A2—Rasterization Parameter Interpolation and Applications

*Due: Wednesday February 3 at 7am*

1. Expand your rendering engine with screen space rasterization parameter interpolation
2. Use it for Z-buffering (screen space interpolation of “1/w”).
3. Use it for vertex color interpolation.
4. Expand your rendering engine with model space rasterization parameter interpolation
	1. Use it for texture mapping (nearest neighbor lookup is OK)
	2. Use it for per pixel lighting based on vertex normal interpolation; use a simple lighting equation such as Color = Color0\*(ka + (1-ka)\*kd), where Color0 is the material color given as input, ka is the ambient factor (e.g. 0.25), and kd is the diffuse factor, computed as the dot product between the light vector and the surface normal (negative values are clamped to 0).
5. Render a 20s 30Hz 720p video sequence illustrating your scene.
	1. First 5s should show the teapot (i.e. the teapot loaded from the “teapot1K” file, see tmesh.\* files) with z-buffering and vertex color interpolation (using the vertex colors loaded from the file).
	2. Second 5s sequence should show a red version of the teapot with per pixel lighting (i.e. Color0 in the expression above should be (1.0f, 0.0f, 0.0f) for all teapot vertices, discarding the colors loaded from the file). Show the point light source with a big white dot. The camera should move but the light source should not.
	3. Third 5s sequence same as b, but now the camera should be stationary and the light source should move.
	4. Fourth and last 5s sequence should show the cube from A1 (i.e. assignment 1) texture mapped and spinning.
6. Extra credit 1%: Bilinear interpolation lookup
7. Extra credit 1%: Lighting with specular highlights
8. Extra credit 3%: Mipmapping
9. Extra credit 2%: Sprites (i.e. textures with transparent texels to cut out the background, e.g. to model a game character)
10. Turn in via blackboard one zip archive that contains
	1. Source code
	2. Executable
	3. Video file

© Popescu 2016, popescu@purdue.edu