing a fully professional group activation/deactivation system, that let's you save lots of system resources by deactivating currently unused and unimportant code.

You can play a bit with groups, if you wish, practicing the art of dragging and dropping conditions into them and creating new groups with just a single click (well... almost a single click). When you'll be ready – delete all the groups and events that you've created by far and let's proceed to part VI, where we'll finally get to do some coding! You'll find out how to use groups to create a fully operational turn-based game in a jiffy!

Part VI: Programmer's delight

OK, let's start our little programming experience, shall we? This will really be as simple as possible. Allow me to introduce you to an event-recording system that I'm going to use in this tutorial:

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

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

Not too complicated, is it? All the conditions will be marked in red color, while actions are written in a fancy blue one. If we'll have a multi-condition event (for example: the condition is true only when a counter is equal to zero and the selected object collides with backdrop), then we'll have:

IF (Condition 1): **[Object for condition 1] > Condition group 1 > Condition 1**

IF (Condition 2): **[Object for condition 2] > Condition group 2 > Condition 2**

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

If we'd have a multi-action event, it would look 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**

It's pretty simple, right? Let's see it in action, then.

1) Create a **"RedPlayer collides with background"** condition. To do so, create a new condition, go to the "RedPlayer" object, right-click on it, go to the "Collisions" condition group and select "Backdrop". Your condition is ready. Now go to the corresponding white tile in the "RedPlayer" object's column, right-click on it, go to the "Movement" group and select the "Stop" action. You currently have created such an event:

IF: **[RedPlayer] > Collisions > Backdrop**

THEN: **[RedPlayer] > Movement > Stop**

This event will stop the "RedPlayer" object, when the "RedPlayer" collides with backdrop. Let's do the same for the "BluePlayer" object... Create a new event:

IF: **[BluePlayer] > Collisions > Backdrop**

THEN: **[BluePlayer] > Movement > Stop**

Pretty simple. There will be more advanced events later on, though, keep that in mind.

2) Now, let's create a condition that is true when the "BluePlayer" and "RedPlayer" objects collide:

IF: [RedPlayer] > Collisions > Another object > [BluePlayer]

THEN: [RedPlayer] > Movement > Bounce

THEN: [BluePlayer] > Movement > Bounce

Hooray! We've got a basic collision system ready!

3) Now, let's create a "Start of Frame" condition that sparkles quite a lot of actions:

IF: [Storyboard Controls] > Start of frame

THEN: [Sound Object] > Music > Play music (select any music from the MMF2 midi library)

THEN: [Player 1] > Player Control > Ignore control

THEN: [Player 2] > Player Control > Ignore control

THEN: [Grenade] > Destroy

THEN: [BluePlayer] > Alterable Values > Set (set the Alterable Value B to 16)

THEN: [WhichPlayer Counter] > Set Counter (set the Counter to 2)

THEN: [ThrowStrength Counter] > Visibility > Make object invisible

OK, now let's have a little review of what we have just done... As soon as the frame starts, the game:

- **Plays the selected MIDI file** (I'd suggest looping it continuously);
- Starts to **ignore players' control** (players can't move their Globbs until each player's turn starts – we'll find out more about this later on, while coding the turn system);
- **Destroys the original "Grenade" object** (we don't need it wandering around the frame, as we will create new instances of the grenade whenever the player will command his Glob to throw one, using the "create new object" command);
- **Changes the Alterable Value B of "BluePlayer" to "16"** (this has a lot more sense if you remember that the Alterable Value B is the AV that we wanted to use to store the direction that the rotating targeting-marker is set to – in other words: this little action makes the targeting-marker look to the left when BluePlayer will start his first turn);
- **Sets the "WhichPlayer" counter to "2"** (you'll find out more about this soon enough, but let me just say that this means that player 1 will have his turn as soon as the "stand by" period is over)
- **Makes the "ThrowStrength" counter invisible** (this is purely aesthetical).

Note that if you don't have the "Player 2" object in your objects list in the Event editor, you've most probably forgot to set the "BluePlayer" object's control to "Player 2".

Take a look at the image below, this is how our Event editor should look like by now (note that it can differ, since you could have created all the objects and/or events in a different order – so don't panic if it doesn't look exactly like this):

hot-spot trick - You can take a look at it's animations a bit later, just to fully understand how this was possible). Take a look at the image below to have a visual hint on what we've just created:



This was quite easy, wasn't it? Save your project, then.

5) Let's move along – time to decide what will happen to our Globs when they will be hit by a grenade blast (by the “**Boom**” object – note that only a collision with the “Boom” object will make our Globs sustain damage – nothing will happen on collision between the Globs and the grenade itself):

IF: [Boom Object] > Collisions > Another object > [RedPlayer]

THEN: [Sound object] > Samples > Play sample (pick something from MMF2's sound libraries)

THEN: [RedPlayer] > Animation > Change > Animation sequence > Hit

THEN: [RedHealth counter] > Subtract from counter (subtract 15 from this counter)

As you can see, a collision between the “RedPlayer” and the “Boom” object will result in subtracting 15 points from the “RedHealth” counter. Now it's time to do the same for the “BluePlayer” object. Create a new event:

IF: [Boom Object] > Collisions > Another object > [BluePlayer]

THEN: [Sound object] > Samples > Play sample (pick something from MMF2's sound libraries)

THEN: [BluePlayer] > Animation > Change > Animation sequence > Hit

THEN: [BlueHealth counter] > Subtract from counter (subtract 15 from this counter)

Don't worry that we haven't got the “Boom” object on stage yet – it will be created in due time, when we'll establish all the needed groups.

6) Another event – this one plays a sample (only once, thanks to the limiting condition of the “Only one action when event loops” – this condition can really be quite useful, so get to know it a bit better at your free time) and destroys the Glob if it's health goes all the way to zero:

IF: [RedHealth counter] > Compare the counter to a value (counter's value is equal 0)

IF: [Special Object] > Limit conditions > Only one action when event loops

THEN: [Sound object] > Samples > Play sample (pick something from MMF2's sound libraries)

THEN: [RedPlayer] > Destroy

And here's the same thing for Player 2:

IF: [BlueHealth counter] > Compare the counter to a value (counter's value is equal 0)
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [Sound object] > Samples > Play sample (pick something from MMF2's sound libraries)
THEN: [BluePlayer] > Destroy

7) These two events control what to do when one (or both) of the Globs has been destroyed (the game just goes to the next frame – if there is a next frame, that is. If there's not – the game will most probably just exit the application):

IF: [RedPlayer] > Pick or count > Have all "RedPlayer" been destroyed
THEN: [Storyboard Controls] > Next frame

IF: [BluePlayer] > Pick or count > Have all "BluePlayer" been destroyed
THEN: [Storyboard Controls] > Next frame

OK. We're now ready to play with our groups. We shall create four – one called "Stand by – change seats" (which will contain all the events that "happen" in-between the players' turns), the second one called "RED player's turn", "BLUE player's turn" as the 3rd one and the last one: "Throwing a grenade".

8) Create a new **event group**, name it "**Stand by – change seats**" and make sure that the "**Active when the frame starts**" option is **ON**. When the game starts, this will be the only group activated, giving the players' a couple more seconds to prepare for combat. It would be best to create the three other groups now as well: create three more groups, calling them "**RED player's turn**", "**BLUE player's turn**" and "**Throwing a grenade**", but remember to switch the "Active when the frame starts" option **OFF** in all of them.

Let's get back to the first group we've created. As soon as you're ready, create these events inside the "**Stand by – change seats**" group:

IF: [Special Object] > Group of events > On group activation
THEN: [Sound object] > Samples > Play sample (select the "Teeth chatter" file from MMF2's sound libraries)
THEN: [Player 1] > Player Control > Ignore control
THEN: [Player 2] > Player Control > Ignore control
THEN: [Targeter Object] > Visibility > Make Object Invisible
THEN: [Countdown Counter] > Set Counter (set the Counter to 3)
THEN: [ThrowStrength Counter] > Visibility > Make object invisible
THEN: [Displayer String Object] > Text > Set font size (set font size 8, keep current border size)
THEN: [Displayer String Object] > Text > Set bold (set bold effect ON)
THEN: [Displayer String Object] > Change alterable string (input: "GET READY!")

This will set all these parameters as soon as this group is activated – just inspect them on your own, they're pretty easy to understand. Let's go further on, and create another event, shall we?

IF: **[The Timer Object] > Every** (set the timer to "every 1.00 second")

THEN: **[Countdown Counter] > Subtract from Counter** (subtract 1 from counter)

Now we have our countdown timer ready, as far as this group goes. There are still two events to program before we can move to the next group, though:

IF: **[Countdown Counter] > Compare the counter to a value** (counter is equal 0)

IF: **[WhichPlayer Counter] > Compare the counter to a value** (counter is equal 1)

THEN: **[Special Object] > Group of events > Activate** (select the "BLUE Player's turn" group)

THEN: **[Special Object] > Group of events > Deactivate** (select the "Stand by – Change seats" group)

IF: **[Countdown Counter] > Compare the counter to a value** (counter is equal 0)

IF: **[WhichPlayer Counter] > Compare the counter to a value** (counter is equal 2)

THEN: **[Special Object] > Group of events > Activate** (select the "RED Player's turn" group)

THEN: **[Special Object] > Group of events > Deactivate** (select the "Stand by – Change seats" group)

And that's all here, folks! Remember that all these events have to be **INSIDE the group!** Don't forget!

9) Having the "Stand by – Change seats" group properly set up, let's go to the second group on our list – the one called "**RED Player's turn**". Create this event inside:

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

THEN: **[Sound object] > Samples > Play sample** (select a simple "pop" sound from MMF2's sound libraries)

THEN: **[Player 1] > Player Control > Restore control**

THEN: **[Player 2] > Player Control > Ignore control**

THEN: **[RedPlayer] > Alterable Values > Set** (set the Alterable Value A to 0)

THEN: **[Targeter Object] > Direction > Select direction** (a dialogue box will open – press the "calculate direction" button and input: **Alterable Value B("RedPlayer")** - this will retrieve RedPlayer's Alterable Value B – if you have changed it's name, either input that name instead of "Alterable Value B" in this expression, or use the "Retrieve data from object" button and retrieve the AV directly from "RedPlayer")

THEN: **[Targeter Object] > Visibility > Make Object Reappear**

THEN: **[Countdown Counter] > Set Counter** (set the Counter to 12)

THEN: **[WhichPlayer Counter] > Set Counter** (set the Counter to 1)

THEN: **[ThrowStrength Counter] > Set Counter** (set the Counter to 1)

THEN: **[Displayer String Object] > Text > Set font size** (set font size 24, keep current border size)

THEN: **[Displayer String Object] > Text > Set bold** (set bold effect OFF)

Quite massive, isn't it? It sets up all the parameters that this group needs to function correctly.

Now, let's create another event, this time with the "Always" condition (this will – as long as this group is active - control our vertical scrolling, position of the "Targeter" object and displaying the "Countdown" counter's value in the "Displayer" string):

IF: [Special Object] > Always

THEN: [Storyboard Controls] > Scrollings > Center horizontal position of window in frame (input in the Expression editor: `X("RedPlayer")` – this will retrieve RedPlayer's X position)

THEN: [Targeter Object] > Position > Select position (select [2, -18] relatively to RedPlayer)

THEN: [Displayer String Object] > Change alterable string (input in the Expression editor: `Str$(value("Countdown"))` – this will display the "Countdown" counter's value in our string)

Note that you can recreate all these expressions by using the "retrieve value from object" button and browsing the objects list for yourself.

Let's get another event working in this group (this will make the "Countdown" counter work properly):

IF: [The Timer Object] > Every (set the timer to "every 1.00 second")

THEN: [Countdown Counter] > Subtract from Counter (subtract 1 from counter)

Here's an event that will help us control the cycle of turns in our game:

IF: [Countdown Counter] > Compare the counter to a value (counter is equal 0)

THEN: [Special Object] > Group of events > Activate (select the "Stand by – Change seats" group)

THEN: [Special Object] > Group of events > Deactivate (select the "RED Player's turn" group)

We're almost at the end (as far as this group is concerned)! It's time to create three little events that will control the process of throwing grenades at your opponent:

IF: [Keyboard & Mouse Object] > The Keyboard > Upon pressing a key (press the "CTRL" key)

THEN: [RedPlayer] > Alterable Values > Set (set the Alterable Value A to 1)

THEN: [RedPlayer] > Alterable Values > Set (set the Alterable Value B using this expression: `Dir("Targeter")` – this will save the current direction number of "Targeter" to the Alterable Value B of the "RedPlayer" object)

THEN: [ThrowStrength Counter] > Visibility > Make object Reappear

IF: [Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed (press the "CTRL" key)

THEN: [ThrowStrength Counter] > Add to Counter (Add 1 to counter)

IF: [RedPlayer] > Alterable Values > Compare to one of the Alterable Values (Alterable Value A is equal 1)

IF: [NEGATE] [Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed (press "CTRL")

THEN: [Player 1] > Player Control > Ignore control

THEN: [Special Object] > Group of events > Activate (select the "Throwing a grenade" group)

THEN: [Special Object] > Group of events > Deactivate (select the "RED Player's turn" group)

Note: to negate, firstly create the condition that you want to negate - such as our "repeat while key is pressed" condition - then right-click on it and select "negate" from the pop-up menu. Not all conditions can be negated, but a lot of them can (i.e. "Check for mouse pointer in a zone", "Check for mouse pointer over an object", "Overlapping another object", all conditions that come from the Sound object, the Array object etc.). To learn more – just check your MMF2 help file.

Let's save our project and move on...

10) We've just finished creating the "RED Player's turn" group. Now let's do the same for the "**BLUE Player's turn**" one – it consists of the same events, but there are some conditions and actions that differ a bit from the "RedPlayer" version (which is pretty obvious). If something's not working the way it should, double-check if the actions correspond to the right object:

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

THEN: [Sound object] > Samples > Play sample (select a simple "pop" sound from MMF2's sound libraries)

THEN: [Player 1] > Player Control > Ignore control

THEN: [Player 2] > Player Control > Restore control

THEN: [BluePlayer] > Alterable Values > Set (set the Alterable Value A to 0)

THEN: [Targeter Object] > Direction > Select direction (a dialogue box will open – press the "calculate direction" button and input: **Alterable Value B("BluePlayer")** - this will retrieve BluePlayer's Alterable Value B)

THEN: [Targeter Object] > Visibility > Make Object Reappear

THEN: [Countdown Counter] > Set Counter (set the Counter to 12)

THEN: [WhichPlayer Counter] > Set Counter (set the Counter to 2)

THEN: [ThrowStrength Counter] > Set Counter (set the Counter to 1)

THEN: [Displayer String Object] > Text > Set font size (set font size 24, keep current border size)

THEN: [Displayer String Object] > Text > Set bold (set bold effect OFF)

IF: [Special Object] > Always

THEN: [Storyboard Controls] > Scrollings > Center horizontal position of window in frame (input in the Expression editor: **X("BluePlayer")** – this will retrieve BluePlayer's X position)

THEN: [Targeter Object] > Position > Select position (select [2, -18] relatively to BluePlayer)

THEN: [Displayer String Object] > Change alterable string (input in the Expression editor:
Str\$(value("Countdown")) – this will display the "Countdown" counter's value in our string)

IF: [The Timer Object] > Every (set the timer to "every 1.00 second")

THEN: [Countdown Counter] > Subtract from Counter (subtract 1 from counter)

IF: [Countdown Counter] > Compare the counter to a value (counter is equal 0)

THEN: [Special Object] > Group of events > Activate (select the "Stand by – Change seats" group)

THEN: [Special Object] > Group of events > Deactivate (select the "BLUE Player's turn" group)

IF: **[Keyboard & Mouse Object] > The Keyboard > Upon pressing a key** (press the "CTRL" key)
THEN: **[BluePlayer] > Alterable Values > Set** (set the Alterable Value A to 1)
THEN: **[BluePlayer] > Alterable Values > Set** (set the Alterable Value B using this expression: **Dir("Targetter")** – this will save the current direction number of "Targetter" to the Alterable Value B of the "BluePlayer" object)
THEN: **[ThrowStrength Counter] > Visibility > Make object Reappear**

IF: **[Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed** (press the "CTRL" key)
THEN: **[ThrowStrength Counter] > Add to Counter** (Add 1 to counter)

IF: **[BluePlayer] > Alterable Values > Compare to one of the Alterable Values** (Alterable Value A is equal 1)
IF: **[NEGATE] [Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed** (press "CTRL")
THEN: **[Player 1] > Player Control > Ignore control**
THEN: **[Special Object] > Group of events > Activate** (select the "Throwing a grenade" group)
THEN: **[Special Object] > Group of events > Deactivate** (select the "BLUE Player's turn" group)

OK, only one group to go! I hope that you remembered to save your app from time to time?

11) Let's get to the grenade-throwing part. Create a new event inside the "Throwing a grenade" group:

IF: **[Special Object] > Group of events > On group activation**
IF: **[WhichPlayer Counter] > Compare the counter to a value** (counter is equal 1)
THEN: **[Sound object] > Samples > Play sample** (select a sound from MMF2's libraries, i.e. "Whoosh quick 2")
THEN: **[Create new objects] > Create object > [Grenade]** (create it at [0,-25] relative coordinates from the "RedPlayer" object)
THEN: **[Player 1] > Player Control > Ignore control**
THEN: **[Player 2] > Player Control > Ignore control**
THEN: **[Grenade] > Movement > Set speed** (use this expression: **value("ShotStrength")+30**)
THEN: **[Grenade] > Direction > Select direction** (a dialogue box will open – press the "calculate direction" button, input this: **Dir("Targetter")** and press "OK")
THEN: **[Grenade] > Alterable Values > Set** (set the Alterable Value A to 0)
THEN: **[Targetter Object] > Visibility > Make Object Invisible**
THEN: **[ThrowStrength Counter] > Visibility > Make Object Invisible**
THEN: **[Displayer String Object] > Text > Set font size** (set font size 8, keep current border size)
THEN: **[Displayer String Object] > Text > Set bold** (set bold effect ON)
THEN: **[Displayer String Object] > Change Alterable String** (change Alterable String to "FIRE IN THE HOLE!")

When that's done, let's create another event, similar to the one above:

IF: **[Special Object] > Group of events > On group activation**
IF: **[WhichPlayer Counter] > Compare the counter to a value** (counter is equal 2)
THEN: **[Sound object] > Samples > Play sample** (select a sound from MMF2's libraries, i.e. "Whoosh quick 2")
THEN: **[Create new objects] > Create object > [Grenade]** (create it at [0,-25] relative to the "BluePlayer" object)
THEN: **[Player 1] > Player Control > Ignore control**

THEN: [Player 2] > Player Control > *Ignore control*

THEN: [Grenade] > Movement > *Set speed* (use this expression: `value("ShotStrength")+30`)

THEN: [Grenade] > Direction > *Select direction* (a dialogue box will open – press the “calculate direction” button, input this: `Dir("Targeter")` and press “OK”)

THEN: [Grenade] > Alterable Values > *Set* (set the Alterable Value A to 0)

THEN: [Targeter Object] > Visibility > *Make Object Invisible*

THEN: [ThrowStrength Counter] > Visibility > *Make Object Invisible*

THEN: [Displayer String Object] > Text > *Set font size* (set font size 8, keep current border size)

THEN: [Displayer String Object] > Text > *Set bold* (set bold effect ON)

THEN: [Displayer String Object] > *Change Alterable String* (change Alterable String to “FIRE IN THE HOLE!”)

This little event will control the horizontal scrolling while the “Throwing a grenade” group is active:

IF: [Special Object] > *Always*

THEN: [Storyboard Controls] > Scrollings > *Center horizontal position of window in frame* (input in the Expression editor: `X("Grenade")` – this will retrieve “Grenade” object’s X position)

And here comes the “Grenade” object’s collision detection:

IF: [Grenade] > Collisions > *Backdrop*

IF: [Grenade] > Movement > *Compare speed of “Grenade” to a value* (is greater or equal 4)

IF: [Special Object] > Limit conditions > *Only one action when event loops*

THEN: [Sound object] > Samples > *Play sample* (select a sound from MMF2’s libraries, i.e. “IMPACT02.wav”)

THEN: [Grenade] > Movement > *Bounce*

THEN: [Grenade] > Alterable Values > *Add to* (add 2 to the Alterable Value A)

Note that the event above is true ONLY when our Grenade’s speed is greater or equal 4. Here’s an event that will control the situation when it’s speed is a bit lower:

IF: [Grenade] > Movement > *Compare speed of “Grenade” to a value* (is lower than 4)

IF: [Grenade] > Collisions > *Backdrop*

IF: [Special Object] > Limit conditions > *Only one action when event loops*

THEN: [Sound object] > Samples > *Play sample* (select a sound from MMF2’s libraries, i.e. “IMPACT02.wav”)

THEN: [Grenade] > Movement > *Stop*

THEN: [Grenade] > Movement > Multiple movements > *Select movement* (select Movement #2)

Here’s an event that will bounce the Grenade when it will hit the “Interface” object:

IF: [Grenade] > Collisions > *Another object* > [Interface]

THEN: [Sound object] > Samples > *Play sample* (select a sound from MMF2’s libraries, i.e. “IMPACT09.wav”)

THEN: [Grenade] > Movement > *Bounce*

Another event, this time it's for keeping the "Grenade" object inside the frame:

IF: [Grenade] > Position > Test position (a dialogue box will open – select the "leaves in the right", "leaves in the left", "leaves in the top" and "leaves in the bottom" buttons and click "OK" to create a "leaves the play area" condition)

THEN: [Sound object] > Samples > Play sample (select a sound from MMF2's libraries, i.e. "IMPACT02.wav")

THEN: [Grenade] > Movement > Bounce

The grenade will explode when its Alterable Value A will be equal or greater than 5. I guess that you've already noticed that bouncing of the backdrop makes it a bit closer to explosion? Let's set up an event that will also trigger the explosion on timer:

IF: [The Timer Object] > Every (set the timer to "every 1.00 second")

THEN: [Grenade] > Alterable Values > Add to (add 1 to the Alterable Value A)

And here comes the "big boom event"! As soon as the grenade bounces a bit and the time passes, it's Alterable Value A comes to the "explosion point" – becomes equal or greater than 5. And then...

IF: [Grenade] > Alterable Values > Compare to one of the Alterable Values (Alterable Value A is equal or greater than 5)

IF: [Special Object] > Limit conditions > Only one action when event loops

THEN: [Sound object] > Samples > Play sample (select a sound from MMF2's libraries, i.e. "EXPLOD03.wav")

THEN: [Create new objects] > Create object > [Boom Object] (create it at [3,12] relative to the "Grenade" object)

THEN: [Grenade] > Destroy

THEN: [Grenade] > Visibility > Make Object Invisible

THEN: [Displayer String Object] > Change Alterable String (change Alterable String to "BOOOM!")

Nice explosion, was it? Sure it was. Now it's time to wrap it up with these here little fellows, which are intended to finish what the "Boom" event started:

IF: [Boom Object] > Animation > Has an animation finished? (select the "Appearing" animation)

THEN: [Boom Object] > Destroy

IF: [Grenade] > Pick or count > Have all "Grenade" been destroyed

THEN: [Special Object] > Group of events > Activate (select the "Stand by – Change seats" group)

THEN: [Special Object] > Group of events > Deactivate (select the "Throwing a grenade" group)

And that's it!

We're finally over!

Take a look at your application – save it, run it and play it! Isn't it fun? Sure it is! Now go and find someone who will play the "Glob Wars" game with you. Either that, or you may want to enhance this game a bit, adding new features, weapons, making either the window size or the frame size bigger, adding jet-packs and weapon supplies that drop down from the sky... Take it to a whole new level, if you wish!

Have fun!

Cheers!

Koobare

marchewkowy@gmail.com

