GETTING The most OF...

The Animation

Welcome to the eight issue of our Multimedia Fusion tutorials. This issue will discuss how the Fusion runtime handle sprites animations. All the content of this tutorial is applicable to Click & Create and The Games Factory.

☑ Animations and directions in runtime.

Fusion's runtime does its best to match the animation with the movement of the character. It does this job at three different levels :

- Animation
- Direction
- Speed

For example, if a character is moving slowly to the right, then it will prefer the Walking animation, in the East (0) direction, and it will adapt the speed of the animation to the speed of the movement. Most of the times, the author of the application does not draw all the animations, or all the 32 directions. When this is the case, Fusion has to approximate and find the best animation and direction possible.

Matching the animation.

If the exact animation is defined, then Fusion will use it. If this animation is not available, then Fusion will revert to the **closest** animation possible. If this one is not available, to the **stopped** animation. If the stopped animation is not available, then to the **first available** animation...

Fusion only does this process of animation approximation with the default animations, numbered 0 to 11. The following list describes the correspondance between the action of the sprite and the animation chosen in order of preference...

□ Static : stopped, appearing, walking, running, any available.
□ Moving at speed below 80 :: walking, running, stopped, any available
□ Moving at speed above 80 : running, walking, stopped, any available
□ Just created : appearing **(1)**
□ Just destroyed : destroyed **(2)**
□ A bounce action has occured : bouncing, stopped, walking, running, any available **(3)**
□ A shoot object action has occured : shooting **(4)**
□ Platform movement jump : jumping, walking, running, any available
□ Platform movement fall : falling, stopped, walking, running
□ Platform movement climb : climbing, walking, running, stopped, any available
□ Platform movement crouch : crouching, stopped, walking, running, any available
□ Platform movement standup : standup, stopped, walking, running, any available

(1) If the Appearing animation is not defined, the object is immediately created with the first image of the stopped animation. The object does not generate any collision while the creation animation is being played. If the object has at the same time a creation animation and a fading effect, the fading effect will come first, then the appearing animation.

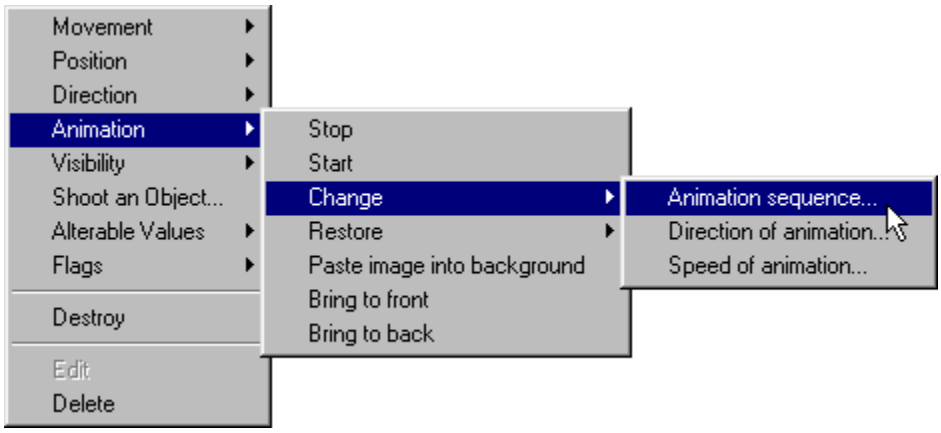
(2) If the disappearing animation is not defined, the object is immediately destroyed, and disappears from the screen (with an eventual fade out effect). The object does not generate collisions while the disappearing animation is playing, it is though still present in the list of object, take that into account if you are counting the number of objects in the application. If both a disappearing and fade out are defined, then the disappearing animation will **not** be played : the sprite will simply fade out. At the end, the object is actually destroyed from memory.

(3) You should avoid looping the bounce animation, if should be short and effective to get the best effect on the screen. This also applies to the shoot action.

(4) The shooted object is created at the position of the action point of the first frame of the shoot animation. Care should be taken to properly define the action points of this animation.

Preventing the automatic animation selection from working.

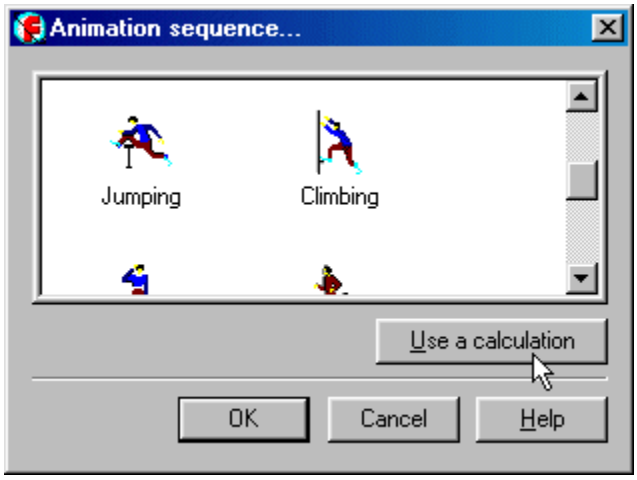
This automated system is very handy for many situations, but some you might want to enforce a specific animation at keypoints of an application. Just use the **Animation** / **Change** / **Animation sequence** action in the action editor. This action is also the only way to play the extra animations. Note that this actions does **not** prevent the automated direction matching and the automated speed matching to occur.



The enforced animation will remain enforced until it finishes, or until you use the action **Animation** / **Restore** / **Animation sequence.** If the animation loops, it will stay enforced forever.

Tip: if you want you animation to stay enforced but "stop", make it loop on the last image.

You can select the animation via a calculation. In this case refer to the numbers from the above table for correspondance between numbers and default animations. Numbers above 12 refers to the user animations, 12 being the first one in the animation selector box, 13 the second etc... C&C and Games Factory only allow four user animations, whereas Fusion allow any number. A bug in Fusion up to Build 93 makes it impossible to select any animation after the four user animations. This bug is corrected in Build94 and after.



Note that using a calculation is the only way to specify an animation for behaviours and global events : in such list of events, Fusion is dealing with shortcuts to objects and not the real objects themselves, and cannot enumerate the available animations.

Matching the direction.

Once Fusion runtime is happy with the animation, it has to choose the direction of the animation. An important thing is not to confuse the direction of the movement (where the object is going) and the direction of the animation (what the sprite is facing).

Fusion does its best to find a direction in the animations that matches the direction of the movement. Most of the time, an object does not contain all the 32 directions for a given animation (to save some space) so Fusion has to find the most approaching one.

The current direction of the movement gives the base, from 0 to 31. If the exact direction exists for the current animation, then Fusion chooses it. If not, it starts exploring the circle from the wanted direction, both clockwise and anti clockwise. The closest direction is then chosen.

Fusion will choose direction 4

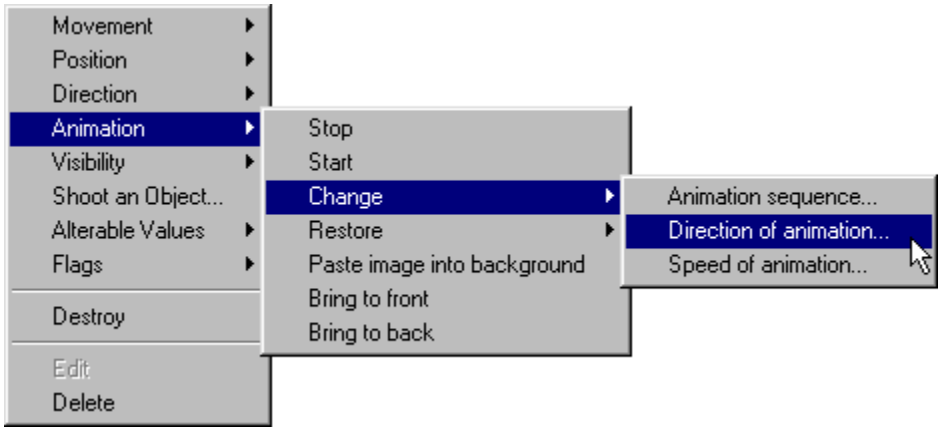
to replace direction 0.

Special case of the platform movement jumps : when jumping, the platform movement will take into account if the object was facing on the left or on the right before the jump.

The direction of the mouse movement is computed from the position at the previous frame: small movement will make the object rotate very fast.

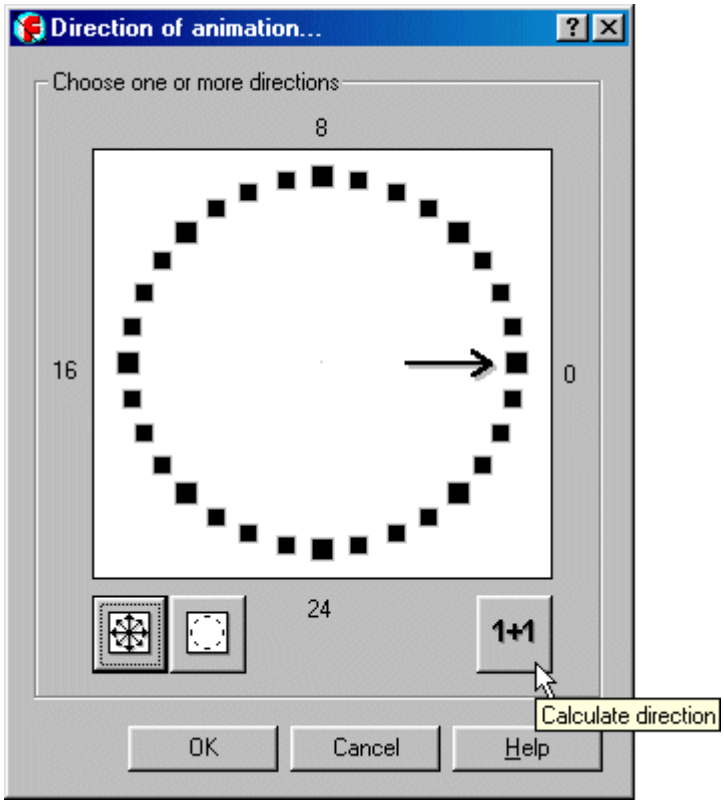
Preventing the automatic direction from working.

Just like you can prevent the automated animation selection from working, you can prevent the automated direction selection from working. Use the **Animation** / **Change** / **Direction of animation** action in the event editor. This action will show you a direction selector, in which you can select more than one direction. If you do, Fusion will choose one direction at random between the selected direction. If you leave everything blank or select the 32 directions, then it will choose a direction at complete random.



Note that enforcing a direction does **not** prevent the automated speed matching to work.

A very interesting possibility is to calculate the direction of the object with the **Use expression** button. This opens the evaluation editor. Despite the message displayed on the top of the editor, Fusion will happily accept values **above 32**, or **below 0** : it will simply get do a modulus on the number. For example a direction of 46 will turn the object in direction 46 modulus 32 = 14...

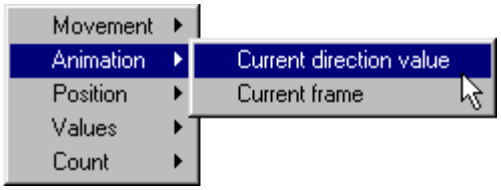


An enforced animation direction will stay like this forever. You have to use the **Animation** / **Restore** / **Direction of animation** direction action to get things back to normal.

Note : to further demonstrate the fact that you must not confuse the direction of the movement and the direction of the animation, here is a small example that will make an object face the opposite direction of where it is going :

Create an object with a stopped animation containing the whole 32 directions (click on **Create rotated directions**). Affect a movement to the object (anyone you like!). In the event editor, insert the following :

Always
  + Object : Change animation direction to Dir("object")+16



How to recover the direction of an object in the expression editor...
Current direction value should actually be transfered
in the **Movement** sub menu...

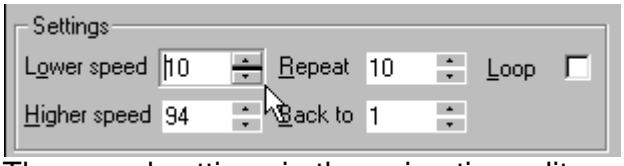
Warning : I just realised while writing these lines that there is a small inconsistence in the menu from the expression editor. The **Current direction value** choice should

be found in the **Movement** submenu, and not in the **Animation** submenu, as it depicts the direction of the movement. I also realise that we should add an entry in the **Animation** submenu to recover the current direction used in the animation (which may be different from the one used for moving)... Eee for a next version.

Matching the speed of the animation.

The last thing to know for Fusion before actually animating the object, is the speed of the animation. Here too, Fusion runtime does its best to produce the best result, by computing the animation speed so that it matches the movement speed.

You must have noticed the two values in the animation editor : **Lower speed** and **Higher speed**. Lower speed will be used when the object is at its minimum speed : stopped. Higher speed will be used when the object moves at the maximum possible speed for this movement.



The speed settings in the animation editor...

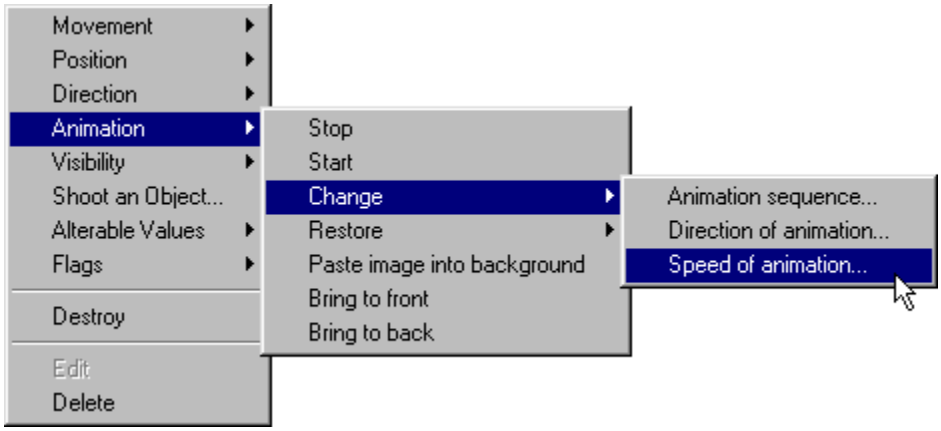
Example : a race car with a speed of 75.

When stopped, the stopped animation will be played, at the Lower speed (Higher speed is not taken into account).
When moving on the screen at maximum speed, the animation played will be Walking, with an animation speed equal to Higher speed.
When moving on the screen at an intermediate speed, Fusion calculates the animation speed from the Lower and Higher values (...)

As a result, if you choose carefully the Lower and Higher values of the animation speed, you can have the animation follow exactly what your character does: very important for platform characters.

Preventing the automated animation speed calculation from working.

You should use the action **Animation** / **Change** / **Speed of animation** to set the speed of the animations to a preset speed, regardless of the speed of the object. This action will stay on until you perform the action **Animation** / **Restore** / **Speed of animation.**



☑ Frames and animation changes.

Fusion does its best to keep the animation continuous on the display by using the same frame number when the animation or direction changes.

Imagine your character is going east (direction 0). It is in the middle of the Walking animation, direction 0, at frame number 5 (leg up). Now you face North. Fusion will take the Walking animation, direction 8, and will first try to position itself at frame 5, to keep the animation running smoothly. If a frame 5 does not exist, then it will revert to frame1.

The same applies when changing the animation themselves, for example walking to running : Fusion tries to position itself at the same frame in the new animation.

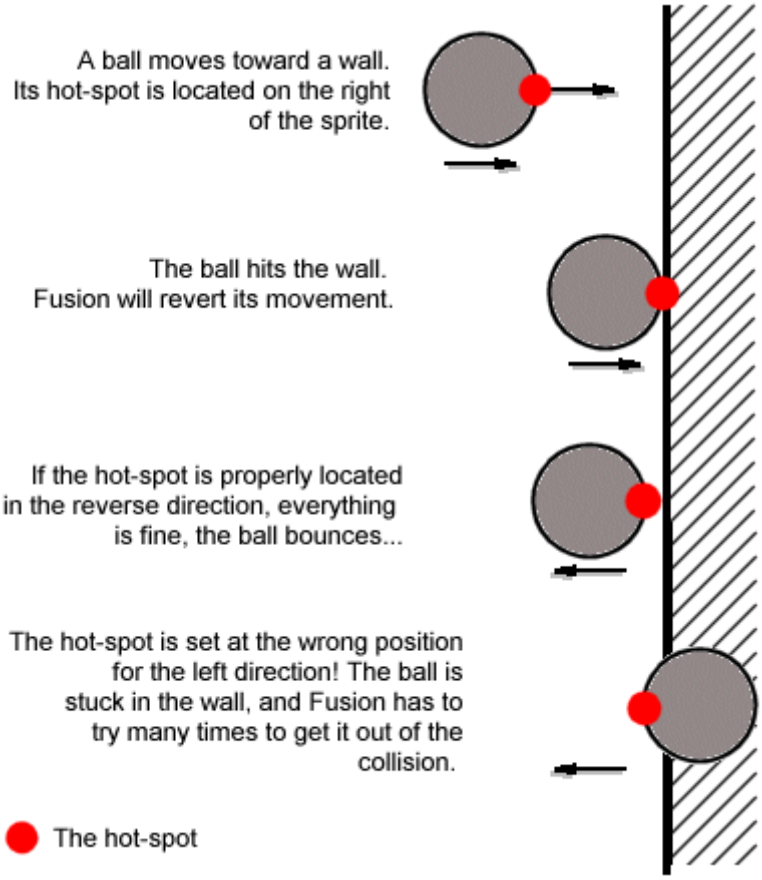
Therefore, it is very important that you program your sprite with the **same** number of frames in **each direction**, or **each animation** if you want them to link nicely.

☑ Movements and hot-spots.

Another very important point to acheive good movements and good bounces is to position the hotspot at sensible location for all the frames of all the directions of all the animation of one object (pfew!).

Imagine the following scenario.

- A ball is bouncing on a wall, with its hotspot located on the right-edge of the sprite.
- Fusion will make this ball bounce, but the hotspot of the ball facing west is located on the **left** side of the sprite!
- The result is an instant collision immediately after the bounce, that will generate strange effects.



Fusion is programmed to take this kind of things into consideration when computing the bounces, but it will make it task a lot simpler, and thus save **a lot** of processing time while bouncing if you make sure to position the hot-spot at the center.

Objects moving with a platform movement should **always** have the hot-spot located in the middle of the feet, on the bottom line of the sprites.

Tip #1 : in the picture editor, you can position the hotspot instantly by pressing the numeric keypad keys : 1 is for bottom left, 2 for bottom-middle... etc.

Tip #2 : in the picture editor, you can position the hotspot of all the frames of one animation-direction by holding the ALT key while positionning the hotspot with the mouse. You can of course use the combination : ALT+NUMERIC KEY.

Written by François Lionet, edited by Mike Bibby.
Please **do not copy or re-use** without written permission.