The AIR format and standard

M.U.G.E.N, (c) Elecbyte 1999-2010

Documentation for version 1.1 (2010)

Updated 30 November 2010

Contents

Introduction

AIR is a text file format that describes a set of animations. The AIR format is used widely throughout M.U.G.E.N for animating characters, backgrounds, life bars and more.

An animation (specifically called an animation action in M.U.G.E.N) describes a single sequence of sprites to display. A character's animation file (.air) can contain as many animation actions as necessary.

The format

Here's a typical entry from an .air file. We will explain the format next.

```
; Standing Animation
[Begin Action 000]
Clsn2Default: 2
 Clsn2[0] = -10,  0, 10,-79
 Clsn2[1] =  -4,-92,  6,-79
0,1, 0,0, 7
0,2, 0,0, 7
0,3, 0,0, 7
0,4, 0,0, 50
0,5, 0,0, 7
0,6, 0,0, 7
0,7, 0,0, 7
0,8, 0,0, 60
```

Semicolon (;) is used for comments. They can be used anywhere and everything after the semicolon on the same line is ignored.

`[Begin Action n]`
: This defines an action. The number n is called the action number. Each action must have a unique action number in order to be distinguishable from other actions. You can have many actions, but keep in mind that there are reserved actions numbers. These numbers and their associated actions are listed below.

`Clsn2Default: 2`
: This says that there are two collision boxes being defined and they will be defined for all the entries below the Clsn definition. In this case for every sprite in this action. If you don't want to have a default collision box, then change Clsn2Default to Clsn2. You must put the Clsn definition before a sprite entry. Each sprite entry can have its own collision boxes. Clsn2 refers to a plain collision box and Clsn1 refers

to an "attacking" box. You use attacking boxes when making attacking actions such as punching and kicking or special moves. To define collision boxes, use AirEdit, a program included in the M.U.G.E.N development toolset.

`0,3, 0,0, 7`

An entry looking like this is called an "animation element", or just "element" for short. 1st number is the sprite's group number. 2nd number is the sprite's image number. You assign these numbers to your sprites when you make an .sff file.

So group #0 and image #3 were defined with sprmaker with an entry such as `0,3, mypic.png, 40, 40`.

The 3rd and 4th numbers of the element represent the X and Y offsets to the sprite's position. For example, the pair `15,-10` would shift the sprite's position 15 pixels forwards and 10 pixels upwards. This offset is affected by scaling operations but not rotation.

5th number is the length of time to display the element before moving onto the next, measured in game-ticks. There are 60 game-ticks in one second at normal game speed. For the 5th number, you can specify "-1" if you want that element to be displayed indefinitely (or until you switch to a different action). If you choose to do this, do it only on the last element of the action.

The "looptime" of an action is the sum of all the times of each element. Basically, it is the total length of the action in game-ticks. If the last element has a time of "-1", the looptime will be considered infinite. We call that "no loop". Actions with a finite loop time will go back to the first element after it has finished displaying the last element, and keep looping until the you switch to another action.

In the example action above, the looptime would be: 7 + 7 + 7 + 50 + 7 + 7 + 7 + 60 = 152 ticks

```
; Standing Animation
[Begin Action 000]
Clsn2Default: 2
 Clsn2[0] = -10,  0, 10,-79
 Clsn2[1] =  -4,-92,  6,-79
0,1, 0,0, 7
0,2, 0,0, 7
Loopstart
0,3, 0,0, 7
0,4, 0,0, 50
0,5, 0,0, 7
0,6, 0,0, 7
0,7, 0,0, 7
0,8, 0,0, 60
```

Adding a line with the word `Loopstart` will make the animation begin looping from the element on the following line. In the example above, when the element with sprite 0,8 is displayed, the next element to be shown will be the one with sprite 0,3.

When testing out your animations, you may find the pause and frame-advance functions useful. To pause, hit the "Pause" button. To advance the game by one game-tick, press Scroll Lock. Frame- advance only works when the game is paused.

Optional parameters

Here's an entry:

```
15,1, 0,0, 5
```

If you want to flip the sprite horizontally and/or vertically, you will need to use the "flip" parameters: V for vertical flip, and H for horizontal flip. These parameters will be 6th on the line. For example:

```
15,1, 0,0, 5, H   ;<-- This flips it horizonally
15,2, 0,0, 5, V   ;<-- This flips it vertically
15,3, 0,0, 5, VH  ;<-- This flips it both ways, ie. rotates 180 deg
```

For certain things such as hit sparks, you might like to use color addition to draw the sprite, giving it a transparent effect. This is known as transparency blending. The parameters for color addition and subtraction are `A` and `S` respectively, and should go as the 7th on the line. For example:

```
15,4, 0,0, 5, ,A   ;<-- Color addition (flip parameter omitted)
15,4, 0,0, 5, H, S ;<-- Flips horizontally and does color subtraction
```

If you wish to specify alpha values for color addition, use the parameter format `AS???D???`, where `???` are replaced with values of the source and destination alpha respectively. Values range from 0 (low) to 256 (high). For example, `AS64D192` stands for "Add Source_64 to Dest_192". Also, `AS256D256` is equivalent to just `A`. A shorthand for `AS256D128` is `A1`.

```
15,4, 0,0, 5, ,A1  ;<-- Color addition to 50% darkened dest
15,4, 0,0, 5, ,AS128D128 ;<-- Mix 50% source with 50% dest
```

To scale a sprite, use the 8th and 9th parameters for x-scale and y-scale respectively, e.g. this will scale the sprite by 1.5 and 2 respectively:

```
15,4, 0,0, 5, ,A, 1.5,2
```

To rotate a sprite about its axis, specify the angle in degrees in the 10th parameter:

```
15,4, 0,0, 5, ,A, 1,1, 45
```

Note: scale and angle parameters are not implemented in M.U.G.E.N versions prior to 1.1.

Interpolation

Certain parameters can be interpolated from one element to the next. Parameters supported by interpolation are the offset, blending, scale and angle.

Interpolation is supported in M.U.G.E.N 1.1 and higher.

Currently only linear interpolation is supported.

The syntax is:

```
Interpolate <type>
```

where `<type>` is one of `Offset`, `Blend`, `Scale` or `Angle`.

`Offset`
: Interpolates the offset.

`Blend`
: Interpolates the transparency blend parameters. Both elements must have the same transparency function, i.e. `A`. To interpolate to or from a fully opaque element, use `AS256D0`.

`Scale`
: Interpolates the scale.

`Angle`
: Interpolates the angle.

One or more `Interpolate` lines can be inserted between two elements to enable interpolation between them. `Interpolate` lines before the first element will be used to interpolate from the last element to the first element in a looping animation.

The following example interpolates the offset between 0,0 and 100,0, and the scale between 1,1 and 1.5,1.5:

```
Interpolate Offset
Interpolate Scale
20,0, 0,0, 60, , , 1,1
```

```
Interpolate Offset
Interpolate Scale
20,0, 100,0, 60, , , 1.5,1.5
```

The following example animates a spinning object with a rotational period of 60 ticks:

```
20,0, 0,0, 59, , , 1,1, 0
Interpolate Angle
20,0, 0,0, 1, , , 1,1, 354
```

Character Reserved Action Numbers

M.U.G.E.N's common state engine (data/common1.cns) requires several animation actions to be present in your character (a warning will be logged if not all required animations are present). There are also some actions that are optional, which will be made use of if present in your character.

If you are unsure of how any of these actions should look, take a look at chars/kfm/kfm.air.

All action numbers in the 5000-5999 range not listed below are reserved for possible future use, so avoid using those numbers for custom actions.

An "opt" besides the number means the animation is optional.

Number	Description	Comments
0	Standing	
5	Stand turning	Must have finite looptime
6	Crouch turning	Must have finite looptime
10	Stand to crouch	Finite looptime
11	Crouching	
12	Crouch to stand	Finite looptime
20	Walking forwards	
21	Walking backwards	
40	Jump start (on ground)	Shown just before player leaves ground
41	Jump neutral (upwards)	Shown when player is going up
42	Jump forwards (upwards)	Shown when player is going up-towards
43	Jump back (upwards)	Shown when player is going up-back
44 opt	Jump neutral (downwards)	Activated when Y-velocity > -2
45 opt	Jump fwd (downwards)	Same as above
46 opt	Jump back (downwards)	Same as above
47	Jump landing	Shown as player lands on ground
100	Run fwd/hop forward	
105	Hop back	
120	Start guarding (stand)	Finite looptime
121	Start guarding (crouch)	Finite looptime
122	Start guarding (air)	Finite looptime
130	Guard (stand)	
131	Guard (crouch)	
132	Guard (air)	
140	Stop guarding (stand)	Finite looptime
141	Stop guarding (crouch)	Finite looptime
142	Stop guarding (air)	Finite looptime

150	Guarding a hit (stand)	Finite looptime
151	Guarding a hit (crouch)	Finite looptime
152	Guarding a hit (air)	No loop
170 opt	Lose	(See Note 1)
175 opt	Time Over drawgame	(See Note 1)
180 opt	Win	No loop (181-189 for multiple) (See Note 1)
190 opt	Intro	No loop (See Note 1)
195 opt	Taunt	Finite looptime (See Note 1)
5000	Stand/Air Hit high (light)	Looptime around 10-20
5001	Stand/Air Hit high (medium)	" (See Note 2)
5002	Stand/Air Hit high (hard)	"
5005	Stand Recover high (light)	No loop (See Note 3)
5006	Stand Recover high (medium)	"
5007	Stand Recover high (hard)	"
5010	Stand/Air Hit low (light)	Looptime around 10-20
5011	Stand/Air Hit low (medium)	"
5012	Stand/Air Hit low (hard)	"
5015	Stand Recover low (light)	No loop
5016	Stand Recover low (medium)	"
5017	Stand Recover low (hard)	"
5020	Crouch Hit (light)	Looptime around 10-20
5021	Crouch Hit (medium)	"
5022	Crouch Hit (hard)	"
5025	Crouch Recover (light)	No loop
5026	Crouch Recover (medium)	"
5027	Crouch Recover (hard)	"
5030	Stand/Air Hit back	Looptime around 10-20
5035 opt	Stand/Air Hit transition	Looptime around 5-15 (See Note 3)
5040	Air Recover	No loop
5050	Air Fall	No loop
5060 opt	Air Fall (coming down)	No loop
5070	Tripped	
5080	LieDown Hit (stay down)	(See Note 4)
5090	LieDown Hit (hit up into air)	
5100	Hitting ground from fall	Looptime around 3
5160	Bounce into air	
5170	Hit ground from bounce	Looptime around 3 or 4
5110	LieDown	
5120	Get up from LieDown	
5140 opt	LieDead (first rounds)	
5150 opt	LieDead (final round)	
5200	Fall-recovery near ground	
5210	Fall-recovery in mid-air	
5300	Dizzy	
5500 opt	"Continue?" screen	If omitted, plays dizzy anim
5510 opt	"Yes" to "Continue"	If omitted, plays first win anim (not yet implemented)

5520 opt	"No" to "Continue"	(not yet implemented)

Optional Hit Up animations (see Note 5):

Number	Description	Comments
5051 opt	Air fall -- hit up	
5061 opt	Coming down from hit up	
5081 opt	LieDown Hit (stay down)	
5101 opt	Bounce from ground into air	Looptime around 3
5161 opt	Bounce into air	
5171 opt	Hit ground from bounce	Looptime around 3 or 4
5111 opt	LieDown	
5121 opt	Get up from LieDown	
5151 opt	LieDead (first rounds)	
5156 opt	LieDead (final round)	

Optional Hit Up-Diagonal animations (see Note 6):

Number	Description	Comments
5052 opt	Air fall -- hit up	
5062 opt	Coming down from hit up	
5082 opt	LieDown Hit (stay down)	
5102 opt	Bounce from ground into air	Looptime around 3
5162 opt	Bounce into air	
5172 opt	Hit ground from bounce	Looptime around 3 or 4
5112 opt	LieDown	
5122 opt	Get up from LieDown	
5152 opt	LieDead (first rounds)	
5157 opt	LieDead (final round)	

Note 1: These do not actually have to use only the stated numbers. If is
more of a recommended number than a M.U.G.E.N one. If necessary, feel free to use any other action numbers.

Note 2: If medium and heavy hits are omitted, they default to the light hits

Note 3: "No loop" means last frame has a time of -1.
For recover animation, the first frame of each recovery should be the last frame of the corresponding hit. Eg. If action 5000 has frame 5000,0 and 5000,1, then action 5005 should start with frame 5000,1. This is because the animation will be locked in the first frame of the recovery after the hit animation is over, but before the player has recovered from the hit. If you have a Stand/Air Hit transition animation, then the first frame of Air Recover and Air Fall should be the last frame of the transition animation.

Note 4: The Stand/Air Hit transition animation is played back after each
hit animation in (or into) the air, but before the Air Recover and Air Fall animations.

Note 5: You can loop the LieDown Hit if you want the player to look like he
is "twitching" while being hit

Note 6: This set of animations is optional. It is an alternate set of
falling animations, which is used if hit by a HitDef with "Up" as the animtype.

Note 7: This set of animations is optional. It is an alternate set of
falling animations, which is used if hit by a HitDef with "DiagUp" as the animtype.

Recommended Action Numbers

You do not have to follow this exactly, but it should serve as a guideline. In general, the states in the CNS should have the same numbers as the animation they use, to reduce confusion.

You might want space out your animation and state numbers. This gives room for you to add in more actions as necessary (some attacks can use multiple states and animations). For instance, Standing Light Kick and Standing Strong Kick could have action numbers 200 and 210 respectively.

Number	Description
0-199	*reserved*
200-299	Standing attacks
300-399	More standing attacks, running attacks
400-499	Crouch attacks
500-599	More crouch attacks
600-699	Air attacks
700-799	More air attacks
800-999	Unused - use if you need more states
1000-2999	All special attacks
3000-4999	All hyper attacks
5000-5999	*reserved*