Student no.	point total	req total	extra total	R1 Texture sampling (1p)	R2 diffuse shading (1p)	R3 D (1p)	R4 G (1p)	R5 Fr (1p)	mod	notes / wtf / ...	World space (2p, 3p if optimized)	Point lights (3p)	Moving lights (1p)	SSS (2p)	Color/position variations (1.5p)	Shadow maps (4p)	Shadowmap SSS (2p)	Envmap (3+p)	Other extras (?p)	What other extras
218096	0	0	0																	
225157	0	0	0																	
270034	2	2	0	1	1															
292009	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
292326	5	5	0	1	1	1	1	1												
292986	0	0	0																	
293545	0	0	0																	
295323	0	0	0																	
297606	0	0	0																	
311210	0	0	0																	
347022	0	0	0																	
350006	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
350475	0	0	0																	
353692	6	4	2	0.5	1	0.5	1	1		R1: DiffuseColor shouldn't be multiplied by old color; R3: The specular component should still be multiplied by color, dot(N,L) and Li. PositionVarying is already in camera space (converted in vertex shader), therefore V is simply normalize(-positionVarying); R5: n should be squared in beta; World space: V calculated incorreclty. You should have positionVarying and the camera position in world space. If you had changed the code so that positionVarying is always in world space then your V for R3 would have been correct; SSS: Taking the code straight from the website does not work because your code already implements diffuse and specular shading. Approximation does not work, only lights up the face and there is no red tint anywhere.	2			0						
353757	0	0	0																	
357083	0	0	0																	
362256	0	0	0																	
401311	0	0	0																	
424615	9	5	4	1	1	1	1	1								4				
425494	0	0	0																	
425614	7.5	5	2.5	1	1	1	1	1				0.5		2						
426419	0	0	0																	
426736	0	0	0																	
427492	0	0	0																	
427793	2	2	0	1	1					note that redefining mappedNormal in the scope leads to it not being actually used for rendering.										
427845	0	0	0																	
428381	0	0	0																	
428789	2	2	0	1	1	0				R2: Diffuse not written to the final result; R3: V and H not calculated, CookTorrance not called, D incorrect.										
430324	0	0	0																	
430463	1.5	1.5	0	1	0.5					R2: Diffuse not added to light_contribution;										
431857	0	0	0																	
432241	5	5	0	1	1	1	1	1												
437631	0	0	0																	
438397	0	0	0																	
472379	0	0	0																	
473158	0	0	0																	
473420	0	0	0																	
473637	0	0	0																	
474380	0	0	0																	
475389	0	0	0																	
475758	0	0	0																	
475910	0	0	0																	
476498	0	0	0																	
477170	0	0	0																	
477400	7.5	4.5	3	1	1	1	0.5	1		R4: Remove clamp;	3									
477617	0	0	0																	
477659	3	3	0	0.5	1	0.5	1			R1: Normal read from texture should not be normalized before scaling to range [-1,1]; R3: V and H incorrect, H is just normalize(L+V). Specular not added to light contribution;										
477701	5	5	0	1	1	1	1	1					0							
478328	0	0	0																	
478470	0	0	0																	
478687	0	0	0																	
479505	0	0	0																	
479576	5	5	0	1	1	1	1	1												
479725	0	0	0																	
480248	0	0	0																	
480303	0	0	0																	
480730	21	5	16	1	1	1	1	1			3	3	1	2		4			3	PCF(2p), Separate controllable wraps for SSS (1p)
481014	4.5	4.5	0	1	1	0.5	1	1		R3: D should return 0 if cos_lambda is negative.										
481441	0	0	0																	
493578	0	0	0																	
506355	0	0	0																	
508793	0	0	0																	
514020	0	0	0																	
516109	5	5	0	1	1	1	1	1		R3: Your results are correct but the debug render makes an assumption that the specular variable doesn't contain the dot product and light intensity. The example does (diffuse+specular)*max(dot(n, l),0.0)*Li at the end of the loop body;										
519656	0	0	0																	
525653	0	0	0																	
525666	5	5	0	1	1	1	1	1												
525792	0	0	0																	
525925	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
526490	5	5	0	1	1	1	1	1												
526717	5	5	0	1	1	1	1	1												
526746	0	0	0																	
527143	0	0	0																	
527347	2	2	0	1	1															
527389	0	0	0																	
527444	0	0	0																	
527923	37.5	5	32.5	1	1	1	1	1		Shadow maps work here (not for the animated lights though, as the CPU-side position info isn't updated). Trackball still a bit iffy; negating v.yz and the rotation angle and switching the multiplication order of dR and camera_rotation_ seems to fix most of this. Local coordinate frames not smooth over curve segments (for example, see weirder.swp)	3	3	1	2		4			19.5	Rotate and scale (+.5p), Trackball(+.5p), Normal transforms(+1p), FOV(+1.5p), Animation (0.5p), PLY(3.5p), Local coordinate frames(.5p), surfaces of revolution (3p), gencyls (3p), Cloth tearing (1p), Poking (+.5p), GPU wind (2p), Frictionless collision (2p)
528634	8.5	5	3.5	1	1	1	1	1		Point light: h incorrect, you should normalize L before calculating h. The point lights don't behave as expected because you're lacking an attenuation term that depends on the light distance.		2	1						0.5	Ambient light & more pleasant look (0.5p)
528883	0	0	0																	
529293	0	0	0																	
529303	0	0	0																	

Student no.	point total	req total	extra total	R1 Texture sampling (1p)	R2 diffuse shading (1p)	R3 D (1p)	R4 G (1p)	R5 Fr (1p)	mod	notes / wtf / ...	World space (2p, 3p if optimized)	Point lights (3p)	Moving lights (1p)	SSS (2p)	Color/position variations (1.5p)	Shadow maps (4p)	Shadowmap SSS (2p)	Envmap (3+p)	Other extras (?p)	What other extras
529617	0	0	0																	
529992	7	4.5	2.5	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying); World space: PositionVarying not in world space	2.5									
530185	0	0	0																	
530907	4.5	4.5	0	1	1	0.5	1	1		R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.										
530981	0	0	0																	
540094	0	0	0																	
540311	5	5	0	1	1	1	1	1												
541543	0	0	0																	
544375	4.5	4.5	0	1	1	0.5	1	1		R3: Shading loop never adds specular to the result. PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
544566	0	0	0																	
549749	0	0	0																	
552969	0	0	0																	
556347	0	0	0																	
561578	0	0	0																	
563068	0	0	0																	
570116	0	0	0																	
586210	5	5	0	1	1	1	1	1		R3: Your results are correct but the debug render makes an assumption that the specular variable doesn't contain the dot product and light intensity. The example does (diffuse+specular)*max(dot(n, l),0.0)*Li at the end of the loop body;										
587170	10	5	5	1	1	1	1	1		No toggle for world space lighting	2	3								
587921	12	5	7	1	1	1	1	1			3	3	1							
588137	0	0	0																	
588441	0	0	0																	
589291	0	0	0																	
589437	4	4	0	1	1	1	1			G is correct; the visualization loop is not. specular should only store the result of CookTorrance to show up properly.										
589848	0	0	0																	
590112	4.5	4.5	0	1	1	0.5	1	1		Shading loop never adds specular to the result.										
590332	3	3	0	0.5	1	0.5	1			the texture normal is never actually used; some mistakes in the D term and the general structure of the shading loop.										
590426	5	5	0	1	1	1	1	1												
593177	0	0	0																	
593452	4.5	4.5	0	1	1	0.5	1	1		R3: V incorrect. Diffuse and specular should be added to light_contribution, not to answer;										
593876	6	5	1	1	1	1	1	1			1									
594367	2.5	2.5	0	1	1	0.5				R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.										
594590	0.5	0.5	0	0.5						Normal read from texture is not normalized.										
594930	0	0	0																	
595201	0	0	0																	
595612	3.5	3.5	0	1	1	0.5	1			R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
596048	4	4	0	0.5	1	1	1	0.5		Readme not filled! R1: Normal not normalized; R5: Missing parentheses for 0.5*(Rs+Rp)										
596242	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying). Roughness should be squared in the numerator of D;										
596747	0	0	0																	
596789	0	0	0																	
596792	8.5	5	3.5	1	1	1	1	1		Point lights: Positions should be transformed with w = 1.		2.5	1							
596857	4.5	4.5	0	1	1	0.5	1	1		R3: Shading loop never adds specular to the result.										
597445	5	5	0	1	1	1	1	1												
598088	6.5	4.5	2	1	1	0.5	1	1		R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.				2						
598318	0	0	0																	
602851	0	0	0																	
603067	6	4.5	1.5	1	1	1	1	0.5		R5: In return, you have Rs*Rp, when you should have Rs+Rp					1.5					
603096	2.5	2.5	0	1	1	0.5				R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
603326	0	0	0																	
604105	9	4.5	4.5	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying). SSS: Applying the dot(N,L) to the scatter doesn't seem necessary since the scatter is already based on the dot(N,L) value.			1	2	1.5					
606064	0	0	0																	
606268	0	0	0																	
608952	3.5	3.5	0	0.5	1	0.5	1	0.5		R1: Matrix multiplication isn't generally commutative, the order of multiplication is wrong when transforming the normal; R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying). The illumination model isn't correct, should be (kd+ks*Si)*dot(n,l)*Li; R5: You're doing integer division when calculating the average, i.e. 1/2 = 0;										
609142	11	5	6	1	1	1	1	1				3	1	2						
609155	0	0	0																	
609168	5	5	0	1	1	1	1	1												
610827	5	5	0	1	1	1	1	1												
612155	0	0	0																	
612540	0	0	0																	
612812	0	0	0																	
621308	0	0	0																	
647175	0	0	0																	
647502	17	5	12	1	1	1	1	1		Note: Using variable names that are also GLSL function names seems a dangerous practice! (e. g. tan & distance); SSS: Should multiply the scatter by Li, otherwise the effect shows up even if there's no incoming light! A4: Constant hard coded wind	3	3	1	2					3	A1: Viewport & Perspective(1.5p), Mesh simplifier attempt(1p); A4: Wind (0.5p)
648080	4.5	4.5	0	1	1	0.5	1	1		R3: positionVarying is already in camera coordinates; should just normalize and negate to get V.										
648569	4.5	4.5	0	1	1	0.5	1	1		R3: When calculating V, should use positionVarying, not posToCamera[3]; R5: Using n2 instead of n1/n2										
648860	0	0	0																	
649458	0	0	0																	
650191	0	0	0																	
650560	0	0	0																	
650829	4	4	0	1	1	0.5	0.5	1		R3: view direction isn't constant; it should be positionVarying, not the camera direction. R4: should get angle from dot product result before calling tan (essentially tan(acos(dot(v))))										
651640	4.5	4.5	0	1	1	0.5	1	1		The specular component should also be multiplied by dot(N,L)										

Student no.	point total	req total	extra total	R1 Texture sampling (1p)	R2 diffuse shading (1p)	R3 D (1p)	R4 G (1p)	R5 Fr (1p)	mod	notes / wtf / ...	World space (2p, 3p if optimized)	Point lights (3p)	Moving lights (1p)	SSS (2p)	Color/position variations (1.5p)	Shadow maps (4p)	Shadowmap SSS (2p)	Envmap (3+p)	Other extras (?p)	What other extras
651802	0	0	0																	
652209	31	5	26	1	1	1	1	1		World space lighting doesn't work when the lights are animated because you apply the rotation in the shader. You could just rotate the lights in the C++ code right after the definition of the rotation matrix; Viewport & perspective: Weird matrix, many terms cancel out (r+l)/(r-l)=(t+b)/(t-b)=0, 2*n/(r-l)=n/r, 2*n/(t-b)=n/t; GPU collision seems a bit bugged	3	3	1	2		4			13	A1: Normal transform(1.5p), Viewport & Perspective(2p), PLY Ascii & Quads (2.5p); A4: CPU: Collision(2p), Tearing(1p), GPU: Collision(1p), Tearing(1p), Poke(1p), Wind(1p)
652584	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
653156	0	0	0																	
653347	4	4	0	0.5	1	0.5	1	1		R1: Normal not transformed to camera space; R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying); R4&5: Your results are correct but the debug render makes an assumption that the specular variable doesn't contain the dot product and light intensity. The example does (diffuse+specular) *max(dot(n,l),0.0)*Li at the end of the loop body;										
654142	5	5	0	1	1	1	1	1												
654294	0	0	0																	
654618	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
655109	0	0	0																	
655361	0	0	0																	
655390	0	0	0																	
656014	0	0	0																	
657068	0	0	0																	
657181	9.5	4.5	5	1	1	0.5	1	1		R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.			1			4				
657437	4.5	4.5	0	1	1	0.5	1	1		R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.										
657767	18	5	13	1	1	1	1	1			3	3	1	2		4				
657893	0	0	0																	
663434	0	0	0																	
665173	5	5	0	1	1	1	1	1		R1: The debug image of normal map texture does not show the correct image because normalFromTexture is multiplied by normalToCamera;										
665678	24.5	5	19.5	1	1	1	1	1			3	3	1	2		4			6.5	PCF(2p), floor(.5p), post processing (4p)
666208	0	0	0																	
666211	5	5	0	1	1	1	1	1												
666253	0	0	0																	
710015	12	4.5	7.5	1	1	0.5	1	1		R3: D should return 0 if cos_lambda is negative; there also seems to be some other NaN-type issue. In the subsurface scattering, the specular should not change (we already have a different specular model and specular reflections are strictly unrelated to subsurface effects), and the ndotl/ndoth variables are missing the mapping from [0,1] to [-1,1].				1.5		4			2	PCF(2p)
715298	5	5	0	1	1	1	1	1		R3: SpecularColor multiplied by Si twice										
716734	0	0	0																	
717377	0	0	0																	
717539	0	0	0																	
718020	0	0	0																	
718208	5	5	0	1	1	1	1	1												
718512	4.5	4.5	0	1	1	0.5	1	1		R3: H incorrect due to missing parentheses around L+V;										
718826	19	5	14	1	1	1	1	1		Color variations: Positions should be transformed with w = 1; SSS: Specular shouldn't change. Maybe have the scatter color be more red so that the effects can be seen properly.	3	3	1	2	1	4				
719032	8.5	5	3.5	1	1	1	1	1		SSS: Diffuse should use NdotL_wrap, not NdotL, and the whole algorithm has not been implemented (e.g. missing scatter which uses smoothstep)			1	1	1.5					
721619	0	0	0																	
721923	0	0	0																	
723154	0	0	0																	
723329	4.5	4.5	0	1	1	0.5	1	1		R3: No check for cos_theta_h being negative (should return 0 if it is), specular and diffuse uninitialized in lighting computations (should set them instead of adding to)										
723468	6.5	4.5	2	0.5	1	1	1	1		R1: normals shouldn't be normalized before being mapped into the [-1,1] range and the texture normal is not assigned to the shading normal in the actual rendering path. World space shading requires you to also change the view and light vectors. Point light is really just another directional light.	0.5	0.5	1							
723484	4.5	4.5	0	1	1	0.5	1	1		R3: should check if cosin_theta negative (and return 0 if so)										
723565	6	5	1	1	1	1	1	1					1							
723976	0	0	0																	
724483	0	0	0																	
726915	5.5	5	0.5	1	1	1	1	1			0.5									
728696	0	0	0																	
729297	0	0	0																	
732323	0	0	0																	
737551	5	5	0	1	1	1	1	1												
765714	0	0	0																	
765756	0	0	0																	
765785	4.5	4.5	0	1	1	0.5	1	1		R3: You should return 0 when dot(n,h) < 0										
765882	4.5	4.5	0	1	1	0.5	1	1		R3: V should just be -positionVarying; its value in principle is normalize(cameraPos-position), but since we're shading in camera space, camera is at the origin.										
766108	0	0	0																	
767136	0	0	0																	
769396	0	0	0																	
772419	0	0	0																	
784465	0	0	0																	
784847	0	0	0																	
784902	0	0	0																	
785053	0	0	0																	
785134	5	5	0	1	1	1	1	1												
785163	0	0	0																	
785228	4.5	4.5	0	1	1	0.5	1	1		R3: PositionVarying is already in camera space, therefore V is simply normalize(-positionVarying).										
785257	0	0	0																	
785325	0	0	0																	
785354	5	5	0	1	1	1	1	1												
785367	5	5	0	1	1	1	1	1												
785435	0	0	0																	
785448	0	0	0																	
785451	4	4	0	1	1	0.5	0.5	1		R3: V incorrect. R4: You set specular to zero when dot(n,l) < 0, this is not incorrect but breaks the debug visualization for r4; Specular not written to the final result;										

Student no.	point total	req total	extra total	R1 Texture sampling (1p)	R2 diffuse shading (1p)	R3 D (1p)	R4 G (1p)	R5 Fr (1p)	mod	notes / wtf / ...	World space (2p, 3p if optimized)	Point lights (3p)	Moving lights (1p)	SSS (2p)	Color/position variations (1.5p)	Shadow maps (4p)	Shadowmap SSS (2p)	Envmap (3+p)	Other extras (?p)	What other extras
785493	2.5	2.5	0	0.5	1	0.5	0.5			R1: Use the normalToCamera transformation matrix on normalFromTexture to get mappedNormal; R3&R4: You haven't called the CookTorrance function, that's why you don't see any results. V and H vectors not defined, otherwise D and G functions implemented correctly.										
785503	0	0	0																	
785516	0	0	0																	
795551	0	0	0																	
795577	0	0	0																	
795593	2	2	0	1	1															
795629	0	0	0																	
795658	0	0	0																	
795674	8	5	3	1	1	1	1	1					1	2						
795713	0	0	0																	
795865	0	0	0																	
796178	5	5	0	1	1	1	1	1												
798257	0	0	0																	
801131	0	0	0																	
804646	5	5	0	1	1	1	1	1												
807711	0	0	0																	
809609	4.5	4.5	0	1	1	0.5	1	1		Shading loop never adds specular to the result.										
811383	0	0	0																	
814872	0	0	0																	
818315	0	0	0																	
821289	0	0	0																	
822709	0	0	0																	
46596K	0	0	0																	
55055P	0	0	0																	
62727K	5	5	0	1	1	1	1	1												
64879R	14	5	9	1	1	1	1	1		Point lights: Are there two quadratic attenuation terms on purpose? Point lights are a hacky anyways so you're free to do anything that looks good of course	3	3	1	2						
65451T	0	0	0																	
67932J	2.5	2.5	0	1	1	0.5				R2: You should use diffuseColor here, which you read in R1. The model is implemented otherwise correctly however; R3: V is just normalize(-positionVarying) since the camera is located at the origin. The head is black because the specularUniform isn't set and happens to be 0, you were "meant" to use specularColor which incidentally is just set to vec3(1) though.										
69246M	7.5	5	2.5	1	1	1	1	1		World space: PositionVarying still in camera space	2.5									
77241H	0	0	0																	
77388B	0	0	0																	
83107B	0	0	0																	
83854J	0	0	0																	
84171B	5.5	4.5	1	1	1	1	0.5	1		R4: Your formula for G is not equal to the one given in handout (even if we account for dot(n,x) < 0.0). It's similiar in shape but not the same.			1							
84805K	0	0	0																	
k28342	0	0	0																	
k90624	0	0	0																	
k93517	0	0	0																	