---------- (RMotion) ----------

Animation tool for the powerful RAPTOR Engine

---------- (RMotion) ----------

Version 1.1

Code: Matthieu Barreteau (Matmook)

Date: 01/12/2012

Content

Description

Rmotion is a complete new little language for the Raptor Engine (API for the Atari Jaguar platform designed to allow rapid development of games without compromising code execution speed). RMotion offers facility to create and manage animation using Raptor Engine. RMotion is able to modify all Raptor's object's property (See Raptor's documentation for more information).

RMotion is :
- A macro-compiler who transform your animation sequence in binary form.
- A GPU addon who read your binary animation and execute them.

Usage

How to use RMotion:

Rmotion.exe (the macro compiler) will take your animation script and transform it to binary code (RMotion will produce an error if a command is not understood):
Rmotion.exe source file destination file

Overview

RMotion use his own language for animation declaration.

Example:
myanimation:
gfx(mybitmap,WAIT)
...
stop()

An animation is placed on raptor's object using two ways:
- Directly in object's declaration:

```
dc.l    ...
dc.l    single          ; sprite_bboxlink       ; single for normal bounding box, … to table
dc.l    myanimation     ; sprite_rmotion        ; used for R-Motion animation
```

- By code at runtime:

```
lea     object,a0
move.l  #myanimation,sprite_rmotion(a0)
```

General instruction option

WAIT

All "instructions" are executed in the same frame but if there is a WAIT option specified (some instruction could not use the WAIT option):
- x(50|16.16) will set x=50 and the next instruction will be executed
- x(50|16.16,WAIT) will set x=50 and no more instruction will be executed in this frame, the animation will continu at the next screen refresh (depending of this object rspeed).

UPDATE

Most instruction has the "UPDATE" options who add your value to the current one instead of replacing it:
- x(50|16.16) will set x=50
- x(50|16.16,UPDATE) will add 50 to the current x value (x=x+50)

UPDATE and WAIT could be used the same time:
x(50|16.16,UPDATE,WAIT) will add 50 to the current x value (x=x+50) and will WAIT for the next frame (depending of this object rspeed).

Source type declaration

There is multiple ways to specify instruction source:
- x(5) will set x with the value 5
- x(@address) will set x with the value found at address "address".
- x(@$FF3234) will set x with the value found at address $FF3234
- x(userdat1) will set x with the value found in current object's userdat1 register.
- x(@userdat1) will set x with the value found at address in current object's userdat1 register.
- x([240) will set x with the value found in current object's address+240.
- x(@[240) will set x with the value found at address in current object's address+240.

Pre-formatted data

When using direct value (5 for example), it's possible to ask RMotion to transform in different format:
16.16 pixel format precision: 5.45|16.16
3.5 pixel format precision: 5.54|3.5

Basic instructions

* x(source), y(source), xadd(source), yadd(source)

x(1.5|16.16) set object's x position to 1.5 (in 16.16 pixel format).

* gfx(source)

gfx(mybitmap) set mybitmap as the new bitmap displayed.

gfx(mybitmap+1024) set mybitmap + 1024 octets (16*16 pixels in 8bits) as the new bitmap displayed.
gfx(1024,UPDATE) add 1024 to the current bitmap displayed.

* inactive(), active()

inactive() disable this object (not displayed)
active() enable this object (displayed)

* state(source)

state(5) change the state of this object to 5 (could be "monster is running"). Values are defined by the user. state() must use a value superior to 1 and make an implicit active().

* flipped(), normal()

flipped() the bitmap will be displayed mirrored.
normal() the bitmap is not mirrored.

* stop()

stop() stop the current animation, no more instruction will be executed. This instruction is the same as JUMP(0,WAIT)

* scale(), unscale()

scale() enable zoom function
unscale() disable zoom function

* speed(source)

speed(5) change Raptor's frame delay (how many frames to skip before next change : Raptor specific usage).

* rspeed(source)

rspeed(5) change RMotion's frame delay (how many frames to skip before executing another instruction in the animation).

* userdat1(source), userdat2(source), …, userdatB(source

userdat1 (50) will set the value 50 to the multipurpose register 1

* loop()

This instruction enable you to repeat many time a single or a bunch of instructions. To use loop(), you must define a loop point in your animation.

Example:
Myanimation:
...
...
userdat1(50)

```
mylooppoint: [declare a new loop point]
        x(50,UPDATE)
        gfx(1024,UDPATE,WAIT)
        ...
loop(userdat1,mylooppoint)
```

The piece of animation from "mylooppoint:" to " loop(userdat1,mylooppoint)" will be executed 51 times (defined in userdat1).

Note: loop() instruction does not WAIT. It's up to you to include WAIT option in at least one instruction inside the loop.

*jump(source)

Jump(source) with this instruction, you could jump on another animation of another portion of an animation

* wmem(source,destination)

wmem(userdat1,score,UPDATE) will update the value found in userdat1 at address "score"
wmem(@userdat1,score) will update the value found in the address pointed by userdat1 at address "score"

* scalex(source), scaley(source)

scalex(0.8|3.5) Set a 0.8 zoom factor (in 3.5 pixel format) to this object.

* clut(source), noclut()

clut(1) use palette number 1 for this object
noclut() disable palette usage (bitmap > 8bit)

* test(source, value, condition, destination) : conditional jumping

test(@ninjaxpos,200|16.16,PL,anim_walk_right) will look at value store in ninjaxpos and compare it with 200 (in 16.16 pixel precision) and if greater then animation will jump to anim_walk_right otherwise animation will continue with the next command

test(x,200|16.16,PL,anim_walk_right) will do the same but will look in the current object's x instead of loading value from memory

For the comparative value, you can use different format:
- free : 5 / $3245 / ACONSTANT
- fixed 16.16 : 4.5|16.16
- fixed 3.5 : 4.5|3.5
- from an internal register : x

condition field available :
- EQ : equal
- NE : different
- MI : inferior
- PL : superior

* Specific instruction

For those instructions, you will need to read the Raptor's documentation (but they work like basic instruction):
width(), height(), vbox(), hbox(), framesz(), framedel(), curframe(), maxframe(), animloop(), wrap(), timer(), track(), colchk(), tracktop(), washit(), coffx(), coffy(), remhit(), bboxlink()

Examples

Moving an object

```
RMA_move_example:          [declare a new animation]
Rspeed(2)                  [animation instruction will be proceed every 3 frames]
x(2|16.16,UPDATE)          [add 2 to object's x (in 16.16 pixel precision)]
y(2|16.16,UPDATE)          [add 2 to object's y (in 16.16 pixel precision)]
stop()                     [don’t do anything else]
```

*** SEE SVN EXAMPLE ANIMATION