Subject: Human Animations System
Posted by Jerad2142 on Wed, 31 Dec 2014 15:40:57 GMT
View Forum Message <> Reply to Message

So as some of you know I've made full custom animation sets for cats, bears, and deer. However, I've run into an issue which has me stumped.

As you may know from watching AI, sometimes AI will look left or right at where they heard a sound from or things like that (basically their head will turn where they are looking).

My issue is I have yet to find an animation that they use to blend in to make this look left and look right happen. They have animation sets for tilting the player's 3rd person gun up and down when they aim up and down. So it seems like it would be possible they blend an animation in; however, I suspect its actually done in the engine code itself.

If anyone has stumbled across at least 2 animations where the only thing that was changed on the t-pos was the head was turned left or right let me know. Or if anyone has stumped across engine code that would be responsible for this please let me know as well.

Thanks

Subject: Re: Human Animations System
Posted by danpaul88 on Wed, 31 Dec 2014 16:36:39 GMT
View Forum Message <> Reply to Message

Theres a bunch of code in soldier.cpp that rotates the head to "look" at targets.

Toggle Spoiler

```
Vector3 desired_head_rotation( 0,0,0 );
if ( !returning ) {
if ( HeadLookAngle.Length() > 0.001f ) {
HeadLookAngleTimer -= TimeManager::Get_Frame_Seconds();
if ( HeadLookAngleTimer < 0 ) {
HeadLookAngle = Vector3(
FreeRandom.Get_Float( -HEAD_TURN_LIMIT, HEAD_TURN_LIMIT ),
FreeRandom.Get_Float( -HEAD_TILT_LIMIT, HEAD_TILT_LIMIT ) ,0);
// Debug_Say(( "New Look Turn %f Tilt %f\n", RAD_TO_DEG(HeadLookAngle.X),
RAD_TO_DEG(HeadLookAngle.Y) ));
HeadLookAngleTimer = FreeRandom.Get_Float( 2, 5 );
}
desired_head_rotation = HeadLookAngle;
} else {
//
// Release the captured bone, this will have the effect of causing the Get_Bone_Transform ()
// methods to return the un-modified (due to being controlled) transform of the bone.
//
/*if ( Peek_Model()->Is_Bone_Captured( head_bone ) ) {
```

```
Peek_Model()->Release_Bone( head_bone );
}*/
///
// Get the transform that has been used to modify the head bone...
//
const HTreeClass *htree = Peek_Model()->Get_HTree();
WWASSERT( htree != NULL );
Matrix3D bone_control_tm( 1 );
htree->Get_Bone_Control( head_bone, bone_control_tm );
//
// Get the inverse of the head-bone transform
//
Matrix3D inv_bone_control_tm;
bone_control_tm.Get_Orthogonal_Inverse (inv_bone_control_tm);
//
// Get the head to world and neck to world transforms
//
Matrix3D cur_head = Peek_Model()->Get_Bone_Transform( head_bone );
Matrix3D cur_neck = Peek_Model()->Get_Bone_Transform( neck_bone );
//
// Strip off the control transform from last frame
//
cur_head = cur_head * inv_bone_control_tm;
//
// Get the world to neck transform
//
Matrix3D world_to_neck_tm;
cur_neck.Get_Orthogonal_Inverse (world_to_neck_tm);
//
// Build a head to neck transform
//
Matrix3D head_to_neck_tm = world_to_neck_tm * cur_head;
//
// Get the target relative to the head
//
Vector3 relative_head_target;
Matrix3D::Inverse_Transform_Vector( cur_head, HeadLookTarget, &relative_head_target );
//
// Determine the 'twist' and lookup/down angles.
// Note: Currently in the head bone coordinate system, the X axis is the same
// as the Z axis in object space, the Y axis is the same as the X axis in object space,
// and the Z axis is the same as the Y axis in object space.
//
desired_head_rotation.X = WWMath::Atan2( relative_head_target.Z, relative_head_target.Y );
desired_head_rotation.Z = -WWMath::Fast_Asin( relative_head_target.X /
relative_head_target.Length() );
desired_head_rotation.Y = 0;
//
```

```
// Determine how far to allow the character to turn and tilt his/her head.
// These boundaries are based on the "absolute" amount the person can turn
// their head, this has to take into consideration the amount that the current
// animation is turning the head and the amount we need to turn to look at the target.
//
Vector3 temp_vec = head_to_neck_tm.Get_Y_Vector ();
float curr_rot_x = ::atan2 (temp_vec.Z, temp_vec.Y);
float curr_rot_z = ::atan2 (temp_vec.X, temp_vec.Y);
float min_twist = ((-HEAD_TURN_LIMIT) - curr_rot_x);
float max_twist = (HEAD_TURN_LIMIT - curr_rot_x);
float min_tilt = ((-HEAD_TILT_LIMIT) - curr_rot_z);
float max_tilt = (HEAD_TILT_LIMIT - curr_rot_z);
//
// Clamp the rotations
//
desired_head_rotation.X = WWMath::Clamp( desired_head_rotation.X, min_twist, max_twist);
desired_head_rotation.Z = WWMath::Clamp( desired_head_rotation.Z, min_tilt, max_tilt);
}
}
#define HEAD_TURN_RATE (DEG_TO_RAD( 360 )/2)
#define HEAD_TILT_RATE (DEG_TO_RAD( 180 )/2)
float max_turn = HEAD_TURN_RATE * TimeManager::Get_Frame_Seconds();
float max_tilt = HEAD_TILT_RATE * TimeManager::Get_Frame_Seconds();
HeadRotation.X += WWMath::Clamp( (desired_head_rotation.X - HeadRotation.X), -max_turn,
max_turn );
HeadRotation.Z += WWMath::Clamp( (desired_head_rotation.Z - HeadRotation.Z), -max_tilt,
max_tilt );
Matrix3D head(1);
head.Rotate_X( HeadRotation.X );
head.Rotate_Z( HeadRotation.Z );
if ( !Peek_Model()->Is_Bone_Captured( head_bone ) ) {
Peek_Model()->Capture_Bone( head_bone );
}
WWASSERT( Peek_Model()->Is_Bone_Captured( head_bone ) );
if ( Peek_Model()->Is_Bone_Captured( head_bone ) ) {
Peek_Model()->Control_Bone( head_bone, head );
}
HeadRotation.Z = 0;
if ( returning && HeadRotation.Length() > 0.001f ) {
HeadLookDuration = 0.0001f; // maybe done next time...
}
```

Subject: Re: Human Animations System
Posted by Jerad2142 on Wed, 31 Dec 2014 17:17:00 GMT
View Forum Message <> Reply to Message

Well I assume it must be hard coded by bone count, beings my deer rotate in the middle of their

back at c spine2 (which makes their forelegs lift off the ground as they twist to the left or right).

Subject: Re: Human Animations System
Posted by Jerad2142 on Thu, 01 Jan 2015 18:11:21 GMT
View Forum Message <> Reply to Message

Added an update so that skeletons with a NOLOOK bone will be excluded by the engine's attempts to turn the head to look at locations.