ZombieHorse

File Format Specifications

# Animations (*.zha*)

ZHA files are XML files which store sets of animation data and annotations.

## The root element

The root element of a ZHA file is *<AnimationSet>*. It has the following attributes:

* *version* – *string*; file format version; can be 1.0
* *id* – *unsigned long*; animation set ID
* *name* – *string*; animation set name

<AnimationSet version="1.0" id="17" name="Ninja">

...

</AnimationSet>

The root element can have up to one of the following child elements:

* *<Animations>*
* *<AnimationSpaces>*

## Animations

*<Animations>* element has no attributes. It has zero or more *<Animation>* elements as children.

*<Animation>* element has the following attributes:

* *id* – *unsigned short*; animation ID
* *name* – *string*; animation name
* *interpMethod* – (optional) *enum KFInterpolationMethod* (possible values: *Linear*, *Spline*); method used to interpolate between key-frame values on animation tracks

Moreover, it can have the following child elements:

* *<Tracks>* – (optional) specifies a list of animation tracks
* *<Annotations>* – (optional) can be used to specify animation annotations

<Animation id="125" name="WalkStraight" interpMethod="Spline" >

<Tracks>

...

</Tracks>

<Annotations>

...

</Annotations>

</Animation>

### Animation tracks

Tracks are specified in arbitrary sequence; depending on track type, they are specified as *<BoneTrack>* or *<MeshTrack>* elements.

*<BoneTrack>* element specifies a bone animation track. It has the following attributes:

* *boneId* – *unsigned short*; ID of the affected bone

It has one child – *<TransformKeyFrames>* – specifying a list of *<TransformKeyFrame>* elements, each specifying a single animation key-frame. *<TransformKeyFrame>* element has the following attributes:

* *time* – *float*; key-frame time
* *translation* – *Vector3*; translation
* *rotation* – *Vector4*; rotation (in axis-angle form)
* *scale* – *Vector3*; scale

<BoneTrack boneId="0">

<TransformKeyFrames>

<TransformKeyFrame time="0" translation="0.0031 0 -0.00012"

rotation="0 1 0 0.273" scale="1 1 1" />

...

</TransformKeyFrames>

</BoneTrack>

*<MeshTrack>* element specifies a mesh animation track. It has the following attributes:

* *meshId* – *unsigned short*; ID of the affected mesh

It has one child – *<MorphKeyFrames>* – specifying a list of *<MorphKeyFrame>* elements, each specifying a single animation key-frame. *<MorphKeyFrame>* element has the following attributes:

* *time* – *float*; key-frame time
* *MTWeights* – *Vector*, *N*-dimensional (where *N* is the number of morph targets in the mesh); list of morph target weights

<MeshTrack meshId="1">

<MorphKeyFrames>

<MorphKeyFrame time="0" MTWeights="0 0.36 0 0 0 0.07 0 0 0 0" />

...

</MorphKeyFrames>

</MeshTrack>

### Animation annotations

Annotations are specified in arbitrary sequence, as children of the *<Annotations>* element. Each annotation is specified as element *<Annotation>*, with the following attributes:

* *startTime* – *float*; annotation start time (in seconds)
* *endTime* – *float*; annotation end time (in seconds)
* *classId* – *enum AnimationAnnotationClass* (possible values: *Transition*, *PlantConstraint*, *SimEvent*, *GesturePhase*); annotation class ID

<Annotation startTime="0.456" endTime="0.78" classId="Transition">

...

</Annotation>

*<Annotation>* has exactly one child element, which specifies the information specific to the annotation class given in *classId*. There is a different XML subelement for each annotation class: *<TransitionAnnotation>*, *<ParamTransitionAnnotation>*, *<PlantConstraintAnnotation>*, *<SimEventAnnotation>*, *<GesturePhaseAnnotation>*.

*<TransitionAnnotation>* element specifies an animation transition. It has the following attributes:

* *targetSetId* – (optional) *unsigned long*, ID of the target animation set
* *targetId* – *unsigned short*; ID of the target animation
* *targetTime* – *float*; transition start time in the target animation
* *alignTransf* – *ModelSituation*; 2D aligning transformation for the two animations

<TransitionAnnotation targetSetId="2" targetId="37" targetTime="0.4006"

alignTransf="0 0 0" />

*<TransitionAnnotation>* element has no children.

*<ParamTransitionAnnotation>* element specifies a transition to a parametrized animation. It has the following attributes:

* *targetSetId* – (optional) *unsigned long*, ID of the target animation set
* *targetId* – *unsigned short*; ID of the target animation
* *lowerBound* – *Vector*; lower boundary of the transition region of the target animation space
* *upperBound* – *Vector*; upper boundary of the transition region of the target animation space
* *targetTime* – *float*; transition start time in the target animation
* *alignTransf* – *ModelSituation*; 2D aligning transformation for the two animations

<ParamTransitionAnnotation targetSetId="2" targetId="37"

lowerBound="0.14 -0.14" upperBound="2.23 1.27" targetTime="0.4006"

alignTransf="0 0 0" />

*<ParamTransitionAnnotation>* element has no children.

*<PlantConstraintAnnotation>* element specifies a plant constraint on an end-effector bone. It has the following attributes:

* *boneId* – *unsigned short*; ID of the end-effector bone

<PlantConstraintAnnotation boneId="12" />

*<PlantConstraintAnnotation>* element has no children.

*<SimEventAnnotation>* element specifies simulation event that should be triggered by the animation. It has the following attributes:

* *eventClassId* – *unsigned short*; ID of the simulation event

<SimEventAnnotation eventClassId="112" />

*<SimEventAnnotation>* element has no children.

*<GesturePhaseAnnotation>* element specifies a gesture phase, enabling synchronization of gesture animations by the behavior system. It has the following attributes:

* *gesturePhase* – *enum AnimationGesturePhase* (possible values – *Start*, *Ready*, *StrokeStart*, *Stroke*, *StrokeEnd*, *Relax*, *End*); gesture phase ID

<GesturePhaseAnnotation gesturePhase="Stroke" />

*<GesturePhaseAnnotation>* element has no children.

## Animation spaces

*<AnimationSpaces>* element has no attributes. It has zero or more *<AnimationSpace>* elements as children.

*<AnimationSpace>* element has the following attributes:

* *id* – *unsigned short*; ID of the the animation space
* *name* – *string*; name of the animation space

<AnimationSpace id="2" name="MoveForward">

...

</AnimationSpace>

The root element has the following children (up to one of each):

* *<BaseAnimations>* – specifies a list of base (discrete) animations that define the animation space
* *<TimewarpCurve>* – specifies a timewarp curve used for temporal alignment
* *<AlignmentCurve>* – specifies an alignment curve used for spatial alignment
* *<AnimationParametrization>* – specifies the parametrization of the blend weights

### Base animations

*<BaseAnimations>* element has no children, but the following attributes:

* *animations* – *unsigned short list*; list of IDs of base animations in this animation space

<BaseAnimations animations="22 23 28 24 25 12 31 30" />

### Registration curves

The children of *<TimewarpCurve>* are 3 or more *<CtrlPoint>* elements, specifying control points of the spline (therefore, their ordering is mathematically relevant!). Each *<CtrlPoint>* element has the following attributes:

* *position* – *Vector*; *N*-dimensional, where *N* is equal to the number of base animations

Similarly, the children of *<AlignmentCurve>* are 3 or more *<CtrlPoint>* elements. Each *<CtrlPoint>* element has the following attributes:

* *position* – *Vector*; *3N*-dimensional, where *N* is equal to the number of base animations

<TimewarpCurve>

<CtrlPoint position="0 0 0 0 0.12 0.10 0.11 0.10 0.12 ..." />

...

</TimewarpCurve>

<AlignmentCurve>

<CtrlPoint position="0 0 0 12.55 -3.43 0.94 6.25 12.06 1.18 0.04 -0.19 0.01 -8.90 2.20 -0.44 ..." />

...

</AlignmentCurve>

### Animation parametrization

*<AnimationParametrization>* has the following attributes:

* *numParams* – *unsigned int*; number of parameters (*M*)
* *classId* – *enum AnimationParamClass* (possible values: *DenseSampling*); parametrization class ID

It also supports the following children:

* *<Params>*
* *<DenseSamplingParametrization>*
* *<BaseSamples>*

*<Params>* element is mandatory and contains a list of *M* parameters. Each parameter is specified as *<Param>* child element, with the following attributes:

* *name* – *string*; name of the parameter

*<BaseSamples>* element is mandatory and contains a list of *N* sample parameter values (*<Sample>* child elements) which correspond to blends where a base animation has weight 1. *<Sample>* element has the following attributes:

* *paramValues* – *Vector*, *M*-dimensional, where *M* is the number of parameters

Depending on *classId* attribute, exactly one of the following children must be defined in *<AnimationParametrization>*:

* *<DenseSamplingParametrization>* – used when the parameter space is densely-sampled

*<DenseSamplingParametrization>* does not have any attributes, but has one child element, *<Samples>*, which specifies a list of sample parameter value sets and their corresponding blend weights. Each set of sample parameter values is specified as a *<Sample>* element (child of *<Samples>*), with the following attributes:

* *paramValues* – *Vector*, *M*-dimensional, where *M* is the number of parameters
* *weights* – *Vector*, *N*-dimensional, where *N* is the number of base animations

<AnimationParametrization numParams="2" classId="DenseSampling">

<Params>

<Param name="direction" />

<Param name="speed" />

</Params>

<BaseSamples>

<Sample paramValues="0 0.98" />

...

</BaseSamples>

<DenseSamplingParametrization>

<Samples>

<Sample paramValues="0.03 0.98"

weights="0.97 0.02 0.01 0 0" />

<Sample paramValues="0.10 0.97"

weights="0.91 0.04 0.05 0 0" />

...

</Samples>

</DenseSamplingParametrization>

</AnimationParametrization>

## Animation Binary

ZHA can also be binary files which store sets of animation data and annotations.

Data in ZHA binary files is organized into chunks. The following types of chunks are supported:

enum ZHAChunkType

{

ZHAChunk\_Header,

ZHAChunk\_AnimationSet,

ZHAChunk\_Animation,

ZHAChunk\_BoneTrack,

ZHAChunk\_MeshTrack,

ZHAChunk\_TransformKeyFrame,

ZHAChunk\_MorphKeyFrame,

ZHAChunk\_Annotation,

ZHAChunk\_TransitionAnnotation,

ZHAChunk\_ParamTransitionAnnotation,

ZHAChunk\_PlantConstraintAnnotation,

ZHAChunk\_SimEventAnnotation,

ZHAChunk\_GesturePhaseAnnotation,

ZHAChunk\_AnimationSpace,

ZHAChunk\_TimewarpCurve,

ZHAChunk\_AlignmentCurve,

ZHAChunk\_AnimationParametrization,

ZHAChunk\_Param,

ZHAChunk\_Sample,

ZHAChunk\_DenseSamplingParametrization

};

A chunk begins with the following header:

struct ChunkHeader

{

unsigned short type; ///< Chunk type.

unsigned long size; ///< Chunk size (in bytes).

};

The header is followed by a block of chunk data, the size of which is given in the header. Data can include primitive types (integers, floats…), arrays of primitive types, strings or other chunks.

# Animation Tree (*.zht*)

ZHT files are XML files which store animation trees.

## The root element

The root element of a ZHT file is *<AnimationTree>*. It has the following attributes:

* *version* – *string*; file format version; can be 1.0
* *id* – *unsigned long*; ID of the the animation tree
* *name* – *string*; name of the animation tree
* *rootId* – *unsigned short*; (optional) ID of the root node
* *firstId* – *unsigned short*; (optional) ID of the first bone controller
* *applyMover* – *bool*; (optional) if false, anim. mover is applied directly if present, otherwise animations are realigned

<AnimationTree version="1.0" id="6" name="AT\_Ninja" applyMover=“true“>

...

</AnimationTree>

The root element has the following children (up to one of each):

* *<Nodes>* – (optional) specifies a hierarchy of animation nodes
* *<BoneControllers>* – (optional) specifies a chain of bone controllers

## Animation nodes

Animation nodes are specified as *<Node>* elements. Each *<Node>* has the following attributes:

* *classId* – *unsigned long*; ID of the animation node class
* *id* – *unsigned short*; ID of the animation node
* *name* – *string*; name of the animation node

*<Nodes>* element can have zero or more *<Node>* children, one of which is the root animation node.

*<Node>* element supports the following child elements:

* *<Parameters>* –specifies node parameters, such as play state, bone mask etc.
* *<Children>* – (optional) specifies a list of child animation nodes (*<Node>* elements)

*<Children>* element can have any number of *<Node>* elements as children.

*<Parameters>* can have zero or more of the following children:

* *<Parameter>* – specifies a single node parameter

*<Parameter>* element specifies the following attributes:

* *name* – *string*, parameter name

Content of *<Parameter>* element depends on parameter name (different parameters have different content). Parameter set depends on node class – a node can have all the parameters of its superclass nodes + a set of its own parameters.

<Node classId="2" id="17" name="WalkForward">

<Parameters>

<Parameter name=“mainChildId“>

...

</Parameter>

...

</Parameters>

<Children>

<Node classId="1" id="22" name="WalkStraight">

...

</Node>

...

</Children>

</Node>

### AnimationNode

Nodes of class *AnimationNode* have the following parameters (see animation tree spec. for explanations):

* *mainChildId* – *unsigned short*
* *playing* – *bool*
* *paused* – *bool*
* *playTime* – *float*
* *normalizedPlayTime* – *float* (constrained to 0-1 range)
* *playRate* – *float*
* *annotationsEnabled* – *bool*
* *origin* – *Vector3* (position x-z, orientation y)
* *boneMask* – *unsigned short array*

### AnimationMixer

Nodes of class *AnimationMixer* have the following additional parameters (see animation tree spec. for explanations):

### AnimationSampler

Nodes of class *AnimationSampler* have the following additional parameters (see animation tree spec. for explanations):

* *animationSetId* – *unsigned long*
* *animationId* – *unsigned short*

### AnimationBlender

Nodes of class *AnimationBlender* have the following additional parameters (see animation tree spec. for explanations):

* *weights* – *float array*
* *params* – *Vector*
* *animationSetId* – *unsigned long*
* *animationSpaceId* – *unsigned short*
* *useBlendCurves* – *bool*
* *useBlendCurves* – *bool*
* *plantConstrControllers* – *unsigned short-unsigned short map*

### AnimationTransitionBlender

Nodes of class *AnimationTransitionBlender* have the following additional parameters (see animation tree spec. for explanations):

* *transitionQueue* – 0 or more *<Transition>* elements
* *defaultTransitionLength* – *float*
* *defaultNode* – *unsigned short*

*<Transition>* element has the following attributes:

* *startTime* – *float*; transition start time
* *endTime* – *float*; transition end time
* *targetTime* – *float*; transition start time in target node
* *targetNode* – *unsigned short*; ID of the transition target node
* *targetParams* – *Vector*; parameter value set for the transition target node

### FaceController

*<FaceController>* element has the following attributes:

* *morphTargetWeights* – *Vector*; *N*-dimensional vector of morph target weights, where *N* is the number of morph targets in the mesh

## Bone controllers

Bone controllers are specified as *<BoneController>* elements. Each *< BoneController >* has the following attributes:

* *classId* – *unsigned long*; ID of the bone controller class
* *id* – *unsigned short*; ID of the bone controller
* *name* – *string*; name of the bone controller

*< BoneControllers>* element can have zero or more *< BoneController>* children, one of which is the first bone controller in the chain.

*< BoneController >* element supports the following child elements:

* *<Parameters>* –specifies bone controller parameters, such as whether it is active
* *<Next>* – (optional) specifies the next bone controller in the chain (*<BoneController>* element)

*<Next>* element can have up to one *<BoneController>* elements as children.

*<Parameters>* can have zero or more of the following children:

* *<Parameter>* – specifies a single bone controller parameter

*<Parameter>* element specifies the following attributes:

* *name* – *string*, parameter name

Content of *<Parameter>* element depends on parameter name (different parameters have different content). Parameter set depends on bone controller class – a bone controller can have all the parameters of its superclass bone controller + a set of its own parameters.

<BoneController classId="1" id="4" name="RotateHead">

<Parameters>

<Parameter name=“boneId“>

...

</Parameter>

...

</Parameters>

<Next>

<BoneController classId="1" id="3" name="MoveBody">

...

</BoneController>

</Next>

</Node>

### BoneController

Nodes of class *BoneController* have the following parameters (see animation tree spec. for explanations):

* *boneId* – *unsigned short*
* *enabled* – *bool*
* *responseTime* – *float*

### BoneTransformController

Nodes of class *BoneTransformController* have the following parameters (see animation tree spec. for explanations):

* *translation* – *Vector3*
* *rotation* – *Quat*; rotation applied to the bone
* *scale* – *Vector3*; scaling applied to the bone
* *transformMode* – *enum BoneTransformMode* (possible values: *RelToInitial*, *RelToCurrent*, *RelToTarget*)

### BoneIKController

Nodes of class *BoneIKController* have the following parameters (see animation tree spec. for explanations):

* *targetPosition* – *Vector3*
* *offsetRotation* – *axis-angle vector*
* *IKSolver* – *enum IKSolverClass* (possible values: *CCD*)

# Animation Index (*.zhi*)

ZHI files are XML files which store an index of animation data.

## The root element

The root element of a ZHI file is *<AnimationIndex>*. It has the following attributes:

* *version* – *string*; file format version; can be 1.0
* *id* – *unsigned long*; animation index ID
* *name* – *string*; animation index name

<AnimationIndex version="1.0" id="3" name="Walk3">

...

</AnimationIndex>

The root element can have up to one of the following child elements:

* *<Animations>* – indexed raw animation segments
* *<MatchWebs>*

## Animations

*<Animations>* element has no attributes, but it has zero or more of the following children:

* *<Animation>* – indexed animation segment

*<Animation>* element has the following attributes:

* *animationSetId* – *unsigned long*; animation set ID
* *animationId* – *unsigned short*; raw animation ID
* *startTime* – *float*; animation segment start time
* *endTime* – *float*; animation segment end time

<Animations>

<Animation animationSetId="0" animationId="1"

startTime="2.37" endTime="56.01" />

<Animation animationSetId="0" animationId="3"

startTime="0" endTime="13.13" />

<Animation animationSetId="0" animationId="3"

startTime="21.47" endTime="45.5" />

...

</Animations>

## Match webs

*<MatchWebs>* element has no attributes. It has zero or more *<MatchWeb>* elements as children.

*<MatchWeb>* element has the following attributes:

* *animation1* – *unsigned int*; index of the first animation
* *animation2* – *unsigned int*; index of the second animation
* *sampleRate* – *unsigned int*; animation sample rate

Moreover, it can have the following child element:

* *<Paths>* – (optional) specifies a list of paths through the match web

<MatchWebs>

<MatchWeb animation1="0" animation2="0" sampleRate="30">

<Paths>

...

</Paths>

</MatchWeb>

<MatchWeb animation1="0" animation2="1" sampleRate="30">

<Paths>

...

</Paths>

</MatchWeb>

...

</MatchWebs>

### Paths

Match web paths are specified in arbitrary sequence, as children of the *<Paths>* element. Each path is specified as element *<Path>*, with the following attributes:

* *nextPath* – (optional) *unsigned int*; index of the next path that this path merges into
* *nextPoint* – (optional) *unsigned int*; index of the point in the next path that this path merges into

*<Path>* has the following child elements:

* *<Points>* – (optional) list of sequential points that comprise the path
* *<Branches>* – (optional) list of paths that branch off from this one

Points are specified in order of increasing indexes, as children of *<Points>* element. Each point is given as element *<Point>*, with the following attributes:

* *frame1* – *unsigned int*; index of the first frame
* *frame2* – *unsigned int*; index of the second frame
* *distance* – *float*; distance between frames
* *alignTrans* – *ModelSituation*; 2D transformation that aligns the two frames

Branches are specified as children of *<Branches>* element. Each branch is given as element *<Branch>*, with the following attributes:

* *point* – *unsigned int*; index of the point where the branch begins
* *path* – *unsigned int*; index of the branch path

<Path nextPath="13" nextPoint="35">

<Points>

<Point frame1="103" frame2="107"

distance="13.22" alignTransf="-0.0041 0 0.012" />

<Point frame1="104" frame2="107"

distance="13.74" alignTransf="-0.01 0 0.00901" />

...

</Points>

<Branches>

<Branch point="112" path="7" />

<Branch point="126" path="8" />

...

</Branches>

</Path>

## Animation Index Binary (*.zhi*)

ZHI files can also be binary files which store an index of animation data.

Data in ZHI files is organized into chunks. The following types of chunks are supported:

enum ZHIChunkType

{

ZHIChunk\_Header,

ZHIChunk\_AnimationIndex,

ZHIChunk\_Animation,

ZHIChunk\_MatchWeb,

ZHIChunk\_Path,

ZHIChunk\_Point,

ZHIChunk\_Branch

};

As with ZHA files, a chunk begins with the following header:

struct ChunkHeader

{

unsigned short type; ///< Chunk type.

unsigned long size; ///< Chunk size (in bytes).

};

The header is followed by a block of chunk data, the size of which is given in the header. Data can include primitive types (integers, floats…), arrays of primitive types, strings or other chunks.

# config.xml

Configuration of the ZombieHorse system is stored in *config.xml* file.

### The root element

The root element of *config.xml* is *<config>*. It does not have any attributes. Its children are the following:

TODO

# References

TODO