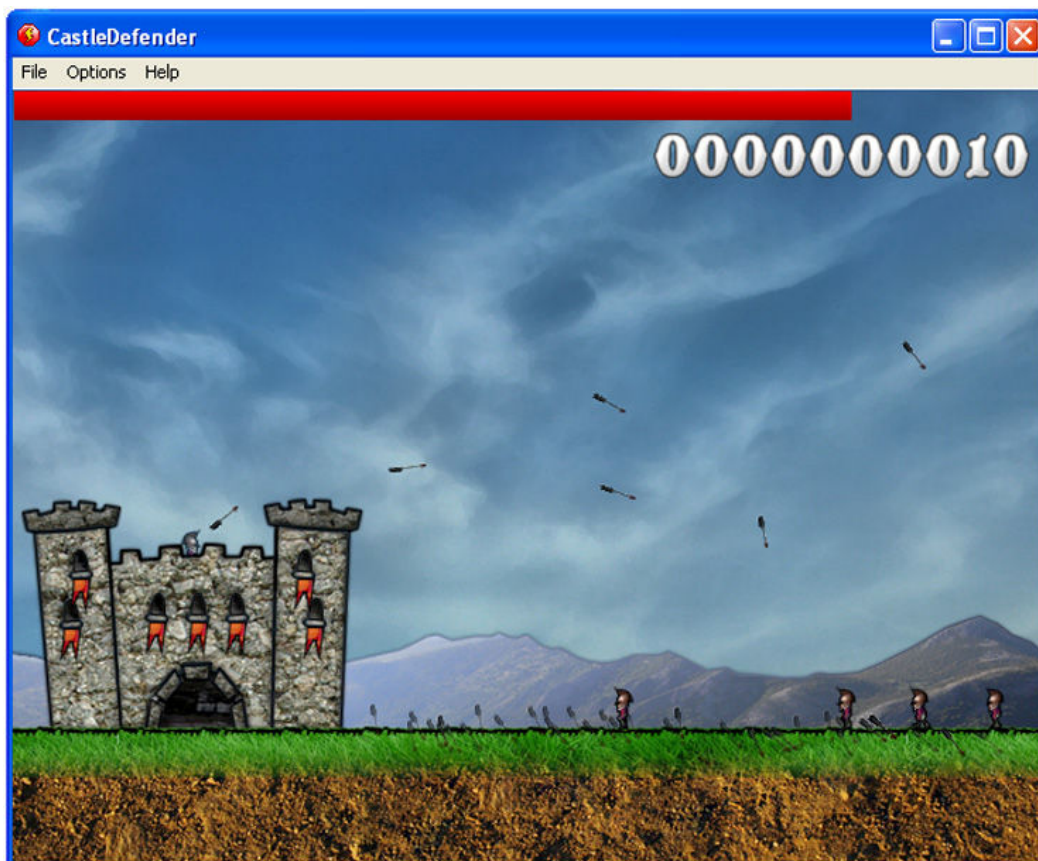


Castle Defender



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

Koobare
marchewkowy@gmail.com

Welcome to another one of Koobare's little tutorials (this time: beginner-intermediate level. Take a look at the "Glob Wars" and "Space Corsair" tutorial first), teaching you – as always – how to effectively and efficiently use the best multimedia authoring tool ever – [Multimedia Fusion 2](#) by Clickteam! Many moons have passed since our last gathering took place in between these cold stone walls, but I am a firm believer that today we shall create something greater than the stars themselves! Well... Maybe not. Let's not get TOO cocky about this, it's just a simple tutorial game after all...

Having said that, there's still a lot to learn from this lil' thing, a quantity of high-quality material to practice. We'll create a small "defend the castle" game, in which our protagonist – a lone archer left behind to guard the fortress of Haylenscalf – will try to hold his post no matter what, fighting off hordes of enemy warriors with his trusty bow and arrows.

The true story of the Battle of Haylenscalf

It isn't easy, being a medieval archer. There you are, sitting with your fellow fighters around the fire, feasting on a roasted pig, drinking ale, polishing your bow, singing some crazy tunes, generally having the time of your life. And then HE arrives at the courtyard, his horse galloping past the gate... Sir Givakiss the Praiseworthy, in his silver armor, all shiny and sparkly, like the prince charming he really is. "Men", he says, "today... We march for glory!" You know what that means: he's in a quarrel with his cousin, Sir Evilian the Dastardly, the malicious ruler of Oldstoryland... Again.

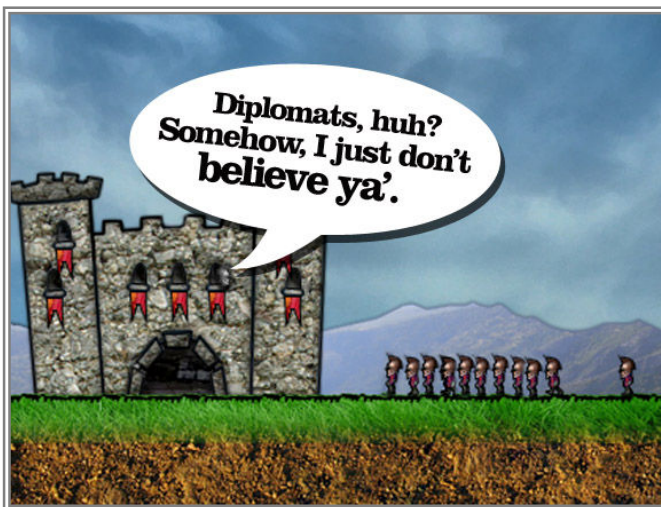
Perhaps Evilian was the one who served tea without a lump of sugar this time, maybe it's all about who's got a better blade hanging above his fireplace... Anyway, it'll play out in the same way as always: they'll both bring their forces to the field of Doggysmells, where another battle will be held. When all the neighboring meadows and rivers will be filled with slain soldiers, the loser – the king with the lesser amount of subjects left alive – will have to apologize to the other one and they will make up like nothing ever happened.

Anyways, you're prepping your weapon and your helmet, ready to march out, when Givakiss walks to you, places his heavy hand on your shoulder and tells you that you will be left behind to guard the castle from potential thieves. Yup, that's exactly what he told you – you're going to

miss out on this little escapade. “Yesss!”, you shouted in your mind. It was like winning the lottery, without even buying a lottery ticket: from the whole army of Haylenscalf you were the one who received a *stay-back-and-enjoy-yourself-while-we-die-on-the-battlefield* free pass. All these fighters were marching out to battle and you’re about to hang out with barrels of ale and tons of delicious roasted meat!

And off they went, to the field of Doggysmells. Givakiss didn’t even lock the main gate, they were in such a hurry. You just chilled out on the wall, catching a nice tan in the midday sun, leaning on your bow. Suddenly... You felt trouble brewing around the hill. It was like spider-sense... But without a radioactive spider-bite, that is. You got up and then... You saw them in the distance. Evilian’s men! He must have decided not to go to Doggysmells, but to organize a sneak attack at Haylenscalf! Thousands of soldiers marching to your position! You had no choice but to defend the fortress, armed only with your bow. This is how it all started, how the long and bloody Battle of Haylenscalf found its way to the annals of history...

What we’re trying to achieve here...

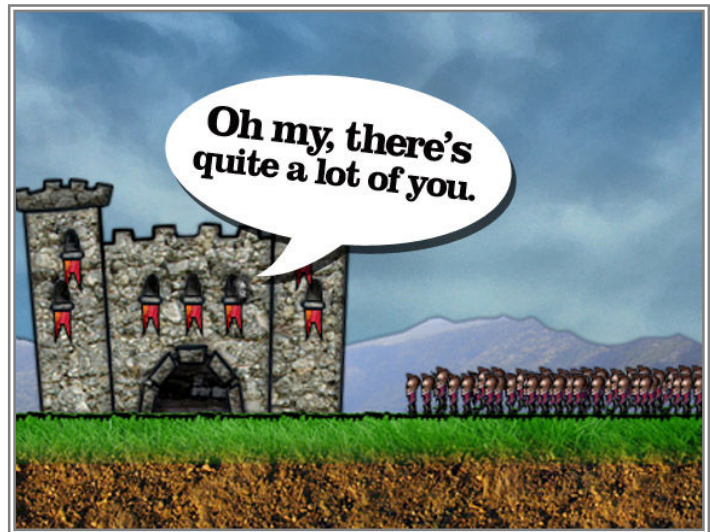


This will be a simple, yet satisfying, game. Our player has to defend the castle of Haylenscalf by shooting hundreds of arrows into the direction of incoming enemies, soldiers of Evilian the Dastardly, obsessed with storming the fortress, no matter what the cost. In this version of the game you won’t find a crystal clear objective to achieve – you just shoot down more and more enemies and they just keep on coming,

multiplying your score until there’s just too many of them to hold the fortress. I encourage you to play a little bit with our creation when you’re done with this tutorial – after you’ve completed it, try to upgrade “Castle Defender” with your own ideas and concepts, add new weapons, functions, levels and objectives. Who knows, perhaps if you hold your ground long enough in your version of the game, Sir Givakiss will return and Haylenscalf will be saved?

Our player will have seven “lives”, symbolized by seven red flags hanging from the windows of Haylenscalf. Every time an enemy enters the castle, one of the flags will turn gray. Seven gray

flags – and you’ve lost the fortress to Evilian. That means Game over, Sire – the whole application restarts itself (in your version you can add a nice “You’ve lost!” screen instead). As for the scoring system: every time you hit the enemy with one of your arrows – you receive 10 points. Every time you shoot an arrow – you subtract 1 from the overall score. The more enemies you shoot with less arrows used to do so – the more points you get in the end.



Let’s take a look at our game’s controls... Arrows will be shot by pressing the space bar. The left and right cursor keys will be used to control the main character (he can only move on the castle’s roof). The up and down arrow keys will be used to control the force of the player’s projectiles – by pressing and holding them you can increase or decrease the strength of the shot (and the resulting trajectory).

Notes about this tutorial

Before we move on – a word about this tutorial itself. Usually I try to get you through the whole creation process step by step, from the very beginning to the very end, including the whole “let’s import all the objects” and “let’s set up the application” routines. This time I’ve decided to skip them – this tutorial is meant for beginner-intermediates and it would be too similar to what you had to do in the “Glob Wars” and “Space Corsair” tutorials (available on Clickteam’s website). That’s why I encourage you to start with those two, and then – upon their completion – move on to this one. Anyways, let’s get moving...

- 📄 **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

 **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 marchewkowy@wp.pl. I'll do my best to check both these e-mails regularly.

Part I: Everything is already set up for us... Ain't life grand?

Usually I would start with the traditional "let's create a new application" text, but this time we're skipping this part and we're moving right to the good stuff. That's why I prepared the whole application for you – it already has all the objects imported into it, all you have to do is to check them out and then program their behavior in the Event Editor.

So... Open up the MMF2 file entitled "**CastleDefender-start.mfa**" – it should be packed in the same archive as this PDF tutorial. It has all the needed objects configured and placed into the right positions, everything's just waiting for you to move in and put some life into it.

Note that I've set my application to display the standard 16 million colors in a 640x480 window. I've also set up two **Global values**: the first one is titled "**Shots**" (it will count how many arrows were fired by the player – this isn't really changing anything in our tutorial game, but you may find it useful when constructing player statistics in your own upgraded versions of this project), the second one is named "**Kills**" (this one actually has an influence on how the game plays out – it counts how many enemies were killed already).

Please note: most of the objects use alpha channels, a feature that is unavailable in Games Factory 2 (TGF2 users should use basic library objects or create their own graphics instead – I generally would advise you to upgrade to MMF2 as soon as possible, since you're missing out on some really good stuff, a lot of quite impressive features).

Part II: Know thy objects, Sire.

It's now time to learn something about the objects that are laying around in our frame. Open up the Frame Editor and take a look at all those actives, backdrops and counters... Wouldn't you like to know more about them? Sure you would! Let's do this in an organized manner, alphabetically:

Object's name:	So... What is it?
arrow	<u>Active object that will be used as our player's projectile.</u> What's worth noticing is that its movement is set to Pinball movement (with Gravity set to 56, Deceleration set to 0, Initial Speed set to 100, Initial directions set to 1,2,3,4,5,6) and that its animation has a single image rotated to 32 directions.
attackers	<u>A hidden counter showing how many attackers have broken through.</u> A counter that is used by our game to count how many enemies have entered the castle (and thus how many lives has the player still left). It is also used to turn the red flags gray in the right moment.
backdrop	<u>Just your usual backdrop picture, 640x480 in size. It's a colorful picture with a castle, grassy ground, mountains in the distance and a bit of blue sky.</u>
castle-element	<u>A piece of the castle walls.</u> Active object displayed in front of the player's character to show that he moves behind a wall.
defender	<u>Player's character – a little blue archer. He moves with the Eight Directions movement, constricted to directions 0 and 16 (left and right).</u>
doorway	<u>A small active object that is hidden throughout the game. If one of the enemies bumps into it – it means that he has just entered the castle.</u>
earth	<u>Active object that acts as the ground. When arrows collide with this object they are immediately pasted onto the background and destroyed.</u>
enemy	<u>Just your standard enemy grunt. These guys are trying to storm the fortress, dying from the arrows shot by our main protagonist. Their movement is set to Path Movement, going all the way left.</u>
flag (and flag 2-7)	<u>Red flags used to display how many lives has the player left. Every time an enemy enters the castle, one of these flags turns gray.</u>

points	<i>A “Numbers” type counter set to display the score of the game. It has its “fixed number of digits” option set to 8.</i>
power-counter	<i>Another counter, this time a horizontal bar. It controls with how much strength are the arrows shot with.</i>
wall	<i>A small active object that is hidden throughout the game. It is used to constrict player’s movements, making it impossible for his character to fall down from the roof, or rather to start “floating” around the screen.</i>

Part III: Programmer’s delight.

It’s time to get to the fun part of this 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 the event-recording system that I use. If you know it already – just skip this frame below and quickly move on to the coding part:

Koobare’s MMF-to-paper coding system

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

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

Seems simple, right? Well, that’s just because IT IS simple.

All the conditions are marked in red, while actions are written in fancy blue. Object names are always put in [square brackets]. The final condition/action is always in *Italic*.

If we’ll have a multi-condition event, then it’ll be like this:

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

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

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

Whereas a multi-action event looks like this:

IF (Condition): **[Object for condition] > Condition group > Condition**
THEN (Action 1): **[Object for the action 1] > Action group 1 > Action 1**
THEN (Action 2): **[Object for the action 2] > Action group 2 > Action 2**

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...

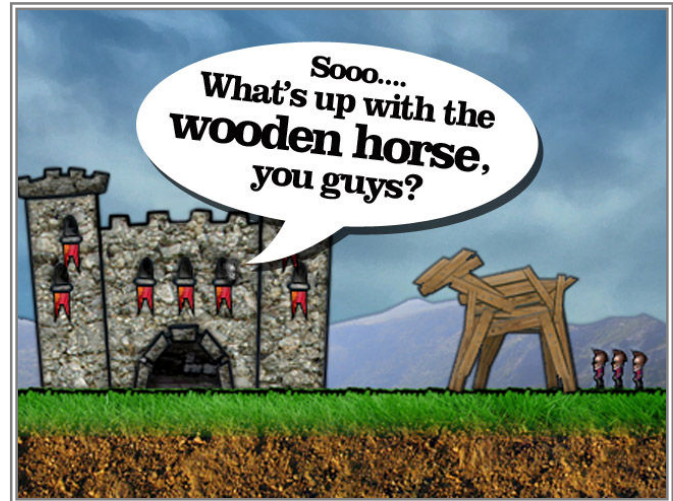
Making it all work...

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 battle tunes in the background.

IF: **[Storyboard Controls] > Start of frame**
THEN: **[Sound Object] > Music > Play and loop music**
<< Select any music from MMF2's library, set the loop to 0 times – a continuous loop.
I chose the "gothic.mid" track, which I think has a nice battle pace to it >>

And that's that – we've got our first event ready & waiting. Wasn't it all too easy? Sure it was, but don't get too cocky about it – there's still much to be done.

2) Time for the second event – this one (and the next six ones) may seem a bit dull and uninteresting, as its only purpose is to control the color of the flag that represents one of our player's lives. In other words – we have to make sure that those flags turn gray when an enemy soldier enters the fortress of Haylenscalf....



Here's how it's done:

```
IF: [attackers] > Compare the counter to a value
< compare whether it is Equal 1 >
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [flag] > Visibility > Change Ink Effect
<< Choose Monochrome >>
```

Now, let's do the same for the next six flags, each one of them representing one of our player's lives:

```
IF: [attackers] > Compare the counter to a value
< compare whether it is Equal 2 >
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [flag 2] > Visibility > Change Ink Effect
<< Choose Monochrome >>
```

```
IF: [attackers] > Compare the counter to a value
< compare whether it is Equal 3 >
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [flag 3] > Visibility > Change Ink Effect
<< Choose Monochrome >>
```

```
IF: [attackers] > Compare the counter to a value
< compare whether it is Equal 4 >
IF: [Special Object] > Limit conditions > Only one action when event loops
THEN: [flag 4] > Visibility > Change Ink Effect
<< Choose Monochrome >>
```

IF: [attackers] > Compare the counter to a value

< compare whether it is *Equal 5* >

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

THEN: [flag 5] > Visibility > Change Ink Effect

<< Choose *Monochrome* >>

IF: [attackers] > Compare the counter to a value

< compare whether it is *Equal 6* >

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

THEN: [flag 6] > Visibility > Change Ink Effect

<< Choose *Monochrome* >>

IF: [attackers] > Compare the counter to a value

< compare whether it is *Equal 7* >

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

THEN: [flag 7] > Visibility > Change Ink Effect

<< Choose *Monochrome* >>



Got it? Great! Let's move on to the next thing. Now, if the "attackers" counter is equal 8 – our application should be automatically restarted:

IF: [attackers] > Compare the counter to a value
< compare whether it is Greater or equal 8 >
THEN: [Storyboard Controls] > Restart the application

3) Once that's done, let's define the keyboard controls for our little project:

IF: [Keyboard & Mouse Object] > The Keyboard > Upon pressing a key
<< press SPACE on your keyboard >>
THEN: [Create New Objects] > Create Object
<< Select the [arrow] object >>
<< Set the coordinates to x=1, y=-15 relatively to the [defender] object >>
THEN: [arrow] > Movement > Set Speed...
< Type this in: value("power-counter") >
THEN: [Sound Object] > Samples > Play sample
<< Choose a quiet "shooting" sound – I chose the "Crunch" sound in my example >>
THEN: [Special Object] > Change a global value > Add to
<< Choose value Shots >>
< input: 1 >
THEN: [points] > Subtract from Counter
<< input: 1 >>

Let's take a look at what we've just done: upon pressing the space bar our protagonist (the "defender" object) shoots an arrow with the speed defined by the "power-counter" (which influences its final trajectory). At the same time a "shooting" sample is played, the game adds 1 to the "Shots" global value and subtracts 1 from the "Points" counter.

Now, let's move on to the next keyboard-related events...

IF: [Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed
<< press UP ARROW on your keyboard >>
THEN: [power-counter] > Add to Counter
<< input: 1 >>

IF: [Keyboard & Mouse Object] > The Keyboard > Repeat while key is pressed
<< press DOWN ARROW on your keyboard >>
THEN: [power-counter] > Subtract from Counter
<< input: 1 >>

4) Once that's done, we should define what happens when active objects in our game collide with other active objects. Firstly, let's make sure that our hero won't escape from the roof of the castle, where we have placed him...

IF: [defender] > Collisions > Another object > [wall]

THEN: [defender] > Movement > Bounce

Yup, now he'll be prisoned on that roof for eternity.

5) Time to instruct our game what to do when one of the enemies enters the castle:

IF: [enemy] > Collisions > Another object > [doorway]

THEN: [attackers] > Add to Counter

<< input: 1 >>

THEN: [enemy] > Visibility > Make object invisible

THEN: [enemy] > Destroy

As you can see, the most important part of the event seen above is the fact that we add 1 to the "attackers" counter. This will be pretty significant a bit later on.

6) Here's what happens when one of our arrows hits the ground:

IF: [arrow] > Collisions > Another object > [earth]

THEN: [arrow] > Visibility > Change Ink Effect

<< Choose Semi-Transparent >>

<< set it to 50% >>

THEN: [arrow] > Animation > Paste into background

THEN: [arrow] > Destroy

THEN: [Sound Object] > Samples > Play sample

<< Choose a quiet "hit" sound – I chose the "CHOP3" sound in my example >>

To sum it up a bit: when the collision takes place the arrow firstly makes itself semi-transparent, pastes itself onto background (creates a backdrop object), then destroys itself. Additionally, a sample is played to illustrate the impact.

7) Yet another collision ahead of us – this time we'll define what should happen when the arrow actually hits the destined target, one of Evilian's men:

[illegible]

9) OK, now it's time to create 5 Event Groups – name them “Starter”, “After 25”, “After 50”, “After 100” and “After 200” respectively. Make sure that all groups – except the first one, “Starter” – are **NOT** set to be active when the frame starts. This should look a bit like this:

Starter	
After 25	
After 50	
After 100	
After 200	
New condition	

10) Now, open up the first group (“Starter”) and create these events inside:

```
IF: [The Timer Object] > Every
<< Set the timer to 1.85 second >>
THEN: [Create New Objects] > Create Object
<< Select the [enemy] object >>
<< Set the coordinates to x=673, y=397 >>

IF: [Special Object] > Compare to a global value
<< Choose value Kills >>
< compare if it is Greater or equal to 25 >
IF: [Special Object] > Limit conditions > Run this event once
THEN: [Special Object] > Group of events > Activate
<< Select the After 25 group >>
```

As you can see here, once the player has killed at least 25 enemy soldiers, the next group of events (“After 25”) will be activated, and enemies will be coming in more often.

11) Open up the next event group – “After 25”. We’ll have to create two events that are very similar to the ones we’ve just completed:

IF: [The Timer Object] > Every

<< Set the timer to 0.90 of a second >>

THEN: [Create New Objects] > Create Object

<< Select the [enemy] object >>

<< Set the coordinates to x=692, y=397 >>

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

<< Choose value *Kills* >>

< compare if it is *Greater or equal* to 50 >

IF: [Special Object] > Limit conditions > Run this event once

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

<< Select the *After 50* group >>

[illegible]

12) It's now time for the next event group – open up “After 50” and create this inside:

IF: [The Timer Object] > Every

<< Set the timer to 2.50 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the [enemy] object >>

<< Set the coordinates to x=686, y=396 >>

IF: [The Timer Object] > Every

<< Set the timer to 3 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the [enemy] object >>

<< Set the coordinates to x=692, y=397 >>

IF: [The Timer Object] > *Every*

<< Set the timer to 4 seconds >>

THEN: [Create New Objects] > Create Object

<< Select the [enemy] object >>

<< Set the coordinates to x=704, y=396 >>

<< Select the *After 100* group >>

[illegible]

<< Set the coordinates to x=701, y=397 >>

<< Set the coordinates to x=692, y=397 >>

<< Select the *After 200* group >>

[illegible]

14) And here are the two final events... Once they are done, you'll have officially completed this tutorial. Just open up the "After 200" event group and create these two inside:

IF: [The Timer Object] > Every
<< Set the timer to 0.62 of a second >>
THEN: [Create New Objects] > Create Object
<< Select the [enemy] object >>
<< Set the coordinates to x=701, y=397 >>

IF: [The Timer Object] > Every
<< Set the timer to 1.22 of a second >>
THEN: [Create New Objects] > Create Object
<< Select the [enemy] object >>
<< Set the coordinates to x=692, y=397 >>

Aaaaand... Mission accomplished!

That's it, Sire! Congratulations! We're done here, you are now ready to hold your ground against Evilian's forces, to defend the castle of Haylenscalf! Feel free to add as many modifications and additional ideas as you wish, try to implement new objectives and features to our little project! Have fun and always remember: with Multimedia Fusion 2, you can create almost anything!

Thanks for your time and see you again soon!

Cheers!

Koobare

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

You have been reading...

Castle Defender



brought to you by
Koobare

Clickteam's funkiest
~~mercenary~~
knight in shining armor



Created for Multimedia Fusion 2 & Multimedia Fusion 2: Developer

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