Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
218096																	
225157																	
270034				1	3	3	3			1							
292009				1	3	3	3			1							
292326				1	3	3	3		R2: You were wondering why normalizing before cross product did not work. Cross product of vectors **a** and **b** produces a vector of length |**a**|*|**b**|*sin(theta) where theta is the angle between **a** and **b**. So the result has length 1 only if **a** and **b** are perpendicular. VCS: claim this point in a later assignment by showing a log where you've made multiple commits;	0							
292986				1	2.5	3	3		R2: normals inverted								
293545				1	2	3	3		R2: height of cone is simply height; the vectors inside the cross product should be the edge vectors of the triangle, not the position of the vertices. The idea in rotate and scaling was to modify modelToWorld (like you did in translation), not to modify the actual positions of the vertices.	1	1					0.5	Animation(0.5p)
295323																	
297606				1	3	3	3		Having only one variable for both rotation angles is kind of weird but not a mistake per se	1	1						
311210				1	2.5	3	3		R2: Normals inverted								
347022	0	0	0														
350006	20	10	10	1	3	3	3		Normals not normalized; Viewport: +0.5p for fixing the appearance of GUI when changing window size; PLY: ascii + smooth normals	1	1	0.5	3		1	3.5	Animation(0.5p), PLY (3p)
350475	11	10	1	1	3	3	3			1							
353692	9.5	9.5	0	1	2.5	3	3		R2: tip not at origin								
353757	11	9	2	1	2	3	3		R2: Tip not at origin, normal should be computed based on edges instead of positions. Scaling should affect a whole row or column of the upper 3-by-3 matrix (also multiply either of the sines with the width_ratio), otherwise scaling and then rotating gives a weird wobbly effect. This is why it's easiest to construct full matrices and multiply them.	1	0.5					0.5	Animation(0.5p)
357083	19.5	10	9.5	1	3	3	3		PLY loader with polygon triangulation, no binary support.	1	1			2	2	3.5	Animation(0.5p), PLY(3p)
362256	0	0	0														
401311	0	0	0														
424615	14.5	9.5	5	1	2.5	3	3		R2: normals inverted	1	1			2	0.5	0.5	Animation (0.5p)
425494	0	0	0														
425614	10	10	0	1	3	3	3										
426419	0	0	0														
426736	0	0	0														
427492	0	0	0														
427793	12	10	2	1	3	3	3			1	1						
427845	0	0	0														
428381	0	0	0														
428789	10	10	0	1	3	3	3		VCS: Please specify where your log or screenshot is, return it next round for points. Rotate: You are rotating the camera, not the object	0	0						
430324	0	0	0														
430463	11.5	9.5	2	0.5	3	3	3		R1: translation only along z-axis; Scaling is uniform;	1	1						
431857	11	10	1	1	3	3	3		R1: z-axis translation missing (-0p); Rotate and scale: You could have easily multiplied modelToWorld with a rotation and a scale matrix to get the desired effect.	1							
432241	9	9	0	1	2	3	3		R2: The vectors inside the cross product should be the edge vectors of the triangle, not the position of the vertices. You are also defining the normal only for vertex v2 (it should be v1.normal = v2.normal = ..., instead of v1.normal, v2.normal = ...).								
437631	16.5	9.5	7	1	3	2.5	3		R3: using same normal for all vertices in a triangle, Normal transform in shader: Normal w component should be 0, normal should be transformed with inverse transpose of the transform and normalized. PLY loader: no normals, non-tris/binary, and the subtle bug; after you change the mode, you continue straight into processing it without reading a new line first. Having else ifs for the modes fixes this.	1	1	0.5			2	2.5	Animation (0.5p), PLY (2)
438397	0	0	0														
472379	13.5	9.5	4	1	2.5	3	3		Could not find a screenshot or log of your version control. Submit it next round for points. R2: Normals a bit off, cone is slightly lit from the back; Rotate & scale: translation, scale and rotation in wrong order; Normal transformation: Normals should have w=0 because they are directions, be transformed with inverse transpose of modelToWorld and normalized. Viewport: No adjustable FoV, your formula for fovYRad is a bit incorrect, it should be angle * pi / 180.	0	0.5	0.5	1		1	1	Animation(0.5p), cone is closed(0.5p)
473158	0	0	0														
473420	0	0	0														
473637	0	0	0														
474380	10.5	9.5	1	1	2.5	3	3		R2: inverted normals	1							
475389	0	0	0						Only .sln file returned. Cannot run code.								
475758	0	0	0														
475910	0	0	0														
476498	0	0	0														
477170	0	0	0														
477400	10	10	0	1	3	3	3										
477617	0	0	0														
477659	11.5	9.5	2	1	2.5	3	3		R2: Inverted normals; Camera: Basic xy-rotation				2				
477701	0	0	0														

Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
478328	9	9	0	1	2	3	3		R2: Normal is a cross product between the edge vectors, not the vertices themselves.								
478470	10	10	0	1	3	3	3										
478687	0	0	0														
479505	0	0	0														
479576	18.5	9.5	9	1	2.5	3	3		R2: normals inverted, Normal transform: normals not normalized afterwads (and w should be 0), Viewport: fov/fov_0 should be multiplied into both the first and second columns, and the first clause of the if always taken. Camera quite simple.	1	1		1.5	1.5	1.5	2.5	Animation(0.5p), Simplifier attempt (3p)
479725	23	10	13	1	3	3	3		Normals not normalized after transform	1	1		3	1.5	2	4.5	Animation(0.5p), PLY(4p)
480248	0	0	0														
480303	9.5	7.5	2	1	2.5	3	1		R2: Normals inverted; R4: Multiple issues causing crashes, e.g. using uninitialized variable without reading positions and normals from the stream; Rotate, scale, translate order of multiplication wrong; Normal trans.: Normal not normalized after transform; Animation: You're only updating the animation if a button is held down; No viewport correction; Did you submit a wrong version of your code?	1	0.5	0.5					
480730	38.5	10	28.5	1	3	3	3			1	1		3	2	2	19.5	Animation(0.5p), FBX (4p), Textures (2p), Drag 'n drop (1p), Simplifier (10p), Progressive LoD rendering (2p)
481014	11.5	9.5	2	1	2.5	3	3		R2: normals inverted	1	1						
481441	11	10	1	1	3	3	3						1				
493578	0	0	0														
506355	14.5	10	4.5	1	3	3	3		Transforming normals: modelToWorldInv should be transposed, normals should have w = 0, and multiplication should be in the order matrix * vector.	1	1			1.5	0.5	0.5	Animation(0.5p)
508793	0	0	0														
514020	0	0	0														
516109	12.5	10	2.5	1	3	3	3		Normals are transformed as positions; should use the inverse transpose of the modeltoworld matrix and normalize after.	1	1	0.5					
519656	0	0	0														
525653	0	0	0														
525666	14.5	10	4.5	1	3	3	3			1	1			2	0.5		
525792	0	0	0														
525925	18.5	10	8.5	1	3	3	3		Camera movement doesn't change with mouse, should probably not multiply r4 to world_to_clip before P (leads to rotation in clip space?), Normals not normalized after transform	1	1		1.5	1.5	2	1.5	Animation(0.5p), PLY attempt (1p)
526490	11	10	1	1	3	3	3			1							
526717	14	10	4	1	3	3	3		Scale, rotation, translation order of multiplication is incorrect; Camera: Basic xy-rotation	1	0.5		2			0.5	Animation(0.5p)
526746	0	0	0														
527143	0	0	0														
527347	11	10	1	1	3	3	3			1							
527389	0	0	0														
527444	0	0	0														
527923	15	10	5	1	3	3	3		Rotate and scale: Rotation and scale in wrong order; Camera: Instead of keeping track of the angle and axis, it would have been easier to just store the camera's current rotation matrix. Then when mouse moves, calculate the rotation needed for that small increment and multiply this with the camera's rotation matrix. This multiplication could have been done in handleEvent() after the axis and angle for this small increment have been calculated. This of course requires that you always update last_pos_ to cur_pos_ after this. In addition, mouse_moved_ should be updated to false after user no longer presses the left mouse.	1	0.5	1	2		0.5		
528634	15.5	10	5.5	1	3	3	3		Camera: not trackball, but somewhat reasonable	1	1	1	2			0.5	Animation(0.5p)
528883	0	0	0														
529293	0	0	0														
529303	8	8	0	1	2	2.5	2.5		R2: p1x should be computed with index (i+1), not n, normals are inverted and tip is at 1 instead of origin. R3: all values read from positions (last three should be normals), R4: obj indexes from 1; you should subtract 1 from each element before pushing the face.								
529617	0	0	0														
529992	14.5	10	4.5	1	3	3	3		Normal trans.: Normals should be transformed using the inverse transpose of our model to world matrix, and w=0 in homogeneous coordinates since normal is a direction, not a point. And don't forget to normalize after transform; Camera: Basic xy-rotation;	1	1	0	2			0.5	Animation(0.5p)
530185	0	0	0														
530907	10.5	9.5	1	1	3	3	2.5		R4: You're not supposed to subtract 1 from the actual vertex and normal values, just the indices.	1							
530981	10.5	9.5	1	1	2.5	3	3		R2: Normals inverted	1							
540094	0	0	0														
540311	11	10	1	1	3	3	3			1							
541543	10.5	10	0.5	1	3	3	3				0.5						
544375	21	10	11	1	3	3	3		Normals not normalized; PLY: No binary or polygon support	1	1		3	1.5	2	2.5	Animation(0.5p), PLY(2p)
544566	0	0	0														
549749	0	0	0														
552969	0	0	0														
556347	0	0	0														

Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
561578	12	10	2	1	3	3	3		VCS: Did not find log or screenshot of version control. Please return it next round for points; Rotate and scale in wrong order; Normals should have w=0 because they are directions and modelToWorldInv is missing transpose.	0	0.5			1.5			
563068	11	9.5	1.5	1	2.5	3	3		R2: normals inverted, No rotation or non-uniform scaling	1	0.5						
570116	0	0	0														
586210	19.5	10	9.5	1	3	3	3			1	1		3	2	2	0.5	Animation(0.5p)
587170	23.5	10	13.5	1	3	3	3			1	1		3	2	2	4.5	Bouncy animation(1p), PLY with non-triangles and normal gen (3.5p)
587921	16.5	10	6.5	1	3	3	3		You were not supposed to multiply vectors with Mat3f:: rotation(Vec3f(0,1,0), camera_rotation_angle_) in translation and rotation (-0p). There really isn't an official policy in naming the submission. The name of your submission was fine. If you have trouble with version. aalto, you can always come to the exercise sessions to ask for help.		1		1	2	2	0.5	Animation(0.5p)
588137	0	0	0														
588441	11	9	2	1	2.5	2.5	3		R2: Normals not normalized; R3: Same normal used for each vertex of a triangle. Note that for most somewhat more detailed meshes, the normals are defined on a per vertex basis.	1					0.5	0.5	Animation(0.5p)
589291	0	0	0														
589437	15.5	9.5	6	1	2.5	3	3		R2: tip not at origin; Transforming normals: Normals should have w = 0 because they are directions, and they should be normalized; Simple camera	1	1		0.5	1.5		2	PLY (2p)
589848	2.5	2.5	0	1	1.5				R2: tip not at origin, normals are a cross product of the edge vectors of the triangle.								
590112	11.5	9.5	2	1	2.5	3	3		R2: v0 and v2 have inverted normals. All vertices could have been assigned the same normal, no need to calculate the cross product 3 times. VCS: claim this point in a later assignment by showing a log or screenshot of your repository history;	0	1				0.5	0.5	Animation(0.5p)
590332	16.5	9.5	7	1	2.5	3	3		R2: Normals not set correctly; Rotation, scale, translation order incorrect, things will break with non uniform scaling; Camera rotation is very clusmy; Your zoom isn't an actual zoom, it's simply moving the camera. Actual zoom could be implemented in the perspective matrix extra; Simple ascii ply loader, no normals.	1	0.5	1	1.5		1	2	Animation(0.5p), PLY(1.5 p)
590426	18	10	8	1	3	3	3		Simple xy-camera. Normals not normalized.	1	1		2	1.5	2	0.5	Animation(0.5p)
593177	0	0	0														
593452	10	9	1	1	2	3	3		R2: Normals incorrect. Use the crossproduct between two **edges** of the triangle to calculate the normal. In this case you can then use the same normal for all three vertices.		1						
593876	12	10	2	1	3	3	3				1	1					
594367	11.5	9.5	2	1	2.5	3	3		R2: tip not at origin, normals inverted;	1	1						
594590	9.5	9.5	0	1	2.5	3	3		Please let us know the hours you spent on the assignment next time. R2: normals inverted;								
594930	0	0	0														
595201	28	9	19	1	2	3	3		R2: Normals: Use the cross product of the edge vectors, not the vertices themselves; Camera could use some some way to move vertically?; Normal transform: Should be transformed with the inverse transpose matrix. Result should be normalized after transformation (also set w-component to 0 when transforming); PLY loader with binary support, no polygon triangulation. Simplifier a bit slow and the results aren't perfect, but it's a solid attempt.	1	1		2.5	1	2	11.5	Animation(0.5p), PLY(3p), Normal generation(1p), Simplifier(7p)
595612	20	10	10	1	3	3	3		Normal not normalized after transform. Camera tries to do a lot but feels very clumsy. See the example executable for a virtual trackball reference.	1	1		2	1.5	2	2.5	Animation(0.5p), Simplification attempt (2p)
596048	14.5	9.5	5	0.5	3	3	3		R1: translation only along z-axis; VCS: claim this point in a later assignment by showing a log where you've made multiple commits; Normal transformation: normals not normalized; Simple camera	0	1	0.5	1		2	0.5	Animation(0.5p)
596242	14.5	10	4.5	1	3	3	3		Normals not normalized after transform	1	1			1.5	0.5	0.5	Animation(0.5p)
596747	0	0	0	0													
596789	0	0	0														
596792	14.5	10	4.5	1	3	3	3		R3: The inner for loop is redundant, the last value of j (=2) is the only one that counts; Normals not normlized; Viewport: Square not in the middle of screen	1	1		0.5	1.5	0	0.5	Animation(0.5p)
596857	10.5	10	0.5	1	3	3	3		Please include the solution files and assets in your submissions; VCS: Claim this point in a later round by showing a log with multiple commits; Rotate and scale don't work correctly together	0	0.5						
597445	13.5	10	3.5	1	3	3	3		Scaling is uniform;	1	1		1			0.5	Animation(0.5p)
598088	22	10	12	1	3	3	3		Simple xy-camera. Normals not normalized.	1	1		2	1.5	2	4.5	Animation(0.5p), PLY(2p), Simplifier attempt(2p)
598318	0	0	0														
602851	1	1	0	1													
603067	11.5	10	1.5	1	3	3	3		R2: inverted normals, Rotate and scale: object can only be scaled, the rotation is another camera rotation	1	0.5						
603096	11	10	1	1	3	3	3			1							
603326	0	0	0														
604105	13	10	3	1	3	3	3		Rotation and scale don't work together; easiest solution is to construct each in their own matrix and multiply. Normal transform should be inverse transpose of position transform (this is computed but not used), and the normal should be normalized after transform.	1	1			1			
606064	0	0	0														
606268	11	10	1	1	3	3	3			1							

Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
608952	5	5	0		2	3			R1 claimed in readme, but no attempt visible in code, R2: the sin and cos draw a circle; they should be in both of the two first vertices (with (i+1) used in the other, so we get two subsequent vertices on the circle)								
609142	21.5	9.5	12	1	2.5	3	3		R2: Inverted normals; Normal trans.: Normals not normalized	1	1		3	1.5	2	3.5	Animation(0.5p), PLY with polygons(3p)
609155	0	0	0														
609168	12.5	10	2.5	1	3	3	3		Please include the solution files and assets in the submission; Normal trans.: Normals not normalized	1	1	0.5					
610827	10.5	9.5	1	1	3	3	2.5		R4: the correct order for the indices for stoi(results[i]) is 1,3,4,6,7,9. What we recommend is reading directly from the stream; iss >> fx[0] >> sink >> fx[1] >> fx[2] ...	1							
612155	0	0	0														
612540	0	0	0														
612812	0	0	0														
621308	0	0	0														
647175	0	0	0														
647502	21.5	10	11.5	1	3	3	3		Trackball increment multiplied from the wrong side; axis of rotation can change	1	1		2.5	2	0.5	4.5	Animation(0.5p), PLY(4p)
648080	13.5	10	3.5	1	3	3	3		Please let us know the hours you spent on the assignment next time; Rotation, scale and translation in wrong order; Camera not a trackball, only 2 axis rotation. Also, rotation around x-axis does not work as it should for a trackball (rotate first around y-axis and then around x-axis to see what I mean)	1	0.5		1		0.5	0.5	Animation(0.5p)
648569	11	10	1	1	3	3	3		Please include the solution files and assets in your submissions;	1							
648860	10	10	0	1	3	3	3										
649458	0	0	0														
650191	1	0	1						all folders are empty, no code to be found :(	1							
650560	11	10	1	1	3	3	3			1							
650829	11.5	9.5	2	1	2.5	3	3		R2: inverted normals; VCS: claim this point in a later assignment by showing a log where you've made multiple commits; Rotation matrix incorrect	0	0.5		0.5		0.5	0.5	Animation(0.5p)
651640	12	10	2	1	3	3	3		Normals should be transformed with inverse transpose of modelTWorld (not worldToClip) and normalized.			0	1			1	Animation(0.5p), cone is closed(0.5p)
651802	6	6	0	1	2	3			R2: normals inverted and should be normalized after cross product								
652209	12	9.5	2.5	1	2.5	3	3		R2: normals inverted, Normals not normalized after transform	1	1	0.5					
652584	18.5	10	8.5	1	3	3	3		The FoV was not supposed to change when the window size changes. Instead you could have controlled it with a slider or buttons. PLY: no normals calculated	1	1		1	2	1.5	2	Animation(0.5p), PLY (1.5)
653156	12	10	2	1	3	3	3			1	1						
653347	14	10	4	1	3	3	3			1	1	1			0.5	0.5	Animation(0.5p)
654142	15.5	10	5.5	1	3	3	3		Simple xy-camera.	1	1	0.5	2		0.5	0.5	Animation(0.5p)
654294	12	10	2	1	3	3	3			1	1						
654618	15.5	9.5	6	1	2.5	3	3		R2: normals inverted and transformed with the direct transformation (should be inverse(transpose (uModelToWorld)), try flattening the gargoyle and rotating to see the difference)		1	0.5	2		2	0.5	Animation (0.5p)
655109	0	0	0														
655361	0	0	0														
655390	4	4	0	1	3												
656014	21.5	9.5	12	1	3	2.5	3		R3: nesting your own for-loop into the given one causes there to be N copies of each N triangle. No FOV control, no binary PLY.	1	1		3	2	1.5	3.5	Animation(0.5p), PLY with non-tris(3p)
657068	11.5	10	1.5	1	3	3	3		Rotation and scale: The idea here was to multiply modelToWorld with a rotation and a scale matrix, and not change v.positions. Now your implementation may cause the object to rotate when you scale the object.	1	0.5						
657181	15	10	5	1	3	3	3		Transforming normals: Normals not normalized	1	1		1	1.5		0.5	Animation (0.5p)
657437	13.5	10	3.5	1	3	3	3		Rotate and scale: Rotation, scale and translation in wrong order; Transforming normals: Normals not normalized; Simple camera	1	0.5	0.5	0.5		0.5	0.5	Animation(0.5p)
657767	22	10	12	1	3	3	3		Normals should have a w-component of 0, also normals not normalized. PLY loader with triangulation, no binary support.	1	1		3	1.5	2	3.5	Animation(0.5p), PLY(3p)
657893	12.5	10	2.5	1	3	3	3		Transforming normals: Normals should have w = 0 and be normalized.	1	1	0.5					
663434	0	0	0														
665173	13	10	3	1	3	3	3		Normals should be transformed with the inverse transpose of the modeltoworld matrix and normalized after. The bottom of the cone seems to have inverted normals (pointing up instead of down.)	1	1	0.5				0.5	Cone is closed(0.5p)
665678	21	10	11	1	3	3	3		Transforming normals: normals not normlized (this is why the lighting changes when object is scaled); Viewport: user should be able to adjust FoV for full points; STL file format: binary and ascii	1	1		3	1.5	1	3.5	Animation(0.5p), STL(3p)
666208	20	10	10	1	3	3	3		Camera: not trackball but nice, Normal transform: normal not normalized afterward, PLY: no non-tris or binary	1	1		2	1.5	2	2.5	Animation(0.5p), PLY (2p)
666211	12	10	2	1	3	3	3			1	1						
666253	0	0	0														
710015	13.5	9.5	4	0.5	3	3	3		R1: translation only along z-axis; Cannot choose FoV		1			2	1		
715298	15.5	10	5.5	1	3	3	3		VCS: Claim this point in a later round by showing a log with multiple commits; Camera: Basic xy-rotation and following the object;	0	1		2.5	2			
716734	0	0	0														
717377	0	0	0														
717539	0	0	0														
718020	0	0	0														

Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
718208	17.5	9.5	8	1	2.5	3	3		R2: tip not at origin; Normal transformation: normals should be transformed with inverse transpose and normalized;	1	1	0.5	1		2	2.5	Animation(0.5p), PLY (2p)
718512	16	10	6	1	3	3	3		Camera: simple and a bit laggy (testing if dragging and directly incrementing the angle and y with the mouse deltas would be smoother), Viewport fix simple but effective	1	1	1	1.5		1	0.5	Animation(0.5p)
718826	21	9.5	11.5	1	2.5	3	3		R2: inverted normals, Camera is fixed-axis instead of trackball,	1	1		2	2	2	3.5	Animation(0.5p), ascii+binary PLY (3p)
719032	18	10	8	1	3	3	3		Normals not normalized and remove uWorldToClip.	1	1		1	1	2	2	PLY (2p)
721619	9.5	9.5	0	1	2.5	3	3		R2: normals inverted								
721923	13	8.5	4.5	1	2	3	2.5		Please let us know the hours you spent on the assignment next time; R1: z-axis translation missing (-0 p); R2: tip not at origin; normals incorrect, should be a cross product of v2.position-v0.position and v1.position-v0.position; R4: Variable n should be pushed into normals instead of v; Transforming normals: Normals not calculated correctly. Resizing window not working. Trackball: When starting trackball, lastPosition should be initialized by trackball_ptov and instead of saving rotation as angle + axis, it would be easier to save it as a rotation matrix.	1	1	0	2		0	0.5	Animation(0.5p)
723154	0	0	0														
723329	11	9.5	1.5	1	2.5	3	3		R2: Normals inverted; VCS: You need to provide some log as a proof of using version control; Normal transform: You should use the inverse transpose of the model to world matrix, not model to clip. If you want help or more detailed answers, you should come ask us in the exercise sessions.		1	0.5					
723468	15	10	5	1	3	3	3		Translation, rotation and scaling in wrong order; Normal transform: Use the inverse transpose of model to world matrix to transform normals, and don't forget to normalize.	1	0.5	0	3			0.5	Animation(0.5p)
723484	17.5	10	7.5	1	3	3	3		Normals should have w-component of 0 in homogeneous coordinates.	1	1		0.5	2	0.5	2.5	Animation(0.5p), PLY(2p)
723565	10.5	9.5	1	1	2.5	3	3		Please include the solution files and assets in your submissions. R2: normals inverted	1							
723976	0	0	0														
724483	0	0	0														
726915	10	10	0	1	3	3	3		Next time return all solution files, not just .cpp and .hpp files, so that the submission can be compiled. And please let us know the hours you spent on the assignment next time. There most likely is a chance to return any assignment extra during round 6. VCS: claim this point in a later assignment by showing a log where you've made multiple commits;	0							
728696	0	0	0														
729297	0	0	0														
732323	0	0	0														
737551	15	10	5	1	3	3	3		Normal transform: not normalized afterwards, trackball: incremental matrix multiplication order should be flipped (Mat3f::rotation()*mouse_rotation_), now the rotation axes can change with subsequent rotations so the object turns into unexpected directions.	1	1	0.5	2.5				
765714	0	0	0														
765756	13	10	3	1	3	3	3		Normals not normalized after transform	1	1	0.5				0.5	Animation(0.5p)
765785	18.5	10	8.5	1	3	3	3		Normal transform: not normalized afterwards. Camera is not a trackball, but does something reasonable. Viewport has no FOV control.	1	1		2	1.5	1	2	PLY (2p)
765882	12	10	2	1	3	3	3		Rotation, scaling and translation in wrong order; Normals should be multiplied with inverse transpose of modelToWorld and they should be normalized	1	0.5	0.5					
766108	0	0	0														
767136	4	3	1	1	2				R2: normals just constant; not computed from geometry	1							
769396	0	0	0														
772419	10	9	1	1	2	3	3		R2: angle increment multiplication wrong (multipying i and adding 1 instead of multiplying i+1), normals inverted	1							
784465	0	0	0														
784847	0	0	0														
784902	0	0	0														
785053	0	0	0														
785134	11	10	1	1	3	3	3			1							
785163	0	0	0														
785228	12.5	10	2.5	1	3	3	3		Normal transform: not normalized afterwards		1			1.5			
785257	9	9	0	1	2.5	2.5	3		R2: Normals not normalized; R3: Normals not set correctly								
785325	8.5	8.5	0	1	2.5	2	3		R2: normals inverted; R3: normals are given values from positions vector; VCS: claim this point in a later assignment by showing a log where you've made multiple commits;	0							
785354	22	10	12	1	3	3	3		Normal transormation: Normals should have w = 0; Simple camera; PLY: Triangulation does not always work	1	1		1	1.5	2	5.5	Cone is closed(0.5p), Animation(0.5p), PLY(2.5 p), Simplifier attempt (2p)
785367	18.5	10	8.5	1	3	3	3		Please specify separately next time how much time you spent on requirements and extras. Transforming normals: modelToWorld_1_final should be transpose of modelToWorld_1_inv, normale should have w = 0, and normalizzata.xyz should be the one normalized; Viewport: Your initial field of view is not actually 90 degrees because tan() takes in radians, so fieldOFV should be initialized as pi / 2, not 90 (-0p). No adjustable FoV; Camera is not actually a trackball; PLY: Triangulation does not always work.	1	1		1	1	1	3.5	Cone is closed (0.5p), Animation (0.5p), PLY (2.5p)
785435	0	0	0														

Student number	point total	req total	extra total	R1 moving (1p)	R2 cone (3p)	R3 unpack (3p)	R4 loader (3p)	mod	notes / comments / ...	VCS (1p)	Rotate and scale (1p)	Normal trans. in shader (1p)	Camera (max 3p)	Normal trans. in uniform (2p)	Viewport & perspective (0.5-2p)	other (?p)	what other extras?
785448	0	0	0	0	0	0	0		Code doesn't compile or make much sense -- you don't have to do any OpenGL directly unless the requirement explicitly asks for it.								
785451	10.5	9.5	1	1	2.5	3	3		R2: normals inverted	1							
785493	10.5	9	1.5	1	2	3	3		R2: normal should be the normalized cross product of edge directions, not positions (for example, e0 = v1. position-v0.position and e1 = v2.position-v0.position and then normal = cross(e0,e1).normalized())	1	0.5						
785503	0	0	0														
785516	0	0	0														
795551	10	9	1	1	2	3	3		R2: Brackets missing from around i+1, and normals inverted	1							
795577	11	10	1	1	3	3	3			1							
795593	12	10	2	1	3	3	3		You can fix the GUI by setting glViewport back to the default size (0,0,width,height) after doing your own rendering. The GUI drawing code doesn't set the viewport itself and just assumes it is set to the default.	1					0.5	0.5	Animation(0.5p)
795629	0	0	0														
795658	0	0	0														
795674	12	9	3	1	2	3	3		R2: Normal is a cross product between the edge vectors, not the vertices themselves.	1	1					1	Simplifier attempt(1p)
795713	15	10	5	1	3	3	3		Normals should be transformed with the inverse transpose of the model to world matrix; PLY without normals and triangulation, binary loader crashes on files other than the provided one.	1	1	0	0.5			2.5	Animation(0.5p), PLY(2p)
795865	17	9	8	1	2.5	3	2.5		R2: normals inverted, R4: normal indices incorrect (to fix, initialize arrIncrementNorm to 1 and increment it by 2 in the loop), Camera is simple but nice, Normals not normalized after transform. The extras are just that; we don't assume you'll have the time or interest to implement every single one.	1	1		2	1.5	2	0.5	Animation (0.5p)
796178	14.5	10	4.5	1	3	3	3		No points reduced for this, but your camera becomes significantly more intuitive if you switch the order of the x and y rotations.	1	1		2		0.5		
798257	0	0	0														
801131	0	0	0														
804646	12	10	2	1	3	3	3			1	1						
807711	7	7	0	1	3	3			VCS: Did not find log or screenshot of version control. Please return it next round for points; R1: z-axis translation missing (-0p);	0							
809609	11.5	10	1.5	1	3	3	3		Please let us know the hours you spent on the assignment next time; Rotation and scale: Scale is uniform and cos(rotate_object_y) is missing from modelToWorld.m00.	1	0.5						
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	14.5	10	4.5	1	3	3	3			1	1					2.5	Animation(0.5p), simple PLY (2p)
64879R	15.5	10	5.5	1	3	3	3		Normals not normalized after transform, peculiar FOV choice	1	1			1.5	1.5	0.5	Animation(0.5p)
65451T	0	0	0														
67932J	10	9	1	1	3	3	2		R4: Values read for positions and normals should have been left unchanged (remove -1s).	1							
69246M	18	10	8	1	3	3	3			1	1		3	2	0.5	0.5	Animation(0.5p)
77241H	0	0	0														
77388B	9	9	0	1	2.5	3	2.5		R2: missing parentheses (multiplying by i and then adding 1), R4: OBJ indexes from 1; you should subtract one from each of the elements in f.								
83107B	0	0	0														
83854J	10.5	10	0.5	1	3	3	3						0.5				
84171B	10	9	1	1	2	3	3		Please let us know the hours you spent on the assignment next time; R2: tip not at origin, normals should be cross product of edge vectors, not positions.	1							
84805K	0	0	0														
k28342	0	0	0														
k90624	0	0	0														
k93517	0	0	0														