Unleashing Aurora Gt

05: GTS Template file

Version

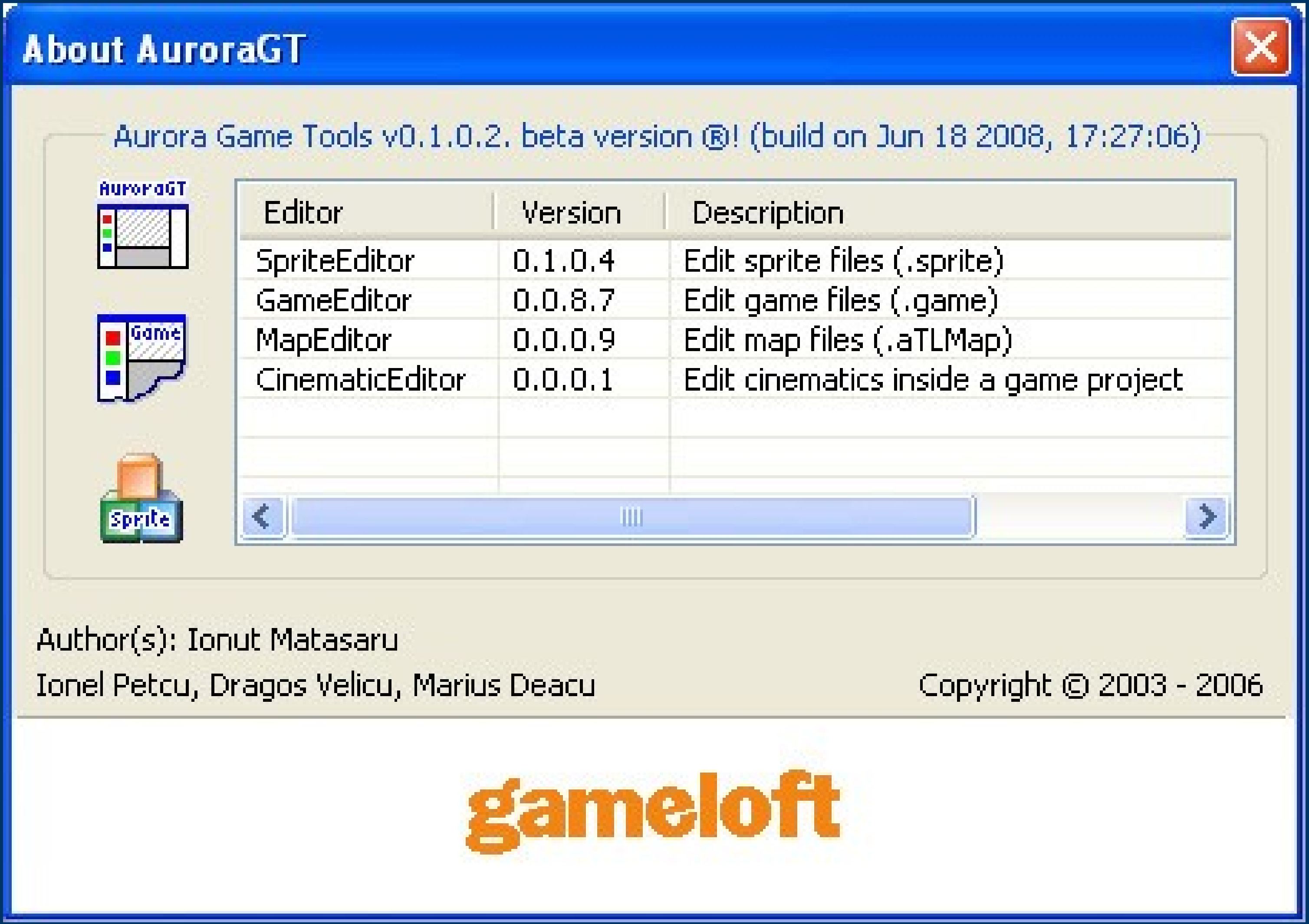
Date	Author	Version	Changelog
21/07/08	gaspar.deelias@gameloft.com	0.0.1	Initial Version

Guideline

Topics of this presentation:

- The GTS Template file:
  - Templates:
    - Object Layers
    - Tiled Layers
  - DataTypes:
    - List
    - BitSet
    - Matrix
  - Cinematics
    - CMD
    - NEW_CMD

Reference Version[1]



About AuroraGT

Aurora Game Tools v0.1.0.2. beta version ®! (build on Jun 18 2008, 17:27:06)

Editor	Version	Description
SpriteEditor	0.1.0.4	Edit sprite files (.sprite)
GameEditor	0.0.8.7	Edit game files (.game)
MapEditor	0.0.0.9	Edit map files (.aTLMap)
CinematicEditor	0.0.0.1	Edit cinematics inside a game project

Author(s): Ionut Matasaru
Ionel Petcu, Dragos Velicu, Marius Deacu

Copyright © 2003 - 2006

gameloft

[1] https://terminus.mdc.gameloft.org/vc/tools/AuroraGT (r1189)

AuroraGT

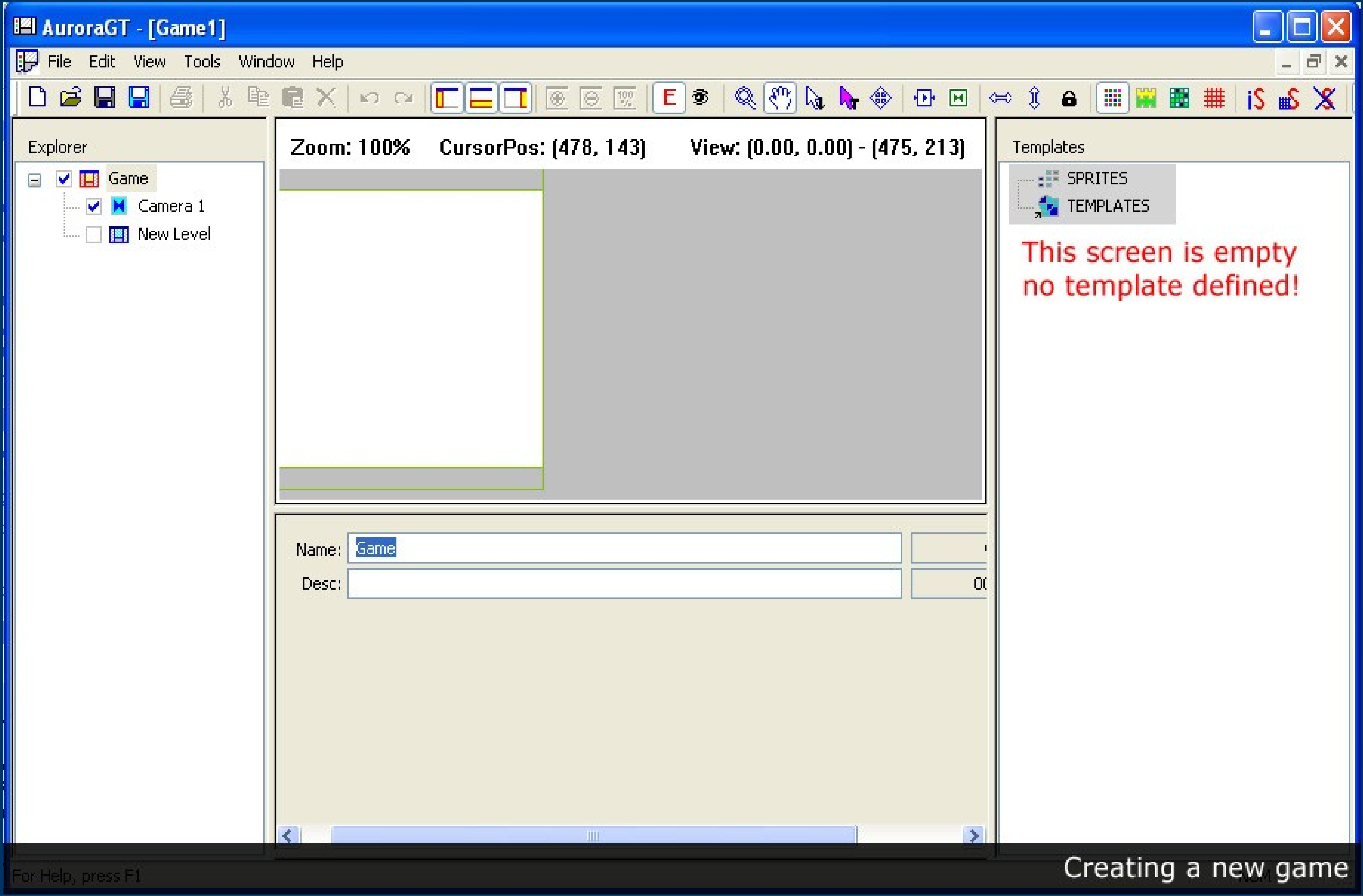
Flexibility

- The .gts file:
  - Is created by hand and maintained by developers.
  - Every game has its own template file.
  - Contains a list of all resources that can be used by the game designer when creating a game using AuroraGT game editor.
  - All game maps, objects, and cinematics properties are defined in this file.
  - We cannot create a game using AuroraGT game editor without a template file.
  - Works like an abstract layer between the specific game and the game editor.

AuroraGT

The templates file (.gts)

- This is an empty created game:


AuroraGT - [Game1]

File Edit View Tools Window Help

Explorer

- Game
  - Camera 1
  - New Level

Zoom: 100% CursorPos: (478, 143) View: (0.00, 0.00) - (475, 213)

Name: Game

Desc:

Templates

- SPRITES
- TEMPLATES

This screen is empty no template defined!

For Help, press F1

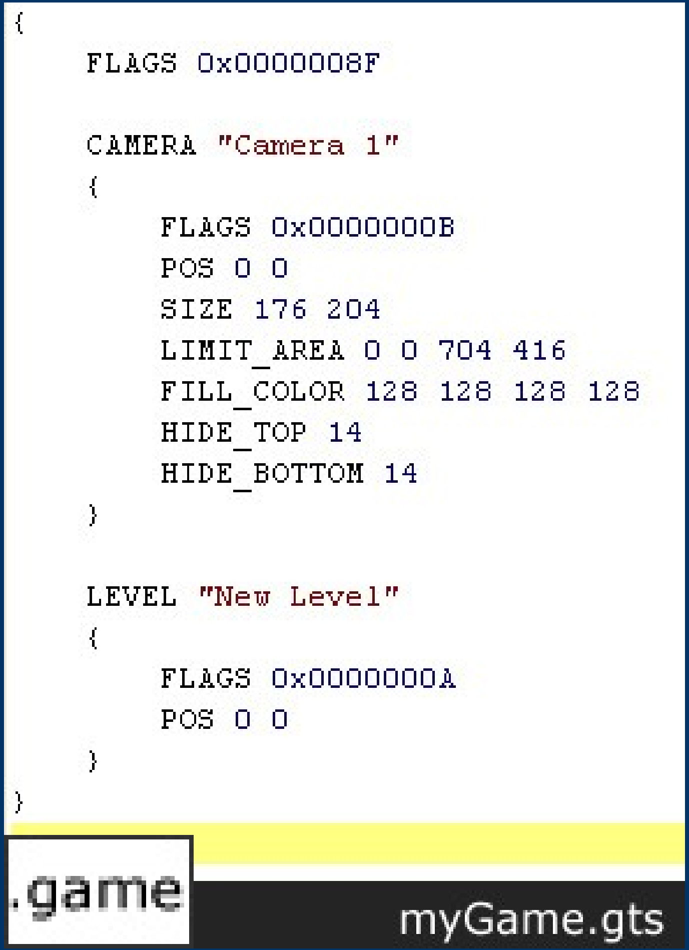
Creating a new game

AuroraGT

The templates file (.gts)

- This is the .game file saved by AuroraGT:

- In the previous screenshot there was no resources defined, so we couldn't insert objects or maps in the game!
- We need to specify a template to use.
- Let's go to the next slide...



```
{
    FLAGS 0x0000008F

    CAMERA "Camera 1"
    {
        FLAGS 0x0000000B
        POS 0 0
        SIZE 176 204
        LIMIT_AREA 0 0 704 416
        FILL_COLOR 128 128 128 128
        HIDE_TOP 14
        HIDE_BOTTOM 14
    }

    LEVEL "New Level"
    {
        FLAGS 0x0000000A
        POS 0 0
    }
}
```

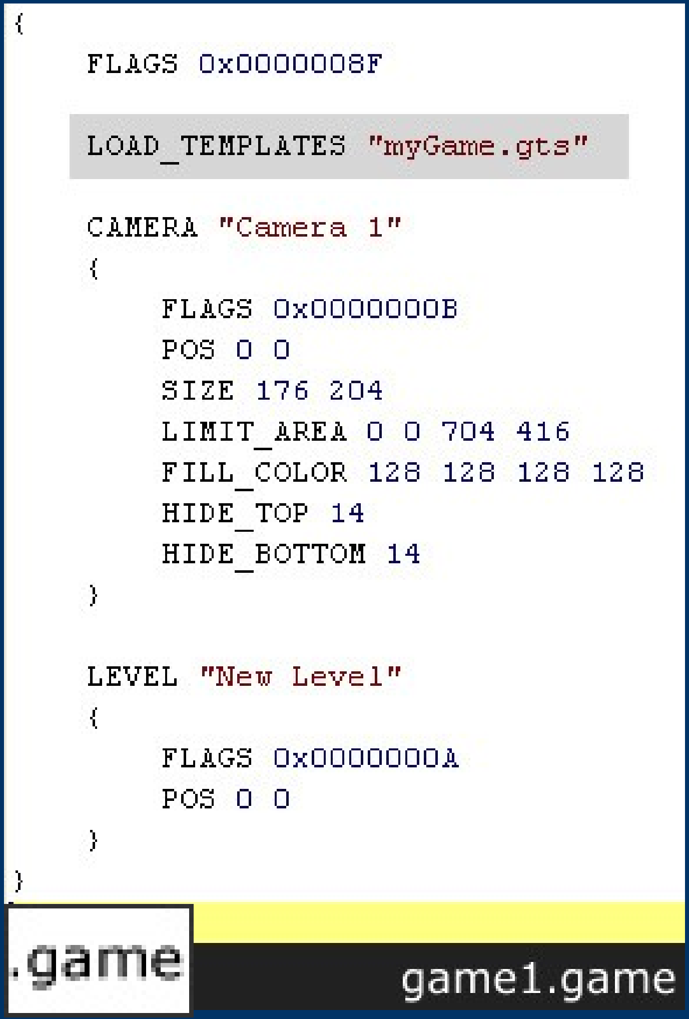
AuroraGT

The templates file (.gts)

Q: Where is defined this template?

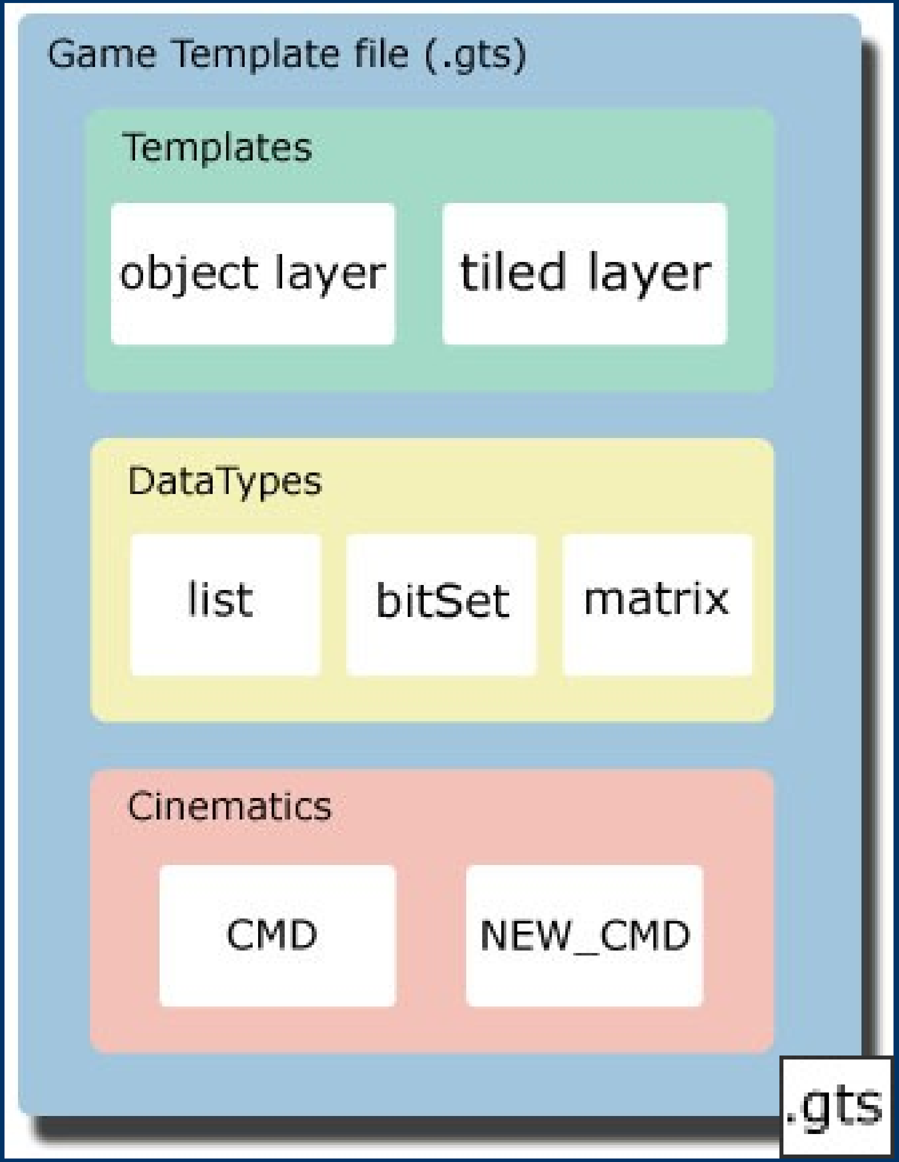
- To specify which template to use, we need to add inside our .game file the command:

LOAD_TEMPLATES “myGame.gts”



```
{
    FLAGS 0x0000008F

    LOAD_TEMPLATES "myGame.gts"

    CAMERA "Camera 1"
    {
        FLAGS 0x0000000B
        POS 0 0
        SIZE 176 204
        LIMIT_AREA 0 0 704 416
        FILL_COLOR 128 128 128 128
        HIDE_TOP 14
        HIDE_BOTTOM 14
    }

    LEVEL "New Level"
    {
        FLAGS 0x0000000A
        POS 0 0
    }
}
```

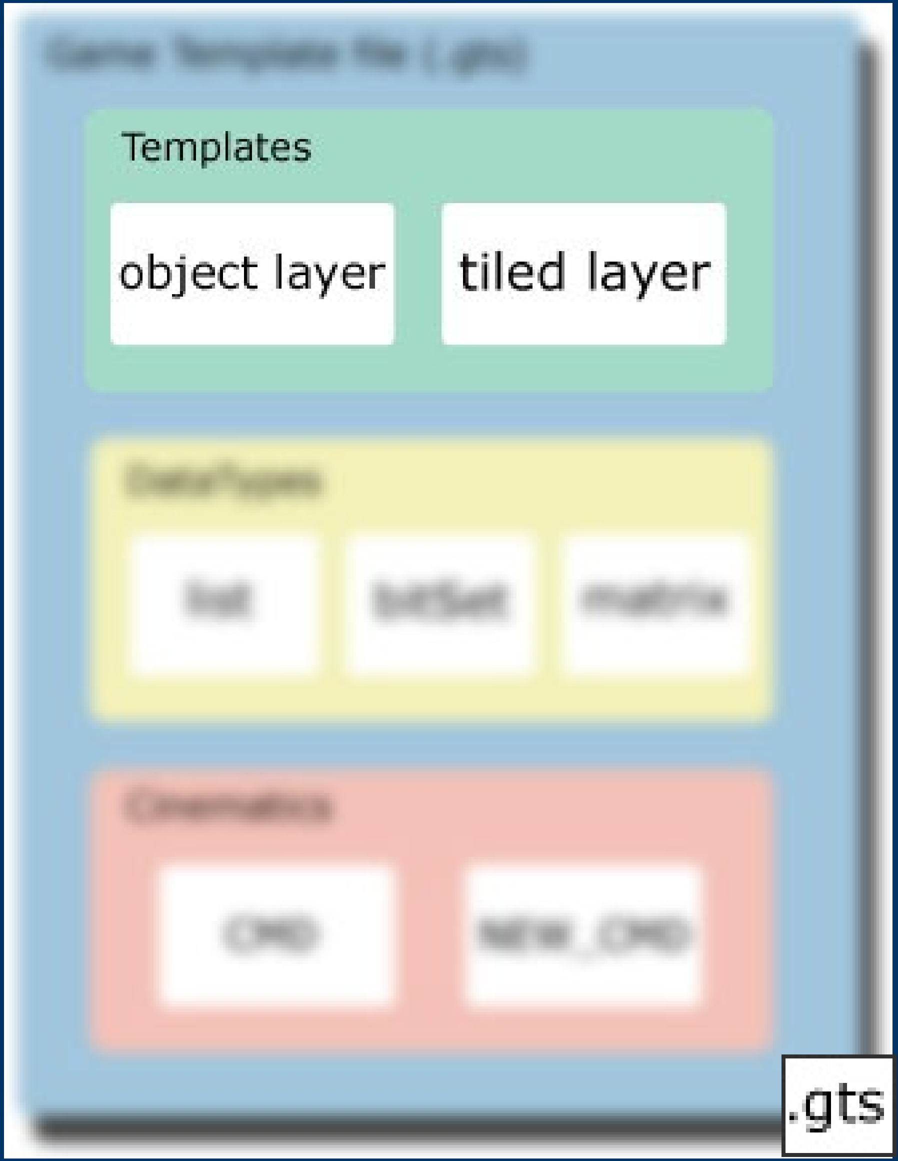
GTS
File Structure



Three main sections

- Templates: most important one. It's mandatory
- DataTypes: Optional, but makes the job easier when creating a game.
- Cinematics: Design of intros, outros, briefings, debriefings templates.

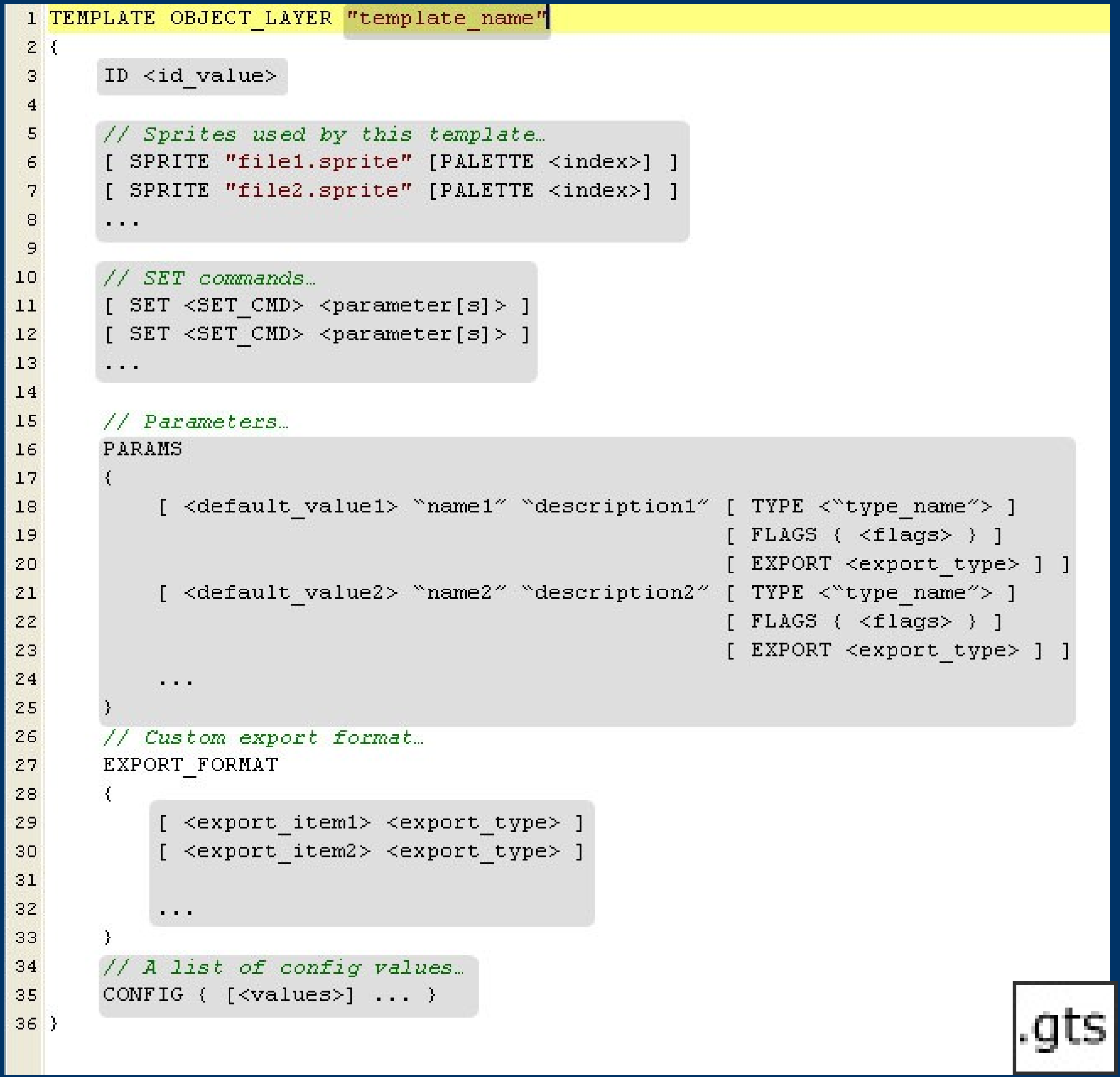
GTS
Templates



- Let's get into the first part, Templates.
- There are two main types:
  - OBJECT_LAYER
    Used for general objects
  - TILED_LAYER
    Used for map layers.

GTS

Templates: Object Layers



```
TEMPLATE OBJECT_LAYER "template_name"
{
    ID <id_value>

    // Sprites used by this template…
    [ SPRITE "file1.sprite" [PALETTE <index>] ]
    [ SPRITE "file2.sprite" [PALETTE <index>] ]
    ...

    // SET commands…
    [ SET <SET_CMD> <parameter[s]> ]
    [ SET <SET_CMD> <parameter[s]> ]
    ...

    // Parameters…
    PARAMS
    {
        [ <default_value1> “name1” “description1” [ TYPE <“type_name”> ]
                                                  [ FLAGS ( <flags> ) ]
                                                  [ EXPORT <export_type> ] ]
        [ <default_value2> “name2” “description2” [ TYPE <“type_name”> ]
                                                  [ FLAGS ( <flags> ) ]
                                                  [ EXPORT <export_type> ] ]
        ...
    }
    // Custom export format…
    EXPORT_FORMAT
    {
        [ <export_item1> <export_type> ]
        [ <export_item2> <export_type> ]

        ...
    }
    // A list of config values…
    CONFIG ( [<values>] ... )
}
```

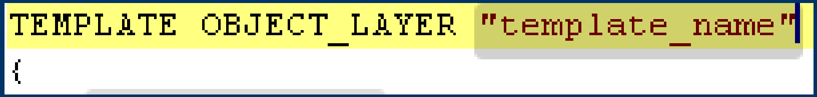
.gts

This is the main structure of an OBJECT_LAYER

Let's take a closer look of every block of code ...

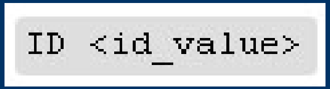
GTS

Templates: Object Layers



```
TEMPLATE OBJECT_LAYER "template_name"
{
```

- Here we define an Object Layer template called “template_name”

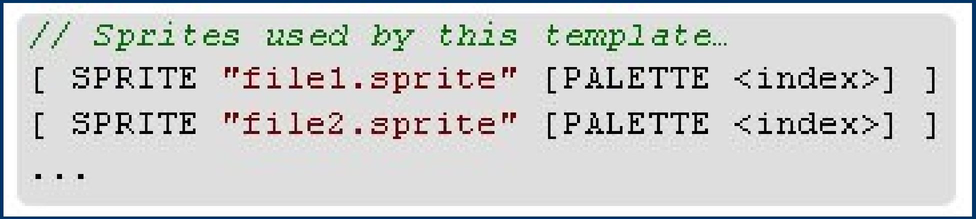


```
ID <id_value>
```

- A Unique ID needs to be addressed to every template, this is used by the code.

GTS

Templates: Object Layers

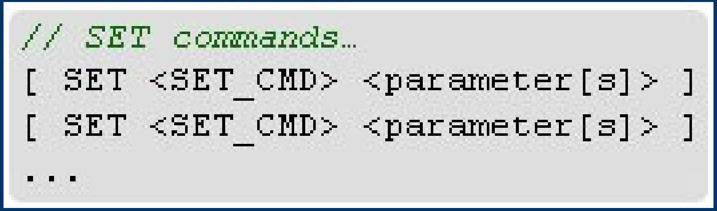


```
// Sprites used by this template…
[ SPRITE "file1.sprite" [PALETTE <index>] ]
[ SPRITE "file2.sprite" [PALETTE <index>] ]
...
```

- We can define zero or more AGT sprites (.sprite files). If there's more than one, we can select one by using commands and parameters explained ahead.
- A palette can be specified too. By default it takes palette zero.

GTS

Templates: Object Layers



```
// SET commands…
[ SET <SET_CMD> <parameter[s]> ]
[ SET <SET_CMD> <parameter[s]> ]
...
```

NOTE: Every CMD command is only reflected in AuroraGT in a graphical way, but this properties are not exported.

- CMD commands are used to show in AGT how the object will behave itself, but they don't change its behaviour.
- Usually these command values take the parameter values to reflect them in the AGT Graphical editor.
- After parameters are explained these concepts will become clearer.

GTS

.gts

Templates: Object Layers

Command	Default	Description
SET SPRITE <int_or_param>	0	sprite that Aurora will show in Preview Window
SET MODULE <int_or_param>	0	module (-1 to disable -> frame)
SET FRAME <int_or_param>	-1	frame (-1 to disable -> aframe)
SET FMODULE <int_or_param>	-1	fmodule (-1 to disable-> frame)
SET ANIM <int_or_param>	-1	anim (-1 to disable-> null)
SET AFRAME <int_or_param>	-1	aframe (-1 to disable -> anim)
SET FLAGS <hex_or_param> <hex>	0	paint flags, <hex> is used as a mask. Aurora flags
SET MM <int_or_param>	-1	module mapping
SET NX <int_or_param>	1	number of item on x
SET NY <int_or_param>	1	number of item on y
SET DX <int_or_param> [<int_or_param>]	0 [0]	space between items
SET DY <int_or_param> [<int_or_param>]	0 [0]	space between items
SET ANGLE <int_or_param>	0	NY items are rotated
SET RANGE_X x1 x2	disabled	draw a "patrol" zone horiz./vert.
SET RANGE_Y y1 y2	disabled	x1, x2, y1, y2 are <int_or_param>;

GTS

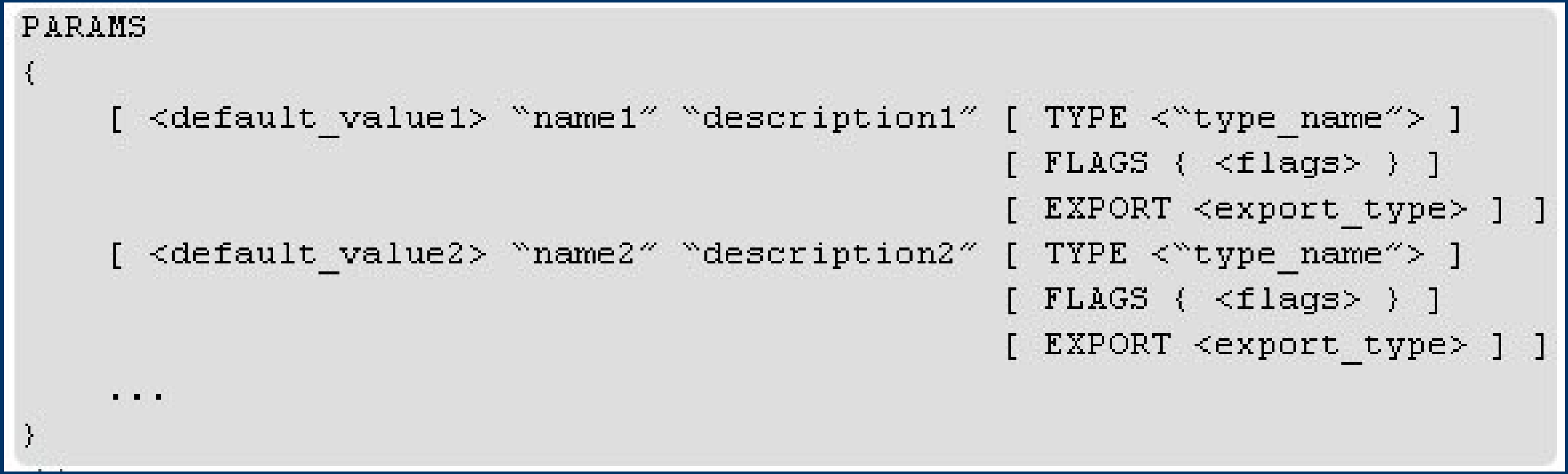
.gts

Templates: Object Layers

SET ANGLE <int_or_param>	0	NY items are rotated
SET RANGE_X x1 x2	disabled	draw a "patrol" zone horiz./vert.
SET RANGE_Y y1 y2	disabled	x1, x2, y1, y2 are <int_or_param>;
SET RECT_AREA ox oy w h RGB(r, g, b)	disabled	draw a relative rectangle area, in a specified color; ox, oy, w and h are <int_or_param>; r, g, b are integer values;
SET TRIANGLE_AREA xa ya xb yb xc yc RGB(r, g, b)	disabled	draw a relative triangle area, in a specified color; xa, ya, xb, yb, xc and yc are <param>; r, g, b are integer values;
SET GRID	disabled	draw a grid (uses NX, NY, DX, DY)
SET SNAP <param>	disabled	set the snap step using the <param> grid
SET PALETTE <int_or_param>	-1	set the current palette for the list of sprites attached to the object layer
SET ROTATE_ANGLE <int_or_param>	disabled	Performs a free rotate with the angle defined by this parameter for the selected object. You must set the sprite and frame number in the ObjectLayerTemplate.

GTS

Templates: Object Layers



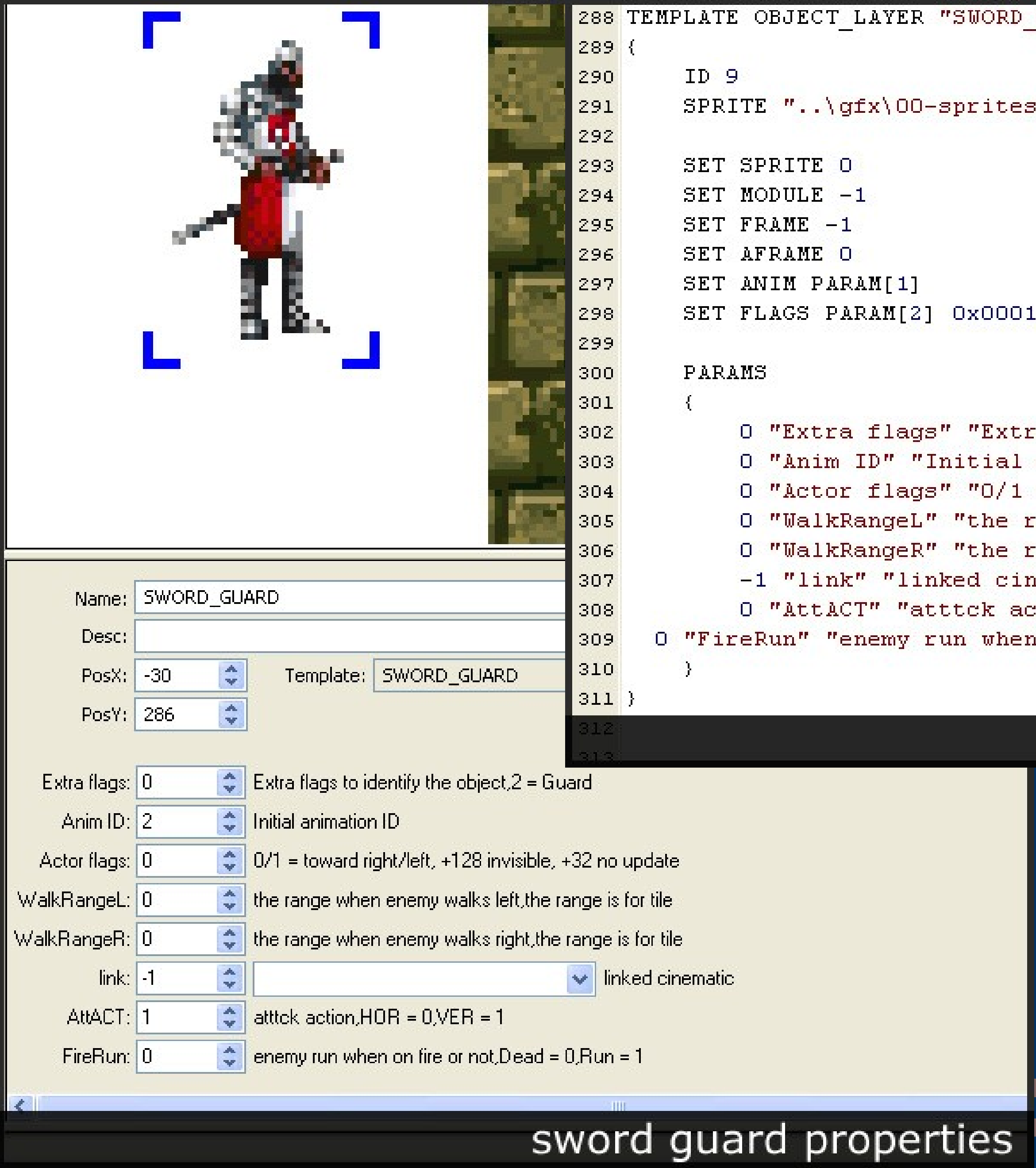
```
PARAMS
{
    [ <default_value1> "name1" "description1" [ TYPE <"type_name"> ]
                                              [ FLAGS { <flags> } ]
                                              [ EXPORT <export_type> ] ]
    [ <default_value2> "name2" "description2" [ TYPE <"type_name"> ]
                                              [ FLAGS { <flags> } ]
                                              [ EXPORT <export_type> ] ]

    ...
}
```

- Defined parameters are shown in the Object Properties Window, and the default values can be changed from there.

- Every parameter defined here is really exported into binaries and can be used by the code to perform different actions.

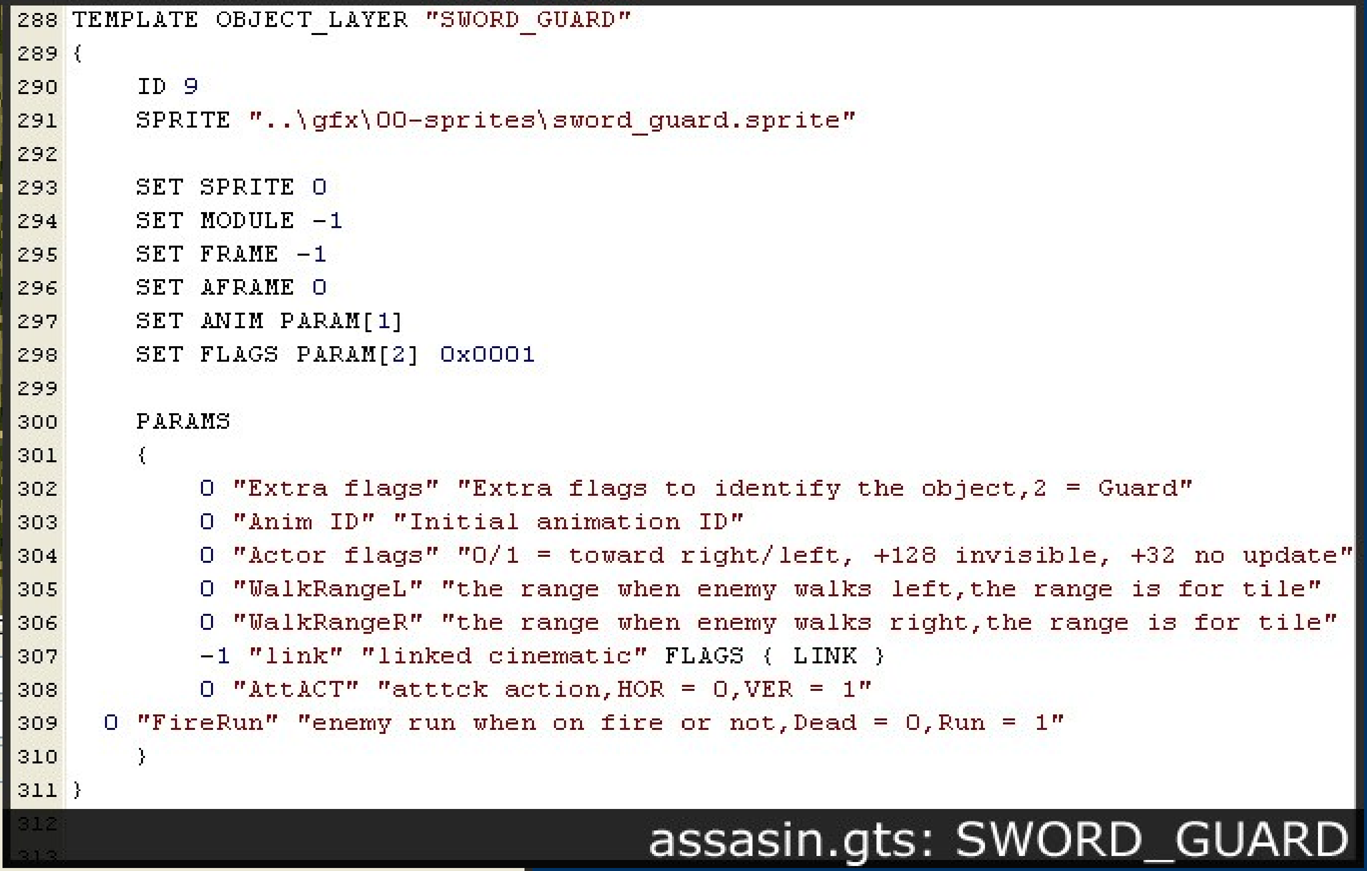
GTS

Templates: Object Layers Example



Name:	SWORD_GUARD	
Desc:		
PosX:	-30	Template: SWORD_GUARD
PosY:	286	
Extra flags:	0	Extra flags to identify the object,2 = Guard
Anim ID:	2	Initial animation ID
Actor flags:	0	0/1 = toward right/left, +128 invisible, +32 no update
WalkRangeL:	0	the range when enemy walks left,the range is for tile
WalkRangeR:	0	the range when enemy walks right,the range is for tile
link:	-1	linked cinematic
AttACT:	1	atttck action,HOR = 0,VER = 1
FireRun:	0	enemy run when on fire or not,Dead = 0,Run = 1

sword guard properties

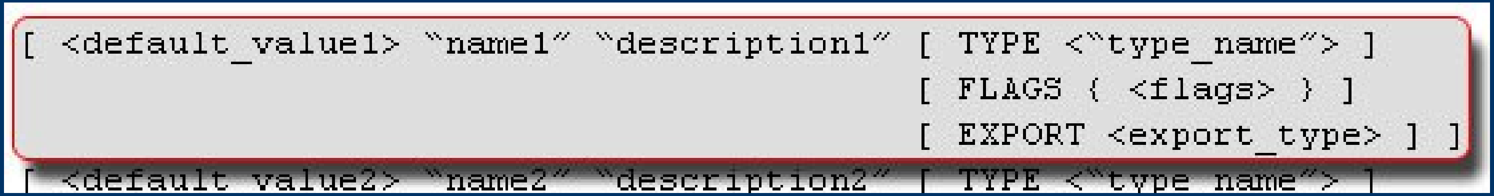


```
TEMPLATE OBJECT_LAYER "SWORD_GUARD"
{
    ID 9
    SPRITE "..\gfx\00-sprites\sword_guard.sprite"

    SET SPRITE 0
    SET MODULE -1
    SET FRAME -1
    SET AFRAME 0
    SET ANIM PARAM[1]
    SET FLAGS PARAM[2] 0x0001

    PARAMS
    {
        0 "Extra flags" "Extra flags to identify the object,2 = Guard"
        0 "Anim ID" "Initial animation ID"
        0 "Actor flags" "0/1 = toward right/left, +128 invisible, +32 no update"
        0 "WalkRangeL" "the range when enemy walks left,the range is for tile"
        0 "WalkRangeR" "the range when enemy walks right,the range is for tile"
        -1 "link" "linked cinematic" FLAGS { LINK }
        0 "AttACT" "atttck action,HOR = 0,VER = 1"
  0 "FireRun" "enemy run when on fire or not,Dead = 0,Run = 1"
    }
}
```

assasin.gts: SWORD_GUARD

GTS

Templates: Object Layers



```
[ <default_value1> “name1” “description1” [ TYPE <“type_name”> ]
                                          [ FLAGS { <flags> } ]
                                          [ EXPORT <export_type> ] ]
[ <default_value2> “name2” “description2” [ TYPE <“type_name”> ]
```

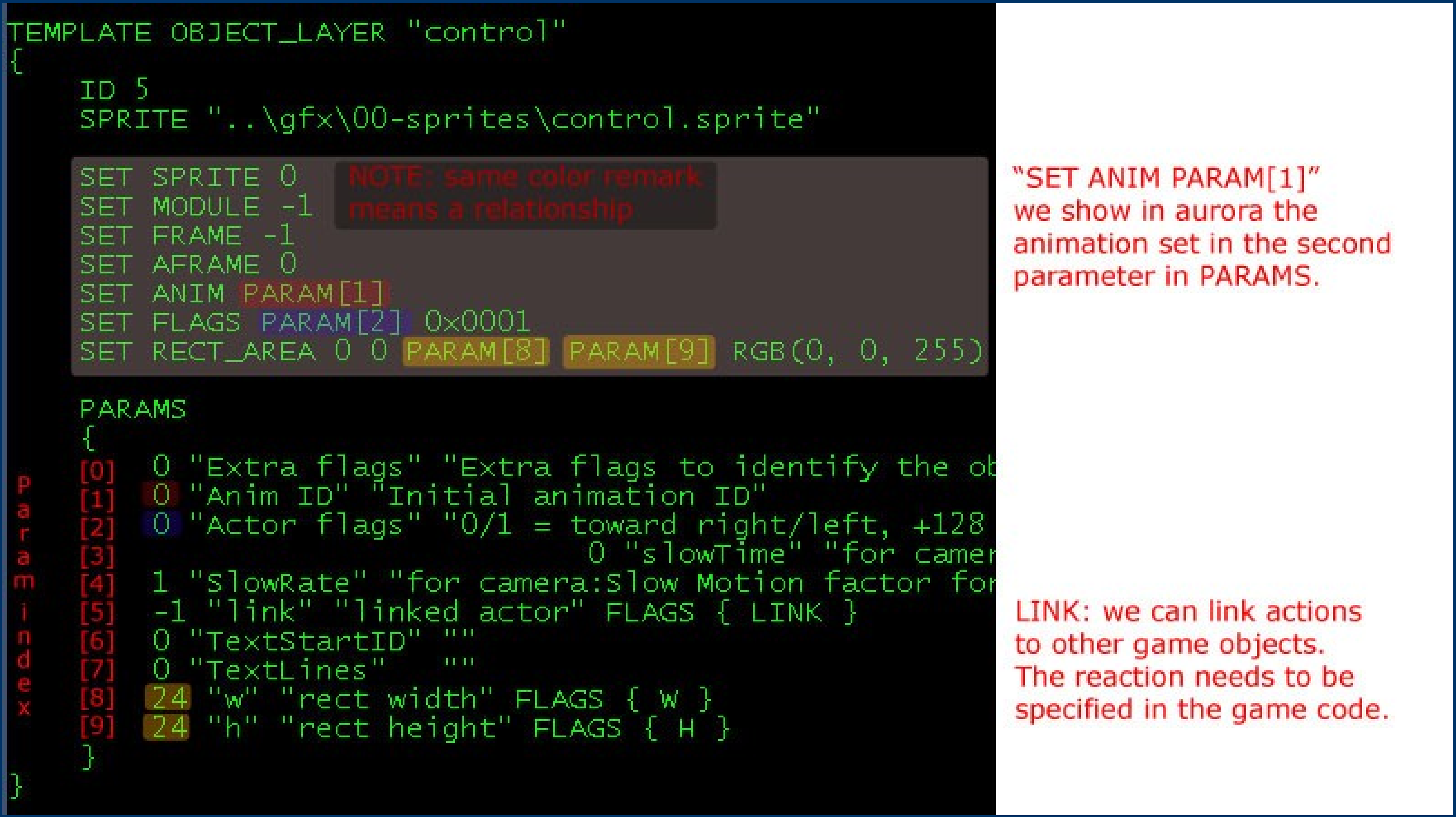
- <default_value1> := integer value, preceded by x-, x+, y- or y+ if the default value for that parameter is relative to position of the object
- <flags> := { X, Y, W, H, LINK } -> to specify what type of param is that. X, Y, W and H are used when we need to scale a level. LINK specifies that the current parameter is an ID link to another object.
- <“type_name”> := the name of a type defined in a “DATA_TYPE” block. Please see DATA_TYPE specifications.
- <export_type> := one of INT8, UINT8, INT16, UINT16, INT32, UINT32 -> used to specify export format for each parameter, if the PARAMS export type is set to CUSTOM (see below)

GTS

.gts

Templates: Object Layers

Object Layer example (screenshot of a .gts file)



```
TEMPLATE OBJECT_LAYER "control"
{
    ID 5
    SPRITE "..\gfx\00-sprites\control.sprite"

    SET SPRITE 0      NOTE: same color remark
    SET MODULE -1     means a relationship
    SET FRAME -1
    SET AFRAME 0
    SET ANIM PARAM[1]
    SET FLAGS PARAM[2] 0x0001
    SET RECT_AREA 0 0 PARAM[8] PARAM[9] RGB(0, 0, 255)

    PARAMS
    {
P   [0]  0 "Extra flags" "Extra flags to identify the ob
a   [1]  0 "Anim ID" "Initial animation ID"
r   [2]  0 "Actor flags" "0/1 = toward right/left, +128
a   [3]                         0 "slowTime" "for camer
m   [4]  1 "SlowRate" "for camera:Slow Motion factor for
i   [5]  -1 "link" "linked actor" FLAGS { LINK }
n   [6]  0 "TextStartID" ""
d   [7]  0 "TextLines"   ""
e   [8]  24 "w" "rect width" FLAGS { W }
x   [9]  24 "h" "rect height" FLAGS { H }
    }
}
```

"SET ANIM PARAM[1]" we show in aurora the animation set in the second parameter in PARAMS.

LINK: we can link actions to other game objects. The reaction needs to be specified in the game code.

GTS

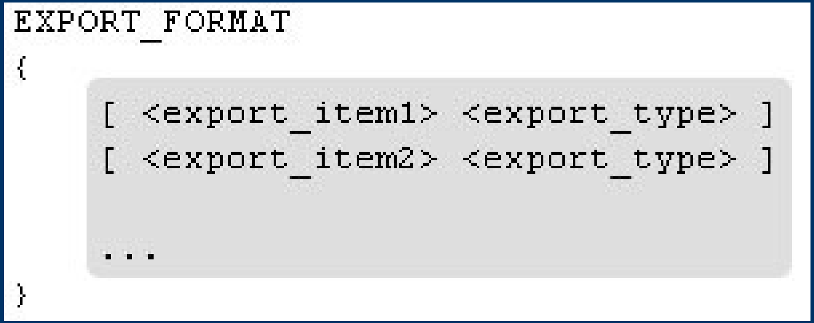
Templates: Object Layers

- See an example of .game and .gts file.

.gts

GTS

Templates: Object Layers



```
EXPORT_FORMAT
{
    [ <export_item1> <export_type> ]
    [ <export_item2> <export_type> ]

    ...
}
```

- <export_item1> := one of the template members: TEMPLATE_ID, LAYER_ID, LAYER_POS_XY, NUM_PARAMS, PARAMS, NUM_POINTS, POINTS_XY, POINTS_PARAM

- <export_type> := one of CUSTOM, INT8, UINT8, INT16, UINT16, INT32, UINT32

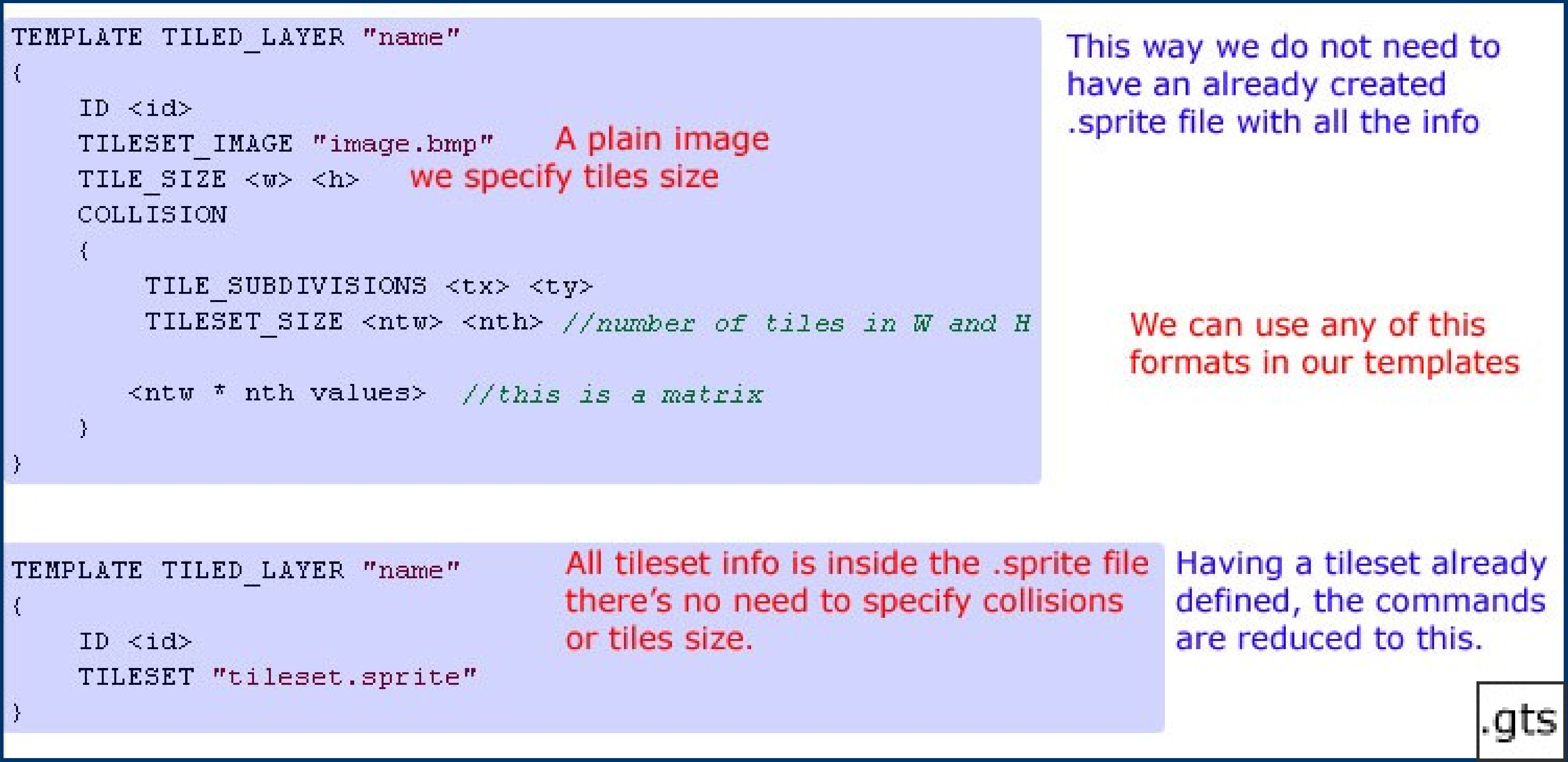
For this section to be used by Aurora, USE_TEMPLATE_EXPORT_FORMAT must be defined in the OBJ_LAYERS section of the .gamecmd file. Template members that are not specified in this section are exported with the defalult Params. If a template member should NOT be exported at all an empty EXPORT_FORMAT should be specified.

```
EXPORT_FORMAT
{
}
```

GTS

Templates: Tiled Layers

- There are two ways of defining this layers:



```
TEMPLATE TILED_LAYER "name"
{
    ID <id>
    TILESET_IMAGE "image.bmp"
    TILE_SIZE <w> <h>
    COLLISION
    {
        TILE_SUBDIVISIONS <tx> <ty>
        TILESET_SIZE <ntw> <nth> //number of tiles in W and H

       <ntw * nth values>  //this is a matrix
    }
}
```

A plain image

we specify tiles size

This way we do not need to have an already created .sprite file with all the info

We can use any of this formats in our templates

```
TEMPLATE TILED_LAYER "name"
{
    ID <id>
    TILESET "tileset.sprite"
}
```

All tileset info is inside the .sprite file there's no need to specify collisions or tiles size.

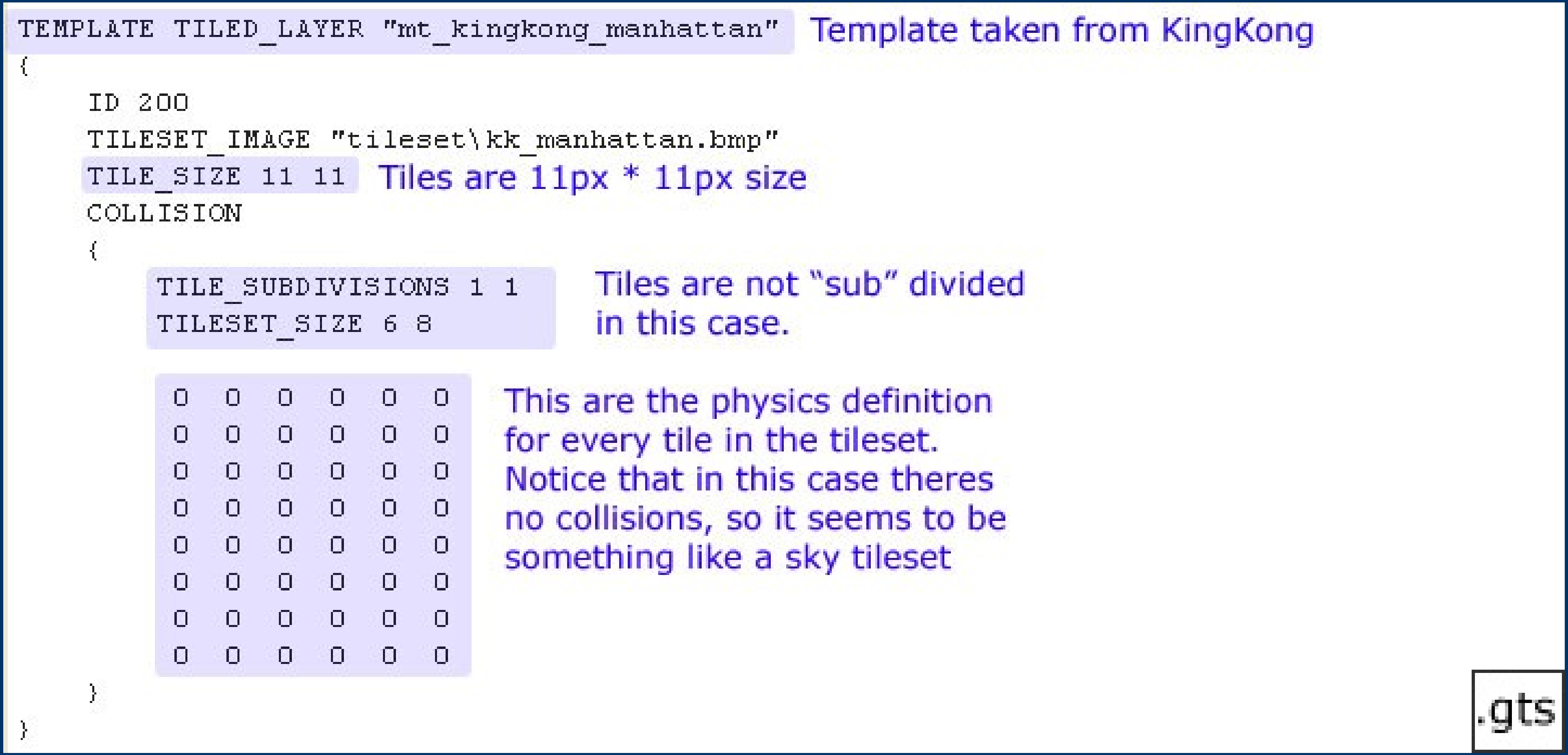
Having a tileset already defined, the commands are reduced to this.

.gts

GTS

Templates: Tiled Layers

- Using an image directly:



```
TEMPLATE TILED_LAYER "mt_kingkong_manhattan"
{
    ID 200
    TILESET_IMAGE "tileset\kk_manhattan.bmp"
    TILE_SIZE 11 11
    COLLISION
    {
        TILE_SUBDIVISIONS 1 1
        TILESET_SIZE 6 8

         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
         0   0   0   0   0   0
    }
}
```

Template taken from KingKong

Tiles are 11px * 11px size

Tiles are not "sub" divided in this case.

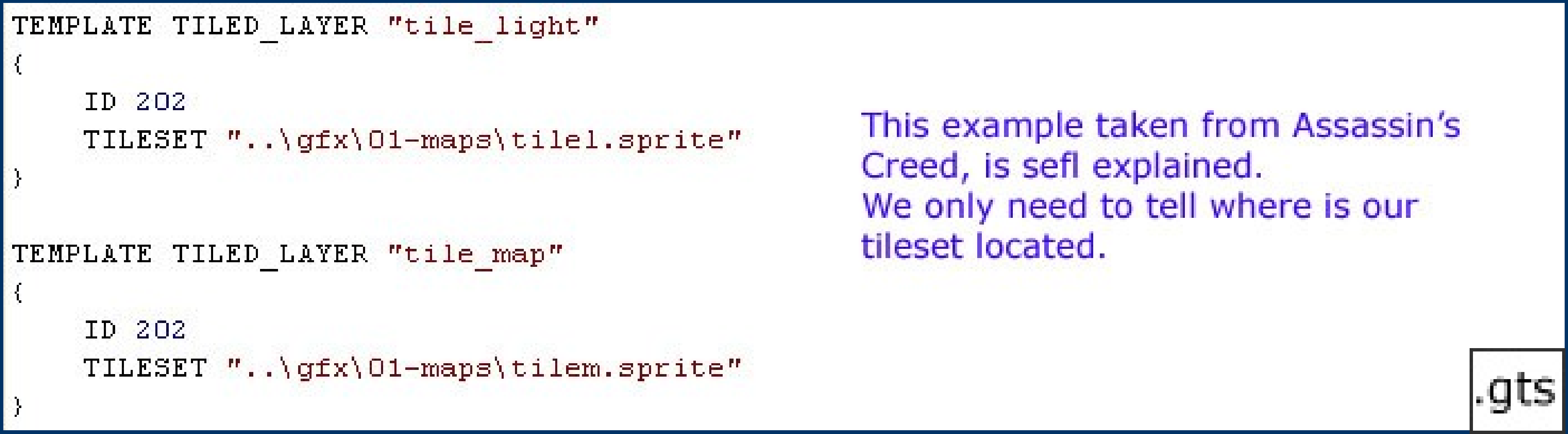
This are the physics definition for every tile in the tileset. Notice that in this case theres no collisions, so it seems to be something like a sky tileset

.gts

GTS

Templates: Tiled Layers

- Using an already-made tileset (.sprite file)

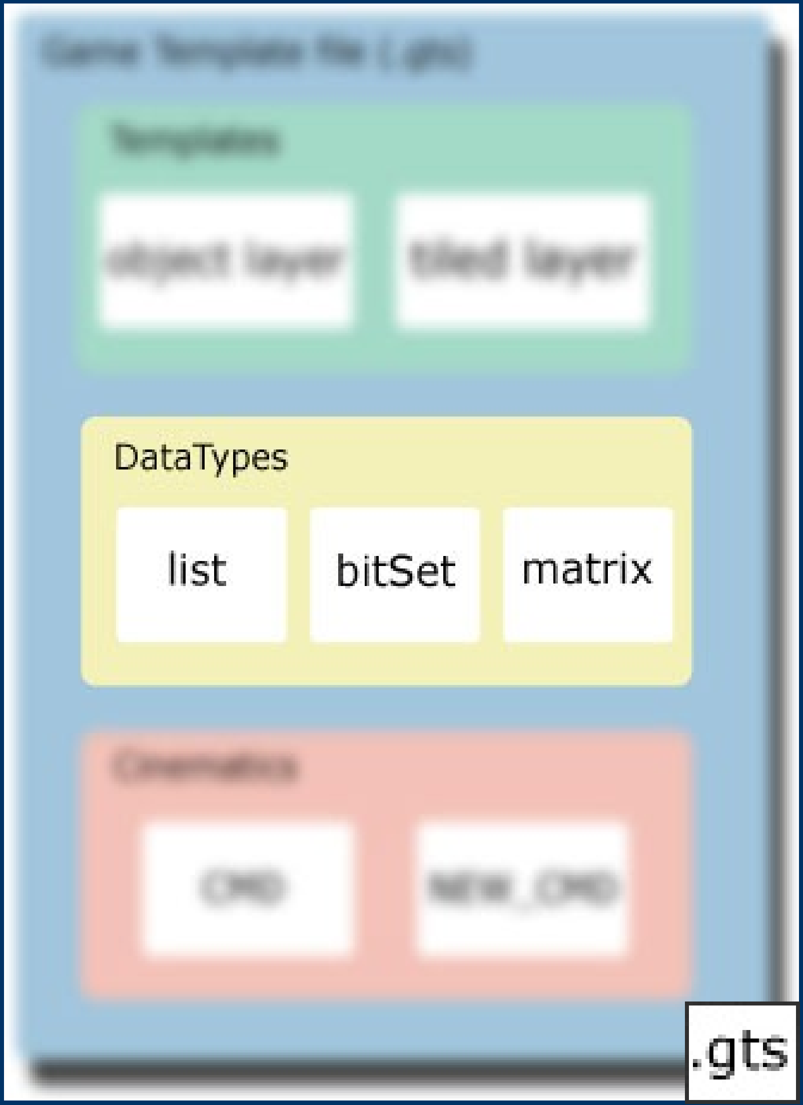


```
TEMPLATE TILED_LAYER "tile_light"
{
    ID 202
    TILESET "..\gfx\01-maps\tilel.sprite"
}

TEMPLATE TILED_LAYER "tile_map"
{
    ID 202
    TILESET "..\gfx\01-maps\tilem.sprite"
}
```

This example taken from Assassin's Creed, is sefl explained.
We only need to tell where is our tileset located.

.gts

GTS
DataTypes

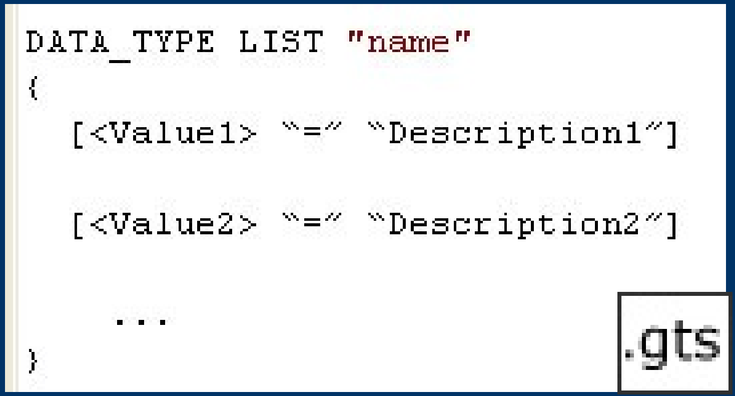


- Three kind of dataTypes:
  - List: Used to show a comboBox in Ojects properties
  - BitSet:
  - Matrix:

GTS

DataTypes: List

- The structure is very simple
- We could say its like a comboBox where:
  - Value: is the index
  - Description: Is the text shown as a description of the option selected.

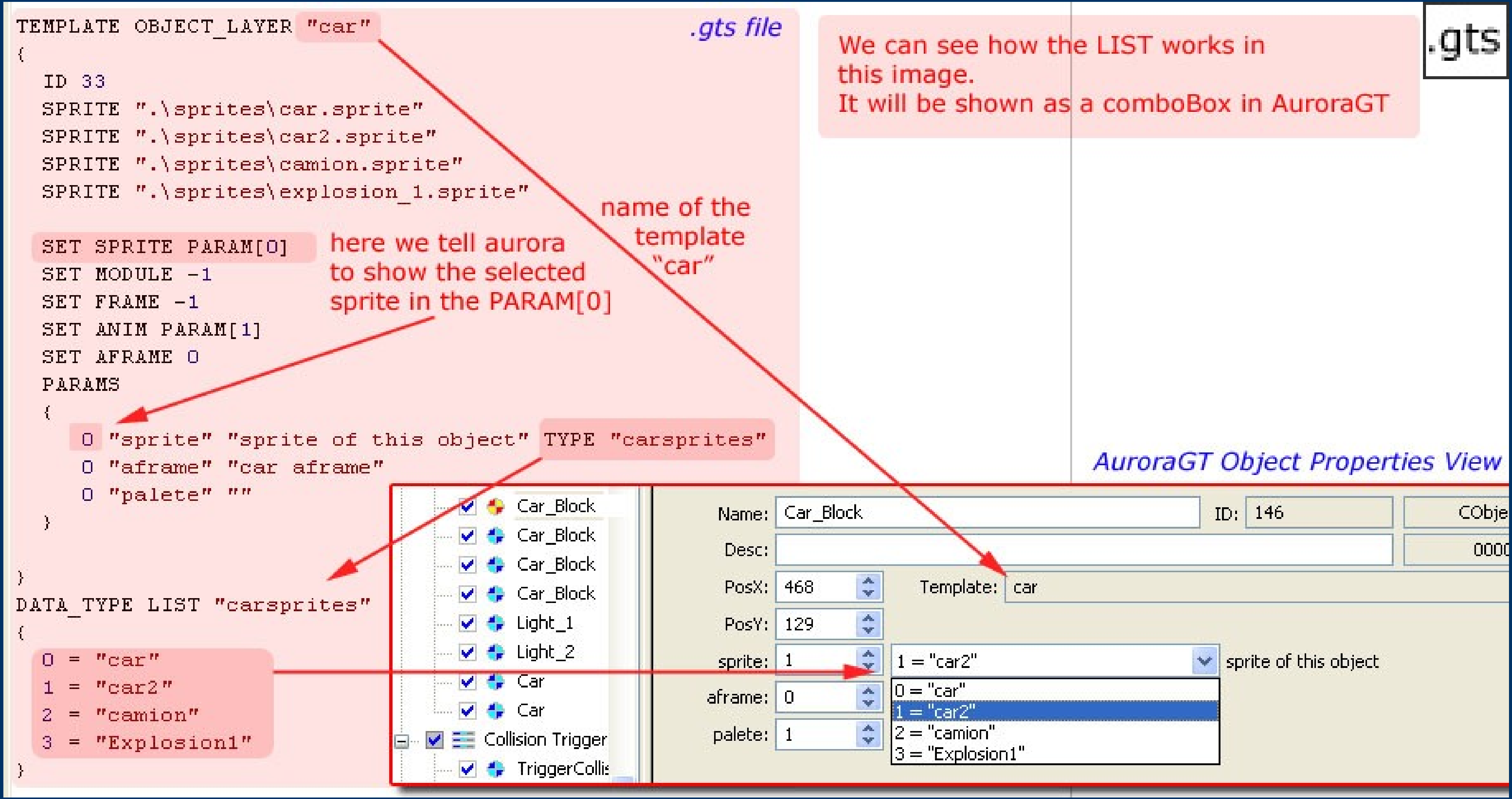


```
DATA_TYPE LIST "name"
{
  [<Value1> "=" "Description1"]

  [<Value2> "=" "Description2"]

     ...
}
```

.gts

GTS
DataTypes: List

- Here is an example of a list taken from Heroes



.gts file

```
TEMPLATE OBJECT_LAYER "car"
{
  ID 33
  SPRITE ".\sprites\car.sprite"
  SPRITE ".\sprites\car2.sprite"
  SPRITE ".\sprites\camion.sprite"
  SPRITE ".\sprites\explosion_1.sprite"

  SET SPRITE PARAM[0]
  SET MODULE -1
  SET FRAME -1
  SET ANIM PARAM[1]
  SET AFRAME 0
  PARAMS
  {
    0 "sprite" "sprite of this object" TYPE "carsprites"
    0 "aframe" "car aframe"
    0 "palete" ""
  }

}
DATA_TYPE LIST "carsprites"
{
  0 = "car"
  1 = "car2"
  2 = "camion"
  3 = "Explosion1"
}
```

here we tell aurora to show the selected sprite in the PARAM[0]

name of the template "car"

.gts

We can see how the LIST works in this image.
It will be shown as a comboBox in AuroraGT

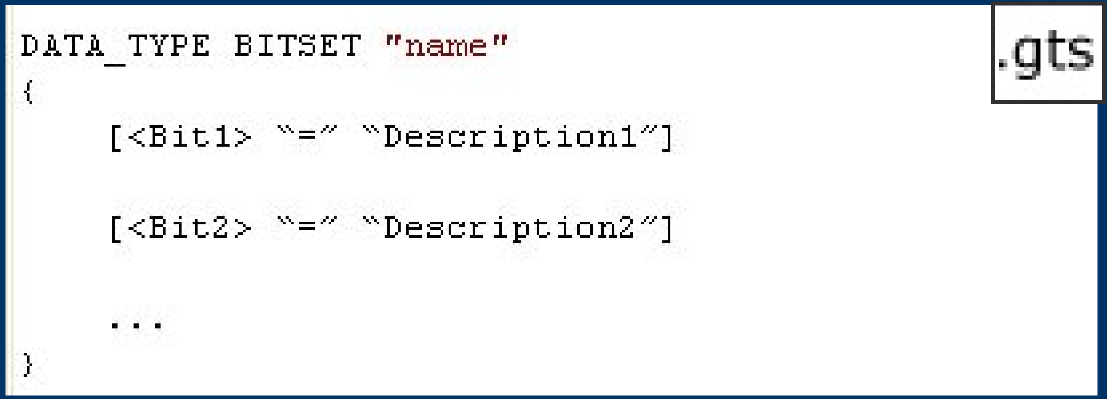
AuroraGT Object Properties View

Name: Car_Block
ID: 146
Desc:
PosX: 468
Template: car
PosY: 129
sprite: 1 — 1 = "car2" — sprite of this object
aframe: 0
palete: 1

0 = "car"
1 = "car2"
2 = "camion"
3 = "Explosion1"

GTS

DataTypes: BitSet



```
DATA_TYPE BITSET "name"
{
    [<Bit1> "=" "Description1"]

    [<Bit2> "=" "Description2"]

    ...
}
```

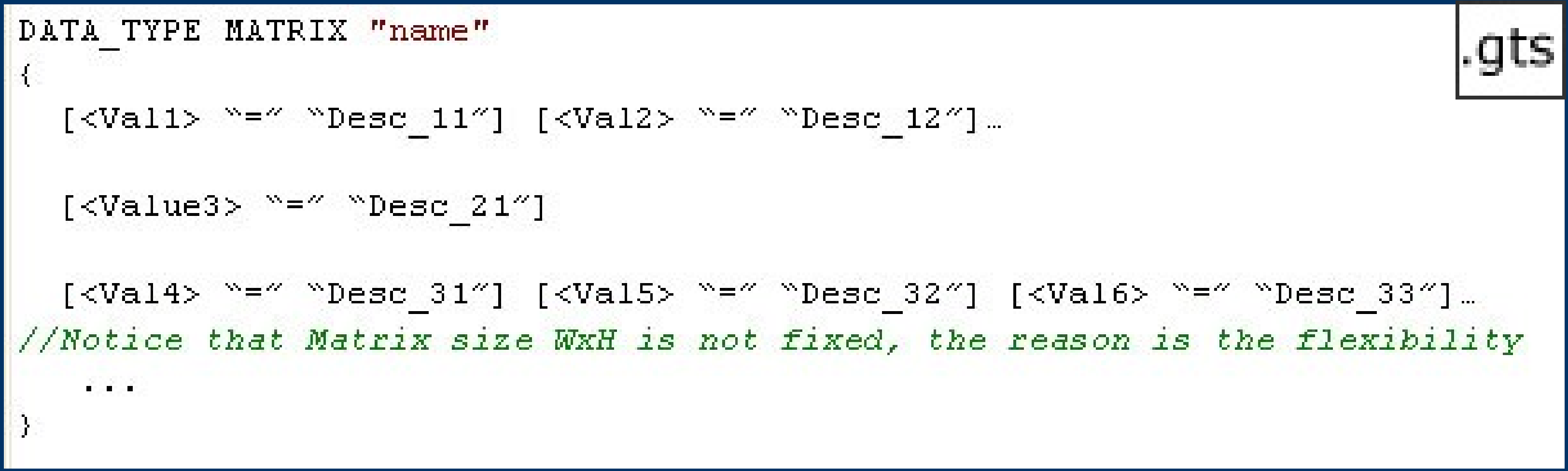
.gts

- BITSET: Used to change certain bits

- Each object can have some flags set. Sometimes is very difficult for the users to set a hexadecimal value for that flag.
- In projects, the flags have a meaning, and now, that meaning can be defined in the Game Editor. A new DATA_TYPE “BITSET” was defined in the GTS, where all possible values of the flags can be added.
- For each template that need those flags, they can be added using a dialog that contains checkboxes.

GTS

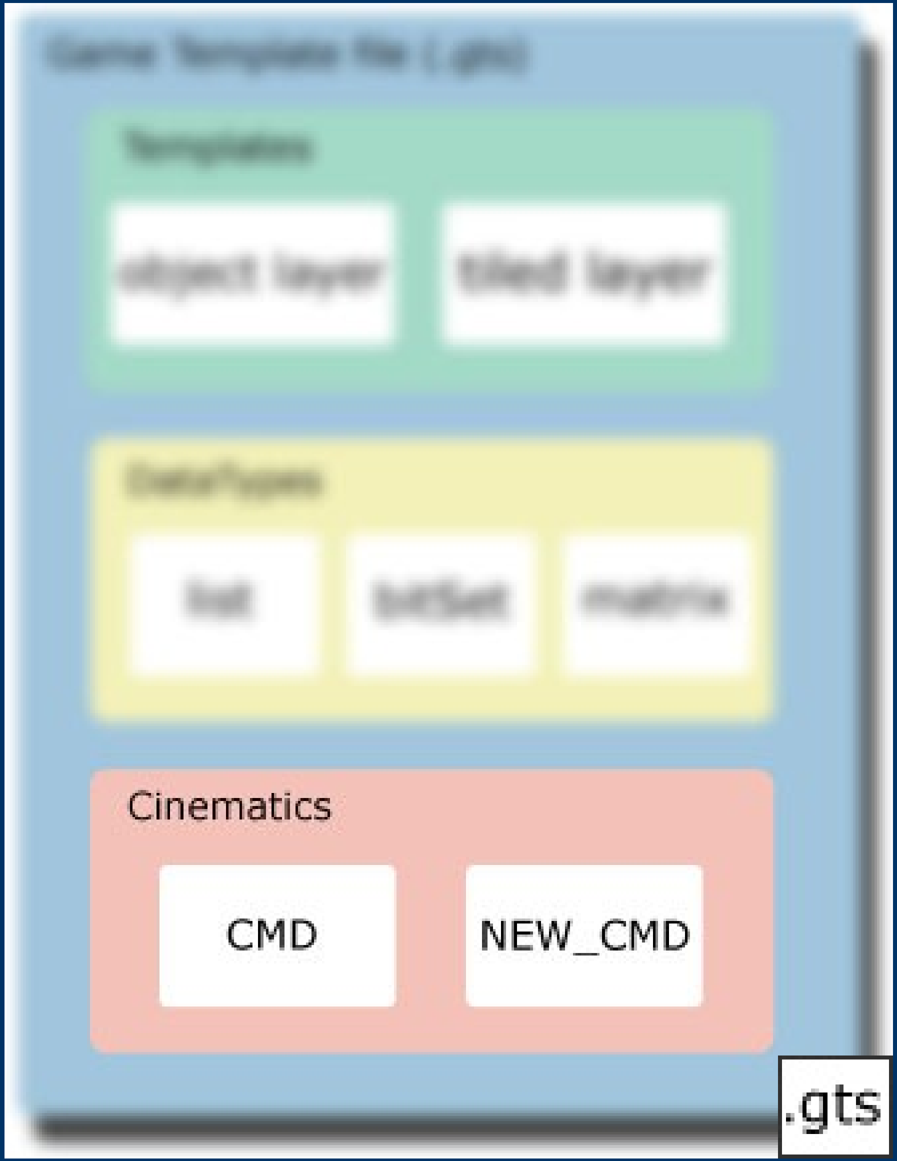
DataTypes: Matrix



```
DATA_TYPE MATRIX "name"
{
  [<Val1> "=" "Desc_11"] [<Val2> "=" "Desc_12"]…

  [<Value3> "=" "Desc_21"]

  [<Val4> "=" "Desc_31"] [<Val5> "=" "Desc_32"] [<Val6> "=" "Desc_33"]…
//Notice that Matrix size WxH is not fixed, the reason is the flexibility
   ...
}
```

- MATRIX:
- One of the latest features added according to the june_aurora_newsletter.
- The size of each row is not fixed.
- The value set for a parameter can be changed from the GameEditor. An edit box will appear with all matrix values, and the user can choose other value.

GTS
Cinematics

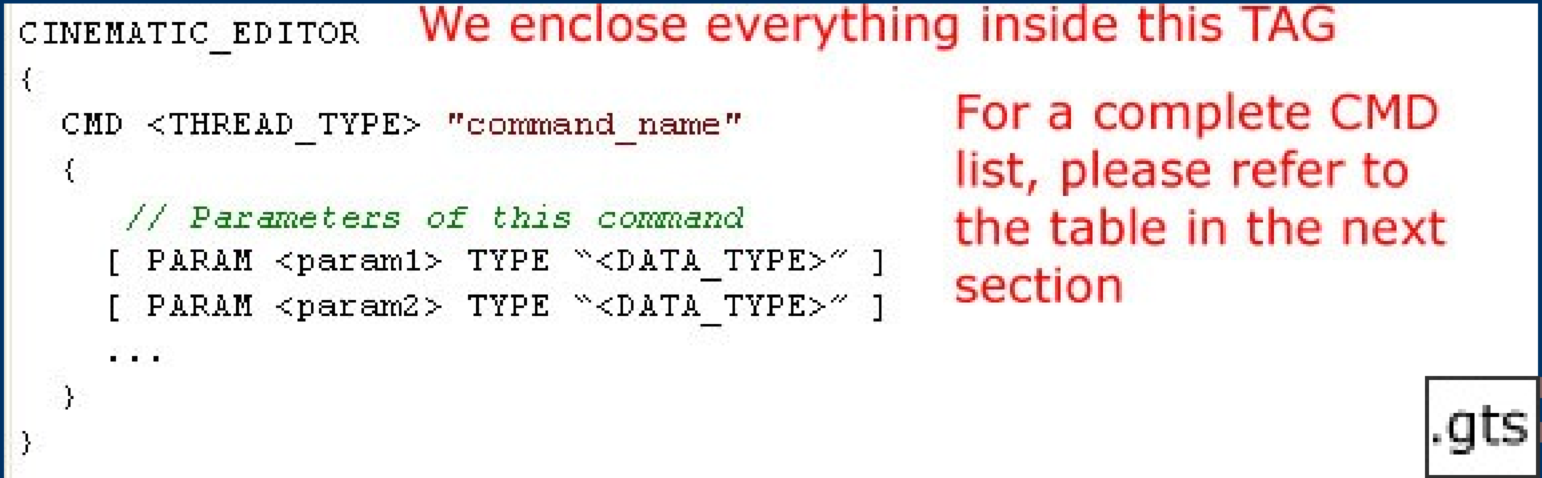


- Contains information regarding the cinematic editor.
- For now there are only two types of script blocks that can be defined CMD and NEW_CMD.
- This Cinematics section is the successor of tasks, much easier since its edition is graphical with keyframes in AuroraGT.

GTS

Cinematics: CMD

- It can have more than one CMD command.
- Each CMD command has parameters and its type.
- PARAM TYPE is one DATA_TYPE.
- CMD for defining TYPEs (comboBox) only.
- For new commands: NEW_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

GTS

Cinematics: CMD

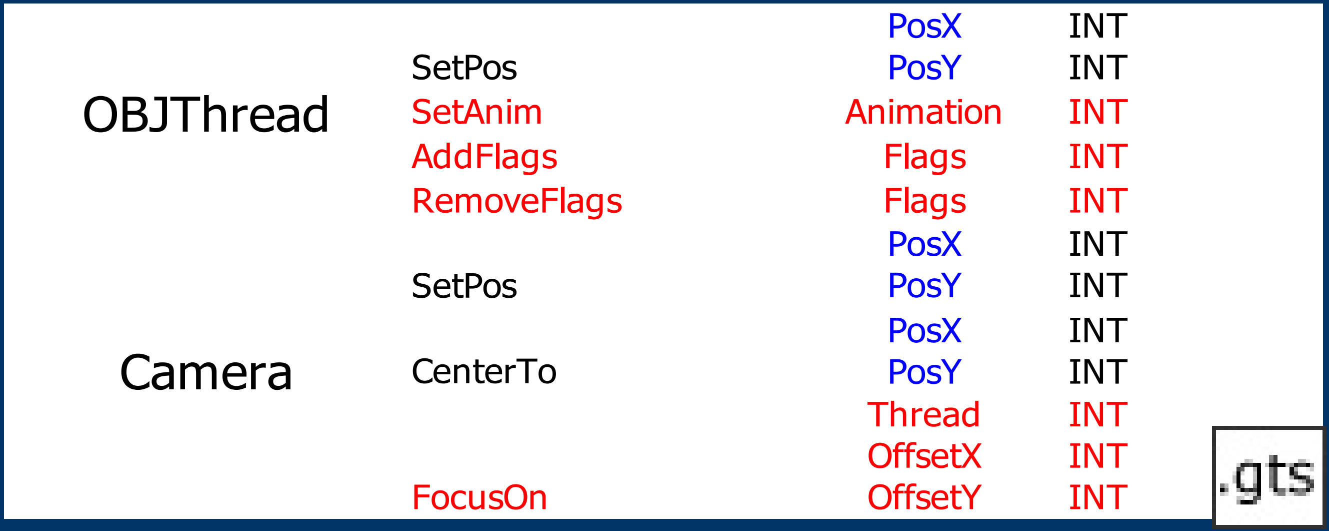
Thread Type	COMMAND	Default	TYPE
Basic	SetPos	PosX	INT
		PosY	INT
	SetAction	Action	INT
	SendObjEvent	ObjectID	LayerLink
		Param	INT
	SendObjEvent2	ObjectID	LayerLink
		Param1	INT
		Param2	INT
	SendObjEvent3	ObjectID	LayerLink
		Param1	INT
		Param2	INT
		Param3	INT
	SendEvent	Param	INT
	SendEvent2	Param1	INT
		Param2	INT
	SendEvent3	Param1	INT
		Param2	INT
		Param3	INT
SpirteInstance	SetPos	PosX	INT
		PosY	INT
	SetAnim	Animation	INT
	AddFlags	Flags	INT
	RemoveFlags	Flags	INT

.gts

We cannot change parameters for commands in RED

GTS

Cinematics: CMD



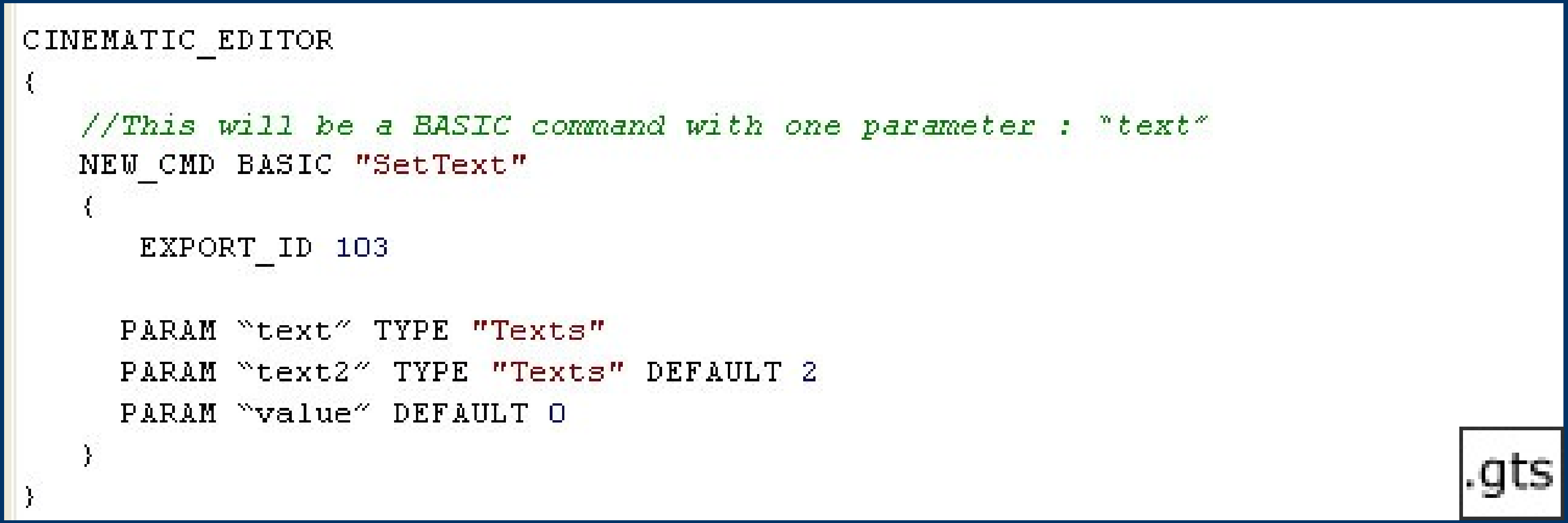
		PosX	INT
	SetPos	PosY	INT
OBJThread	SetAnim	Animation	INT
	AddFlags	Flags	INT
	RemoveFlags	Flags	INT
		PosX	INT
	SetPos	PosY	INT
		PosX	INT
Camera	CenterTo	PosY	INT
		Thread	INT
		OffsetX	INT
	FocusOn	OffsetY	INT

- We cannot change parameters for Commands in RED

GTS

Cinematics: NEW_CMD

- Custom created commands. We define the number of params, EXPORT_ID and the param's types.
- For more examples take a look at “Shrek the third” .gts file.



```
CINEMATIC_EDITOR
{
   //This will be a BASIC command with one parameter : "text"
   NEW_CMD BASIC "SetText"
   {
      EXPORT_ID 103

     PARAM "text" TYPE "Texts"
     PARAM "text2" TYPE "Texts" DEFAULT 2
     PARAM "value" DEFAULT 0
   }
}
```

GTS
Conclusion

- Conclusion:
  - GTS is a list of resources for levels
  - We need a .gts file to create levels.
  - Is a hand made file
  - Connection between designers and developers
  - Maintained by developers
  - Enable designers to create levels with no programming skills.

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

- **Unicode**

http://www.unicode.org/standard/principles.html#What_Characters

AuroraGT
Contact us

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