Blended Inverse Kinematics: Delta3D System Utilization

Michael Guerrero, Chris Darken PhD
MOVES Institute, Naval Postgraduate School
Monterey, CA
mjguerre@nps.edu, cjdarken@nps.edu

ABSTRACT

Traditional inverse kinematics systems are riddled with issues that make their use in real-time simulations prohibitive. Foremost, the computational costs associated with these methods are too high to make their widespread use practical. Furthermore, their usage typically results in the synthesis of animations that fail to impart the sense of weight and timing that would be present in either motion captured or artist created forward kinematic animation. This is a byproduct of using a mathematical technique to solve an artistic problem.

A new method we have developed succeeds in overcoming these shortcomings. By using a database of predefined animation poses, a new animation can be derived to achieve the desired pose. At its simplest, the technique can be used to manually control a character's gaze for simulating environmental awareness as well as directing a character's gun to point in the desired direction for aiming or shooting. It does all of this and more without burdening the CPU and can be easily enhanced using hardware skinning for optimal performance.

ABOUT THE AUTHORS

Michael Guerrero is a research associate at the MOVES Institute of the Naval Postgraduate School in Monterey, California. He specializes in graphics and simulation technologies and has contributed to commercial games on both the Nintendo DS and the PC and is now currently pushing the state of the art in simulation as a co-developer of the popular open source game engine Delta3D.

Dr. Chris Darken is currently an associate professor of computer science at the Naval Postgraduate School in Monterey, California, where he collaborates intensively with the MOVES Institute. He was previously a project manager at Siemens Corporate Research in Princeton, New Jersey, and was on the programming team of what was perhaps the first 3D massively multiplayer online game, Meridian 59. He received his Ph.D. in electrical engineering from Yale University in 1993.

Blended Inverse Kinematics: Delta3D System Utilization

Michael Guerrero, Chris Darken PhD
MOVES Institute, Naval Postgraduate School
Monterey, CA
mjguerre@nps.edu, cjdarken@nps.edu

BACKGROUND

Modern simulations frequently require avatars to assume a variety of different poses that are derived from a set of animations. These animations rigidly define all of the possible ways that the avatar can move. For instance, a basic soldier can be expected to have animations allowing them to "walk", "run", or "aim". This provides three different ways to move which may seem sufficient for providing the basic functionality, and as far as walking and running are concerned, it is sufficient. However, an "aim" animation only allows an avatar to shoot in one specific direction from where they are currently oriented. This is rarely useful since a target is unlikely to be positioned exactly where the avatar's gun happens to be oriented in the animation.

An initial effort to overcome this might be to create more "aim" animations that point in different directions. But how many directions is enough? The real answer to that depends on how much accuracy the application requires and how far away the target is. The useful answer is that a sufficient number may be in the hundreds if not thousands of separate directions. This is unacceptable due to the capacity required to store the animations as well as the time needed to create that many to begin with. A second approach might be to rotate all or part of the avatar so that they can change their aim direction. The main issue with this is the poor aesthetic of seeing an avatar unnaturally rotate to face the desired direction.

A more flexible approach is needed. The approach must operate without the combinatorial explosion of creating discrete animations and must also result in movement that appears natural.

One of the most commonly employed class of methods for dynamically posing an avatar (such as aiming) are those that solve the inverse kinematics (IK) problem (Watt & Watt 1998). In skeletal animation, IK is the process of determining the positions and orientations of the joints in a skeleton with the intent of specifically moving one specific bone (often called the *end-effector*) towards a desired position and/or orientation. For example, if an artist wants to create an animation of an avatar to pushing button, they could move the tip of the index finger out in front of the avatar where the button might be. In the process of doing so, the application determines the position and orientation of all of the joints between the end of the finger and torso (knuckles, wrist, elbow, shoulder, etc.), often referred to as the kinematic *chain* (Lander 1998).

TRADITIONAL SOLUTIONS

A multitude of solutions have been created for IK (along with dozens of variations and enhancements), but most of these have been primarily developed for applications that can tolerate their shortcomings such as digital content creation tools like 3D Studio Max and Maya. These IK solutions are not suitable for real time applications for two main reasons. The first is that IK can produce unnatural poses (Hecker 2002), but in these tools the artist can always manually adjust these poses to meet their needs before they are deployed in a live simulation. This manual adjustment is something that the artist will not have the luxury of doing if the IK were to be applied on-the-fly within the simulation. This is likely to result in the user seeing unnaturally posed avatars and will negatively impact the realism of the game, which is the reason for adding avatar motion to begin with. The second drawback is that the necessary IK computations may take more CPU time than the application can afford to budget. In the relatively forgiving environment of a content creation tool, an unstable frame rate may be acceptable, but the same does not hold true for real-time applications which require a consistently high frame rate.

BLENDED INVERSE KINEMATICS

Our solution to solving the problem utilizes skeletal animation blending and provides an interesting alternative for generating inverse kinematic poses. Given an initial set of poses, it is possible to generate a new set containing all possible blends of the initial set. Consider a set consisting of the three poses: gazing to the right, down and right, and down and left (Figure 1). It would be possible to generate a new gaze pose for

anywhere in between by blending together a percentage of each of the original poses. As an example, two possible poses generated from the blend of the initial set are shown below (Figure 2).

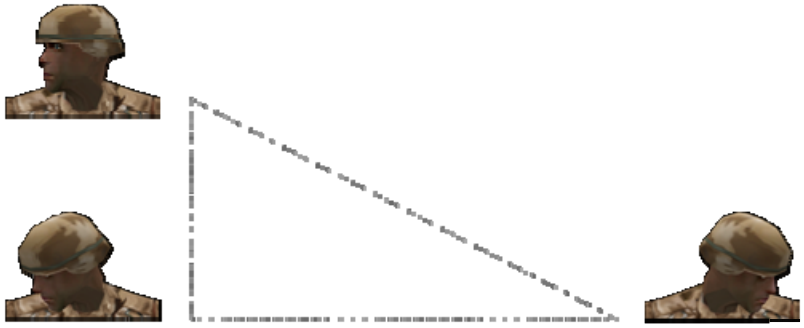

Figure 1. Space defined by a set of poses

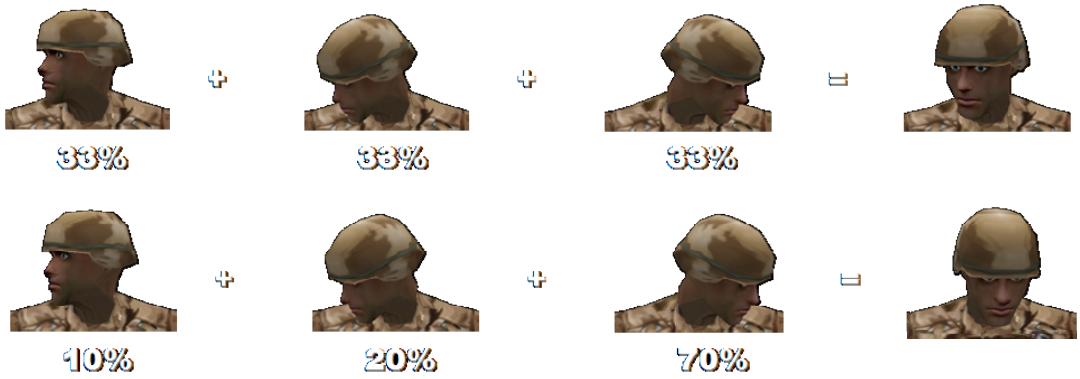

Figure 2. Blending generated poses

Using skeletal blending for inverse kinematics has several advantages that make it an appealing alternative to traditional techniques (Edsall 2003). Foremost, it is computationally inexpensive. A minimal amount of processing is required to determine the correct weighting values (i.e. the percentages) needed to achieve the desired pose. Once the weights have been determined, applying a blend of the poses to the avatar skeleton is also a relatively cheap task, especially since it is likely that some blends would be applied to an avatar regardless of whether or not the inverse kinematic system is used. Furthermore the process is artist driven, as the initial set of poses will be guaranteed to look as desired since they were manually created by an artist rather than by a convoluted series of computations as would be required using traditional IK techniques (Edsall 2003). As long as the set of poses were created to blend well with each other, the poses that are generated from the initial set should also preserve the same artistic quality.

Caveats

The process of determining the blend weights to generate a desired pose is essentially a linear approximation of a non linear space. As such, it is subject to a certain degree of error. The pose mesh provides this linear mapping to the nonlinear result of blending different poses on the avatar skeleton. In other words, the azimuth and elevation values from the pose mesh serve as a guide to what the orientation of the end-effector will be as a result of using a blend at that location in the pose mesh. In many cases the resulting error is negligible but in some use cases, like the aiming of a gun, a higher degree of precision may be required to attain satisfactory results. Fortunately, this inadequacy can be overcome using methods described later.

POSE MESHES

Organizing the set of artist produced poses can be done by constructing a triangulated pose mesh. In this mesh, poses are represented by vertices and each triangle enclosed area represents the poses that can be generated using its surrounding vertices (poses). For example, a set of six poses could be generated to cover all the ways that an avatar might turn their head. Figure 3 shows an example of what this mesh might look like.

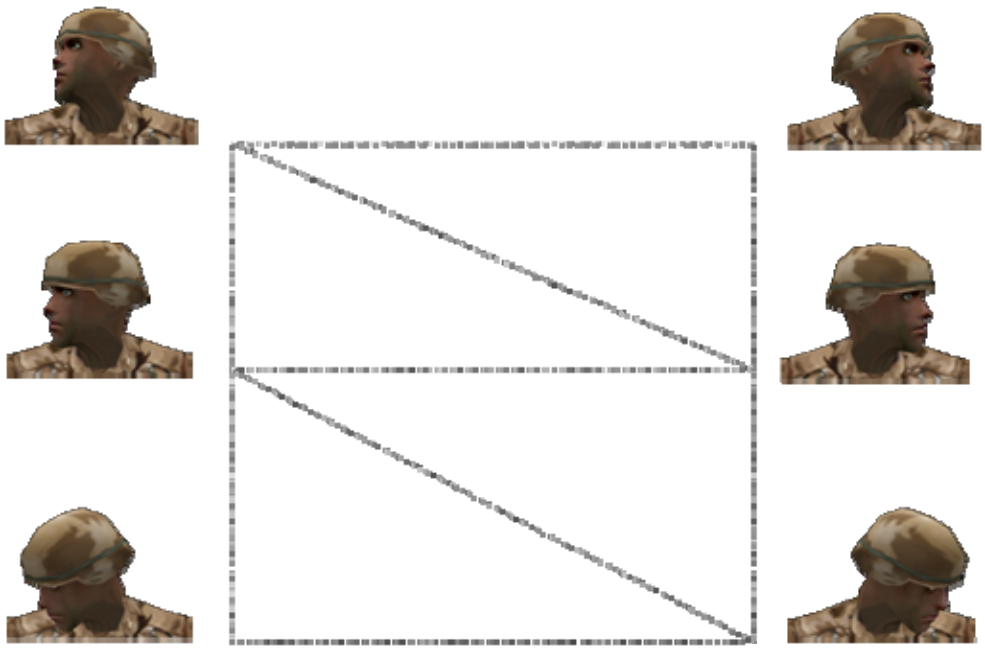

Figure 3. Pose mesh for gaze

Using these six poses, a new "gaze" pose can be generated to orient the head in the desired direction. The six poses used above were created to represent the full range of motion for the human head.

Triangulation

Organizing the poses into non overlapping triangles has several benefits. First, it allows poses that are the most similar to each to other to be grouped together. This is important in order to preserve the quality of synthesized poses and minimize the degree of error to which they deviate from the desired pose. Also using 3 poses provides a large range of possible poses that can be generated by the blend.

DELTA3D SUPPORT

Although the theory behind this system can be applied to any game or simulation engine, it was necessary to choose one to build and test our ideas. We used the popular open source game engine Delta3D which provided us with a robust Cal3D based character animation library. Our system has been integrated and made available to the public in version 2.1 of Delta3D. In addition to the basic system, the Delta3D Animation Viewer tool has been augmented to visualize and interact with the pose meshes in real-time. This makes the system much more accessible to the artists on whom this system depends on for providing the source animations and poses.

Animation Viewer

The animation viewer is a convenient tool for analyzing skeletal meshes and their accompanying animations. It reads in an XML file that specifies all of the meshes, materials, and animations that belong to a given skeletal mesh and displays them.

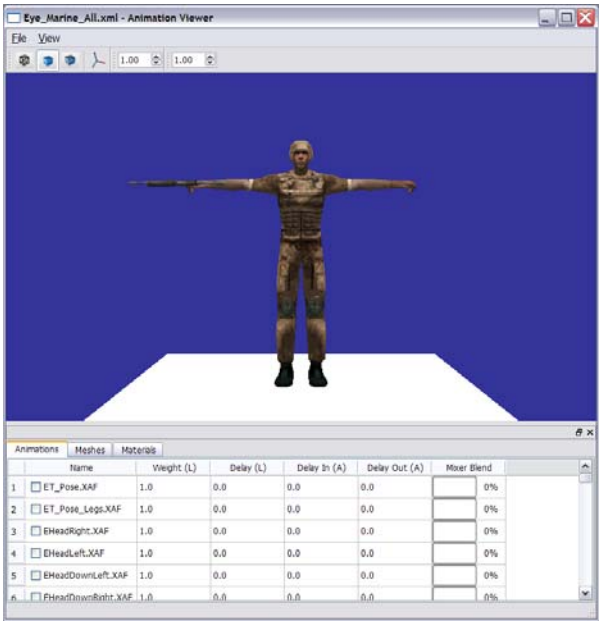

Figure 4. Delta3D Animation Viewer

Pose Mesh Viewer Extension

The tool has also been outfitted with a system that facilitates the usage of the blended inverse kinematics system. Upon the loading of a character XML file, if the file specifies a pose mesh file, the associated data will be automatically loaded in and made available for manipulation through a special panel that appears after loading.

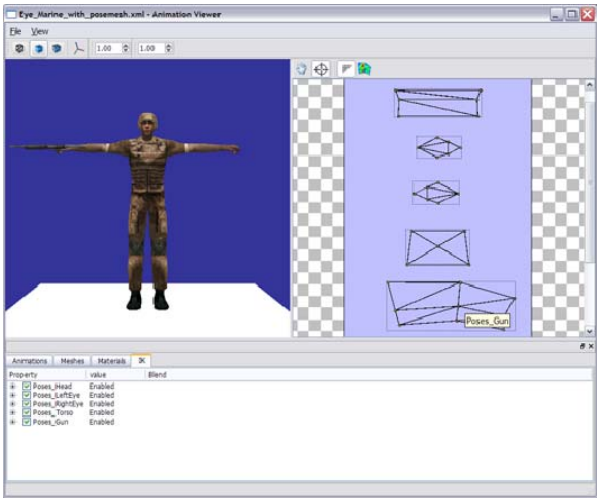

Figure 5. Viewer with IK panel

This panel provides some convenient visualization features. A representation of the pose meshes are displayed with yellow circles as vertices and black lines (edges) connecting them to indicate how they are triangulated. Below is an example of a mesh constructed from the gaze poses mentioned earlier that the viewer might show.

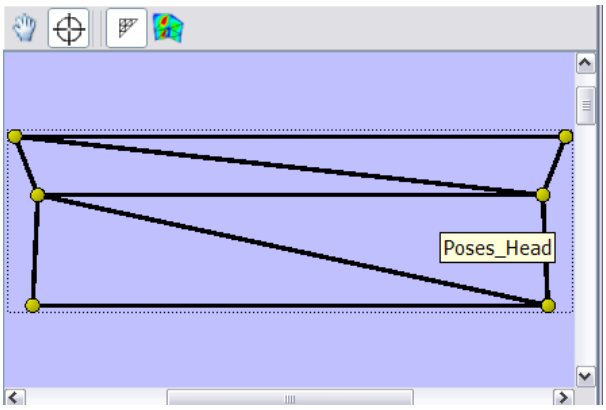

Figure 6. Viewer representation of gaze mesh

Each pose in this example mesh has been deconstructed into an azimuth and elevation value for the direction of the end-effector bone. Since this data is composed of only two values, it easily maps to a traditional two dimensional visualization as shown above.

Interface

Figure 7. IK Panel Toolbar

The IK panel provides a toolbar with four buttons that affect either mouse control or the way the pose meshes are displayed. Using this set of tools enables the user to perform the following useful actions: translate the pose meshes around the panel, pick a location in a pose mesh and view the resulting avatar blend that corresponds to the picked location (figure 8), toggle the display of the pose mesh edges, and toggle the visualization of the error samples taken across each pose mesh.

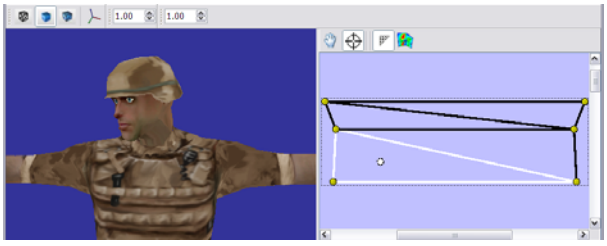

Figure 8. Example of picking a point in blend mode

Error Display Visualization

The rightmost button may be toggled to show the error visualization. It is color coded using a heat map where cooler colors (e.g., blue and green) show little or no error and warmer colors (e.g., yellow and red) show increasing amounts. This provides an intuitive "at-a-glance" view of potential problems in the constructed mesh.

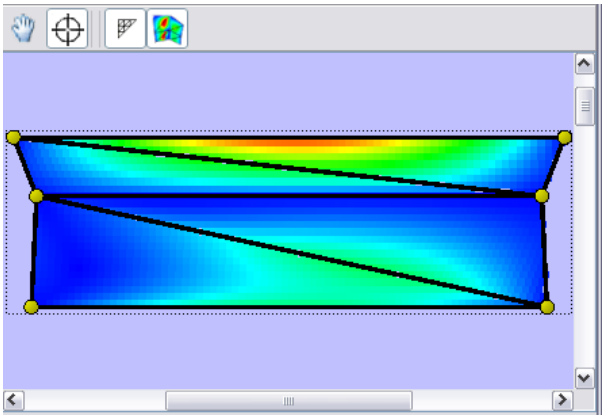

Figure 9. Viewer representation of gaze mesh

Pose Mesh File Specification

Pose meshes are specified via XML files. They contain a list of triangulated poses (the mesh), which bone is designated as the end-effector, and the orientation of the bone when it is considered to be facing forward. An example of what this file might look like for the mesh in Figure 3 is shown to the right (figure 10).

```
<PoseMeshFile>
	<PoseMesh name="Poses_Head" bone="Bip01 Head" forward="0.0 1.0 0.0">
		<Triangle>
			<Animation>EHeadDownRight.XAF</Animation>
			<Animation>EHeadDownLeft.XAF</Animation>
			<Animation>EHeadRight.XAF</Animation>
		</Triangle>
		<Triangle>
			<Animation>EHeadRight.XAF</Animation>
			<Animation>EHeadDownLeft.XAF</Animation>
			<Animation>EHeadLeft.XAF</Animation>
		</Triangle>
		<Triangle>
			<Animation>EHeadRight.XAF</Animation>
			<Animation>EHeadLeft.XAF</Animation>
			<Animation>EHeadUpRight.XAF</Animation>
		</Triangle>
		<Triangle>
			<Animation>EHeadUpRight.XAF</Animation>
			<Animation>EHeadLeft.XAF</Animation>
			<Animation>EHeadUpLeft.XAF</Animation>
		</Triangle>
	</PoseMesh>
</PoseMeshFile>
```

Figure 10. Example pose mesh file

Context Options

Right clicking a point inside the rectangular border of the pose mesh will bring up a context menu which presents the following options: "zoom extents", "disable", and "clear blend". Selecting "zoom extents" will cause the mesh that was clicked over to occupy as much space in the panel as possible, thus allowing you to see it better. Selecting "disable" will gray out the mesh and make it unable to be interacted with. Selecting "clear blend" will remove the current mesh point (shown as a white circle) from the pose mesh and clear any of the mesh's contributing animations from the character in the viewer. This is useful for isolating only the meshes you're interested in testing without being affected by other meshes that are currently applied.

IK Properties Tab

In addition to tabs already provided by the animation viewer consisting of "Animation", "Meshes", and "Materials", the pose mesh extension creates an additional tab titled "IK". An example of the data displayed for each mesh is shown below using the previously discussed "gaze" mesh.

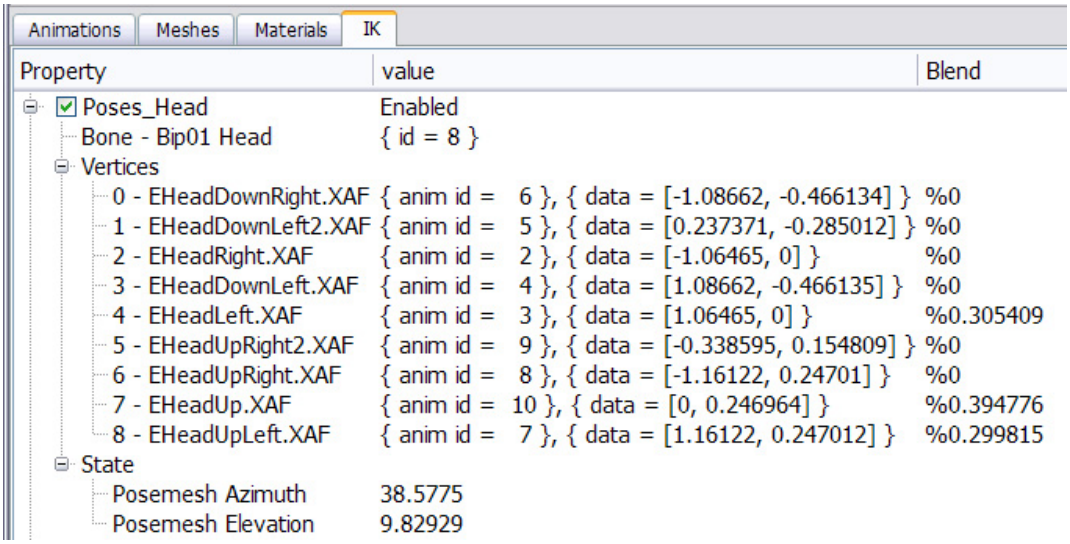

Figure 11. Pose mesh properties tab

The "IK" tab provides detailed information about every loaded pose mesh that is displayed in the pose mesh panel. At the top level, the name of the mesh is displayed along with a checkbox that allows toggling whether the mesh is enabled or disable (this functionality is also provided as a context option by right clicking). At the second level we have the items "Bone", "Vertices", and "State". "Bone" displays the name and ID of the bone in the mesh that represents the end effector. "Vertices" which in this context corresponds to poses, displays their name, ID, data consisting of azimuth and elevation of the "Bone" (end-effector), and the percentage of the vertex (pose) that is currently applied to the avatar. The "State" property consists of the combined azimuth and elevation values that result from currently blended vertices. In figure 11, the "State" shows that the head bone is currently rotated 38.5775 degrees azimuth and 9.82929 degrees elevation as a result of the currently applied blends coming from vertices 4, 7, and 8.

ERROR CORRECTION

Depending upon the application's needs, the degree of error present in the blended inverse kinematic poses may be unacceptable. For instance, the mesh shown in figure 9 shows some regions color coded red. By default, red represents error values of 7.5 degrees or greater from the intended azimuth and elevation angles whereas blue represents 0 degrees of error.

In assessing error, it is important to realize that for every vertex location in the mesh, the error is 0. Error only starts to accumulate as the distance from the vertices increases (see figure 9). This realization leads to one effective strategy for mitigating error, add more vertices!

All of the tools discussed so far work well together in determining where to place additional vertices in the pose mesh. We start by loading in all of our meshes and turning on the error visualization. From here, either identify the point with the greatest error or pick a central location from which error seems to abound. Once you have spotted this point, left click on.

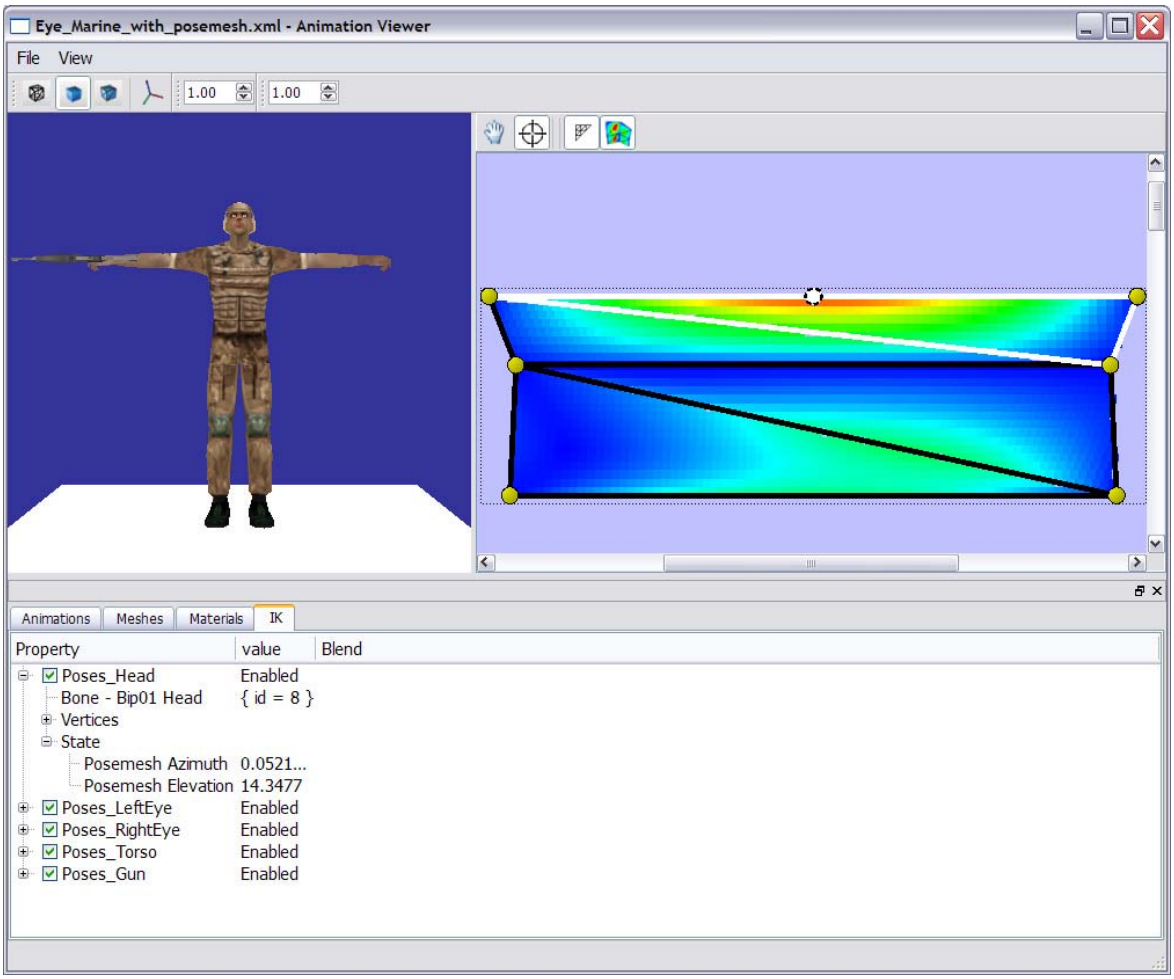

Figure12. Error assessment

Once the error hotspot has been picked, we can determine the azimuth and elevation values corresponding to this location. In this example we are looking at the "gaze" mesh so the required information can be found by looking at its corresponding properties in the "IK" tab (in this case, "Poses_Head/State"). This information can be used to generate a new pose where the end-effector bone is oriented in the same direction as the one where the error hotspot points.

After generating a new pose, the existing mesh must be retriangulated to be properly incorporated. This can easily be done by editing the pose mesh's XML file (see figure 10). Figure 13 shows the result of the addition of this single pose and the drastic reduction of error that occurs as a result.

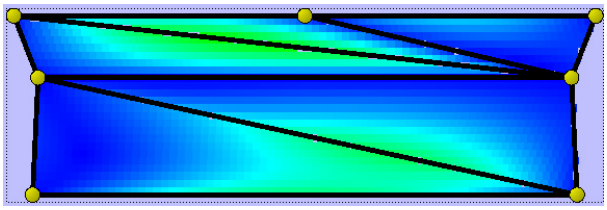

Figure13. Adding a new pose to the mesh

This same approach can be performed iteratively until an acceptable level of error has been attained. Figure 14 shows the results in applying this method a couple through a couple more iterations.

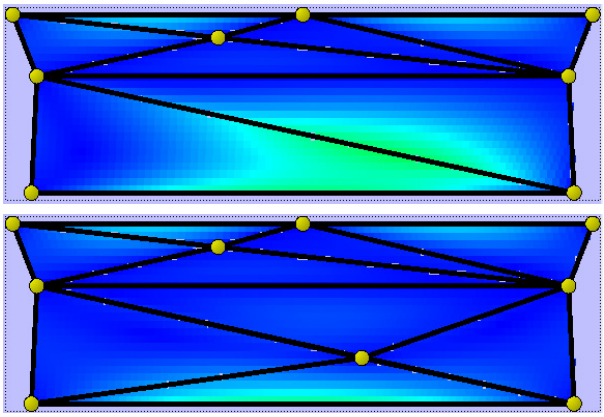

Figure14. Error refinement iterations

CONCLUSION

In this paper, we have demonstrated a unique system that utilizes skeletal animation blending to provide a solution to the inverse kinematics problem. The strengths of the method lie in its ability to produce aesthetically pleasing poses without burdening the CPU. We have also shown one possible method for overcoming its main weakness of not generating poses close enough to the ones desired by adding more strategically placed poses (vertices) to the pose mesh. The system has been successfully deployed in applications to control the direction an avatar gazes as well as the direction of aiming a two handed weapon for acquiring targets. These two cases are simple examples of where the system has proven useful thus far. It is in no way limited by them but they serve as the foundation for the realization of more complex utilization.

ACKNOWLEDGEMENTS

Thanks to the Delta3D team at the MOVES Institute without whose time and support we could have not have produced a tool suite with this level of refinement.

REFERENCES

Edsall, J. (2003). *Achieving Inverse Kinematics and More.* Retrieved June 22, 2008, from http://www.gamasutra.com/features/20030704/edsall_01.shtml

Lander, J. (1998). *Oh My God, I Inverted Kine.* Game Developer Magazine. Retrieved June 22, 2008, from http://graphics.cs.cmu.edu/nsp/course/15-464/Spring07/assignments/jlander_gamedev_sept98.pdf

Watt, A., & Watt, M. (1998). *Advanced Animation and Rendering Techniques*, New York: ACM Press.

Hecker, C. (2002). *My Adventures with Inverse Kinematics.* Game Developer's Conference Proceedings, Retrieved June 22, 2008, from http://chrishecker.com/images/7/76/Gdc2002-ik.ppt