Unleashing AuroraGt

06: Cinematics and Tasks

gameloft

Version

Date	Author	Version	Changelog
21/07/08	gaspar.deelias@gameloft.com	0.0.1	Initial Version

Guideline

Topics of this presentation:

- Tasks:
  - Syntaxes
  - Examples
- Cinematics
  - Syntaxes
  - Examples

AuroraGT

Introduction

- There are two ways of defining automated animations in our games (movies):

- TASKS:
  - Set of commands for creates auto-animations.
  - Deprecated
  - Very complex.

- CINEMATICS:
  - Simpler than tasks.
  - Graphical editing
  - Better previews
  - No need to know programming to create them.

Game Explorer

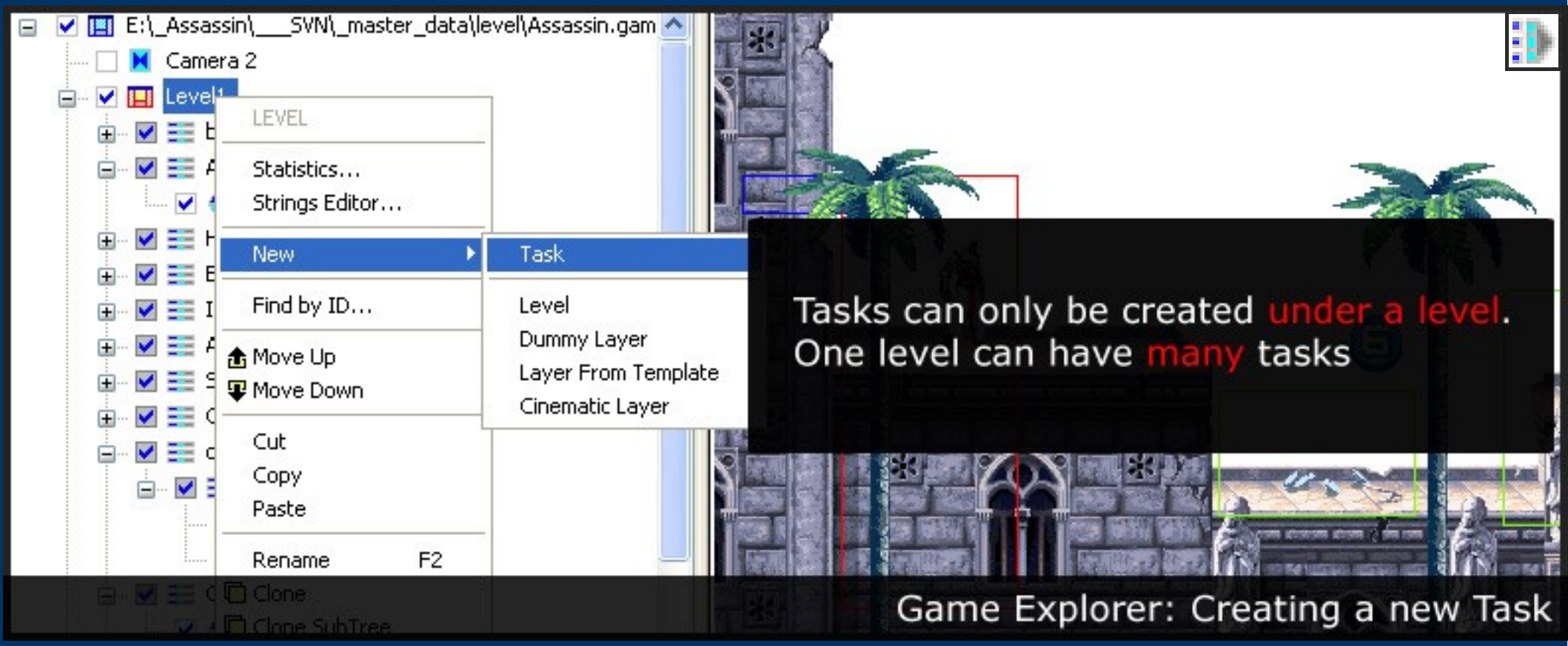
Tasks (deprecated)

- A task is a set of instructions that automates some of the game's behaviors without user intervention.
- Executed sequentially.
- Used for briefings, debriefings, and animations.
- Defined from AuroraGT Game editor, then exported into binaries and finally read by the code.
- An example...

Game Explorer

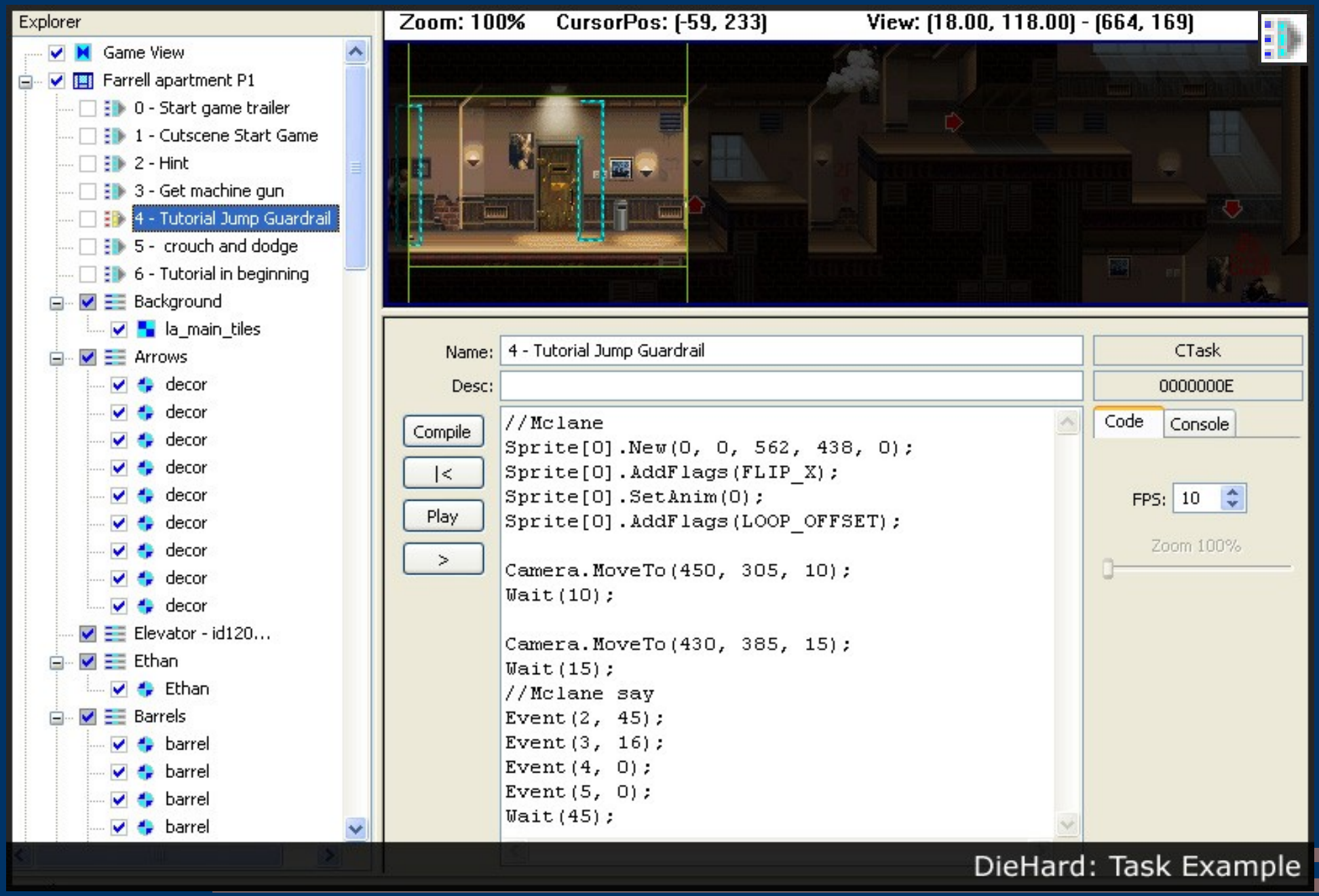
Tasks: Creating

- Using the Level Options we can create a task.



Game Explorer

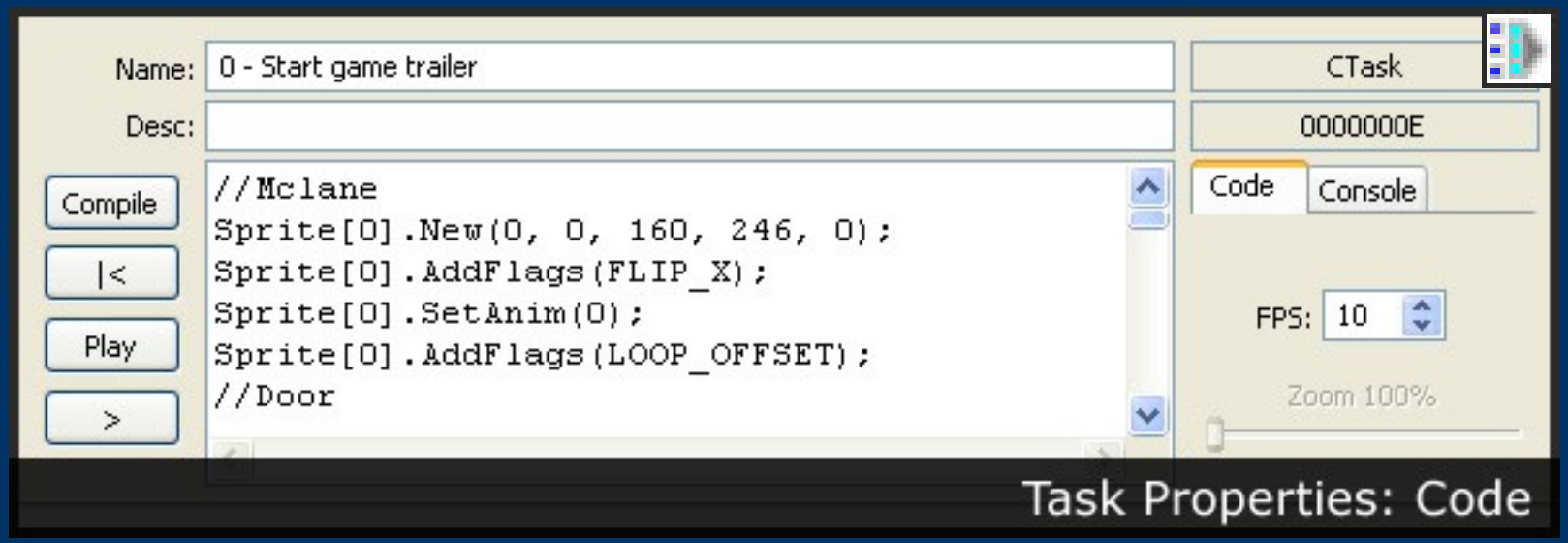
Tasks: Example DIEH



Game Explorer

Tasks: The code tab

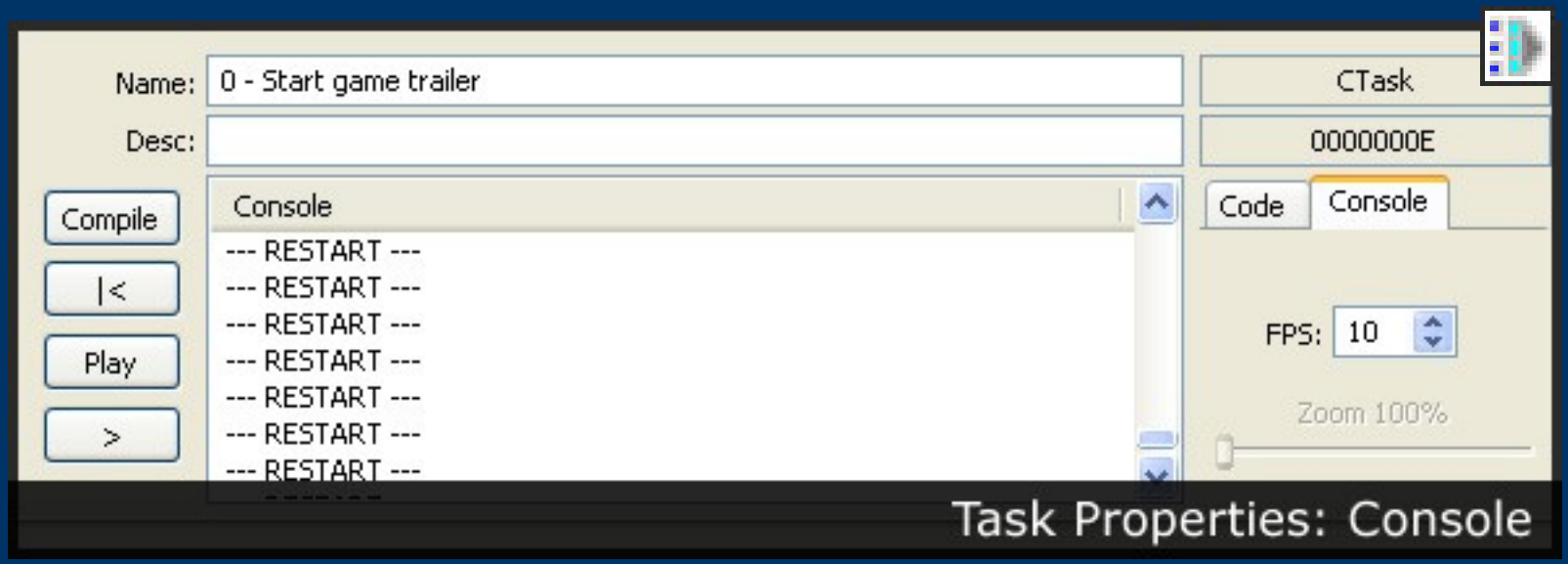
- In the code screen we write every Command.
- Compile before Play.
- Console Screen for debugging.
- FPS for simulation.
- Not all commands are shown in Preview Window.



Game Explorer

Tasks: The console tab

- In the console we can see the compiling result, and is used for debug.
- Here is shown every executed command after clicking on “play” button.



Game Explorer

Tasks: Extra debug info

Extra debug info if checked...

Explorer

- Game View
- Farrell apartment P1
  - 0 - Start game trailer
  - 1 - Cutscene Start
  - 2 - Hint
  - 3 - Get machine gun
  - 4 - Tutorial Jump Gu
  - 5 - crouch and dod
  - 6 - Tutorial in beginr
  - new Task
  - Background
  - Arrows
  - Elevator - id120...
  - Ethan
  - Barrels
  - Doors - id150 ...
  - Boxes

Zoom: 100% CursorPos: (672, 214) View: (-30.00, -47.00) - (695, 588)

Instruction: 17, OpCodePos: 87/120, WaitTime: 9
spr[3]: pos=(50, 277) module=-1 frame=-1 anim=107 aframe=0 time=0
spr[4]: pos=(128, 277) module=-1 frame=-1 anim=103 aframe=0 time=0
spr[5]: pos=(0, 277) module=

Instruction: The actual instruction being executed
OpCodePos: Percentage of opcode executed
WaitTime: Time to wait, defined by "Wait" commands
spr[n]: list of sprite instances created
pos=(x,y) actual position
module: moduleId
frame: frameId
anim: animationId
aframe: animation frameId
time: time left

Tasks: Extra debug info in Preview Window

Tasks

Syntax

```
Task instructions (exported by AuroraGT v0.3.5 or latter)

Camera.MoveTo(x, y, frames);
// Move camera to (x,y), divide this action into "frames" number of frames

Sprite[spr].New(parent, spr_id, x, y, z_order);
// Create a new sprite, where
//spr: identifier, we choose this number
//parent: unused??
//spr_id: Resource ID defined in building scripts (packaging)
// (x,y): initial position
// z_order: paint order

Sprite[spr].Delete();
// Delete the Srite

Sprite[spr].AddFlags(flags);     // DIEH Example: FLIP_X, INVISIBLE, PAUSE, LOOP_OFFSET
Sprite[spr].RemoveFlags(flags); // DIEH Example: FLIP_X, INVISIBLE, PAUSE, LOOP_OFFSET
// Set/Unset sprite flags, each game has its own flags
//this modifies the sprite._flags variable. For further info open the game code.

Sprite[spr].SetPos(x, y); //Set Sprite position
Sprite[spr].SetSpeed(vx, vy); // not used !
Sprite[spr].SetAcc(ax, ay); // not used !

Sprite[spr].SetModule(module, frame);
// Set module, where "module"/"frame" are indexes given by AuroraGT Sprite Editor
Sprite[spr].SetAnim(anim);
//Set animation to play, where "anim" is the anim index in AuroraGT Sprite Editor
Sprite[spr].SetAFrame(aframe, atime);
//Set animation frame "aframe" for "atime" time
```

Tasks

Syntax

```
SetAnimBase(base_index, anim_base);
// If we need to play anims that have correlative indexes (ex: 145,146,147)
// we can define a base to start so we refer to this anims as 0,1,2
// Create an element in the following array base_table[base_index] = anim_base

Sprite[spr].SetAnimEx(anim, base_index); // will do a SetAnim(base_table[base_index] + anim);
// We do SetAnimBase(0,145);
// So doing SetAnimEx(0,0); SetAnimEx(1,0); SetAnimEx(2,0);
// is the same as SetAnim(145);SetAnim(146);SetAnim(147);

Sprite[spr].ApplyAnimOff(); // apply the trajectory offset (if any)

Sprite[spr].SetPalette(pal); //set sprite palette

StartTask(task); // start new task (current one will be stopped) // not tested !
EndTask(); // end current task

WaitEndAnim(spr); // wait until sprite spr ends his anim
Wait(time); // wait time frames, animations will be updated

Event(event, param); // generate an event, with a parameter
//Events are custom for every game,
// The only way to know available events and params is to read the TaskUpdate() method
```

- Almost of all these commands depend on the code and the building system.

Game Explorer

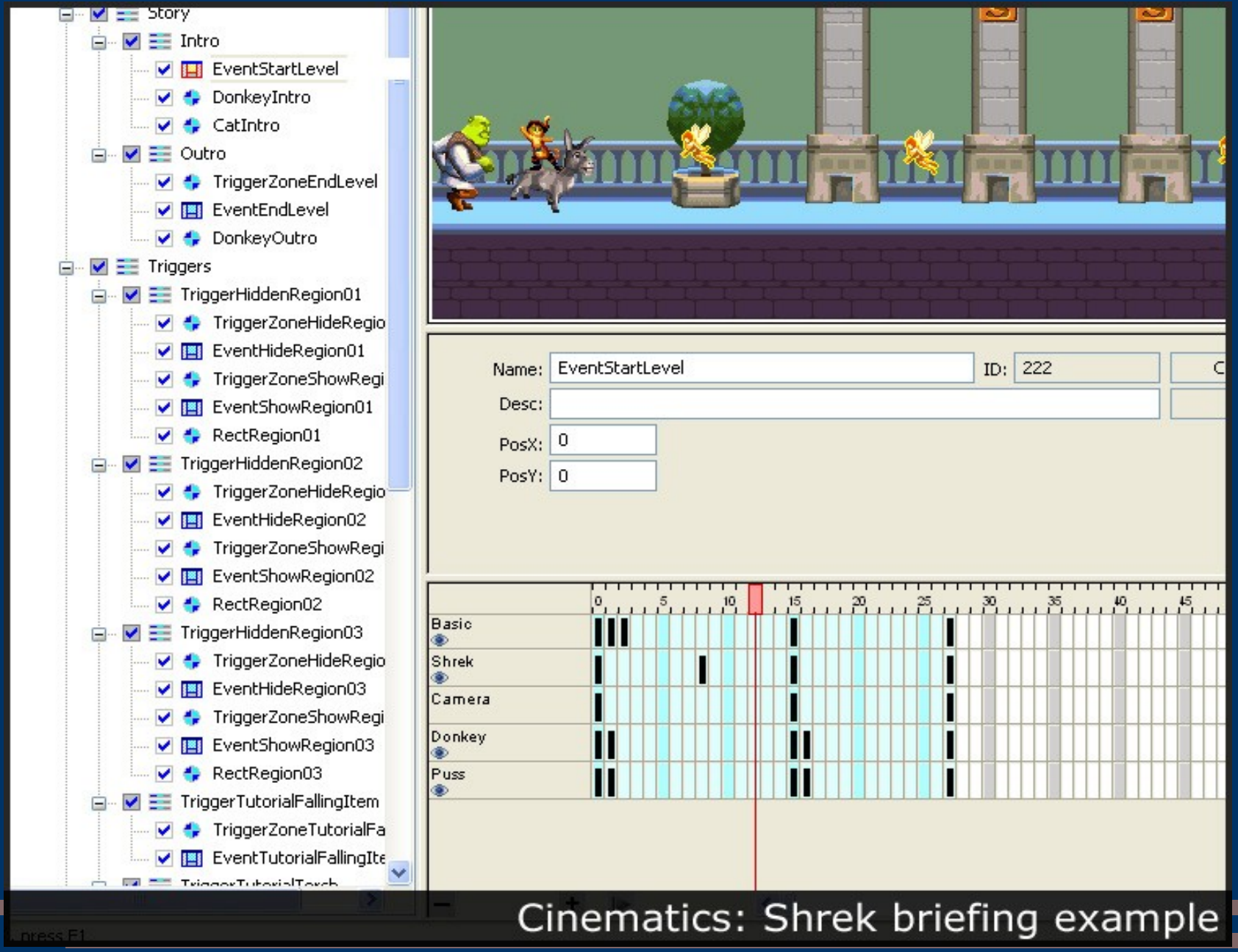
Cinematics

- Available since AuroraGT v0.8.1 beta
- Graphical way of creating tasks, its really easy.
- Events defined in a time line.
- Like flash (frames, keyframes, interpolation...)

- !!!!!!!!REMEMBER: Cinematic Commands are defined in GTS file!

Game Explorer

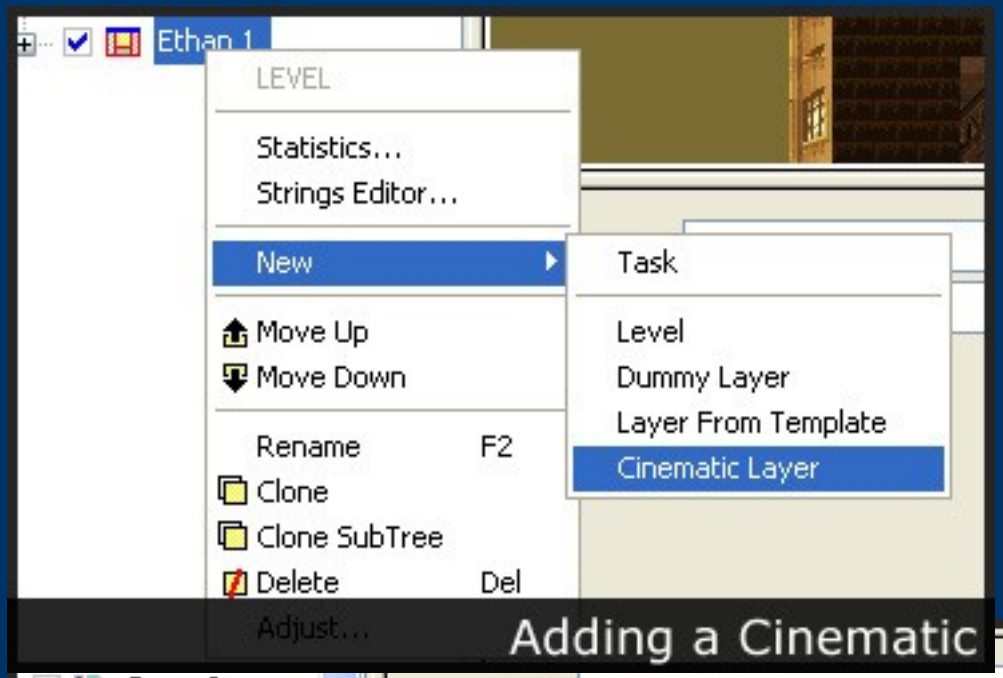
Cinematics: Example



Cinematics: Shrek briefing example

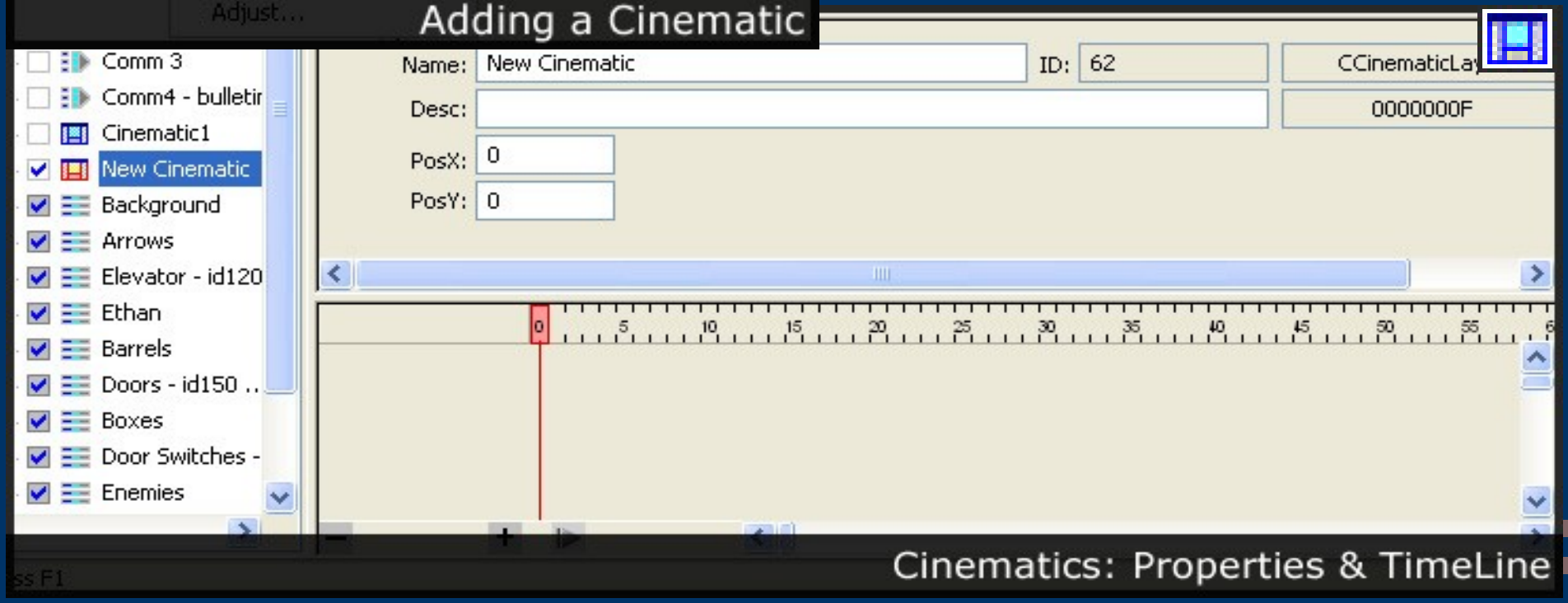
Game Explorer

Cinematics: Creating



Adding a Cinematic

- Added only under a level or a dummy layer.
- Properties and timeline windows in the following image:



Cinematics: Properties & TimeLine

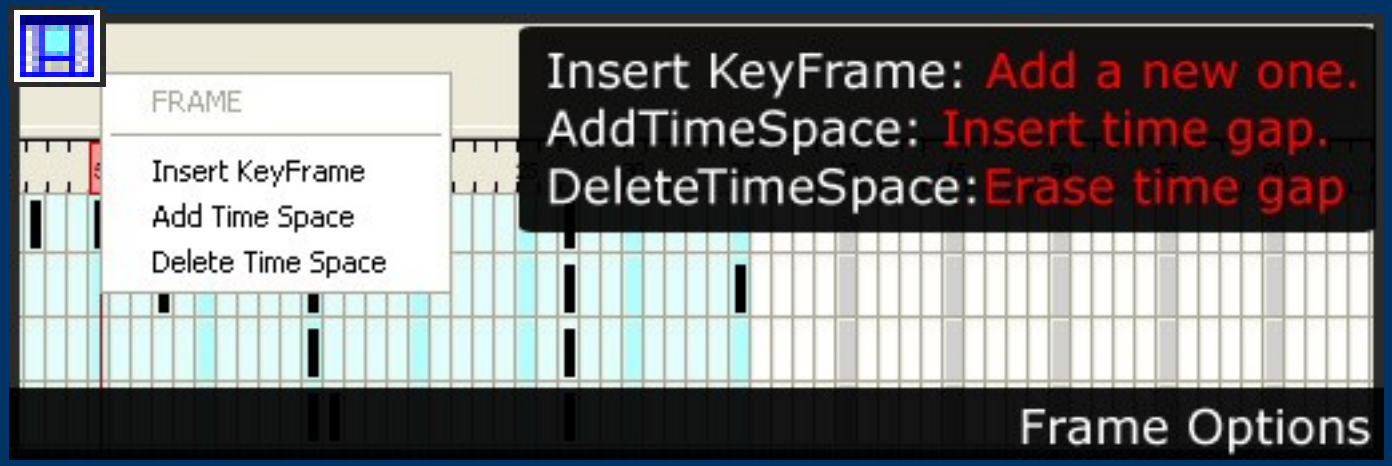
Game Explorer

Cinematics: TimeLine

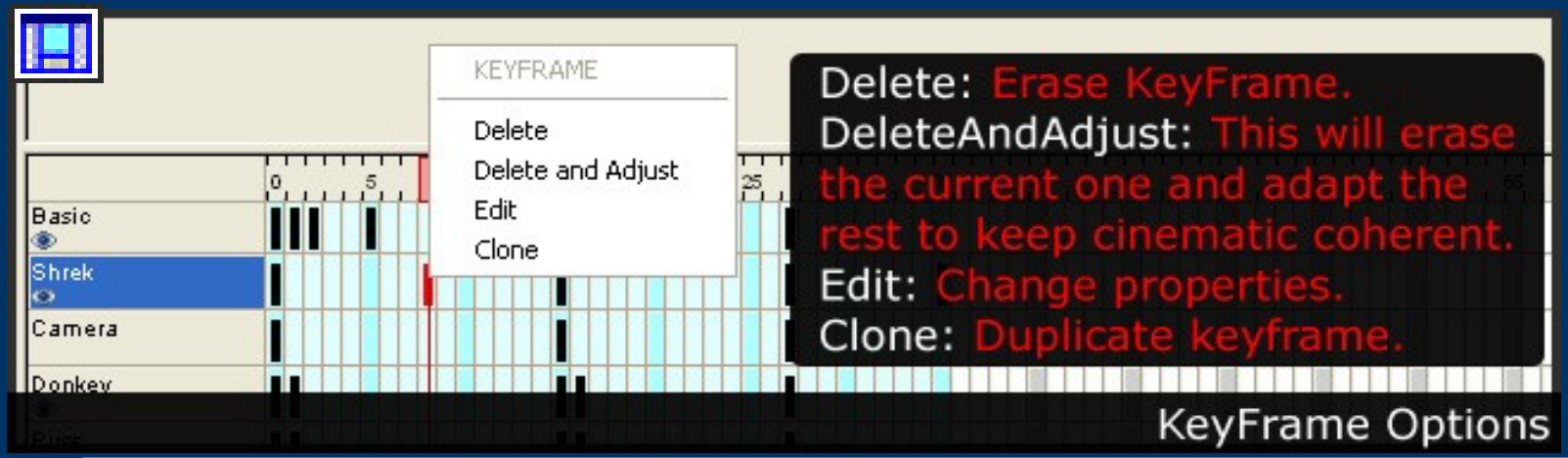
- Every ROW is a thread
- Every column is a frame
- KeyFrames are Special frames
- KeyFrames contain commands inside

Game Explorer

Cinematics: Frames and keyFrames



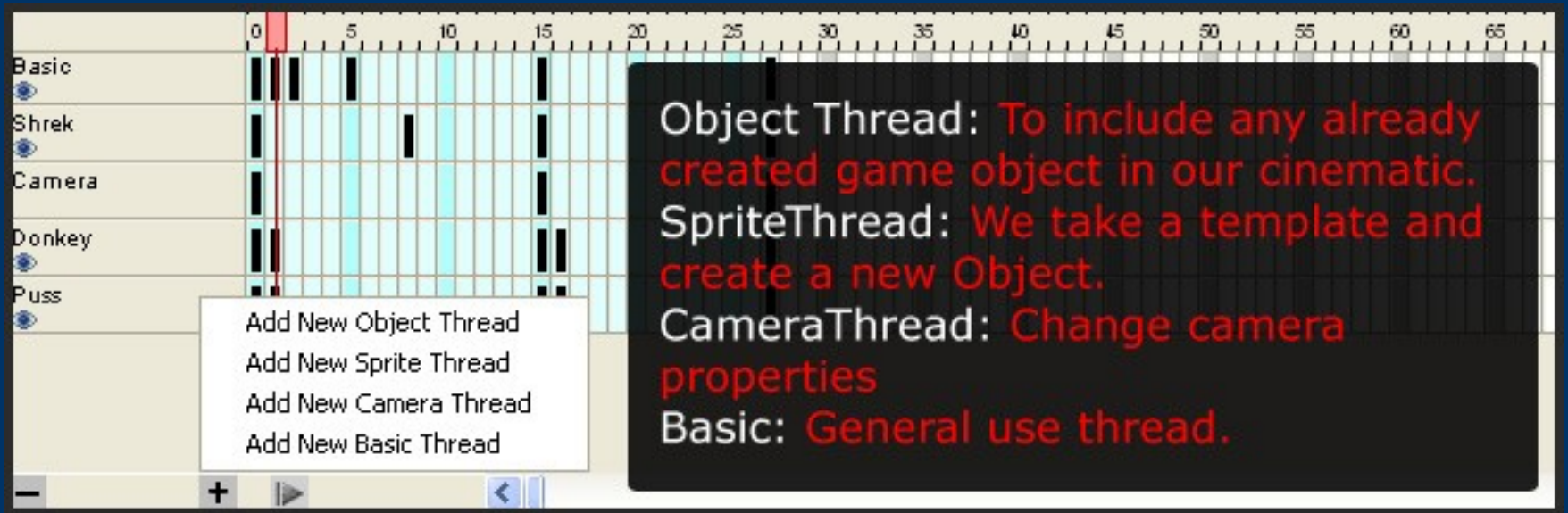
- We get these options menu right-clicking on frames/keyFrames



Game Explorer

Cinematics: Type of threads

- Every row is a “thread”.
- 4 kind of threads
- Different behaviors



Adding a Cinematic Thread

Game Explorer

Cinematics: Type of threads

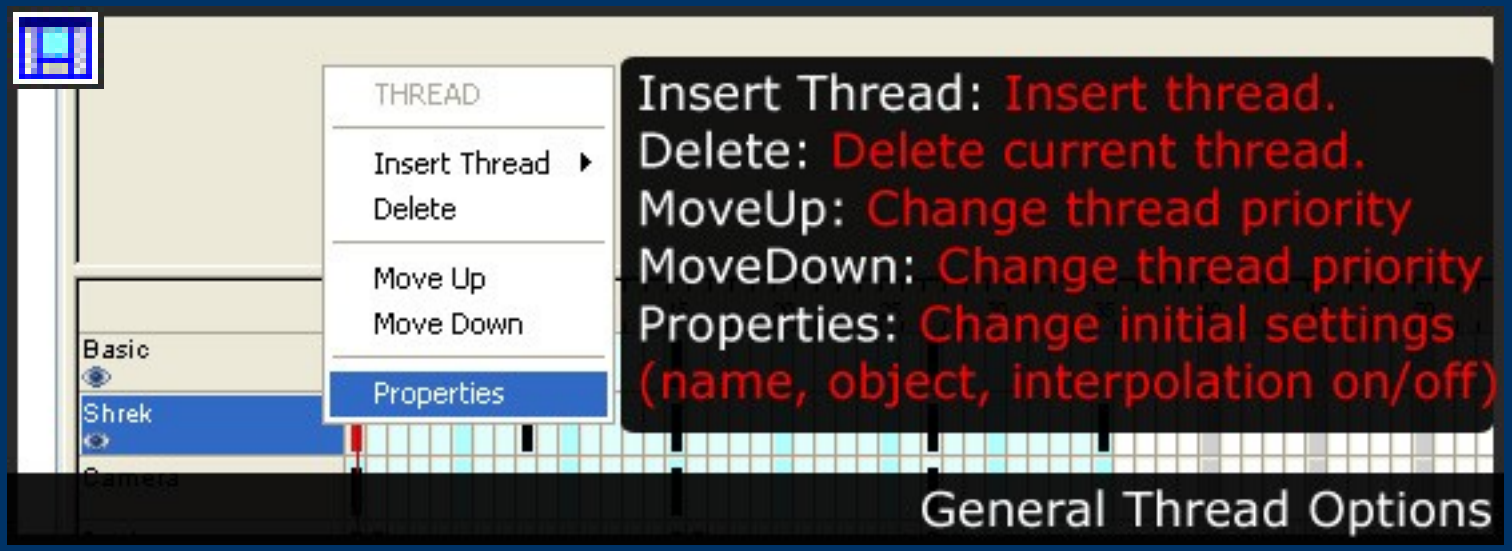
- General Thread Options

- Specific Thread Options:
  - Object Thread
    - Options
    - KeyFrame Options
  - Camera Thread
    - Options
    - KeyFrame Options
  - Sprite Thread
    - Options
    - KeyFrame Options
  - Basic Thread
    - Options
    - KeyFrame Options

Game Explorer

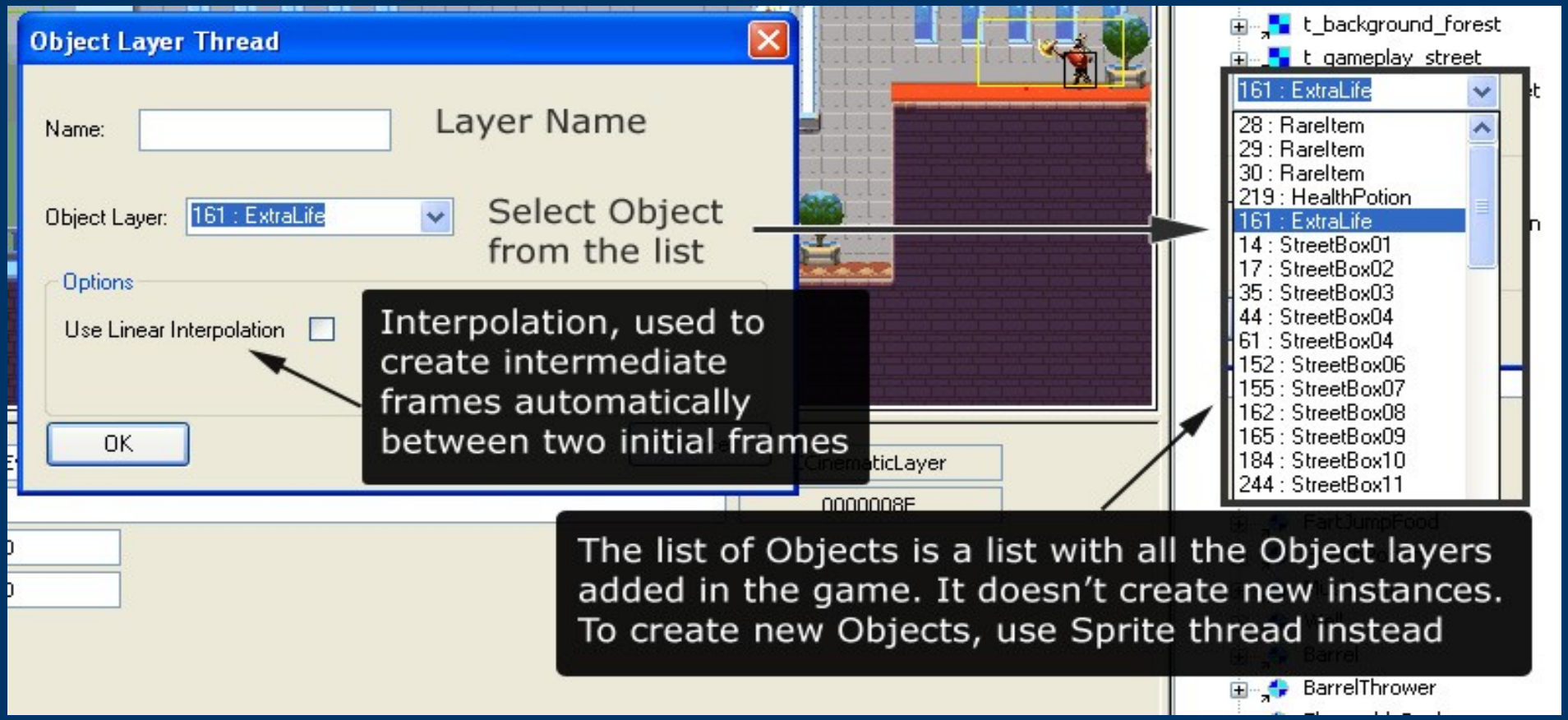
Cinematics: General thread options

- General Thread Options
- This dialog is the same for all kind of threads.



Game Explorer

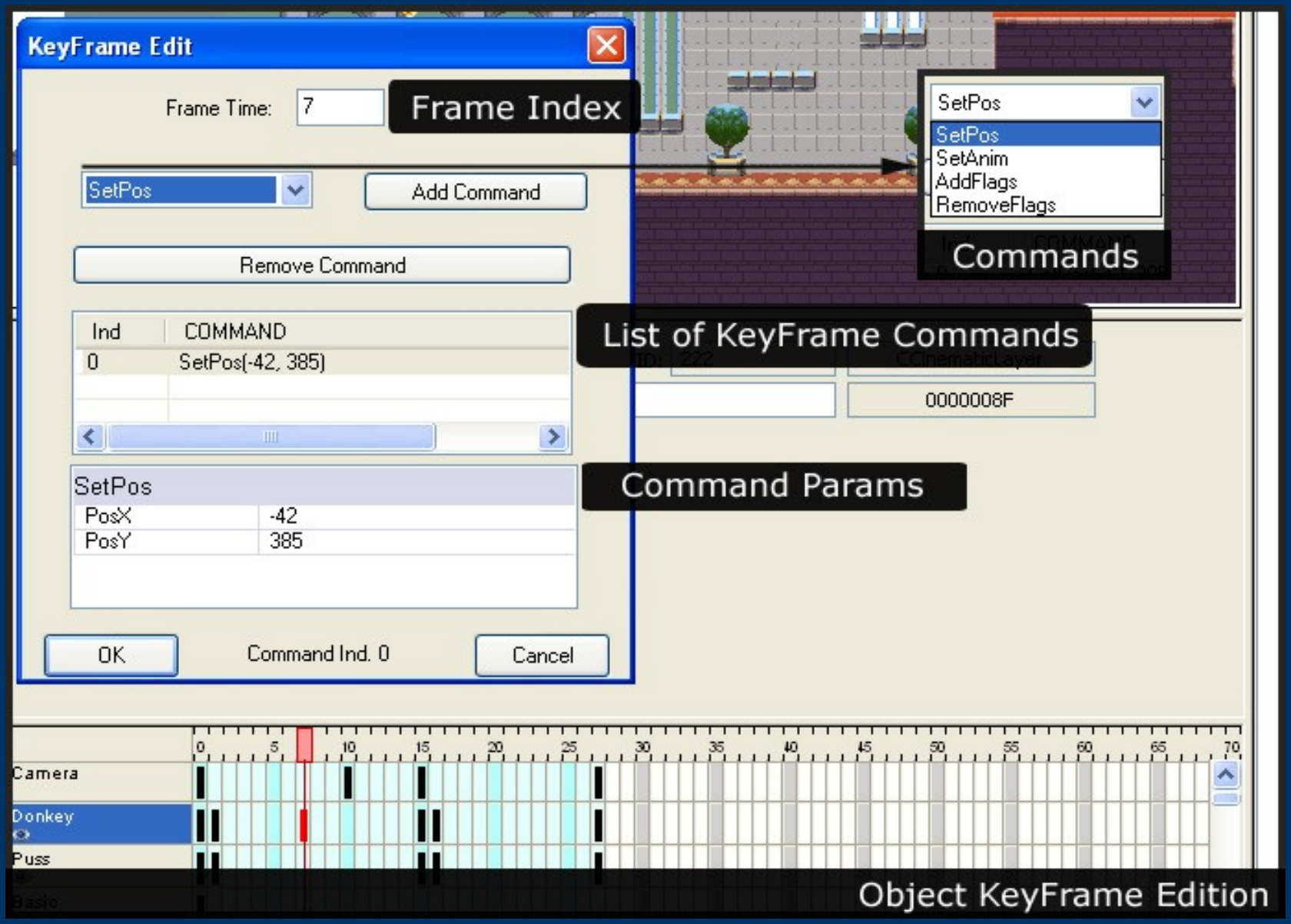
Cinematics: Object Thread Options



- Type of thread used to animate, change or modify objects of this game.
- It won't create new objects.

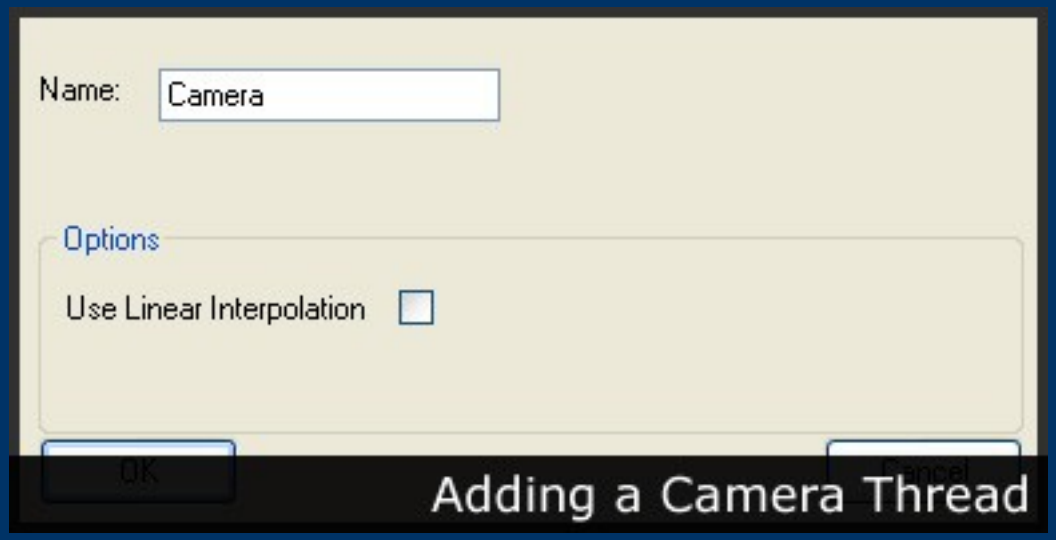
Game Explorer

Cinematics: Object Thread keyframes



Game Explorer

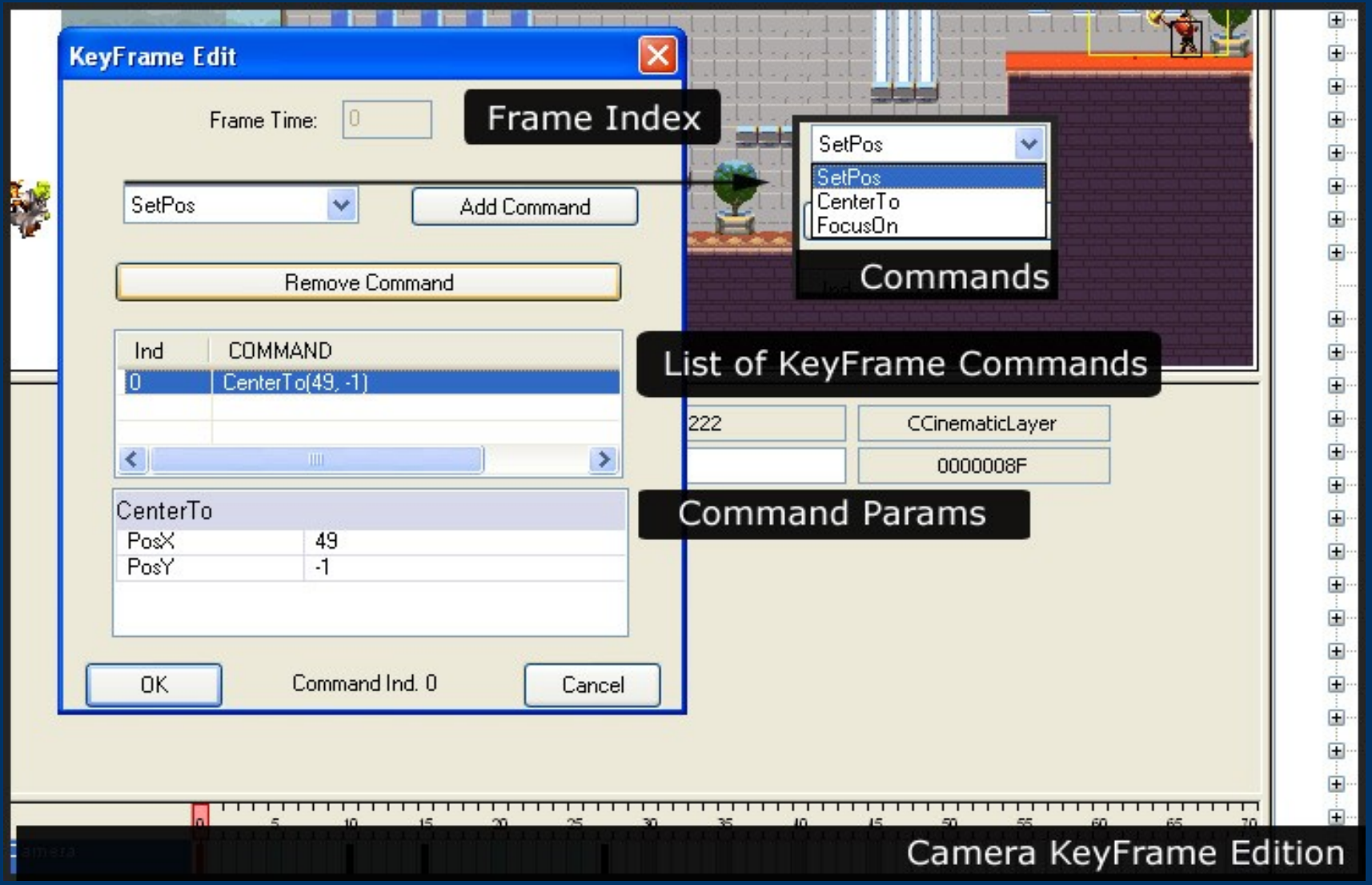
Cinematics: Camera Thread Options



- Name: Name of the Camera
- Interpolation: Transition between frames is made progressively during the time gap between them.

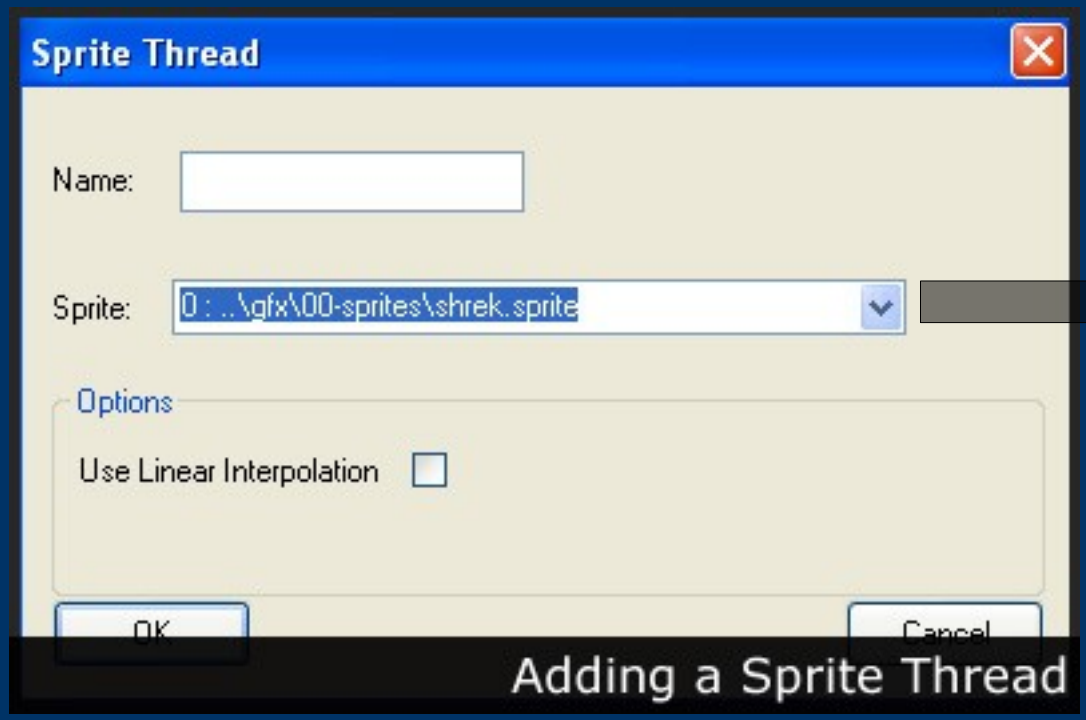
Game Explorer

Cinematics: Camera Thread keyFrames

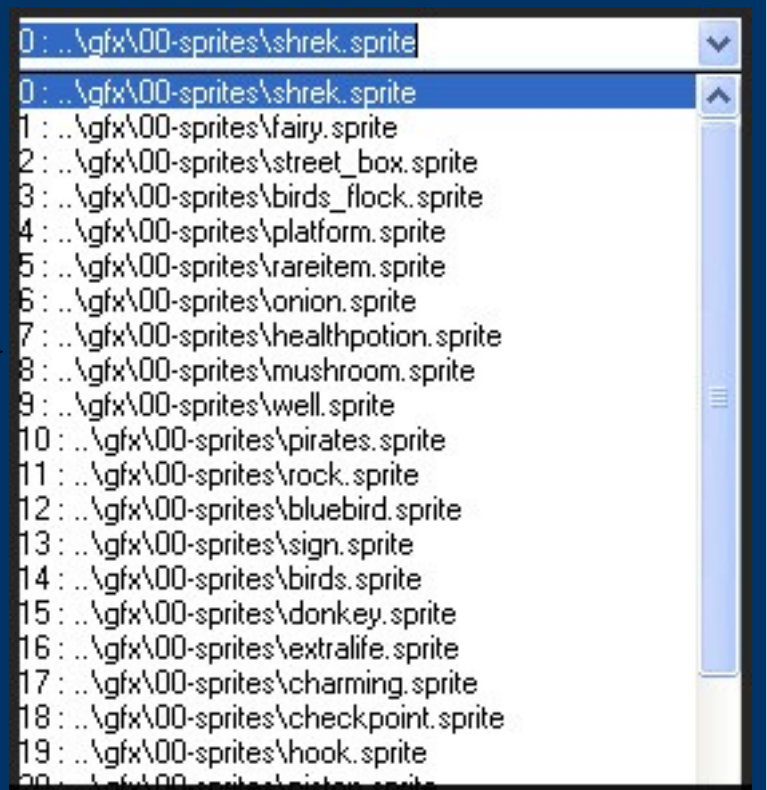


Game Explorer

Cinematics: Sprite Thread Options



Adding a Sprite Thread

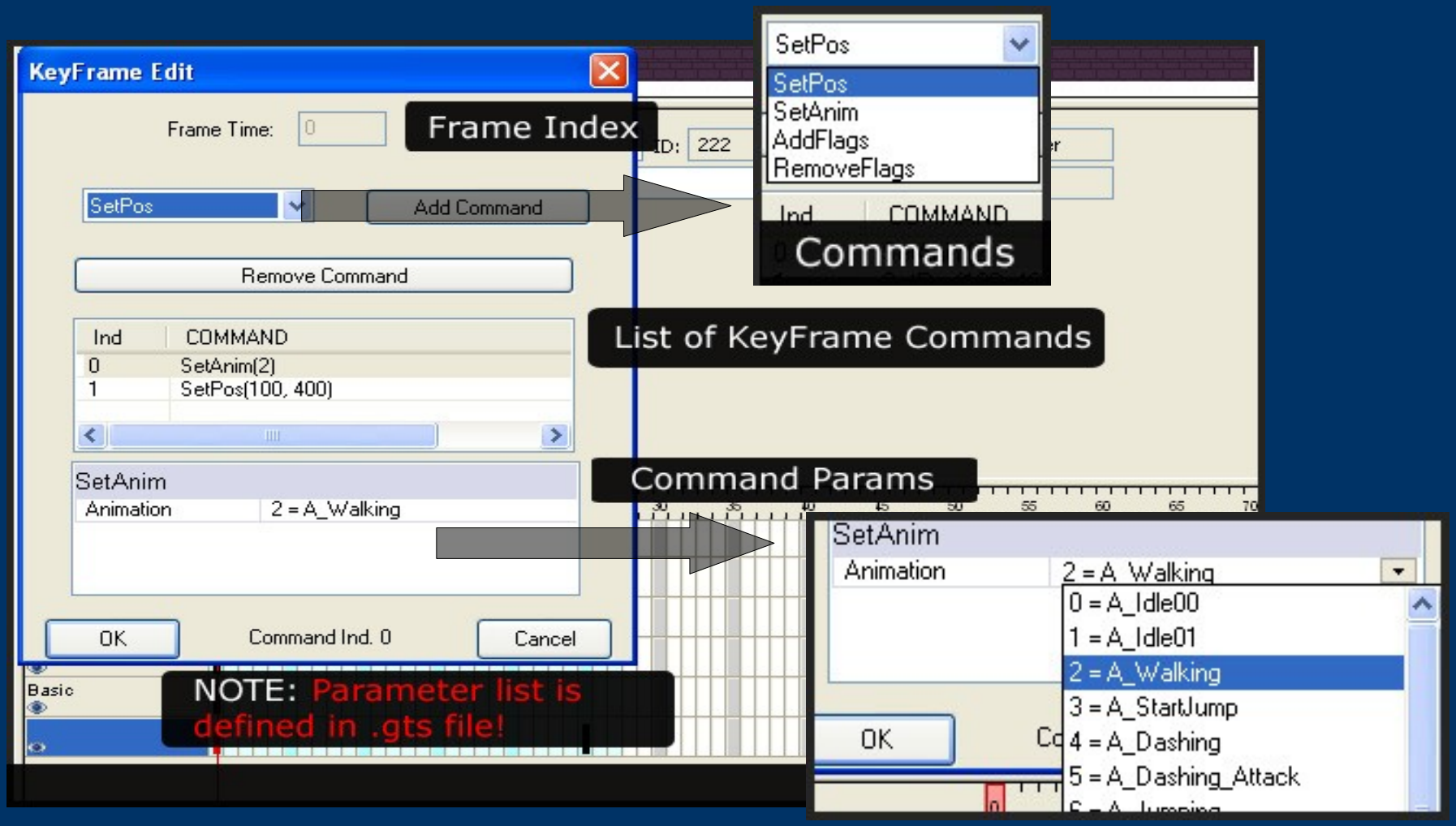


List of available sprites

- The sprite list is populated from the Sprites list in the templates window.
- It enable us to create new objects for the cinematic.
- Interpolation is available too.

Game Explorer

Cinematics: Sprite Thread keyFrames

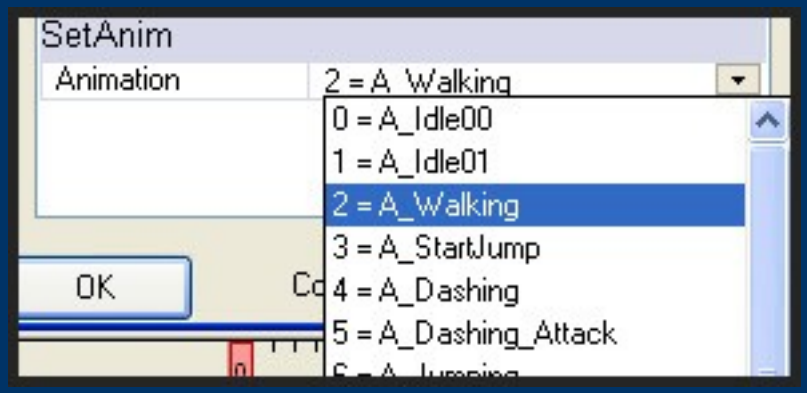


Game Explorer

Cinematics: Sprite Thread keyFrames

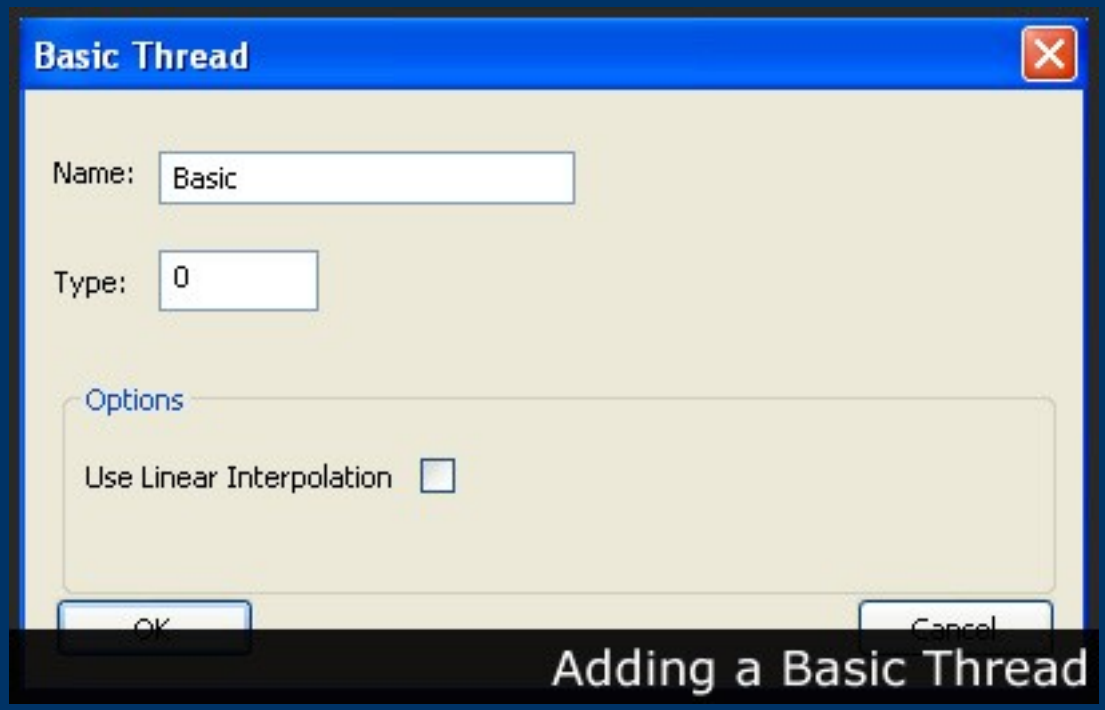
- Commands list by default (CMD)
- User defined commands can be added
- Refer to .gts file specification for user defined commands (NEW_CMD)

- SetAnim command has a comboBox with a list of possible parameters.
- This list is defined in the .gts file using the keyword TYPE



Game Explorer

Cinematics: Basic Thread Options

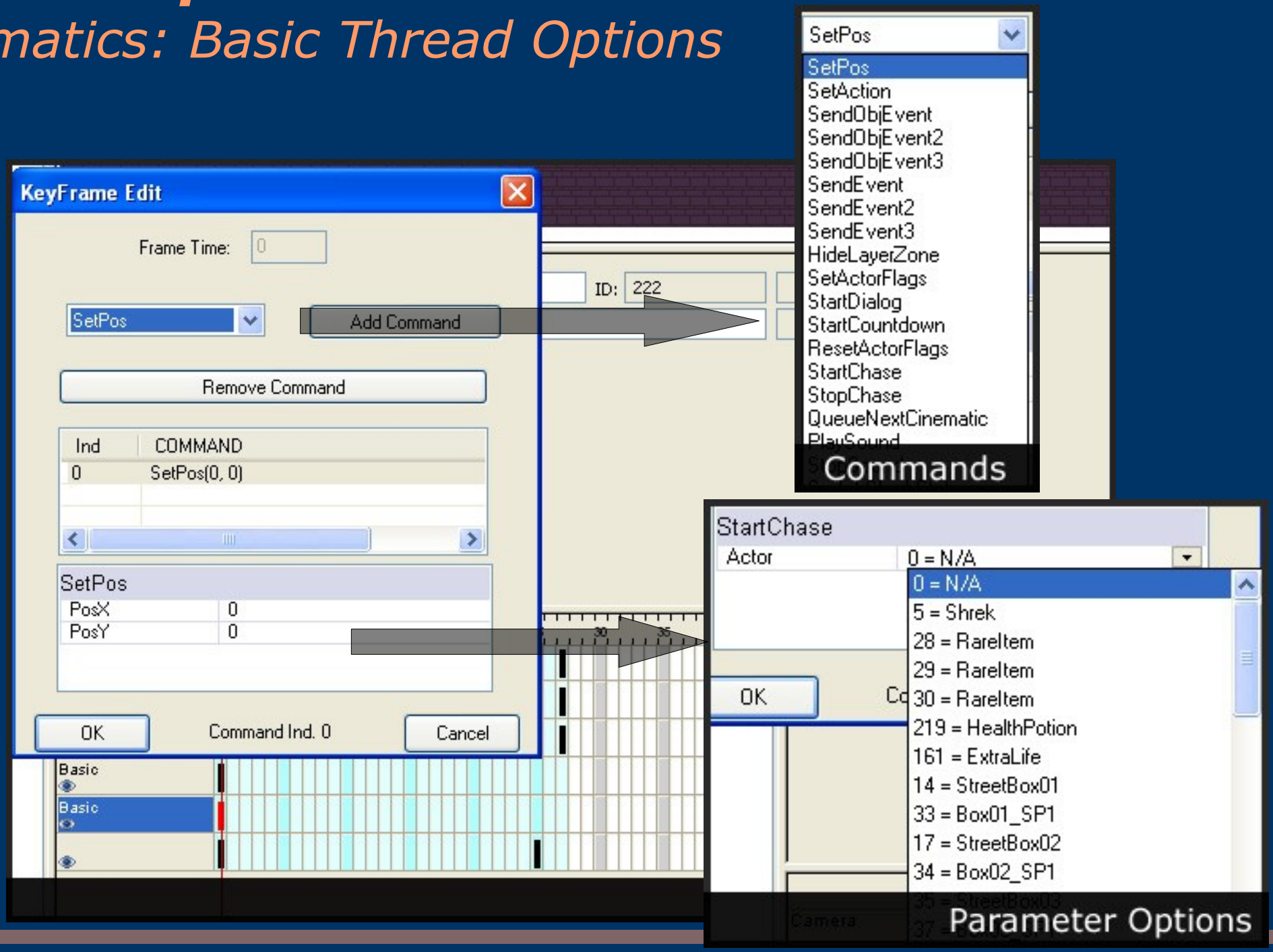


- Linear interpolation is available too.

- Name: we set here the name
- Type: By default set to 0. Can be customized but it needs to have its code counterpart to work.

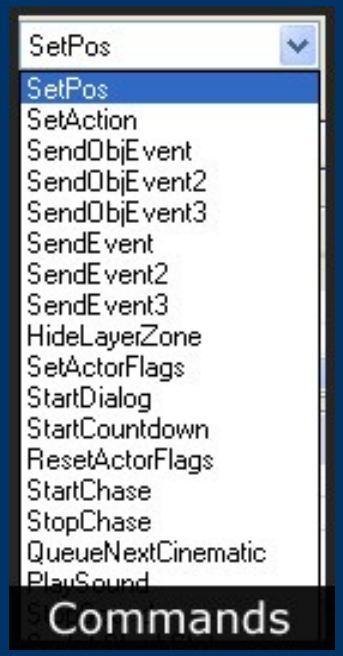
Game Explorer

Cinematics: Basic Thread Options



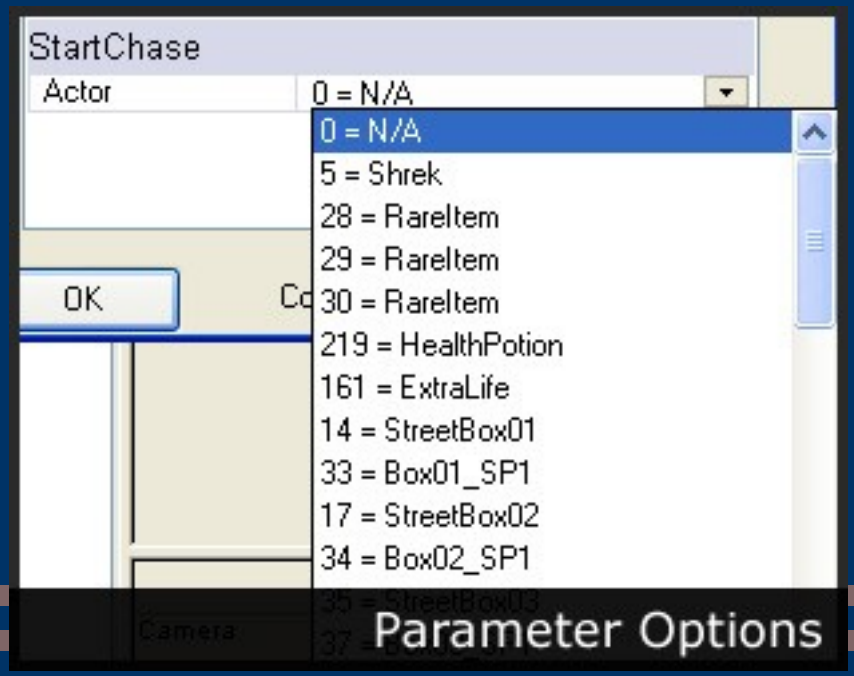
Game Explorer

Cinematics: Basic Thread Options



- List of Available commands.
- CMD + NEW_CMD

- ComboBox options defined in .gts file (TYPE keyword)
- Dev need to specify TYPES for CMD/NEW_CMD commands



Game Explorer

Cinematics: Review .gts file CMD

```
CINEMATIC_EDITOR
{
  CMD <THREAD_TYPE> "command_name"
  {
     // Parameters of this command
    [ PARAM <param1> TYPE "<DATA_TYPE>" ]
    [ PARAM <param2> TYPE "<DATA_TYPE>" ]
    ...
  }
}
```

We enclose everything inside this TAG

For a complete CMD list, please refer to the table in the next section

.gts

- These commands are already defined by default.
- We can not add CMD custom commands !!!!
- So, what's this CMD option for?

- Well, it's intended to be used for parameters specification.
- Basically we first create a DATA_TYPE somewhere in the .gts file.
- Then we add a CMD command to tell aurora to take the parameters list from our DATA_TYPE

Game Explorer

Cinematics: Review .gts file NEW_CMD

- User defined commands: NEW_CMD
- NEW_CMD properties:
  - Command name
  - Export ID (used by the code to identify it!)
  - Parameters TYPE and NUMBER
  - Default value

- NEW_CMD Specification:

.gts

```
CINEMATIC_EDITOR
{
  NEW_CMD <THREAD_TYPE> "command_name"
  {
   EXPORT_ID  <id>

    // Parameters of this command
    [ PARAM "<param0_name>" [ TYPE "<DATA_TYPE>" ] ] [ DEFAULT <value_or_valueIdx>]
    [ PARAM "<param1_name>" [ TYPE "<DATA_TYPE>" ] ] [ DEFAULT <value_or_valueIdx>]
    ...
  }
}
```

Again, this tag is needed for cinematics section

THREAD_TYPE is one of the following:
BASIC | SPRITE_INSTANCE | OBJLAYER | CAMERA

id to be exported

<parami_name>: the name of the i-th parameter
<DATA_TYPE>: an already defined datatype (optional)
<value _or_valueIdx>: an initial value, or a value index (if DATA_TYPE is a LIST)

Game Explorer

Cinematics: Basic Thread Options

- Custom commands defined in GTS
- Number and type of param is defined here
- Commands are finally implemented in the game code

```
3475 CINEMATIC_EDITOR
3476 {
3477   NEW_CMD BASIC "HideLayerZone"
3478   {
3479     EXPORT_ID 100
3480     PARAM "Rect" TYPE "obj_layers"
3481     PARAM "LayerID" TYPE "MapLayers"
3482   }
3483
3484   NEW_CMD BASIC "SetActorFlags"
3485   {
3486     EXPORT_ID 101
3487     PARAM "Actor" TYPE "obj_layers"
3488     PARAM "ActorFlags" TYPE "ActorFlags"
3489   }
3490
3491   NEW_CMD BASIC "StartDialog"
3492   {
3493     EXPORT_ID 102
3494     PARAM "DialogID" TYPE "DialogID"
3495     PARAM "ShowedTimeMS"
3496   }
3497
3498   NEW_CMD BASIC "StartCountdown"
3499   {
3500     EXPORT_ID 103
3501     PARAM "EndAction" TYPE "CountDownEndAction"
3502     PARAM "TimeInSecs"
3503   }
3504
```

.gts file: Cinematics

Game Explorer

Cinematics: Exporting

- Cinematics are exported with the game.
- Export options are defined in a .fft file
  Export Example
- We need to specify datatypes for custom created commands (INT8, INT16, etc...) in .gts file

Game Explorer

Cinematics: Exporting

- %TOOLS%\AuroraGT.exe "myGame.gamecmd"

- Binaries generated:
  - [levelName].cinematics: for each level.
  - Other binaries not under the scope of this document.

- Every game have some generic loading code:
  - loadCinematic();
  - startCinematic();
  - isCinematicRunning();
  - etc...

Game Explorer

Cinematics: Conclusion

- Cinematics are analog to macromedia flash movies.
- Now we can edit them knowing how they really work
- They are related to .gts files as well.
- After a game resize, our cinematics can be adapted too.

AuroraGT
Bibliography

- **AuroraGT official repository**
  https://terminus.mdc.gameloft.org/vc/tools/AuroraGT

- **AuroraGT main wiki**
  https://wiki.gameloft.org/twiki/bin/view/Main/AuroraGT

AuroraGT

Contact us

- Please, we look forward for any suggestions or bug found:
  - send us a mail to World-AuroraSuggestions@gameloft.com