

Interlevel Lumps formal specification 0.99.1

A way of defining background images for victory screens has existed in UMAPINFO for a few years now. But it has never handled the animation capabilities that were on display in the original three episodes from Doom. This specification documents the data format that replicates the capabilities of Doom’s victory screen animations.

Scope of features

This specification has been written to handle exactly what is required of the intermission screens used in Doom and Doom II: Hell on Earth, whether they’re animated or static. Additional features were deemed outside of the scope of this revision of the specification. It has also been authored in such a way that only a single lump is required for both exiting a level and entering a new one, in order to make it easier for content authors to manage the presentation exactly as they deem fit.

JSON lump

This specification uses the JSON Lump 1.0.0 formal specification as the root of its data storage, with a type of “**interlevel**” and a version of “1.0.0”.

Minimum engine featureset

This specification applies to any **limit-removing** featureset or greater. While it is optional for other valid featuresets, it is a requirement of the **ID24** featureset.

Terminology

While the code uses “intermission” to mean the victory screens (ie exiting level tallies and entering level text/”you are here” display) and “finale” to refer to both the end-of-episode text and graphics displayed, community standards have come to term the end-of-episode text as “intermission”. As such, rather than reappropriating the term to match source code the victory screens will now exclusively be referred to as **interlevel** screens. Any terminology that uses **interlevel** refers to this point in the gamesim.

UMAPINFO additions

UMAPINFO previously provided the **exitpic** and **enterpic** lumps to define the background of the **interlevel** screens. Rather than override the behavior of these values, this specification adds two additional fields to any compatible version of UMAPINFO:

Name	Type	Description
exitanim	string	An interlevel background lump used for the score tally screen
enteranim	string	An interlevel background lump used for the “entering” screen

These lumps are considered authoritative and override any **exitpic** and **enterpic** definitions. Content authors are however encouraged to fill out both fields for ports that lack an implementation for this specification. If an **exitanim** or **enteranim**’s target lump is not correctly formed, this is considered an error condition. Falling back on **exitpic** and

enterpic in the event of an error is not allowed and is considered a non-compliant implementation of this spec.

Ports that do not implement this specification are expected to ignore **exitanim** and **enteranim** and present **exitpic** and **enterpic** as normal.

Runtime environment

The **interlevel** backgrounds are to be rendered in a virtual 320x200 resolution (referred to as the virtual environment, presented at a 4:3 aspect ratio with rectangular pixels at a 1:1.2 aspect ratio (ie 1.2 vertical units are required to maintain the 4:3 aspect ratio). This is identical to the original Doom and Doom II presentation. The virtual resolution is enforced to allow equivalent presentations at any actual device output resolutions.

Resources are allowed to render outside of the virtual environment. This allows for the presentation to adapt to widescreen aspect ratios.

Elements are **not allowed** to specify coordinates outside of the virtual environment. Elements are allowed to extend outside of the virtual environment and will be clipped appropriately to the output resolution, but placement of elements must be inside the virtual environment.

As these are backgrounds, every element is rendered under the graphics that the engine renders for stats etc.

The map number accessible for condition checking is dependent on whether the victory screen is currently displaying an exit animation or an enter animation. If it is an exit animation, the map number corresponds to the map being played; if it is an enter animation, the map number corresponds to the map about to be entered. The map number itself is defined through other means, including but not limited to UMAPINFO, and is outside of the scope of this document to define.

Non-deterministic environment

Interlevels are non-deterministic. In practice, this means the following:

- Any value that uses a random number generator (RNG) is not guaranteed to give the same results on every instantiation.
 - Using **M_Random** from the Doom codebase satisfies this requirement.
- Time durations are not considered to be exact; however, implementations are expected to give “as close as feasible” results.
 - Rounding to the nearest tic satisfies this requirement.

Data types

As data is represented using the JSON lump standard, there are several object types that need to be interpreted. The types defined are:

- root
- layer
- anim
- frame
- condition

All values defined in these data types are to be read from the JSON document and set, regardless of whether the declared functionality will use those values or not. If any of these values cannot be read, it is to be considered an error condition.

Root definition

The root object describes a background image to be rendered first; a music track to be played; and an array of layers to be rendered over the background image in sequential order.

The **backgroundimage** lump is always rendered first. This is expected to be centered on the screen, allowing for automatic adjustment for widescreen presentations. If this lump is not defined or not found in the loaded WADs, it is considered an error condition.

The **music** lump plays in the standard music channel of the audio mix. If this lump is not defined or not found in the loaded WADs, it is considered an error condition.

The **layers** array can be zero-length, resulting in an **interlevel** just displaying the background image. This is handled with a **null** value in the JSON document. An array declared with zero elements instead of using **null** is to be considered an error condition.

Layer definition

A **layer** is a collection of **anims** that are displayed in sequential order. Each **layer** can be turned on or off by resolving an **array of conditions**. When a layer is turned off, none of its anims update or render.

A layer must have a minimum of one **anim**. Less than one **anim** is considered an error condition.

Anim definition

An **anim** is a collection of **frames** that are displayed in sequential order at a specified screen coordinate. Each **anim** can be turned on or off by resolving an **array of conditions**. When an **anim** is turned off, none of its frames update or render.

An **anim** must have a minimum of one **frame**. Less than one **frame** is considered an error condition.

Frame definition

A **frame** is a graphic displayed on screen for a defined amount of time.

There are three types of frame times that can be specified:

- **Infinite** - A frame that is always displayed, and will never progress to the next frame in an animation. **duration** and **maxduration** are ignored at runtime.
- **Fixed** - A frame that is displayed for an amount of time no less and no more than the specified **duration**. A fixed frame must have a minimum duration of one rendered image in the runtime environment. As such, a zero value for **duration** is to be considered an error condition. **maxduration** is ignored at runtime.
- **Random** A frame that is displayed for an amount of time no less than the specified **duration** and no more than the specified **maxduration** as determined by the RNG, resulting in a frame that changes its duration every time it is displayed. A

random frame must have a minimum duration of one rendered frame. As such, a zero value for **duration** is to be considered an error condition; and a **maxduration** of lesser value than **duration** is to be considered an error condition.

When discussing the length of time of a **frame**, **duration** is assumed to mean one of these three things and dependent on the values the user has set within the **frame** object.

A frame can also be specified to have a **random first frame offset**. When this functionality is specified, the presentation code will use the RNG to reduce the **duration** of the frame if it is the **first frame displayed** when an **interlevel** is instantiated, i.e. on a **one-time only** basis. The new duration generated must have a minimum of one rendered frame, and a maximum of the original **duration** resolved. When this frame is encountered again, it will follow the normal **duration** rules.

Condition definition

A **condition** is a test that must pass to resolve to **true**; It otherwise resolves to **false**.

An array of **conditions** must all resolve to **true** for that entire collection to resolve to **true**; any failure makes the entire **condition array** resolve to **false** - that is to say that the entire collection resolves with **logical and** operations.

When zero **conditions** are declared in a **condition array**, the **condition array** always resolves to **true**. This is handled with a **null** value in the JSON document. An array declared with zero elements instead of using **null** is to be considered an error condition.

Conditions and **condition arrays** are resolved on **interlevel** instantiation. They do not resolve at any other time, and any implementation using them in such a manner is to be considered a non-compliant implementation of this spec.

Data type definitions

root

Name	Type	Description
music	string	Lump name of music to be looped while this background lump is displayed.
backgroundimage	string	Lump name of the patch to be rendered first before any layers.
layers	array of layer	An array of anim elements to be rendered on top of the backgroundimage. Can be null.

layer

Name	Type	Description
anims	array of anim	An array of anim elements to be rendered on top of the preceding layers.
conditions	array of	An array of conditions that must be met for this layer

	condition	to be displayed. Can be null.
--	------------------	-------------------------------

anim

Name	Type	Description
x	integer	The x position of this animation.
y	integer	The y position of this animation.
frames	array of frame	An array of frame elements that are iterated on according to their parameters, and displayed sequentially over time.
conditions	array of condition	An array of conditions that must be met for this layer to be displayed. Can be null.

frame

Name	Type	Description
image	string	The patch to be rendered
type	integer	<p>Bitfield with the following values:</p> <ul style="list-style-type: none"> • 0x0000: None • 0x0001: Infinite duration • 0x0002: Fixed duration • 0x0004: Random duration • 0x1000: Random first frame offset <p>Infinite duration, fixed duration, and random duration correspond to the definitions found in the Frames subsection and are in an exclusive binary bit group. Only one of these bits can be selected at a time; zero bits selected or more than one bit selected is to be considered an error condition.</p>
duration	number	The number of seconds that this frame is displayed for.
maxduration	number	The maximum number of seconds this frame is displayed for if random duration is selected in the type field.

condition

Name	Type	Description
condition	integer	<p>Enumeration with the following values:</p> <ul style="list-style-type: none"> • 0: None • 1: Current map number is greater than the

		<p>param value</p> <ul style="list-style-type: none"> • 2: Current map number is equal to the param value • 3: The map number corresponding to the param value has been visited • 4: The current map is not a secret map • 5: Any secret map has been visited • 6: The current screen is the tally screen • 7: The current screen is the “entering” screen <p>This enumeration determines the test that will be run when the parent object is instantiated.</p>
param	integer	A value that this condition checks against, as defined in the condition field.

Reference implementation details

The implementation was written to be the bare minimum required to handle Doom’s **interlevel** animations in a generic manner. The specification was derived from this implementation, however there are elements that you will see in the code that fall outside of this specification. This is intentional, and the following elements are described below.

Win screens

The code for the win screens (**wi_*.***) has been entirely rewritten to handle the data structures generated by this specification. How a port implements these screens is entirely up to the code author; likewise, how they translate the data defined by this specification into their own internal formats is beyond the scope of this specification.

Frame type

The **type** field in the **frame** definition has one other bitfield defined internally:

- 0x8000000: Adjust for widescreen, ie centre the image at (160, 0) in the virtual environment

With this, a dummy animation is created solely to render the **backgroundimage** lump from the **root** definition without needing to special case the render loop. As it is an internal flag, it can be changed at any time pending future specifications; and as the JSON Lump specification does not support forward compatibility, this value should be considered entirely free for use in future specifications.

Conditions

There is one additional condition used in the reference implementation:

- Whether this element fits within the virtual space, and is the first element of the group specified by **param** to do so

This is used exclusively as a shortcut for the hardcoded tables to handle the “You are here” pointer graphics in Doom episodes 1 through 3. Rather than painstakingly code each separate element, this shortcut means it can intelligently decide whether to use the left-pointing or right-pointing element. The implementation code however probably breaks for

anything outside of this specific usecase. It is also made entirely redundant by layout tools. Do not attempt to copy the implementation or the condition, as it is precisely the kind of esoteric “implementation specific” feature that should be left entirely out of the collective consciousness.

Durations

Durations have been intentionally specified as seconds for futureproofing purposes. As this specification defines a non-deterministic visual-only element, there is no particular need for the animations to adhere to Doom’s 35Hz tic rate. The reference implementation however does adhere to the tic rate, so values are converted to tics on load and stored in the relevant structures.