orxOBJECT structure

Summary

```
[ObjectTemplate]
Graphic            = GraphicTemplate
AnimationSet       = AnimationSetTemplate
AnimationFrequency = <float>
Body               = BodyTemplate
Group              = <string>
Clock              = ClockTemplate
LifeTime           = anim | child | fx | sound | spawn | track | <float>
Age                = <float>
Color              = <vector>
RGB                = <vector>
HSL                = <vector>
HSV                = <vector>
Alpha              = <float>
AutoScroll         = x | y | both
Flip               = x | y | both
DepthScale         = <bool>
Position           = ((center (truncate | round)) (left | right) (top |
bottom) | <vector>
SphericalPosition  = <vector>
Speed              = <vector>
Pivot              = (center (truncate | round)) (left | right) (top |
bottom) | <vector>
Size               = <vector>
UseRelativeSpeed   = <bool>
Rotation           = <float>
AngularVelocity    = <float>
Scale              = <vector>
Smoothing          = <bool>
BlendMode          = alpha | multiply | add | premul | none
Repeat             = <vector>
FXList             = FXTemplate1 # ... # FXTemplateN
FXRecursiveList    = <bool>
FXFrequency        = <float>
SoundList          = SoundTemplate1 # ... # SoundTemplateN
ShaderList         = ShaderTemplate1 # ... # ShaderTemplateN
TrackList          = TimeLineTrackTemplate1 # ... # TimeLineTrackTemplateN
Spawner            = SpawnerTemplate
ChildList          = ObjectTemplate1 # ... # ObjectTemplateN
ChildJointList     = JointTemplate1 # ... # JointTemplateN
ParentCamera       = CameraTemplate
UseParentSpace     = true | false | both | position | scale | none
IgnoreFromParent   = none | all | rotation | scale<.xyz>
OnPrepare          = <command>
OnCreate           = <command>
```

```
OnDelete              = <command>
```

Details

Here's a list of the available properties for an `orxOBJECT` structure:

- `Group`: GroupId from the GroupList for the object to be rendered into (like a layer). Defaults to `default` group.
- `Clock`: Associates the corresponding clock and creates it if needed. Associated clocks are useful to achieve time localized stretching (only for the corresponding object). When an object is time strectched by an associated clock, all its properties will behave accordingly (spawners, FXs, physics and sound pitch). If no clock is specified the main clock (core) will be used for update calculations.
- `LifeTime`: Multiple literals can be defined at once, in which case the object will be deleted when all of their associated conditions are fulfilled. If the value is numeric, the object will be deleted after that duration (in seconds). If not defined or negative value, infinite life is granted. 0 means it'll be instantly deleted. It's default value is -1.0, which means the object has an infinite life time. `child` keeps the object alive as long as there are owned children; `fx` keeps the object alive as long as there are FXs; `sound` keeps the object alive as long as there are sounds; `spawn` keeps the object alive as long as spawner isn't depleted (based on TotalObject); `track` keeps the object alive as long as there are tracks.
- Age: If defined, object will seem to be [Age] old upon creation, in seconds.
- `Color`: Defines a color (ie. tint) to apply on this object when rendered. Values are RGB from 0 to 255. Official HTML web color names are also supported (this list can be extended through the Color config section). This color will override any color defined in the current rendered orxGRAPHIC for this object. If none is provided, the color of the current orxGRAPHIC will be used is available, the white color (ie. no tint) will be used otherwise.
- RGB: Values are RGB from 0.0 to 1.0. Will be used only if Color isn't defined.
- HSL: Values are HSL from 0.0 to 1.0. Will be used only if Color and RGB aren't defined.
- HSV: Values are HSV from 0.0 to 1.0. Will be used only if Color, RGB and HSL aren't defined.
- `Flip`: Defines if the object should be flipped when rendered. Flipping only affects the visual. Properties like physics body won't be affected. It can be flipped on either X axis or Y axis or on the `both` axes at the same time. If no value is provided, no flipping will occur.
- `Position`: Initial position for this object, in pixels. If an object has a parent, this value will be considered in parent's space instead than in world space. By default its value is (0.0, 0.0, 0.0). An optional placement pivot “override” (literals or vector) can be provided at the beginning, followed by '→'. If a parent is available, the position (literals with optional offset or vector) can be expressed in the parent space with UseParentSpace, ie. [0-1] map to the parent's size on each axis.
- `SphericalPosition`: Theta & Phi components are in degrees. Will be used only if Position isn't defined.
- `Pivot`: Normally this is defined in the Object's `Graphic` section. However, setting Pivot here overrides graphic's value if present. `truncate` and `round` will adjust pivot values if they are not integers. z is ignored for 2D graphics.
- `Size`: Overrides graphic's value if present.
- `UseRelativeSpeed`: If true, the initial speed will be applied relatively to current object rotation & scale. Defaults to false.

- `Rotation`: Initial 2D angle in degrees around z axis. If an object has a parent, this value will be considered in parent's space instead than in world space. By default its value is 0.0.
- `AngularVelocity`: Initial velocity in degrees/second around z axis. By default its value is 0°/s
- `Scale`: Initial scale for this object (Z component being ignored). By default its value is 1.0 (ie. (1.0, 1.0, 0.0)). If it has a valid parent and use its space, this value will be considered in parent's space, ie. [0-1] defines parent size on each axis.
- `Smoothing`: Specifies if the graphic object should be rendered with antialising or not. This property is overridden by the current orxGRAPHIC used for this object. By default its value is default which means the global display setting value will be used.
- `BlendMode`: Defines which blend mode will be used when rendering this object. By default its value is alpha, which means the alpha channel will be used as opacity value. This rendering parameter can be overridden by the current rendered graphic object.
- `Repeat`: Specifies if the object should be repeated (so as to achieve a tiling effect) when rendered. The vector specifies the number of times the object will be repeated on X and Y axes (Z being ignored). If none is provided, the value specified for the current orxGRAPHIC will be used. Please notice that the repeat property does not change the object's size, if you wish the object to be drawn several times (e. g.: you want to render the ground of your game, by having the same graphic several times) you need to set the `Scale` property as well. Other values must be strictly positive. This will be ignored for text objects.
- `FXList`: Specifies a list of orxFX to apply on this object when created. FX will be played immediately. Up to 8 FXs can be defined. If none is provided, no orxFX will be applied.
- `FXRecursiveList`: Should the corresponding FXs in FXList be applied recursively (defaults to false for each of them).
- `SoundList`: List of orxSOUND to apply on the object at its creation. They can be both music or sound. Up to 4 sounds can be defined. If none is provided, no sound will be created with this object.
- `ShaderList`: List of orxSHADER to apply when this object will be rendered. Shader will be activated immediately. The shader is applied after the object's rendering but object without any `Graphic` can also use them. Up to 4 shaders can be defined. If none is provided, no shader will be applied when the object is rendered.
- `TrackList`: Timeline tracks will be played immediately. Up to 16 timeline tracks can be defined.
- `ChildList`: List of objects (config names) to create along this object. They'll be linked as children which means they'll adapt to the object's current position, rotation and scale (they'll 'follow' it). Children will automatically be deleted when the current object (ie. their parent) is deleted. Up to 255 children can be created. If no value is provided for this property, no children will be created.
- `ChildJointList`: Optional list of joints to use to link the children to the parent. If defined, its indices **must** match those of the `ChildList` property. Defaults to nothing.
- `ParentCamera`: Config name of the orxCAMERA that will be used as parent. This is very useful to place objects in the camera space for UI purposes, for example. If none is provided, the object won't have any parent except if it was created through a `ChildList` of another object.
- `UseParentSpace`: If set to true/both and has a parent at creation (camera, object or spawner) [1], its position and scale will be considered in parent's space, ie. values between 0.0 and 1.0 defines parent's size on each axis [2] and using its pivot as origin. It can be applied individually to only position or scale. Only one of the two parent property can be considered by using the value `position` or `rotation`. Defaults to true if ParentCamera is defined, false otherwise.
- `IgnoreFromParent`: Defines which parts of the parent's transformation will be ignored when transmitted to the object. Ignoring any position components will take precedence over placement pivot override (see 'Position') and might yield unexpected results. Defaults to none.
- `OnPrepare`: Command run when preparing an object for creation. If the command returns false

or empty, the object will not be created.
- `OnCreate`: Command run when an object has been created.
- `OnDelete`: Command run when an object will get deleted. If the command returns false or empty, the object will not be deleted, but only deactivated.

Latest config settings for the Development Version

We endeavor to keep the config properties on this page up to date as often as possible. For up to the minute config information for the latest version of Orx, check the most recent published at:

CreationTemplate.ini and

SettingsTemplate.ini

Additionally these files can be found under your orx source tree in the `orx/code/bin` folder.

1)
this won't affect objects created through another object's `ChildList`
2)
mainly used for easiness of placing UI objects relative to camera that will remain resolution independent

From:
https://www.orx-project.org/wiki/ - **Orx Learning**

Permanent link:
https://www.orx-project.org/wiki/en/orx/config/settings_structure/orxobject?rev=1692861870

Last update: **2025/09/30 17:26 (14 months ago)**