Bone(bpy_struct)

base class — bpy_struct

class bpy.types.**Bone(bpy_struct)**

Bone in an Armature data-block

bbone_curveinx

X-axis handle offset for start of the B-Bone's curve, adjusts curvature

TYPE:

float in [-inf, inf], default 0.0

bbone_curveinz

Z-axis handle offset for start of the B-Bone's curve, adjusts curvature

TYPE:

float in [-inf, inf], default 0.0

bbone_curveoutx

X-axis handle offset for end of the B-Bone's curve, adjusts curvature

TYPE:

float in [-inf, inf], default 0.0

bbone_curveoutz

Z-axis handle offset for end of the B-Bone's curve, adjusts curvature

TYPE:

float in [-inf, inf], default 0.0

bbone_custom_handle_end

Bone that serves as the end handle for the B-Bone curve

TYPE:

Bone

bbone_custom_handle_start

Bone that serves as the start handle for the B-Bone curve

TYPE:

Bone

bbone_easein

Length of first Bézier Handle (for B-Bones only)

TYPE:

float in [-inf, inf], default 1.0

bbone_easeout

Length of second Bézier Handle (for B-Bones only)

TYPE:

float in [-inf, inf], default 1.0

bbone_handle_type_end

Selects how the end handle of the B-Bone is computed

- `AUTO` Automatic – Use connected parent and children to compute the handle.
- `ABSOLUTE` Absolute – Use the position of the specified bone to compute the handle.
- `RELATIVE` Relative – Use the offset of the specified bone from rest pose to compute the handle.
- `TANGENT` Tangent – Use the orientation of the specified bone to compute the handle, ignoring the location.

TYPE:

enum in ['AUTO', 'ABSOLUTE', 'RELATIVE', 'TANGENT'], default 'AUTO'

bbone_handle_type_start

Selects how the start handle of the B-Bone is computed

- `AUTO` Automatic – Use connected parent and children to compute the handle.
- `ABSOLUTE` Absolute – Use the position of the specified bone to compute the handle.
- `RELATIVE` Relative – Use the offset of the specified bone from rest pose to compute the handle.
- `TANGENT` Tangent – Use the orientation of the specified bone to compute the handle, ignoring the location.

TYPE:

enum in ['AUTO', 'ABSOLUTE', 'RELATIVE', 'TANGENT'], default 'AUTO'

bbone_handle_use_ease_end

Multiply the B-Bone Ease Out channel by the local Y scale value of the end handle. This is done after the Scale Easing option and isn't affecte by it.

TYPE:

boolean, default False

bbone_handle_use_ease_start

Multiply the B-Bone Ease In channel by the local Y scale value of the start handle. This is done after the Scale Easing option and isn't affected by it.

TYPE:

boolean, default False

bbone_handle_use_scale_end

Multiply B-Bone Scale Out channels by the local scale values of the end handle. This is done after the Scale Easing option and isn't affected b it.

TYPE:

boolean array of 3 items, default (False, False, False)

bbone_handle_use_scale_start

Multiply B-Bone Scale In channels by the local scale values of the start handle. This is done after the Scale Easing option and isn't affected by

TYPE:

boolean array of 3 items, default (False, False, False)

bbone_mapping_mode

Selects how the vertices are mapped to B-Bone segments based on their position

- `STRAIGHT` Straight – Fast mapping that is good for most situations, but ignores the rest pose curvature of the B-Bone.
- `CURVED` Curved – Slower mapping that gives better deformation for B-Bones that are sharply curved in rest pose.

TYPE:

enum in ['STRAIGHT', 'CURVED'], default 'STRAIGHT'

bbone_rollin

–

Roll offset for the start of the B-Bone, adjusts twist

TYPE:

float in [-inf, inf], default 0.0

bbone_rollout

Roll offset for the end of the B-Bone, adjusts twist

TYPE:

float in [-inf, inf], default 0.0

bbone_scalein

Scale factors for the start of the B-Bone, adjusts thickness (for tapering effects)

TYPE:

`mathutils.Vector` of 3 items in [-inf, inf], default (1.0, 1.0, 1.0)

bbone_scaleout

Scale factors for the end of the B-Bone, adjusts thickness (for tapering effects)

TYPE:

`mathutils.Vector` of 3 items in [-inf, inf], default (1.0, 1.0, 1.0)

bbone_segments

Number of subdivisions of bone (for B-Bones only)

TYPE:

int in [1, 32], default 0

bbone_x

B-Bone X size

TYPE:

float in [-inf, inf], default 0.0

bbone_z

B-Bone Z size

TYPE:

float in [-inf, inf], default 0.0

children

Bones which are children of this bone

TYPE:

`bpy_prop_collection` of `Bone`, (readonly)

collections

Bone Collections that contain this bone

TYPE:

`BoneCollectionMemberships bpy_prop_collection` of `BoneCollection`, (readonly)

color

TYPE:

`BoneColor`, (readonly)

envelope_distance

envelope_distance

Bone deformation distance (for Envelope deform only)

TYPE:

float in [0, 1000], default 0.0

envelope_weight

Bone deformation weight (for Envelope deform only)

TYPE:

float in [0, 1000], default 0.0

head

Location of head end of the bone relative to its parent

TYPE:

`mathutils.Vector` of 3 items in [-inf, inf], default (0.0, 0.0, 0.0), (readonly)

head_local

Location of head end of the bone relative to armature

TYPE:

`mathutils.Vector` of 3 items in [-inf, inf], default (0.0, 0.0, 0.0), (readonly)

head_radius

Radius of head of bone (for Envelope deform only)

TYPE:

float in [-inf, inf], default 0.0

hide

Bone is not visible when it is not in Edit Mode (i.e. in Object or Pose Modes)

TYPE:

boolean, default False

hide_select

Bone is able to be selected

TYPE:

boolean, default False

inherit_scale

Specifies how the bone inherits scaling from the parent bone

- `FULL` Full – Inherit all effects of parent scaling.
- `FIX_SHEAR` Fix Shear – Inherit scaling, but remove shearing of the child in the rest orientation.
- `ALIGNED` Aligned – Rotate non-uniform parent scaling to align with the child, applying parent X scale to child X axis, and so forth.
- `AVERAGE` Average – Inherit uniform scaling representing the overall change in the volume of the parent.
- `NONE` None – Completely ignore parent scaling.
- `NONE_LEGACY` None (Legacy) – Ignore parent scaling without compensating for parent shear. Replicates the effect of disabling the original Inherit Scale checkbox..

TYPE:

enum in ['FULL', 'FIX_SHEAR', 'ALIGNED', 'AVERAGE', 'NONE', 'NONE_LEGACY'], default 'FULL'

length

Length of the bone

Length of the bone

TYPE:

float in [-inf, inf], default 0.0, (readonly)

matrix

3×3 bone matrix

TYPE:

mathutils.Matrix of 3 * 3 items in [-inf, inf], default ((0.0, 0.0, 0.0), (0.0, 0.0, 0.0), (0.0, 0.0, 0.0)), (readonly)

matrix_local

4×4 bone matrix relative to armature

TYPE:

mathutils.Matrix of 4 * 4 items in [-inf, inf], default ((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)), (readonly)

name

TYPE:

string, default “”, (never None)

parent

Parent bone (in same Armature)

TYPE:

Bone, (readonly)

select

TYPE:

boolean, default False

select_head

TYPE:

boolean, default False

select_tail

TYPE:

boolean, default False

show_wire

Bone is always displayed in wireframe regardless of viewport shading mode (useful for non-obstructive custom bone shapes)

TYPE:

boolean, default False

tail

Location of tail end of the bone relative to its parent

TYPE:

mathutils.Vector of 3 items in [-inf, inf], default (0.0, 0.0, 0.0), (readonly)

tail_local

Location of tail end of the bone relative to armature

TYPE:

mathutils.Vector of 3 items in [-inf, inf], default (0.0, 0.0, 0.0), (readonly)

tail_radius

Radius of tail of bone (for Envelope deform only)

TYPE:

float in [-inf, inf], default 0.0

use_connect

When bone has a parent, bone's head is stuck to the parent's tail

TYPE:

boolean, default False, (readonly)

use_cyclic_offset

When bone doesn't have a parent, it receives cyclic offset effects (Deprecated)

TYPE:

boolean, default False

use_deform

Enable Bone to deform geometry

TYPE:

boolean, default False

use_endroll_as_inroll

Add Roll Out of the Start Handle bone to the Roll In value

TYPE:

boolean, default False

use_envelope_multiply

When deforming bone, multiply effects of Vertex Group weights with Envelope influence

TYPE:

boolean, default False

use_inherit_rotation

Bone inherits rotation or scale from parent bone

TYPE:

boolean, default False

use_local_location

Bone location is set in local space

TYPE:

boolean, default False

use_relative_parent

Object children will use relative transform, like deform

TYPE:

boolean, default False

use_scale_easing

Multiply the final easing values by the Scale In/Out Y factors

TYPE:

TYPE:
boolean, default False

basename

The name of this bone before any '.' character

(readonly)

center

The midpoint between the head and the tail.

(readonly)

children_recursive

A list of all children from this bone.

Note

Takes `O(len(bones)**2)` time.

(readonly)

children_recursive_basename

Returns a chain of children with the same base name as this bone. Only direct chains are supported, forks caused by multiple children with matching base names will terminate the function and not be returned.

Note

Takes `O(len(bones)**2)` time.

(readonly)

parent_recursive

A list of parents, starting with the immediate parent

(readonly)

vector

The direction this bone is pointing. Utility function for (tail - head)

(readonly)

x_axis

Vector pointing down the x-axis of the bone.

(readonly)

y_axis

Vector pointing down the y-axis of the bone.

(readonly)

z_axis

Vector pointing down the z-axis of the bone.

(readonly)

evaluate_envelope(point)

Calculate bone envelope at given point

PARAMETERS:

point (`mathutils.Vector` of 3 items in [-inf, inf]) – Point, Position in 3d space to evaluate

RETURNS:

Factor, Envelope factor

RETURN TYPE:

float in [-inf, inf]

convert_local_to_pose(matrix, matrix_local, *, parent_matrix=((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)), parent_matrix_local=((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)), invert=False)

Transform a matrix from Local to Pose space (or back), taking into account options like Inherit Scale and Local Location. Unlike Object.convert_space, this uses custom rest and pose matrices provided by the caller. If the parent matrices are omitted, the bone is assumed to have no parent.

PARAMETERS:

- **matrix** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf]) – The matrix to transform
- **matrix_local** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf]) – The custom rest matrix of this bone (Bone.matrix_local)
- **parent_matrix** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf], (optional)) – The custom pose matrix of the parent bone (PoseBone.matrix)
- **parent_matrix_local** (`mathutils.Matrix` of 4 * 4 items in [-inf, inf], (optional)) – The custom rest matrix of the parent bone (Bone.matrix_local)
- **invert** (*boolean, (optional)*) – Convert from Pose to Local space

RETURNS:

The transformed matrix

RETURN TYPE:

`mathutils.Matrix` of 4 * 4 items in [-inf, inf]

This method enables conversions between Local and Pose space for bones in the middle of updating the armature without having to update dependencies after each change, by manually carrying updated matrices in a recursive walk.

```
def set_pose_matrices(obj, matrix_map):
    "Assign pose space matrices of all bones at once, ignoring constraints."

    def rec(pbone, parent_matrix):
        if pbone.name in matrix_map:
            matrix = matrix_map[pbone.name]

            # # Instead of:
            # pbone.matrix = matrix
            # bpy.context.view_layer.update()

            # Compute and assign local matrix, using the new parent matrix
            if pbone.parent:
                pbone.matrix_basis = pbone.bone.convert_local_to_pose(
                    matrix,
                    pbone.bone.matrix_local,
                    parent_matrix=parent_matrix,
                    parent_matrix_local=pbone.parent.bone.matrix_local,
                    invert=True
                )
            else:
                pbone.matrix_basis = pbone.bone.convert_local_to_pose(
                    matrix,
```

```
                    pbone.bone.matrix_local,
                    invert=True
                )
        else:
            # Compute the updated pose matrix from local and new parent matrix
            if pbone.parent:
                matrix = pbone.bone.convert_local_to_pose(
                    pbone.matrix_basis,
                    pbone.bone.matrix_local,
                    parent_matrix=parent_matrix,
                    parent_matrix_local=pbone.parent.bone.matrix_local,
                )
            else:
                matrix = pbone.bone.convert_local_to_pose(
                    pbone.matrix_basis,
                    pbone.bone.matrix_local,
                )

        # Recursively process children, passing the new matrix through
        for child in pbone.children:
            rec(child, matrix)

    # Scan all bone trees from their roots
    for pbone in obj.pose.bones:
        if not pbone.parent:
            rec(pbone, None)
```

classmethod MatrixFromAxisRoll(axis, roll)

Convert the axis + roll representation to a matrix

PARAMETERS:

- **axis** (`mathutils.Vector` of 3 items in [-inf, inf], (never None)) – The main axis of the bone (tail - head)
- **roll** (*float in [-inf, inf]*) – The roll of the bone

RETURNS:

The resulting orientation matrix

RETURN TYPE:

`mathutils.Matrix` of 3 * 3 items in [-inf, inf]

classmethod AxisRollFromMatrix(matrix, *, axis=(0.0, 0.0, 0.0))

Convert a rotational matrix to the axis + roll representation. Note that the resulting value of the roll may not be as expected if the matrix has shear or negative determinant.

PARAMETERS:

- **matrix** (`mathutils.Matrix` of 3 * 3 items in [-inf, inf], (never None)) – The orientation matrix of the bone
- **axis** (*float array of 3 items in [-inf, inf], (optional)*) – The optional override for the axis (finds closest approximation for the matrix)

RETURNS:

result_axis, The main axis of the bone, `mathutils.Vector` of 3 items in [-inf, inf]

result_roll, The roll of the bone, float in [-inf, inf]

RETURN TYPE:

(`mathutils.Vector` of 3 items in [-inf, inf], float in [-inf, inf])

parent_index(parent_test)

The same as ‘bone in other_bone.parent_recursive’ but saved generating a list.

translate(vec)

Utility function to add *vec* to the head and tail of this bone

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- bpy.context.active_bone
- bpy.context.bone
- Armature.bones
- ArmatureBones.active
- Bone.bbone_custom_handle_end
- Bone.bbone_custom_handle_start
- Bone.children
- Bone.parent
- BoneCollection.bones
- PoseBone.bone