PoseBone(bpy_struct)

base class — bpy_struct

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

Channel defining pose data for a bone in a Pose

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:

PoseBone , (readonly)

bbone_custom_handle_start

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

TYPE:

PoseBone , (readonly)

bbone_easein

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

TYPE:

float in [-inf, inf], default 0.0

bbone_easeout

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

TYPE:

float in [-inf, inf], default 0.0

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)

bone

Bone associated with this PoseBone

TYPE:

`Bone`, (readonly, never None)

child

Child of this pose bone

TYPE:

`PoseBone`, (readonly)

color

TYPE:

`BoneColor`, (readonly)

constraints

Constraints that act on this pose channel

TYPE:

`PoseBoneConstraints bpy_prop_collection` of `Constraint`, (readonly)

custom_shape

Object that defines custom display shape for this bone

TYPE:

`Object`

custom_shape_rotation_euler

Adjust the rotation of the custom shape

TYPE:

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

custom_shape_scale_xyz

Adjust the size of the custom shape

TYPE:

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

custom_shape_transform

Bone that defines the display transform of this custom shape

TYPE:

PoseBone

custom_shape_translation

Adjust the location of the custom shape

TYPE:

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

custom_shape_wire_width

Adjust the line thickness of custom shapes

TYPE:

float in [1, 16], default 0.0

head

Location of head of the channel's bone

TYPE:

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

ik_linear_weight

Weight of scale constraint for IK

TYPE:

float in [0, 1], default 0.0

ik_max_x

Maximum angles for IK Limit

TYPE:

float in [0, 3.14159], default 0.0

ik_max_y

Maximum angles for IK Limit

TYPE:

float in [0, 3.14159], default 0.0

ik_max_z

Maximum angles for IK Limit

TYPE:

float in [0, 3.14159], default 0.0

ik_min_x

Minimum angles for IK Limit

TYPE:

float in [-3.14159, 0], default 0.0

ik_min_y

Minimum angles for IK Limit

TYPE:

float in [-3.14159, 0], default 0.0

ik_min_z

Minimum angles for IK Limit

TYPE:

float in [-3.14159, 0], default 0.0

ik_rotation_weight

Weight of rotation constraint for IK

TYPE:

float in [0, 1], default 0.0

ik_stiffness_x

IK stiffness around the X axis

TYPE:

float in [0, 0.99], default 0.0

ik_stiffness_y

IK stiffness around the Y axis

TYPE:

float in [0, 0.99], default 0.0

ik_stiffness_z

IK stiffness around the Z axis

TYPE:

float in [0, 0.99], default 0.0

ik_stretch

Allow scaling of the bone for IK

TYPE:

float in [0, 1], default 0.0

is_in_ik_chain

Is part of an IK chain

TYPE:

boolean, default False, (readonly)

length

Length of the bone

TYPE:

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

location

TYPE:

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

lock_ik_x

Disallow movement around the X axis

TYPE:

boolean, default False

lock_ik_y

Disallow movement around the Y axis

TYPE:

boolean, default False

lock_ik_z

Disallow movement around the Z axis

TYPE:

boolean, default False

lock_location

Lock editing of location when transforming

TYPE:

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

lock_rotation

Lock editing of rotation when transforming

TYPE:

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

lock_rotation_w

Lock editing of 'angle' component of four-component rotations when transforming

TYPE:

boolean, default False

lock_rotations_4d

Lock editing of four component rotations by components (instead of as Eulers)

TYPE:

boolean, default False

lock_scale

Lock editing of scale when transforming

TYPE:

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

matrix

Final 4×4 matrix after constraints and drivers are applied, in the armature object space

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))

matrix_basis

Alternative access to location/scale/rotation relative to the parent and own rest bone

Alternative access to location/scale/rotation relative to the parent and own rest bone

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))

matrix_channel

4×4 matrix of the bone's location/rotation/scale channels (including animation and drivers) and the effect of bone constraints

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)

motion_path

Motion Path for this element

TYPE:

MotionPath, (readonly)

name

TYPE:

string, default "", (never None)

parent

Parent of this pose bone

TYPE:

PoseBone, (readonly)

rotation_axis_angle

Angle of Rotation for Axis-Angle rotation representation

TYPE:

float array of 4 items in [-inf, inf], default (0.0, 0.0, 1.0, 0.0)

rotation_euler

Rotation in Eulers

TYPE:

mathutils.Euler rotation of 3 items in [-inf, inf], default (0.0, 0.0, 0.0)

rotation_mode

TYPE:

enum in Object Rotation Mode Items, default 'QUATERNION'

rotation_quaternion

Rotation in Quaternions

TYPE:

mathutils.Quaternion rotation of 4 items in [-inf, inf], default (1.0, 0.0, 0.0, 0.0)

scale

TYPE:

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

tail

Location of tail of the channel's bone

TYPE:

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

use_custom_shape_bone_size

Scale the custom object by the bone length

TYPE:

boolean, default False

use_ik_limit_x

Limit movement around the X axis

TYPE:

boolean, default False

use_ik_limit_y

Limit movement around the Y axis

TYPE:

boolean, default False

use_ik_limit_z

Limit movement around the Z axis

TYPE:

boolean, default False

use_ik_linear_control

Apply channel size as IK constraint if stretching is enabled

TYPE:

boolean, default False

use_ik_rotation_control

Apply channel rotation as IK constraint

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

(readonly)

children_recursive

A list of all children from this bone.

Note

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

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]

bbone_segment_index(point)

Retrieve the index and blend factor of the B-Bone segments based on vertex position

PARAMETERS:

point (`mathutils.Vector` of 3 items in [-inf, inf]) – Point, Vertex position in armature pose space

RETURNS:

index, The index of the first segment joint affecting the point, int in [-inf, inf]

blend, The blend factor between the given point and the following joint, float in [-inf, inf]

blend_next, The blend factor between the given and the following joint, float in [-inf, inf]

RETURN TYPE:

(int in [-inf, inf], float in [-inf, inf])

bbone_segment_matrix(index, *, rest=False)

Retrieve the matrix of the joint between B-Bone segments if available

PARAMETERS:

- **index** (*int in [0, inf]*) – Index of the segment endpoint
- **rest** (*boolean, (optional)*) – Return the rest pose matrix

RETURNS:

The resulting matrix in bone local space

RETURN TYPE:

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

This example shows how to use B-Bone segment matrices to emulate deformation produced by the Armature modifier or constraint when assigned to the given bone (without Preserve Volume). The coordinates are processed in armature Pose space:

```
def bbone_deform_matrix(pose_bone, point):
    index, blend_next = pose_bone.bbone_segment_index(point)

    rest1 = pose_bone.bbone_segment_matrix(index, rest=True)
    pose1 = pose_bone.bbone_segment_matrix(index, rest=False)
    deform1 = pose1 @ rest1.inverted()

    # bbone_segment_index ensures that index + 1 is always valid
    rest2 = pose_bone.bbone_segment_matrix(index + 1, rest=True)
    pose2 = pose_bone.bbone_segment_matrix(index + 1, rest=False)
    deform2 = pose2 @ rest2.inverted()

    deform = deform1 * (1 - blend_next) + deform2 * blend_next

    return pose_bone.matrix @ deform @ pose_bone.bone.matrix_local.inverted()


# Armature modifier deforming vertices:
mesh = bpy.data.objects["Mesh"]
pose_bone = bpy.data.objects["Armature"].pose.bones["Bone"]

for vertex in mesh.data.vertices:
    vertex.co = bbone_deform_matrix(pose_bone, vertex.co) @ vertex.co

# Armature constraint modifying an object transform:
empty = bpy.data.objects["Empty"]
matrix = empty.matrix_world

empty.matrix_world = bbone_deform_matrix(pose_bone, matrix.translation) @ matrix
```

compute_bbone_handles(*, rest=False, ease=False, offsets=False)

Retrieve the vectors and rolls coming from B-Bone custom handles

PARAMETERS:

- **rest** (*boolean, (optional)*) – Return the rest pose state
- **ease** (*boolean, (optional)*) – Apply scale from ease values

- **offsets** (*boolean, (optional)*) – Apply roll and curve offsets from bone properties

RETURNS:

handle1, The direction vector of the start handle in bone local space, `mathutils.Vector` of 3 items in [-inf, inf]

roll1, Roll of the start handle, float in [-inf, inf]

handle2, The direction vector of the end handle in bone local space, `mathutils.Vector` of 3 items in [-inf, inf]

roll2, Roll of the end handle, float in [-inf, inf]

RETURN TYPE:

(`mathutils.Vector` of 3 items in [-inf, inf], float in [-inf, inf], `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.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.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.type_recast
- bpy_struct.values

References

- bpy.context.active_pose_bone
- bpy.context.pose_bone
- bpy.context.selected_pose_bones
- bpy.context.selected_pose_bones_from_active_object
- bpy.context.visible_pose_bones
- Object.convert_space
- Pose.bones
- PoseBone.bbone_custom_handle_end
- PoseBone.bbone_custom_handle_start
- PoseBone.child
- PoseBone.custom_shape_transform
- PoseBone.parent