**BIO ENGINE Level Parsing Specification**

**Overview**

This specification describes the layout and format of '.level' files and how they should be parsed in the BIO Engine. '.level' files use the xml markup language. XML version 1.0 with the utf-8 encoding. All '.level' files should have '<?xml version=”1.0” encoding=”utf-8”?>' as the first line in the file to ensure portability and compatibility.

'.level' files are meant to hold all of the data needed to create the level in the BIO Engine. All tags and attributes are parsed before the level is actually created. This allows for a flexible structure in which order doesn't matter. As long as it is specified in the '.level' file, it will be loaded.

Note to Developers: If you need to add a tag to the engine and it must be parsed in a specific order, you are doing it wrong. If the new tag and code is submitted for inclusion in the Engine it will be rejected and sent back for re-work if order matters in the parsing of the tag. If for some unforeseen circumstance arises and there is no way to avoid a specific order, then please submit with the code an explanation of why it needs to be parsed in a specific order.

Also '.level' files are case insensitive. All parsing will be done using case insensitive algorithms (This means that Hello is the same as HELlo which is the same as hElLO). Just because it is case insensitive doesn't give you the right to type stupidly (such as the example above). Please make '.level' files readable and try to keep a consistent style (otherwise people might think you are retarded).

---------------------------------------------------------------------

**Basic Types**

These are types that will be used throughout the specification to specify values for certain elements. This will define what these tags are and how they should be parsed.

**VECTOR3**

A VECTOR3 will have an x, y, and z component. The xml tag will parse x, y, and z attributes. All of the attributes will be opional. If the attribute is not present, then the parser will assume a value of 0.0f. Anything between the tag will be ignored. Also the attributes are allowed to be in any order.

Example:

<VECTOR3 /> will return a 0, 0, 0.

<VECTOR3 x=”1.6” /> will return 1.6, 0, 0>

<VECTOR3 z=”2.8” x=”1.6” /> will return 1.6, 0, 2.8

**VECTOR2**

A VECTOR2 will be the same as a Vector3 except there is no z component. Any z attribute shall be ignored.

<VECTOR2 x=”1.2” y=”2.9” /> will return 1.2, 2.9

**COLOR**

A COLOR tag will have 5 attribute arguments, all of which are optional. The attributes will be: type, r, g, b, and a. Type will determine what types of numbers will be accepted. The default will be int. Accepted types will be 'int' and 'float'. int means the values in r, g, b, and a will be integer values between 0 and 255. Float means they will be decimal values between 0.0 and 1.0. If the value in the type attribute it's not recognized, then int will be assumed. The parser must return all values as floating point numbers between 0.0 and 1.0. If r, g, b, or a attribute is not specified then a value of 255 (or 1.0) is assumed.

Example:

<COLOR type=”int” r=”128” b=”64”/> will return a=1.0, r=0.5, g=1.0, b=0.25

**ANGLE**

The ANGLE tag has 2 attributes: type, and value. The type attribute has 2 valid values: Degrees, and Radians. Radians is the default type value(if type is not specified then assume radians). The value attribute is the value of the angle. The Angle parsing function should return a floating point value and the value should always be in radians.

Example

<ANGLE type=”Degrees” value=”90” /> will return 1.5707f

<ANGLE type=”Radians” value=”1.5707” /> will return 1.5707f

<ANGLE value=”3.14” /> will return 3.14f

---------------------------------------------------------------------

**File Structure**

Each '.level' file will have one base tag that surrounds every other tag. This is the <LEVEL></LEVEL> tag. All tags must be between these two tags. Anything outside of these tags will be ignored.

The level tag has 1 optional attribute inside it. The level tag's name attribute will be a way to name the level*(currently this attribute has no specific use).*

Within the level tag there are several substructures. As with all parts of the '.level' file the order doesn't matter. What does matter is that parts are contained within their proper substructures (so don't put something from 'Objects' under the 'Settings' section). The substructures are as follows:

Camera

Objects

Settings

Example File:

<?xml version="1.0" encoding="utf-8"?>

<LEVEL>

<SETTINGS>

...

</SETTINGS>

<OBJECTS>

...

</OBJECTS>

<CAMERA>

...

</CAMERA>

</LEVEL>

Each of the substructures will be described in detail below.

.....................................................................

**CAMERA**

This tag has all of the settings for the level's camera object. Currently 2 types of cameras are supported: Default and Following. A default camera is controlled using the keyboard and mouse as the inputs. A following camera follows an object around the world. The CAMERA tag has 3 attributes that are all optional: type, minview, and maxview. The type will specify what type of camera object will be created and it's values are default and following(default will be the default). Minview specifies how close the closest object is before it gets clipped from the view. This default value will be 0.1f. Maxview is how far away the furthest object is before it gets clipped from view. The default will be 1000.0f.

Example:

<CAMERA type=”default” minview=”1.0” maxview=”100.0”>...</CAMERA>

The types of tags that are expected within the CAMERA tag depends on which type of camera is defined in the 'type' attribute.

**Following Camera Type Tags**

**CAMERAFOLLOW**

The CAMERAFOLLOW tag specifies which object(defined in the objects tags) will be followed by the camera. If none is defined then the camera will stay stationary until an object is set as it's target. This tag has no attributes and the objects is specified between the two tags as follows:

<CAMERAFOLLOW>tank1</CAMERAFOLLOW>

NOTE: The name between the CAMERAFOLLOW tag is case insensitive.

**FOLLOWDISTANCE**

The distance(horozontal to the ground) that the camera will be placed from the followed object. This can be a floating point value. The FOLLOWDISTANCE tag does not have any attributes and the distance is specified between the opening and closing tags.

Example:

<FOLLOWDISTANCE>11.23</FOLLOWDISTANCE> will specify a distance of 11.23 meters horizontal distance from the object.

**FOLLOWHEIGHT**

The FOLLOWHEIGHT tag will specify the height above the object at which the camera will be placed. This is a floating value and the FOLLOWHEIGHT tag does not have any attributes that will be interpreted.

Example:

<FOLLOWHEIGHT>5.6</FOLLOWHEIGHT> will specify a height of 5.6 meters above the object where the camera will be placed.

**Default Camera Type Tags**

**POSITION**

The POSITION tag specifies the x, y, and z position where the camera will be placed in the level. The POSITION tag does not have any attributes that will be parsed. Instead it will expect a VECTOR3 tag between it's opening and closing tags as follows:

<POSITION><VECTOR3 x=”2.1” z=”4.5”/></POSITION>

**LOOKAT**

The LOOKAT tag specifies which specific point in the world the camera will focus on. This tag has no attributes but like the POSITION tag it requires a VECTOR3 tag between its opening and closing tags.

Example:

<LOOKAT><VECTOR3 x=”1.4” z=”2.2”/></LOOKAT> Will look at the point 1.4, 0.0, 2.2

.....................................................................

**OBJECTS**

@TODO FINISH THE OBJECTS SECTION.

.....................................................................

**SETTINGS**

Within this tag are contained all of the details and data needed for the level in general. Things such as sun position, time of day, date, general lighting colors, and the most important; the file that holds the 3D level data. Below all the tags that belong under the settings substructure are detailed. And remember, ORDER DOESN'T MATTER (Have I said it enough).

**LEVELFILE**

This tag contains the name of the file that contains the level's geometry. This file should be a file that the engine can open/load(different topic for a different specification). This will contain all of the static geometry needed for the level(nothing in this file will be separated or placed into separate objects). The tag itself has no attributes(none will be parsed). The file name is stored between the opening and closing LEVELFILE tags as so:

<LEVELFILE>Level1.obj</LEVELFILE> Level1.obj is the level geometry file.

**GRAVITY**

The GRAVITY tag specifies how strong gravity is and which direction the gravity will be in. This tag has no attributes and must have a VECTOR3 tag between it's opening and closing tags. Gravity will be interpreted in the units m/s2.

Example:

<GRAVITY><VECTOR3 x=0 y=-9.8 z=0/></GRAVITY> will set gravity to be 9.8 m/s2 in the negative y direction.