SIKAAAGS: A CS184 Final Project: Spring 2009

by Loring Scotty Hoag and Zelam Ngo.

Login: cs184-ai and cs184-bb

The SIKAAAGS project had three main goals. The first was to use the GLSL hardware shading language to produce several impressive effects beyond just the standard Phong shading model. Thus, the "S" in SIKAAAGS stands for "Shaders". The most notable effect is the highly polished cartoon/cell shader. When activated, the entire scene renders like a cartoon with flattened colors and dark, thick outlines. Each model has a base RGB material color that is then scaled by the intensity of the light that hits it. The scaling is determined by taking the dot product of the model's surface normal with the direction of incomming light. The value produced is then used as a coordinate for a 1-dimensional texture map which encodes RGB scalers for the material color. Larger dot products produce larger scaler values, and thus brighter colors. Smaller dot products produce smaller scaler values resulting in darker colors. This has the effect of producing various "bands" of similar colors like one would see on a cartoon character. The number of bands can be set seperatly for each model. Thus, some models may be given two-toned effects while others can have a far greater number of bands. Since the scalers affect RGB seperatly, it is also possible to shade a model in colors vastly different from its base material color if so desired.

A feature which may go unnoticed is that the toon shader supports varrying line width for the outlines. In artistic drawings and painting, line width is important for conveying wieght, direction, and curvature. Non-varying line width by comparison looks flat and less interesting. The varying width was achieved by using two common algorithms simultaneously. First, lines were drawn wherever front and back facing polygons shared edges. This was done by rendering the model twice. On the first pass, the front-facing polygons are culled away and the remaining back-facing polygons are rendered as a wire

frame mesh. On the second pass, the model is rendered normally with back-facing polygons culled away and front-facing polygons filled in. The wireframe bleeds out over the edges of the second render, producing a thin dark line of uniform length around the model. The second method involves a special case of color banding. If the dot product between the light direction and the surface normal is sufficiently close to zero, then the pixel is rendered black instead of the material color. This creates thick dark bands around darker areas on the model, but does not garuntee continuous outlines around the entire model. Used together, these two techniques give the model stylistic lines of varying width.

There are several other graphics effects in use. By saturating the material color, the models can vaguely resemble that of a watercolored painting. This effect is done by using the light intensity to inversely scale a vector with coordinates at <1, 1, 1>. The result is added to the color previously computed at the current pixel to saturate it. Another effect is texture blending. By specifying a second texture map, the two maps can fade from one to the other through linear interpolation. This can be used to create effects like camoflauge in a video game where a player model can fade to match a background texture to conceal himself or herself from other players. Also of note is that texture maps are swapped out with greyscale equivalents when toon shading is enabled. This allows textures to naturally blend with model base colors. The last interesting graphical effect in SIKAAAGS is motion blur. This effect was achieved by sending pixel information to and subsequently drawing the accumulation buffer over the current frame.

The "IK" in SIKAAAGS stands for "Inverse Kinematics". The project makes use of two different IK systems. The first system is a closed two-link system, while the other is a 6-link system that is computed using a Jacobian matrix. The chains are fixed to the origin of thier local coordinate space and can solve the inverse kinematics equation for targets in their two-dimensional X/Y plane. To simulate the system in 3D, the target is first translated into the system's local coordinate space. Then, the target is rotated into the plane of the kinematics system. The angle of this rotation is then used as

reference for the kinematics system to function in three dimensions while only seeking for targets in two. The benefit of the two-link system is that it was simpler to implement and it provided instantaneous solutions that were very helpful in creating specific animations. The six-link system was more difficult to implement, but allowed for more links.

Finally, the "AAAGS" stands for "...and AAAAAGH! A GIANT SCORPION!" Viewers may stare in wonder and amazement at the behemoth that parades accross the desert sands, but be weary: The beast uses inverse kinematics and physics-based animation cycles to terrorize the land with its razor edged feet and deadly barbed stinger. Watch out or you'll be stunned by its poisonous venom!