FIGHTING GAME ENGINE

Developers Manual

Brief

This is a developers manual for the Fighting Game Engine

Author	Last changed	Version	Page
Jesper Klingvall, XBrix	2011-12-09	0.1.0	1 (15)

Innehållsförteckning

1 Description

This document describes how to make your own game using this Fighting Game Engine platform. In the future there might be a GUI assisted tool to make it easier to make your own game but in this document we describe the process without this tool.

1.1 Scripting

If you feel satisfied with the functionality in the platform as it is you can just edit script files, add graphics and sound to create a fighting game of your own.

1.2 Coding

If you feel that you need to add or alter the functionality in the platform you are more than welcome. You need to have some C++ coding skills and some know how about the developing environment in AmigaOS 4.x.

1.3 Porting

If you are porting this platform to an alternative platform you must state somewhere visible that it was first created on and for the Amiga OS 4.x.

Also you must give credit to the original authors of this project.

2 Directory structure

In the root drawer of the platform you find the following files and drawers.

config	Contains the main description file of the game
gfx	All the graphics in the game
sfx	All the sound effects and music
objects	The main characters and other objects on screen
scripts	All the level/screen descriptor files
fighter.exe	The runnable exe file for the game. Rename this to your game title

3 Config

The file game.txt is the first file the game engine is looking for that in turns points out the rest of the files the game needs.

In all the script files in you can add comments using the # sign. This is a good way for you to document the different bits and bytes of your game.

Lets break down the different parameters in this file using the supplied game as an example.

Title:Barbarian+

This is the title of your game and will be visible in windowed mode.

ScreenWidth:768

ScreenHeight:224

These two variables describe the width and height of the window or resolution in full screen mode.

Depths:32

This is the depth of your screen.

8 means 256 colours

16 means 64.000 colours

24 means Millions of colours divided into 8 bits (1 byte) per channel. Red, Green and Blue

32 is the same as 24 except that you now have access to an Alpha channel that controls the intensity of the pixel. Red, Green, Blue and Alpha.

FirstScript:waterfall.txt

This points to the first script file in the scripts drawer. In this case a playable level. It could also point to a intro screen IF the platform have support for it.

4 Scripts

In the scripts drawer all the level descriptor files resides. If the platform supports it all the intro, cut scenes and end scenes is also located here.

These are the supported commands in a script file:

Name:waterfall

Name of the level

Bitmap:mfb.bmp

The background graphics. If animated all the frames should be on this bitmap.

Keycolor:ff00ff

The colour that is not rendered on the screen making it transparent. In this case its magenta.

GravityY:10

Makes all object that has the gravity flag turned on fall down based on their mass.

GravityX:0

In this case we have no horizontal gravity. We could use a positive value to emulate hard wind going from left to right.

GroundX:0

GroundY:200

The position of the ground. In this case 200 pixels from the top.

Frames:4

The number of frames in the background animation.

4.1 Frame description

Frame:0,2,0,0,768,224

Frame:1,2,0,224,768,224

Frame:2,2,0,448,768,224

Frame:3,2,0,672,768,224

Description of each background frame:

The parameters given are: ID,Delay,CutX,CutY,CutW,CutH

ID: The ID of the frame. The first one numbered 0

Delay: How many game frames should pass between each background frame incrementation. 50 game frames each second. One game frame is 20 milliseconds.

CutX and **CutY:** The top left cutting position in pixels.

CutW and **CutH:** The size of the cutting area.

Note that the cutted area is always pasted on the position 0,0 of the render screen.

4.2 Object description

Object:1,1,1,0,0,0,30,0,0,0,0,barbarian.txt

Object:2,1,1,0,0,0,730,0,0,0,0,barbarian.txt

Object:3,1,0,1,0,0,0,-20,1,1,1,head.txt

Object:4,1,0,1,0,0,0,-20,2,1,1,head.txt

Object:5,1,0,1,0,0,300,-180,0,0,head.txt

Object:6,1,0,0,0,0,200,-150,0,0,0,head.txt

The object description defines which object should be placed where and how its attached in the level.

The parameters are: ID, Visible, Playable, Gravity, SpeedX, SpeedY, XPos, YPos, AttachedToObjectNr, ShareInvX, ShareInvY,Scriptfile

ID: The number of object on the screen

Visible: Should this be visible initially on the screen

Playable: Should this be a playable character either by the user or by the computer AI. If set to zero it's just a object that is controlled by other factors.

Gravity: If set to one the object reacts to the levels gravity.

SpeedX and **SpeedY:** Initial speed of the object.

XPos and **YPos**: Defines where the object should be placed in

AttachedToObjectNr: Which object should this object be attached to. If attached to the ground give value 0.

ShareInvX and **ShareInvY**: Set to 1 if the object should invert itself when the object it is attached to gets inverted.

Scriptfile: The script file that describes the object.

5 Objects

In the objects drawer, all the game objects are stored. These objects can either be the main characters, enemies or other objects needed by the game.

5.1 Main parameters

Name:Barbarian

This is the name of the object

Bitmap: barbarian_fixed.bmp

The first and foremost bitmap from where the object gets its graphics from.

BitmapInvX:barbarian_fixed_flipped.bmp

If we are using the ability to turn around horizontally we enter the bitmap to use here.

BitmapInvY:null

The vertically inverted bitmap.

BitmapInvXY:null

If we flip in both horizontal and vertical we use this bitmap. Otherwise null

Keycolor:ff00ff

Which color pattern should be transparent. In this case Magenta.

InvX:1

The object should be able to flip horizontally.

InvY:0

The object should NOT be able to flip vertically.

InvKeysX:1

When the object flips and its playable it should also invert the controls I horizontal. This can be denied later in the moves section. Like in Barbarian we base our moves based on which way the character I facing except for the walking left and right. This is overrided later in the walk left and right moves.

InvKeysY:0

Since we are not given the character to flip vertically we don’t need to invert the controls either.

InvMoveX:18

Tells which move to be used when the object flips horizontal

InvMoveY:

Tells which move to be used when the object flips vertically.

FacingOpponentX:1

If set to one the object will flip horizontal when passing the opponent.

FacingOpponentY:0

Since this is a 2d fighting game we don’t need to flip I Y Axis

Mass:90

The weight of the object

Visible:1

Should this object be visible. The Visibility in the level descriptor and this parameter should be 1 for it to be visible. This parameter can be altered during the game via a triggered event.

Playable:1

Tells if this object is a playable character either by the player or by a computer AI.

5.2 Movements

Movements or moves describe how the object should behave based on input from the player or by interaction from other objects movement.

5.2.1 Parameters

ID, Baseframe, Topframe, SpeedX, SpeedY, InvFilter, TriggerType, AnimationType, HoldOnFrame, TriggerKey, TriggerKey2, TriggerKey3

ID The identification number of the move

Baseframe Animation starts from this frame

Topframe …and ends with this frame.

SpeedX This is the speed the object should have whilst performing this move.

SpeedY Same as above except that it is the vertical speed in pixels.

InvFilter If set to 1 it denies the inverted X controls when the object is flipped. If set to 2 it denies the inverted Y controls and set to 3 denies both X and Y inverted controls.

TriggerType Set to 0 means its not triggered by a key. 1 means triggered by a players input. If set to 2 it is triggered by another player objects move when collision have occurred. In that case TriggerKey defines which move.

AnimationType 0 means that when the current animation reaches the topframe it starts over from the bottom frame. 1 Cycles up to the frame entered in HoldOnFrame and when the key input from the user stops it continues to the TopFrame. 2 means it cycles through the frames one time and then stops.

HoldOnFrame Tells which frame the animation should stop before continue if the AnimationType parameter is set to 1.

TriggerKey The first key that has to be activated to get this move to occur.

TriggerKey2 and **TriggerKey3** if set to something else than 0 these keys all have to be activated in order for the move to occur.

5.2.2 Trigger keys

This is the keymap for the moves. If two players are selected the platform automatically calculates the rest.

This routine will be changed in the future when I someone else figures out a better way to do it.

0	No key		
Player 1		**Player 2**	
1	Button 1	21	Button 1
2	Button 2	22	Button 2
3	Button 3	23	Button 3
4	Button 4	24	Button 4
5	Button 5	25	Button 5
6	Button 6	26	Button 6
7	Button 7	27	Button 7
8	Button 8	28	Button 8
9	Button 9	29	Button 9
10	UP W	30	UP Arrow_Up
11	DOWN S	31	DOWN Arrow_Down
12	RIGHT D	32	RIGHT Arrow_Right
13	LEFT A	33	LEFT Arrow_Left
14-20	Reserved	34-40	Reserved

5.2.3 Examples

```
#Stand still

Move:0,62,73,0,0,0,1,0,0,0,0,0

# Kick

Move:1,36,38,0,0,0,1,1,37,12,11,1

# Fall backwards when getting kicked by opponent

Move:17,74,78,0,0,0,2,2,0,1,0,0

# Turn around with jump when players character have passed the opponent

Move:18,39,41,0,0,0,0,2,0,0,0,0
```

5.3 Frames

Frame is a description of an objects movement animation.

5.3.1 Parameters

ID,Delay,CutX,CutY,CutW,CutH,ZeroX,ZeroY,LethalX,LethalY,Damage,HitX,HitY,HitW,HitH,Shield

ID The number of the frame

Delay This tells how many game cycles should pass before the next frame

CutX and **CutY** The top left pixel on the source bitmap where this frame should be cut from.

CutW and **CutH** The width and height of the cutting area

ZeroX and **ZeroY** The zero point on this frames.

LethalX and **LethalY** When this point occurs within the opponents HitBox the opponent looses energy.

Damage The damage this frame inflicts on the opponent when in collision

HitX and **HitY** The top left coordinate of this objects hitbox

HitW and **HitH** Width and height of the hitbox

Shield If opponent hit this object, this value defines how well this object can withstand the hit.

5.3.2 Examples

```
# Walk right

Frame: 0,7,8,9,55,62,30,61,0,0,0,0,0,0,0,0

Frame: 1,7,72,9,55,62,30,61,0,0,0,0,0,0,0,0

Frame: 2,7,136,9,55,62,30,61,0,0,0,0,0,0,0,0

Frame: 3,7,200,9,55,62,30,61,0,0,0,0,0,0,0,0
```