LipSyncing in the CryEngine®

2004-09-08

How it works

Preprocess

- Phonemes are extracted and edited using the program MagPie.
- For each phoneme a morph-target is specified.
- A dialog is compiled into a sync-sequence of phonemes.

Real-time

- The dialog is played and according to the sync-sequence the appropriate morph-targets are blended together to reflect the facial-expression. Additionally expressions can be defined via script or animation-sequences (CryMovie).

Preparing the character model

The character-model must provide all morph-targets used for lip-syncing/expressions. For the system it doesn't matter which phonemes you choose, as long as the phonemes match correctly in MagPie. The morph-target names for the phonemes **must** be `Phoneme_<phonemename>` (e.g. "`Phoneme_th`").

Creating sync sequences

1. Start MagPie and make sure the **framerate** is set to **30**.
2. Create a new expression-set, add all you phonemes to the set (the phoneme name must match the one in the morph-targets, but without the prefix '`Phoneme_`'. For example, if the morph-target is called '`Phoneme_th`', the matching expression in MagPie must be called '`th`'.
3. If you want another expression to trigger (which is not prefixed) you must name it `#[MorphTargetName]`. The hash (#) indicates that no prefix should be attached.
4. Additionally to the phonemes you will need a base-expression (no expression) which **must** be called '**<base…>**' (where … indicates an arbitrary suffix).
5. You can have up to 3 expression-sets to support facial-expressions bound to the dialog. Normally you would create one expression-set for all the phonemes to do the lip-sync-part, and another expression-set for all expressions like angry, sad, etc.
6. After syncing you wave-file you can export the data to the **TextFile**-format and save it **in the same folder as the wave-file** which will be loaded by the game.

For details on how to use MagPie consult the MagPie reference.

How to show LipSyncing in the CryEngine

Every entity which has a character loaded can play **one dialog at a time** by calling `self:SayDialog()` (where self is a CScriptObjectEntity).

The parameters to this function are the following:

SayDialog() parameters

Parameter	Parameter name (internal)	Description
1	pszFilename	Path the wave file to be played
2	nVol	Volume of sound (0-255)
3	fMin	Minimum sound attenuation radius
4	fMax	Maximum sound attenuation radius
5 (opt)	nFlags	Sound-Flags (analogous to Load3Dsound)
6 (opt)	fClipDistance	ClipDistance of sound

This function will load and start playback of the wave file and, if a sync-file is found, will automatically be lip-synced.

Note: If you make another call to `SayDialog()` while the character is still busy saying something, the system will stop and unload the first dialog then load and play the new one instead.

To abort a dialog you can call `self:StopDialog` (where self is a CScriptObjectEntity).

About expressions

There are 2 different kinds of expressions available:

- Automated, random expressions (e.g. eye-twitching, blinking etc.)
- Scripted expressions (e.g. Sadness, happiness etc.)

How to do automated, random expressions

Create a new LUA script-file and define a table called 'RandomExpressions'.

In this table you can define sub-tables named by the morph-target with a set of parameters which define how this expression should be triggered.

Example:

```
RandomExpressions={                 -- main table
      rndexpr_blink={               -- morph-target name
            -- parameters for expression "blink" start here
            Interval=5,
            IntervalRandom=1,
            Amp=1,
            AmpRandom=0,
            BlendIn=0.1,
            Hold=0,
            BlendOut=0.1,
      },
      -- next expression goes here
}
```

The following parameters are available for use in the LUA script.

Scripting parameters for random expressions

Parameter name (internal)	Description

`Interval`	Time in seconds this expression in repeated
`IntervalRandom`	+/- random time difference in interval
`Amp`	Amplitude of expression (0-1)
`AmpRandom`	+/- random amplitude difference
`BlendIn`	Blend in time, in seconds
`Hold`	Hold-time (no change in amp) in seconds
`BlendOut`	Blend out time, in seconds

To activate random-expressions for a character, call `self:DoRandomExpressions()` (where self is a CScriptObjectEntity).

The parameters to this function are as follows:

Parameters for DoRandomExpressions()

Parameter	Parameter name(internal)	Description
1	`pszFilename`	Path the script file to be used
2	`bRaiseError`	1 if an error should be raised when a morph-target isn't found, 0 otherwise.

To stop random expressions, call `self:DoRandomExpressions("");` or `self:ReleaseLipSync();` (where self is a CScriptObjectEntity).
The latter function will stop **all** Lip Sync activity (useful if the character dies).

How to do scripted expressions

To trigger specific expressions you can call `self:DoExpression()` (where self is a CScriptObjectEntity).
The parameters to this function are as follows:

Parameters for self:DoExpression()

Parameter	Parameter name (internal)	Description
1	`pszMorphTarget`	Name of morph-target
2	`fAmplitude`	Amplitude of expression (0-1)
3	`fBlendIn`	Blend in time, in seconds
4	`fHold`	Hold-time (no change in amp) in seconds
5	`fBlendOut`	Blend out time, in seconds

Expressions can overlap. That means that if you start a new expression before the previous one is finished, the first one will **not** stop playing. The expressions will be additively blended. There is no limit, apart from the available processing power, to the amount of simultaneous expressions that can be replayed.

Scripted expressions in cut sequences (CryMovie)

To trigger expressions in cut sequences, follow these steps:

1. Start the Sandbox editor.
2. Show the trackview and open the cut sequence.
3. Add a character entity to your sequence.
4. Create an expression track for the character.
5. Place keys at the desired times on the expression track.
6. In the key properties dialog you will find the same parameters as described in 'How to do scripted expressions'. Edit these properties for each expression key.