

Game creation tutorial

Bermuda Triangle

Welcome to the game creation tutorial and thanks for reading. I will step you through the creation of a complete game from concept to final release. This tutorial is more than just instructions to build a game. It can tell you about why things are done that way and what is going on behind the scenes.

Program: MMF2

Difficulty: Beginner

What you will learn in this tutorial:

- to build a one level game with MMF2 that has a menu, scores, lives, autonomous enemies, scrolling, particle effects and more
- to work in the event editor
- to work in the level editor
- understand how most things work in MMF2
- solutions to some common problems

What you have to know:

- to press F1 and look something up in the help if you don't understand what I'm talking about or where to find something.
- if you are new to MMF2, you should have completed the [Pong tutorial](#). It teaches you the very first steps and gives you a rough idea how MMF2 works.
- while optional, it is very helpful to read the [Interface Guide](#). It tells you about some basic tricks and shows you where everything can be found. It also establishes an event documenting convention that we are going to use in a slightly modified way.

Section 1

What you do in this section:

- Set up all the options for the game and the objects
- Create a moving and colliding player object
- Make the game scroll

What you learn while doing so:

- The structure of object animations
- What are backdrops?
- What are events? Conditions? Actions?
- About the default MMF objects

If you already know this and prefer to start at a higher difficulty and not from scratch, you should read the concept and then skip to the next section of the tutorial.

[✕ Last section](#)

[Next section ➡](#)



1. The concept

1.1. Description

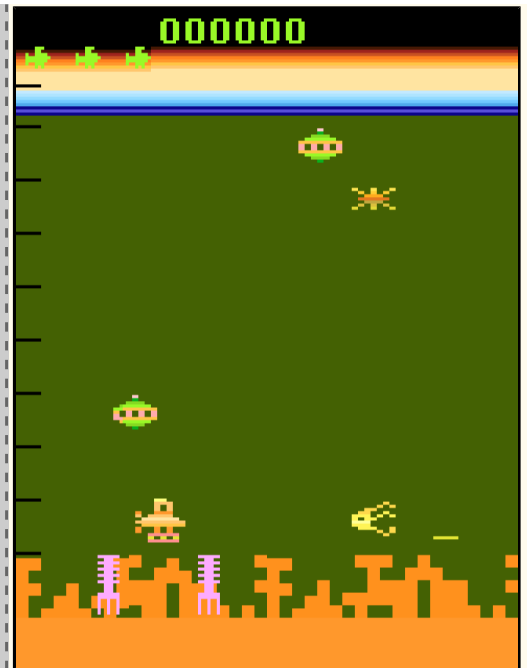
The game we're going to create is a remake of an old Atari game called "Bermuda Triangle". In fact I have never played the game myself, but I let a screenshot and a description inspire myself for designing it. This is the original description of the game:



Bermuda Triangle (1982)

From the outset, Bermuda Triangle appears to be a fun shooter in the same vein as Seaquest (Activision, 1983). Its colorful graphics include the best looking sub I've ever seen in a 2600 game. The object is to collect relics from the ocean floor and return them to a research ship patrolling the water surface. As the screen scrolls, you'll need to avoid mines, UFOs, squid, sharks, and enemy vessels. Your sub moves swiftly and shoots rapidly, but despite the fast action, Bermuda Triangle is lacking in terms of fun. For one thing, it's entirely too easy. When you collide with most obstacles, they just rattle you a bit. The real danger are laser beams that appear at random and are impossible to avoid. Why these unfair devices even exist is a mystery on par with the actual Bermuda Triangle. They seem to serve no purpose except to make the game artificially harder. I have a few other gripes as well. While returning items to the surface should net you 600 points, your score doesn't register immediately, which is disconcerting. I also hate how you can lose points by taking a hit while transporting an item to the surface. All in all, Bermuda Triangle doesn't play nearly as good as it looks.

1 or 2 players

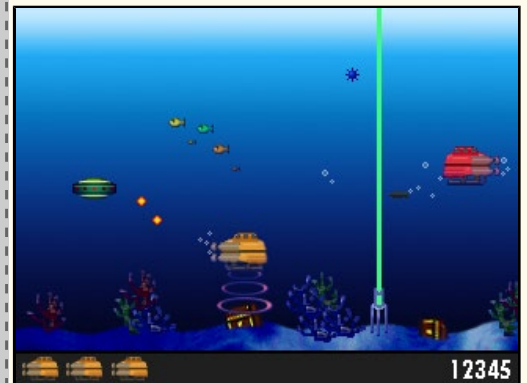


Thinking about the flaws that are pointed out in this description, I modify it to what I would like it to be when it's done. This will be the game concept that we work towards. I also make a mock-up of how I'd like it to look like. Some of the mock-up graphics might make it into the game straight away or at least server as dummy graphics. Here goes:

Bermuda Triangle (2007)

Bermuda Triangle is a remake of the Atari game of the same name. The object is to collect treasures from the ocean floor and return them to the water surface. As the screen scrolls, you'll need to avoid mines, UFOs, enemy vessels, coral reefs and laser beams. Your sub can shoot and moves swiftly while no treasure is being carried. When you collide with an obstacle or touch an enemy or their shots they damage you and make you drop your treasure. Three hits lead to destruction of your vessel. As opposed to the original game, laser beams fire at intervals and can thus be avoided. Also upon returning treasures to the water surface, score is instantly processed.




1 player



1.2. Functional units

Before we start setting things up, we make a list of all elements of the game and a short description of what they do.

Primary Actors

Submarine (player):		Purpose: pick up treasures and bring them to the top of the screen Abilities: can hold treasures with a magnet (casts magnet effect); can shoot torpedos in a straight line; can be destroyed (3 hitpoints) Behavior: controlled by the player; fast when not holding a treasure; else slow
Submarine (enemy):		Purpose: attack the player Abilities: can shoot torpedos in a straight line; can be destroyed (1 hitpoint) Behavior: tries to reach the height of the player and then fires in his direction, tries to avoid player torpedos; gives score when destroyed
UFO:		Purpose: attack the player Abilities: can shoot energy balls up or down; can be destroyed (1 hitpoint) Behavior: moves through the screen in a horizontal line; fires at the player if below or above him; gives score when destroyed
Sea mine:		Purpose: damage the player Abilities: can explode and deal damage



Behavior: floats at one point; explodes on touch; can not be hit with torpedos

Laser cannon:



Purpose: fire laser

Abilities: charge energy; fire laser beam in a vertical line

Behavior: stands on the ground; repeatedly charges and then fires a laser beam

Treasure:



Purpose: gives score when brought to the top of the screen

Abilities: none

Behavior: sticks in the sand; drops down when released

Secondary Actors

Torpedo:



Purpose: damage target

Abilities: can explode and deal damage

Behavior: moves in a straight horizontal line (casts bubbles); explodes on touch

Energy balls:



Purpose: damage target

Abilities: can explode and deal damage

Behavior: moves in a straight line; explodes on touch

Laser beam:



Purpose: damage the target

Abilities: deal damage when touched

Behavior: flashes up, deals damage when possible and fades away again

Background

Coral reef:



Purpose: obstacle

Behavior: does not move; deals damage upon touching

Sea floor:



Purpose: obstacle

Behavior: does not move; defines lower playfield border

Eye candy

Bubbles:



Behavior: moves upwards until out of view

Explosion:

Behavior: animation; removed after playing; casts bubbles

Fishes:



Behavior: swim around; try to cluster; scatter when scared

Debris:

Behavior: describe a parabola movement; removed when touching ground

Interface

Score:



Purpose: display player's points

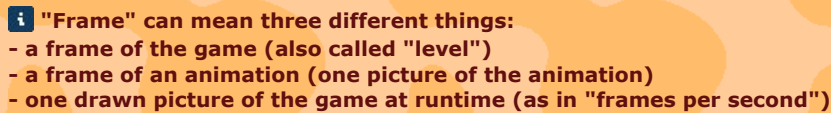
Lives:



Purpose: display player's lives



If you only see grey background, look to the left, where it says "Workspace Toolbar" and **double click on "Frame 1"**. This should open the level editor, where you can see all the objects. I have already sorted them in folders: "Backdrops", "Objects", "Interface" and "System". In the "Frame" you see a white box, this is the "playfield" or "frame area". If something is outside of this box, it will not be visible in the game unless you move it in. Your screen should look like this:



The coral reefs are "Backdrops". These can not be tiled, but are better suited for individual obstacles that will not move.

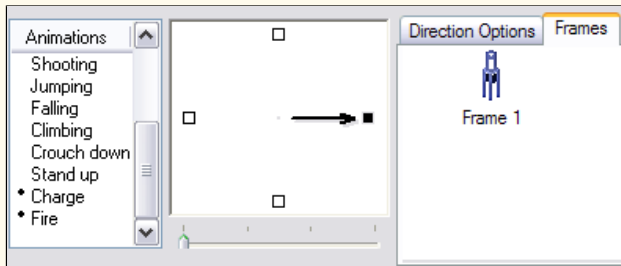
i Quick backdrops and backdrops are best suited for static obstacles.

The player's score is a special object called "Score". Basically it is like a counter, but it automatically keeps track of the player's score. The same is the case with the player's lives. That object is called "Lives" and keeps track of the player's lives. It can use numbers, text or images to display it.

The majority of objects are "Active Objects". These are very versatile. They are animated, can be moved, rotated, resized, they can get transparent, can collide and store their own values and strings and a lot more. There is an internal order between objects. Most important:

i Backdrops and Quick Backdrops are always behind Active Objects.

Let's take a look at the animations of some objects. The Laser cannon should be most interesting. You can find it in the "Objects" folder and also in the frame area. **Double click the objects folder, then double click the laser cannon.** This opens the animation editor.



Here you see a list of all animations. The dot in front of the animation means that it contains frames. **Scroll down the list and click on "Charge"**. This is a custom animation. I have added it, because the default animations did not seem appropriate.

To the right of the animation list you see the direction. This object only contains one direction. Some others, like the enemy contain two directions - one for left and one for right. You can have up to 32 directions.

If you select a direction, you can see the frames in it, to the right. If you press the play button, you can see how it looks.

Okay. **Press enter to close the animation editor.**

i Active objects contain animations, which contain directions, which contain frames.

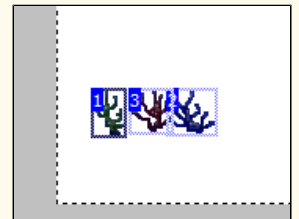


3. Settings

We now have to dig through all the options, so everything works properly. I'll explain the ones that we have to change.

i If you are interested what the other options do, press F1 and look at the "The properties" and "The objects">"The common properties" sections.

Let's start with the objects. You can select multiple objects by dragging over them with the mouse. If they are all of the same type, you can change their options all at once. Begin with the backdrops. **Select all the coral objects in the frame area.**

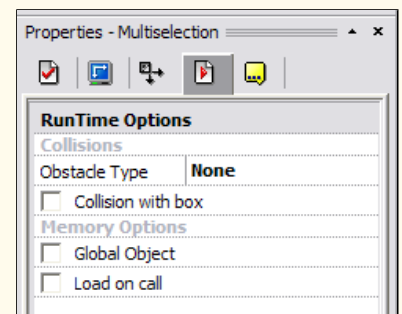


To the left, you should see the "Properties" dialog. If you don't, **right click on the corals and select "Properties"**.

At the top of the properties dialog you find five icons for different tabs. We'll go through each of them. For backdrops, the "Settings" tab is not very interesting. **Click on the fourth tab, "RunTime Options"**. This is where we need to make a change. "Obstacle Type" defines, how the object is seen by other objects. While it is set to "None", it can not cause any collisions. So **set it to "Obstacle"**.

If you want to know more about a certain type of object,

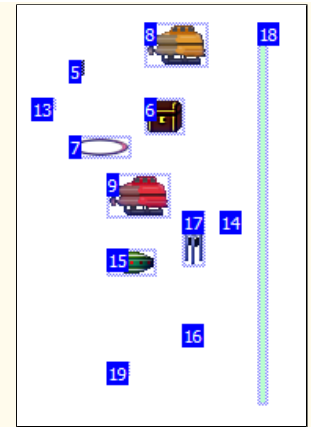
i You can get help for an object's type in the "About" tab of the properties.



Now **select all active objects in the frame**. That are objects like the laser beam, the player, the fishes, etc.

Do you still **have the "RunTime Options" tab open**? If not, click on it. You should have noticed that Active Objects have different objects than Backdrops. In our case, **uncheck "Destroy object if too far from frame"**. We don't want that.

Also **change "Inactive if too far from window" to "No"**. This may cause objects not to respond when we need them to. We sacrifice a little bit of performance with this. **Turn off "Use fine detection"** too. This will change collision detection to "box" mode. This is not pixel perfect but a lot faster. Most objects don't need it on. We will turn it on for those that do later.



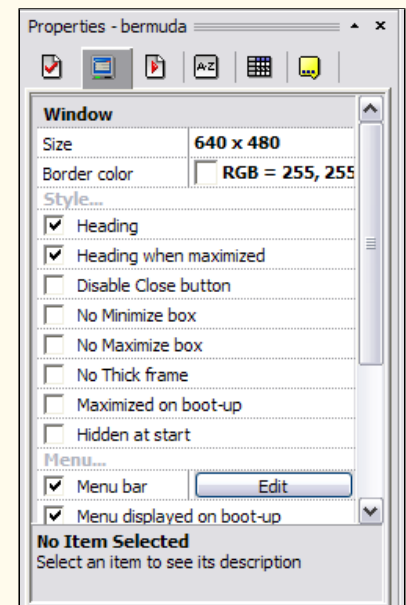
Now **select the Status Bar object**. It is actually also an Active Object, but being part of the interface, it serves a different purpose. In it's "RunTime Options" tab, **unchecked "Follow the frame"**. This means when we scroll in the game, it will not stick with the background but move with the view. For lives and score this is unchecked by default. Also **turn off "Destroy if too far from frame" and "Use fine detection"**. In it's "Size and Position" tab **change X to 0 and Y to 216**.

We're through with the objects for now. Later we'll visit some objects individually, to give them a proper movement.

In the Workspace Toolbar to the left, **select "bermuda"**. This represents the application or game and allows us to change global settings. **Open the "Window" tab**. Here we can specify the size of the game window. We want this one to be small and cozy. It could be put on a website later or be displayed fullscreen for a super-retro feeling. So **set the size to "320 x 240"**. We work with that size, so don't get creative on these numbers. Objects like the background or the status bar are sized to fit with this.

Disable "Heading when maximized". This is for nice fullscreen. **Uncheck "Menu bar"**. The game can still be closed by pressing Alt-F4. **Scroll down. Check "Allow user to switch to/from full screen", "Resize display to fill window size" and "Do not center frame area in window"**. The first two options allow you to have fullscreen if you click the maximize button of the heading. The last option makes sure that the view starts at the very left of a level.

Open the "Runtime options" tab. Check "Machine-independent speed". This will make the game run the same speed on all computers. If the computer is very slow, it can run very choppy though. **Check "Do no share data if run as sub-application"**. This is a security setting. **Check all the sound options**. If have already modified the Initial score of the player to "12345", so I could see how the score counter would look like. You should not rely on those settings, as they only get set when the application is started. So if the player loses and starts a new game, they will not be set to this again. You better initialize the score and lives with some code later. In "Default Controls" you can change the player controls, if you are not happy with the arrow keys and shift and control.



Select "Frame 1" in the Workspace Toolbar. In the "Settings" tab of the frame, you can change the size of the playfield. With "Virtual Width" set to -1, you could create a playfield where you can scroll infinitely long to the right. This is nice if you create the level contents at runtime. This is a bit tricky, so in our game we will set the level up ourselves. **Set the size of the frame to 6400 x 240**. This means the playfield is 20 screens wide. You can make it bigger if you want, but you'll have to fill it with content later.

Open the "Runtime options" tab. Change "Number of objects" to 3000. This is the maximum number of objects at any time. If you reach that amount, the game can become instable and weird things may happen. But if you set it much higher than you need it, it will make the game slower, even if there are not so many objects. So if you actually reach that limit, you should change it to a higher value later.

While we're at it, let's **rename the frame**. Right click on it in the Workspace Toolbar and select "Rename". **Call it "Level"**. You might add a startup frame or menu frame for the game later, then it is nice to have it named properly already.

As we have changed the dimensions of the frame, we should now adapt the size of the blue Background gradient. **Select it, open the "Size / Position" tab in the Properties and change "Width" to 6400** (or whatever size you chose earlier).

The Background is done for now. If you haven't moved it, it should still be aligned nicely. So **right click on it and select "Lock"**. This prevents you from accidentally moving it, if you don't want to. You can still change it's properties through the list of objects or the Workspace Toolbar.

Now we **do the same for the Sea Floor object**. Open the "Size / Position" properties, **change X to 0, Y to 182 and width to 6400 and lock it**.

i If you want to unlock a locked object, hold down Ctrl-Shift and left click it.

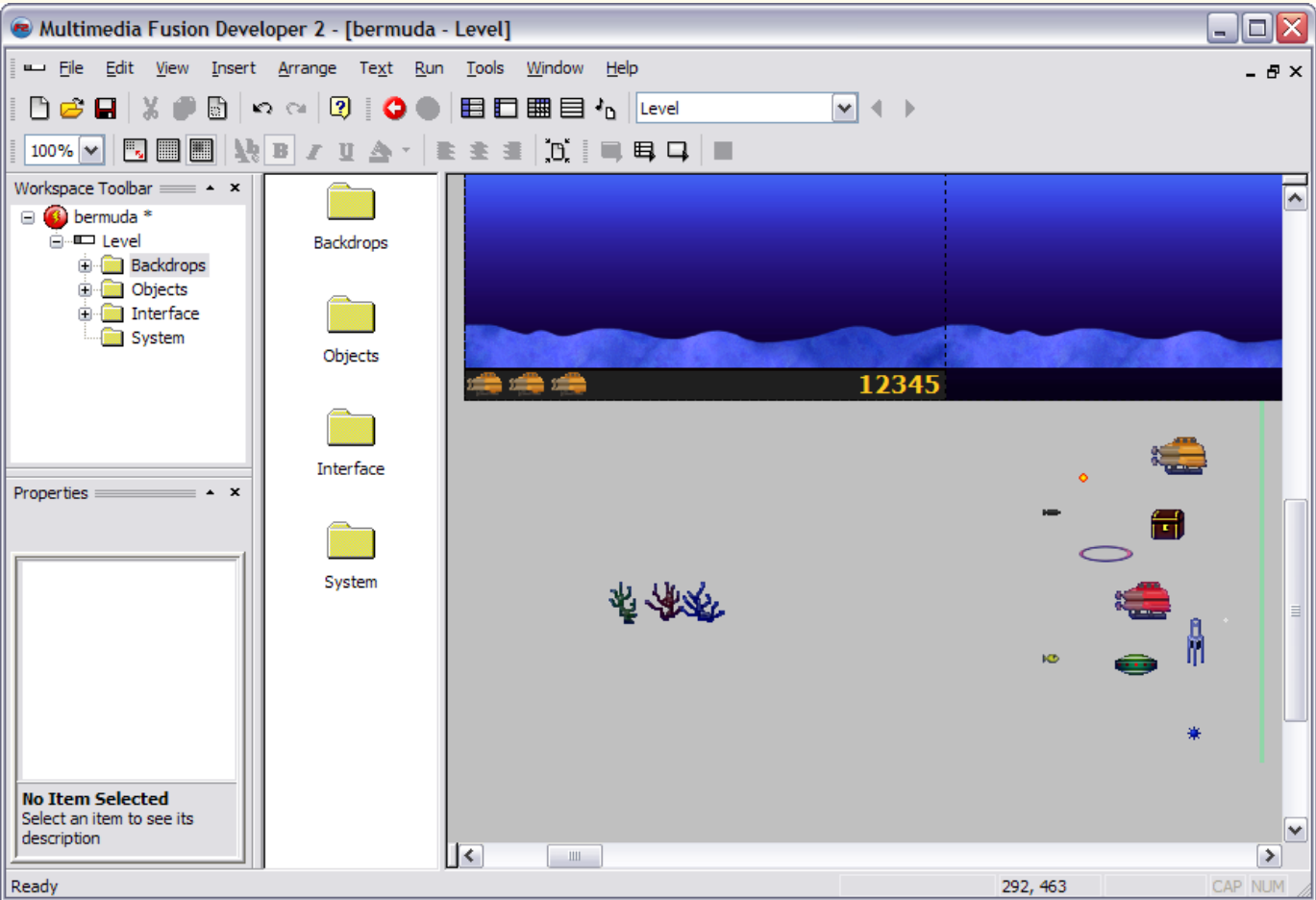
It might be a good idea to **reorder the other objects** in the frame so they get out of the way. Drag a box around the objects to select them and **move them down**, so they are not on the background anymore. While you move them, check if they snap to the grid. If they do, open the "View" menu and select "Snap to grid" to turn it off. We don't need the grid at this point.

i When moving objects, you can use the arrow keys to move them one pixel at a time to get them exactly where you

want them.

While you're at it, **change the position of the lives object. Set X to 2 and Y to 221.** Also **position the score object. Set X to 316 and Y to 237. Lock the lives and the score.**

You should now have something like this:



I have saved this as **bermuda_02.mfa**. If you think you did something wrong you can continue from it.



4. Intermission - time to have fun



Finally it is time to make this into a game. **Drag the yellow submarine in the center of the frame area. Select it and open the "Movement" tab** of its properties. **As "Type" select "Eight Directions". Set Speed to 30 and click "Try Movement"**. Nice, isn't it? Play around with it. When you're done, **press Escape to return**.

5. Mighty Event Editor



5.1. Event system

Let's make this whole thing scroll. **Open the "View" menu and select "Event Editor"**. This is where all the power of Multimedia Fusion comes together and where you will be working most of the time. Before we can do anything with it, we have to understand how it works.

And it works like this:

On the left, where it says "New condition", you create so called "Events". An event is one or more conditions and / or an immediate condition (sometimes called "true event"), combined with and one or more actions. A condition is something that MMF checks for you. For example "Is the player holding down the up arrow?" or "Is the submarine in the playfield?". "Always" or "3 = 2 + 1" are also conditions. If the answer to a condition is "Yes", the condition is called "true". Else it is called "false". "immediate conditions" are treated almost like conditions, but work differently. An event may only have up to one immediate condition. Immediate conditions can be questions like "Did the submarine touch a mine?" or "Did the player click with the mouse?". They only happen once at a time while conditions can be true for a while. When they happen, immediate conditions are called "triggered". If all conditions in an event are true, the event is called "triggered".

Events are triggered, when all conditions are true.

Actions are things that shall happen, when the event is triggered. So if the immediate condition is "Did the submarine touch a mine?" an action could be "Destroy submarine" and "subtract one of player's lives".

"Immediate conditions" may only be used once per event and have to be in the first line.

Here's an example:

To the right it is as seen in the "Event List Editor". Below it is as seen in the "Event Editor".

If the immediate condition triggers, all the actions below it are executed.

Immediate conditions are colored red, when they are placed correctly.

1

- Collision between (Player) and (Mine)
 - (Mine) : Destroy
 - (Player) : Destroy
 - : Play sample explode
 - (Player 1) : Subtract 1 from Number of Lives

All the events All the objects																	
1	• Collision between (Player) and (Mine)																
2	• New condition																

Another example:

Here an immediate condition is coupled with a condition. If the immediate condition is triggered, MMF2 will check if the Player is actually overlapping the Treasure. If not, nothing will happen.

2

- (Player 1) Pressed fire 1
- (Player) is overlapping (Treasure)
 - (Treasure) : Destroy
 - : Play sample collect01
 - (Player 1) : Add 100 to Score

One last example:

In this event we have no immediate conditions. Only a condition. This means that in theory it can be true all the time! But what will happen?

3

- Score of (Player 1) < 100
 - (Player 1) : Add 1 to Score

Here it is important to know how MMF2 works behind the scenes. You might have noticed, when setting up the game / application properties, that the frame rate was set to 50. For every frame, MMF2 runs through a cycle. Simplified it goes like this:

- Check immediate conditions and if one is true, run all events containing it
- Go through one event after the other, ignore events with immediate conditions
- Draw the current frame to the screen
- Next cycle

So what will happen in the third example? The condition is checked every cycle. And when it is true, 1 is added to the players score. As the framerate is set to 50, MMF2 runs 50 cycles per second. So after two seconds the score of the player is 100 and the condition is false.

i Events with immediate conditions are processed before all other events.

i MMF2 processes all normal events in order, every frame.

As can be seen in the first example, there are two editors to modify events. While the Event List Editor lists all the actions and is very useful for finding problems and checking and changing the order of actions, the Event Editor is faster to use and can filter information. If you click on one of the objects in the Event Editor, it will show only the events containing or affecting that object. Very useful.

i Double click on a checkmark to edit the order of actions in the Event Editor.

5.2. Notation

In this game, I will only explain things for the Event Editor. But if you want, you can also do it in the Event List Editor or both.

As there will be a lot of events in this tutorial, I will use a special notation for them. It is very similar as described in the **Interface Guide**. So this would be the same:

```
+ Player 1 - Joystick - Read joystick state: pressed fire 1
+ Player - Collisions - Overlapping another object: Treasure
> Treasure - Destroy
> Sound - Samples - Play sample: collect01
> Player 1 - Score - Add to score: 100
```



If you are familiar with the notation from the Interface Guide, here's what I do different:

Instead of "IF" I write "+" for conditions and instead of "THEN" I write ">" for actions. The first name defines the object that is responsible for the condition or action. Then I list the submenus of the object surrounded by "-" instead of ">" in the Interface Guide. This lets you easily find the condition or action. Then comes the actual condition or action. If it requires you to enter values or other instructions are necessary, I write those after the ":".

If you read events on the Clickteam forums, they will usually write what is seen in the event (list) editor instead of naming the submenus and conditions/actions. But this is much harder for beginners to find. For some of the conditions and actions I may the submenus and object away after a while (e.g. "Always" or "Create Object") if I think they should still be easy to find.

Note that most conditions can also be negated and that there are so called "OR operators". There are two of them, one called "OR (filtered)", the other called "OR (logical)". If I use negation, I write it like this:

```
+ Player - Collisions - NOT overlapping another object: Treasure
```

If I use "OR (filtered)", I write it like this, as it is the default "OR":

```
+ Player 1 - Joystick - Read joystick state: pressed fire 1
OR
+ Player - Collisions - Overlapping another object: Treasure
```

If I use "OR (logical)", I write it like this:

```
+ Player 1 - Joystick - Read joystick state: pressed fire 1
OR (logical)
+ Player - Collisions - Overlapping another object: Treasure
```

i A condition can be negated by right clicking on it and selecting "Negate".

i An OR operator can be inserted by right clicking on a condition and selecting "OR operator".

i More conditions can be added to one condition by right clicking on it and selecting "Insert".

5.3. Abstract objects

Maybe you have noticed that in the event editor there are some more objects than you could see in the frame area.



These are abstract objects that are included in MMF2. The first is "Special conditions". It contains some of the most powerful actions and conditions. Here you would find mathematical functions, access to global values, fast loops, comparison of two formulas and special conditions like "Always", "Never", "Run this event once", "Repeat" or "Only one action when the event loops". These are very important and used a whole lot.

i Most unrelated conditions, actions and functions are in the "Special conditions" object. If objects and submenus are not specified, look there first.

The second is "Sound". With this object you can play sounds and music.

The next is "Storyboard Controls". It has events like "Start of frame" and actions like "Next frame", "End the application", "Pause the application" or scrolling control.

The fourth object is "The timer". It provides functions to retrieve the time that the application ran and events that are based on it.

The next object is "Create new objects". It has only one action, but an important one: "Create object". It also provides conditions to select certain objects, but this is rarely used.

The sixth is "The mouse pointer and keyboard". It has conditions related to input, like "User clicks", "Repeat while key is pressed" or "Upon pressing key".

The last object is "Player 1". If you had a second player, there would be another one. Here you can modify the player's lives, score and controls.



6. Let's get started

6.1. Scrolling

Did I mention that we make the game scroll now? Good. That's what we do. Here's the event:

+ Special - Always
> Storyboard controls - Scrollings - Center window position in frame: 0,0 from Player

Looks simple, but it may give you a hard time to find everything. So I will step you through this one.

Right click on the box with the "1". Choose "Insert">"A new event". Right click on the "Special conditions" object and choose "Always". Nice. Now we have a condition that will be true every frame. **Right click on the box under the "Storyboard Controls" and choose "Scrollings">"Center window position in frame".**

A dialog box will show up that lets you select a position in the frame area. While you could chose a fixed position, you can also chose a relative position. This is a very powerful function! **Click on the yellow submarine.** A dashed box will show up around it, connecting the Player object with the position marker. You can still drag the position marker with the mouse. **Enter 0 in the X field and 0 in the Y field.** This makes sure that the position we want is exactly on the Player object. **Click "OK".**

You're done! Your first event! In fact it should not be, as you have already completed the Pong tutorial, right?



Add another one:

+ The mouse pointer and keyboard - The keyboard - Upon pressing a key: "Escape"
> Storyboard controls - End the application

Can you find it? The condition "Upon pressing a key" will ask you for a key - press the "Escape" button on your keyboard.

Done? Good. Let's see what you have just programmed. There is a tiny toolbar somewhere, containing four buttons:



Interesting for us are the second and third buttons: "Run Application (F8)" and "Run Frame (F7)". One runs the entire game if you have multiple frames, the other one only runs the current frame. For now **click that one or press F7** on the keyboard.

When it moves! Note how it only scrolls to the right. The view can never leave the borders, so it can't go up and down in our game. Note how the player can leave the screen at the left, top or bottom. We will have to restrict that. When you're done, **click the "X" of the game header or press the "Escape" button on your keyboard**.

6.2. Restrict movement

We'll now restrict the player's movement a little. **Do this:**

```
+ Player - Position - Test position of "Player": leaves the play area on the left or right
> Player - Movement - Stop
```

The condition comes from the yellow submarine Player object and is called "Position">"Test position of "Player"". There **select the arrows that point outwards to the left and the right**. Why not up? Because I already have a nice idea how we can use the top for gameplay. Why not down? Because we want to have a border at the floor instead of the bottom of the window. The action should be fairly easy to find.

Because we don't want the player to get "through" the floor, **here's another event:**

```
+ Player - Position - Compare Y position to a value: Greater 200
> Player - Position - Set Y coordinate: 200
```

Here we have "Position">"Compare Y position to a value" and "Position">"Set Y Coordinate...". Both will open a dialog box called the "Expression Editor". It allows you to enter values, but also formulas. For example "2 + 2". Or "10 mod 3". In a formula you can also use functions. A function lets you do more advanced calculations or allows you to get information from the game or an object. For example "xmouse" would give you the X position of the mouse. "Sqr(9)" would give you the square root of 9, "Frame Width" would give you the width of the frame area - in our case that would be 6400. "Speed("Player")" would give you the speed of the Player object on a scale from 0 to 100. "NObjects("Fish")" would give you the number of fish in the game. Of course you don't have to remember all of those. You can easily get them by clicking on "Retrieve data from an object". Here you can right click on one of the objects to get lists of all their functions.

MMF2 allows objects to have multiple movements that you can activate, one at a time. We use this to create a special reaction for the case that the player leaves the top.

i Choose functions by clicking "Retrieve data from an object" in the Expression Editor.

i For some conditions the Expression Editor has a "comparison method" drop down box. ">" is "Greater", while ">=" is "Greater or equal".

Now we handle the top border. Do this: open the "View" menu and **go to the Frame Editor. Select the Player. Open the "Movement" tab** of its properties. See where it says "Movement #1"? **Click on it, then click on the "+/-" button next to it.** On the dialog that opens, **click on the third button from the top** ("Create new movement"). **Click "OK". Set "Type" to "Pinball movement". Click "Try movement"** to see what it does. Looks like Super Mario! **Press "Escape". Set "Gravity" to 25, "Deceleration" to 0. Tick "Move at start", set "Initial speed" to 50. Click in the box of "Initial direction".** A dialog opens to select directions. **Press the "Reset" button** in the lower left corner of it. **Click on the black box of the "Up" direction (8).** Now **click "Try Movement"** again. We'll have something like that happen, when the player leaves the top of the frame area.

Go back to the Event Editor. Add the following events:

```
+ Player - Position - Compare Y position to a value: Lower or equal 12
+ Special - Limit conditions - Only one action when event loops
> Player - Movement - Multiple movements - Select movement: Movement #2
```












```
+ Player - Position - Compare Y position to a value: Greater 12
+ Special - Limit conditions - Only one action when event loops
> Player - Movement - Multiple movements - Select movement: Movement #1
```

i You can create a new event from existing conditions by drag and drop.

In this case, **drag the "Y position of Player > 200" condition and drop it on "New Condition". Right click on it and select "Edit".** Now you can **change the value to 12 and change the comparison method to "Lower or equal". Get the "Only one**

action when event loops" from the "Special Conditions" object. This makes sure that when the first condition is true, the action is only executed once and not every frame. The movement action can be found in the Player object: "Movement">"Multiple movements">"Select movement".

If you **open the Event List Editor**, it should look like this:

1	<ul style="list-style-type: none">• Always  : Center display at (0,0) from  (Player)
2	<ul style="list-style-type: none">• Upon pressing "Escape"  : End the application
3	<ul style="list-style-type: none">•  (Player) leaves the play area on the left or right  (Player) : Stop
4	<ul style="list-style-type: none">• Y position of  (Player) > 200  (Player) : Set Y position to 200
5	<ul style="list-style-type: none">• Y position of  (Player) <= 12 ✦ Only one action when event loops  (Player) : Select movement Movement #2 (number 2)
6	<ul style="list-style-type: none">• Y position of  (Player) > 12 ✦ Only one action when event loops  (Player) : Select movement Movement #1 (number 1)

Press F7 to run the frame. If you've done everything correctly, the player should "jump out and back in" when you touch the top of the window. Later when the player is carrying a treasure, we can use this to remove the treasure and give score.

If you had trouble getting it done exactly like described, you can continue from the file "**bermuda_03.mfa**".

This is the end of this section. I hope you have enjoyed it so far. At this point we have set up most of the options and I have explained some basics, so we can tackle more gameplay elements in the next section.

What you will do in the next section:

- Fill the level with some basic objects
- Let the player collide with obstacles
- Let the player pick up treasures
- Award score for retrieving treasures
- Add a simple particle effect
- Shake the screen
- Let the player shoot torpedos

 **Last section**

Next section 