Guidelines for Character Physics Authoring

2004-09-08

In order for characters to have physicalized skeleton their model files should contain additional information. For characters, cgf file for LOD0 (main LOD) should contain physics for alive body (that is used for hit determination and bullet impact swaying animation), and cgf for LOD1 should contain dead body skeleton. Bones should have meshes that loosely follow shape of the character parts assigned to them. These meshes will be used for collision detection and mass distribution computation, thus they should be relatively simple and non-degenerate (closed, w/o degenerate triangles, etc). These meshes should also have *one smoothing group* so that they are exported as continuous surfaces rather than separated faces.

Besides mesh information, physics should also know joint limits between bones and their parents. These limits are extracted as IK rotational limits from a node named like the corresponding bone plus " Phys". Geometries and positions of these nodes are not taken into account, only IK limits (important: in order for limit to be exported "limited" option should be checked for the corresponding angle). However, having these nodes resembling main skeleton helps to edit and preview angles. It is important that IK Y angle limits lie in (-90,90) range, and do not come too close to its ends. If it's not the case, an additional parent node for bone can be created, and IK rotational limits will be relative to it. The node should be named bone name + " Phys ParentFrame". It should be inserted into the "phys" hierarchy between the child node and it's original physicalized parent. Generally it is advised to use ParentFrames when coordinate frame changes abruptly from parent to child, even if Y angle limit is in –90..90 range.

The recommended sequence of actions is as follows:

- A decision on what bones should be physicalized is made;
- The geometries of these bones are edited to comprise a valid skeleton. For instance, when a marginal bone (palm with fingers) is decided to be excluded from the physicalization, its shape should be included into its last physicalized parent (usually a forearm). Intermediate bones excluded from physicalization (i.e., the ones that have physicalized bones both above and below in the hierarchy) are treated in the code as attached to their last physicalized *parent* (and are oriented as in character default position wrt it), thus this parent's geometry should be modified to encompass them in their default position;
- Entire biped is copied to a new hierarchy and all bones that are to be physicalized are renamed to "*original_bone_name* Phys" and have their IK limits set up. Names of the non-physicalized bones don't matter, as long as they don't follow this naming pattern (so it's ok to just keep them as they are after copying; however, in order for limits of physicalized bones to represent actual values, *these bones should be removed from the hierarchy*, thus deleting them altogether may prove convenient). Note that if a bone has a valid Phys name, but has all its limits at 0, it's treated as non-physicalized, *so some limits should be entered without checking "limited" option* (this is particularly the case with pelvis bone; another important thing about pelvis is *not* to check "limited" option for any of its angles for dead body). ParentFrames are injected into phys hierarchy where necessary.

Some guidelines for designing physical skeletons:

- The recommended number of bones in a dead body skeleton is 10-14, and in alive skeleton 20-23 (less is ok);
- Geometries have their shapes and masses taken into account, so skeleton should be physically valid in order to behave valid;
- Heavy-light-heavy patterns should be avoided (this mostly matters for dead bodies). A typical example from biped is spine-neck-clavicle-upper arm connection with heavy spine, light neck and clavicle, and relatively heavy upper arm. For dead bodies it's recommended to exclude neck (not necessarily neck1,2, though) and clavicles;

- Geometries of dead skeleton should be simpler than that of alive one (if the latter is not yet simple enough), around 12-35 polygons (eventually they may be replaced with primitives), and should not have sharp corners near joints (it's better to have them narrow near the joint and wider in the middle part);
- Avoid using joints with all 3 axes locked whenever possible, because they slow down the simulation w/o doing any useful work

Alive skeleton specific

Stiffness of an angular spring at a joint can be adjusted via "Spring Tension" parameter. A value of 1 means acceleration of 1 radian/second^2 (1 radian ≈ 57 degrees). Damping should also be set to some reasonable value (1.0 corresponds to fully dumped oscillations for an isolated joint, it's a recommended default value).

If a bone is designed to be fully controlled by physics, it (or it's phys alias) should contain the word "physical" in its user-defined properties. In this case spring is applied as a force to a bone. Otherwise spring tension is treated like a hint of what acceleration character muscles should achieve at this joint (there are 2 simulation modes for this parameter, they can be dynamically switched during the game). If a bone is "physical", one can specify whether gravity should be applied to it by adding "gravity" keyword to the user-defined properties.

End alive skeleton specific

Note: if .max file does not contain " phys" counterpart of the main skeleton, all bones are exported as physical (i.e., together with their geometries). To avoid exporting a bone in this situation, one should put the word "nonphysical" in its user-defined properties.

Ropes

In order to have some character bones simulated using rope physics one has to name the corresponding bones '*ropename*[space]*ropesegmentname*'. *ropename* can be any name starting with 'rope' (internally all bones that share one rope name are united into one rope), and *ropesegment* can be any text (internally it is discarded, it should be used just to avoid naming collisions among bones that belong to one rope). All segments in one rope should have approximately equal length. Note that a rope bone should never be the leaf bone (w/o children), since the bone after the last rope bone is used to provide terminal rope vertex. Also note that currently ropes cannot connect 2 physicalized parts of a skeleton; they can be either 'roots' (ex: hanging light cord), or 'leaves' (ex: character's ponytail or braid).

In script one should first create some physical entity, then load and physicalize the character model. By default all bones are physicalized (unless " Phys" counterpart skeleton is present, which will most likely be the case only for characters). Note, however, that rope bones normally should not be a part of main character physics (which for hanging lamp will most likely be a rigid body), thus *they should have 'nonphysical' keyword in their user-defined properties* (as well as other bones that do not represent physical geometry). During the physicalization ropes are created and stored internally in the character system. The 1st end of the rope is the one that is closer to the root bone. Both the 1st and the 2nd ends are automatically tied to character physical entity if they are connected to its physicalized parts (not necessarily directly).

In order to alter parameters of rope physics one should call SetCharacterPhysics(slot_number,"ropename", PHYSICSPARAM_ROPE,param_table). param_table can contain the following fields:

- *length* - full length of the rope;
- *entity_id_1* - entity id the 1st rope end is tied to (-1 if statically tied world, -2 if not tied at all);
- *entity_part_id_1* - entity part id the 1st rope end is tied to
- *end1* (vector) – the 1st end attachment point (if unspecified, it is assumed to be current end point position);
- *end2* (vector) – the 2nd end attachment point;
- *check_collisions* – 1 if the rope should collide with the environment (0 otherwise);
- *coll_dist* – rope thickness for collision detection.

Currently ropes cannot apply forces or otherwise alter the behavior of the objects they are colliding with, save the objects they are directly tied to (and tied objects receive impulses only when the rope is fully strained along a straight line).

A helpful tip: 'stiff' ropes can be mimicked by having few rope segments (bones) and setting up mesh skinning so that visually the rope bends smoothly rather than abruptly at joint points.

