Ονοματεπώνυμο	Νικόλαος Γέροντας
Αριθμός Μητρώου	1092813

Περιεχόμενα

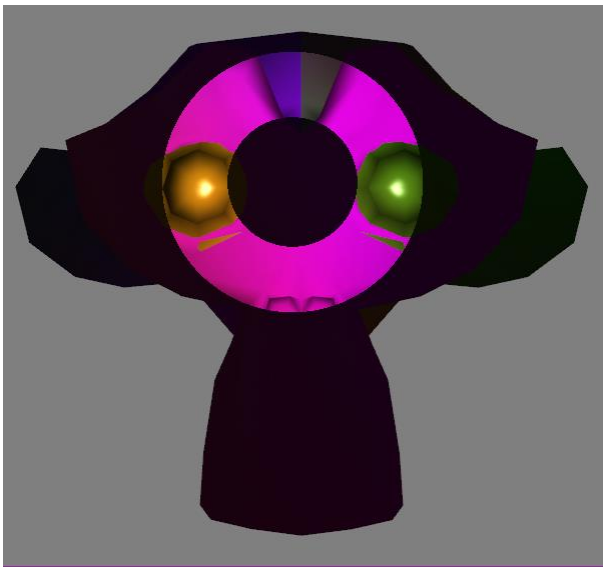
Lab - 05

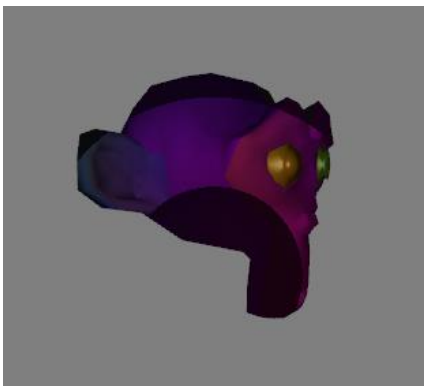
HOMEWORK 1

Modify the code for the spotlight in order to create a “ring light”. Use rough attenuation.

Απάντηση:

Στο παρόν ερώτημα επεκτείνεται η λογική του spotlight από το Task 7 ώστε να δημιουργηθεί ένα ring light. Ορίζονται 2 cutoff, ένα εσωτερικό και ένα εξωτερικό (cos(radians(8.0)) = 0.9903 & cos(radians(4.0)) = 0.9976). Στη συνέχεια, μέσω μιας διακλάδωσης, ελέγχεται αν η τιμή spotlight_arc βρίσκεται μεταξύ των 2 ορίων. Αν το spotlight_arc ανήκει σε αυτό το διάστημα, το spotlight_factor τίθεται ίσο με 1, διαφορετικά παραμένει 0 (rough attenuation). Έτσι φωτίζεται μόνο η δακτυλιοειδής περιοχή που θέλουμε. Τέλος, το spotlight_factor χρησιμοποιείται ως συντελεστής στους όρους diffuse και specular, ώστε το τελικό fragment_color να ενισχύεται μόνο μέσα στο δαχτυλίδι που πρέπει να φωτίζεται.



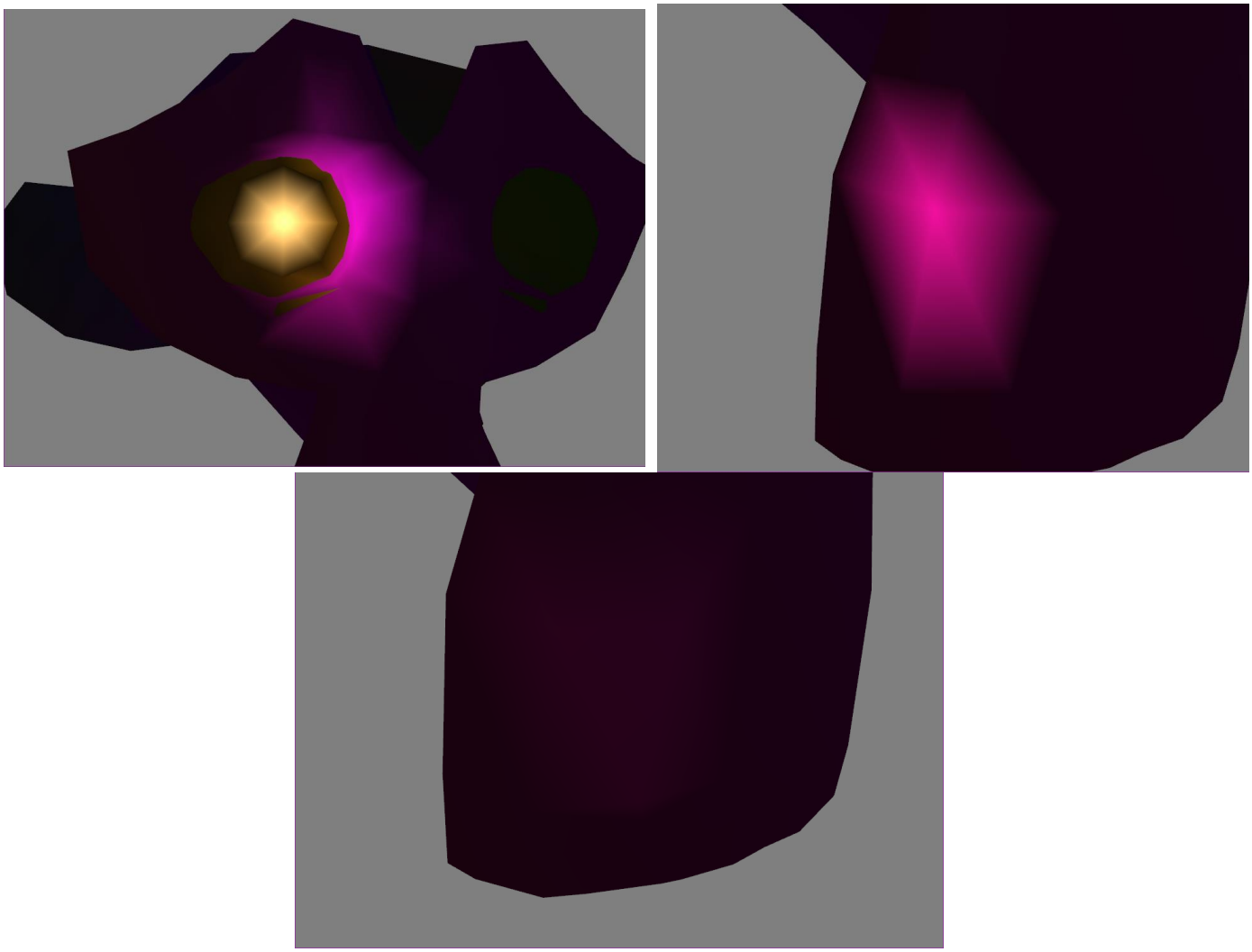


HOMEWORK 2

Implement Gouraud shading. Transfer all the shading calculations from “PhongShading.fragmentshader” over to “GouraudShading.vertexshader”. Calculate the vertex color (instead of the fragment color) and send the interpolated color data to the fragment shader.

Απάντηση:

Για την υλοποίηση του ερωτήματος, μετέφερα τη λογική φωτισμού του Phong Shading από το PhongShading.fragmentshader στο GouraudShading.vertexshader, έτσι ώστε ο φωτισμός να υπολογίζεται πλέον ανά κορυφή (vertex_color). Επιπλέον, υλοποίησα το εφέ φακού και στην εκδοχή με Gouraud, ώστε να γίνονται πιο εμφανή τα προβλήματα της συγκεκριμένης προσέγγισης!

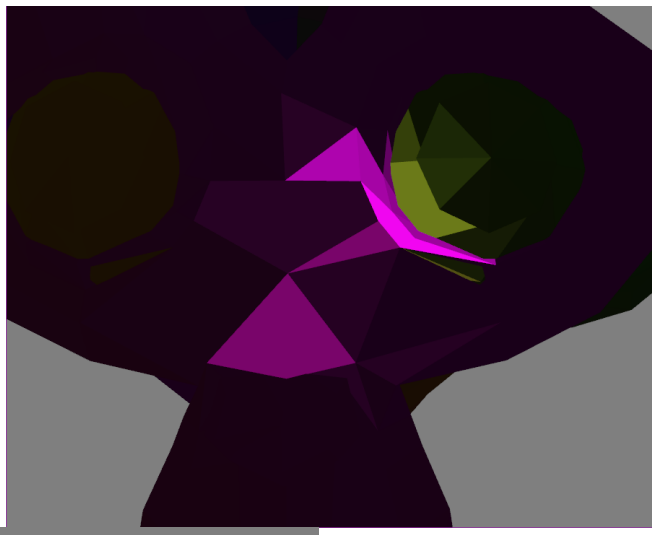


HOMEWORK 3

Implement flat shading. Transfer all the shading calculations from “GouraudShading.vertexshader” over to “FlatShading.vertexshader”. Using the “flat” auxiliary qualifier, disable the interpolation of the color data that is being sent to the fragment shader.

Απάντηση:

Για την υλοποίηση του flat shading ακολούθησα την ίδια λογική με το Gouraud Shading, μεταφέροντας δηλαδή όλους τους υπολογισμούς του φωτισμού (vertex_color) στο FlatShading.vertexshader. Η βασική διαφορά είναι ότι εδώ δήλωσα το vertex_color **ως flat out** στο vertex shader (και αντίστοιχα flat in στο fragment shader), ώστε να απενεργοποιηθεί η παρεμβολή (interpolation) του χρώματος ανάμεσα στις κορυφές. Έτσι, σε κάθε τρίγωνο δίνεται ένα ενιαίο χρώμα!



HOMEWORK 4

Make model materials as uniform variables and display multiple instances of the model with different materials.

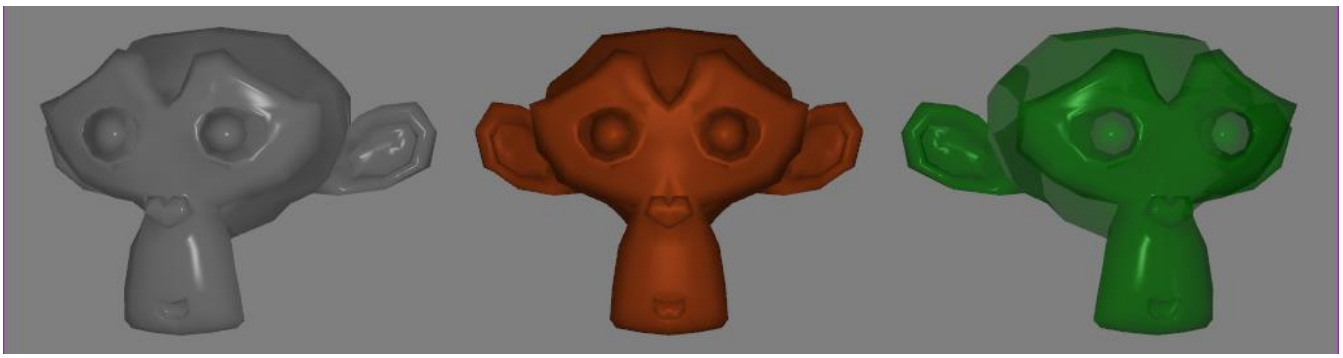
Απάντηση:

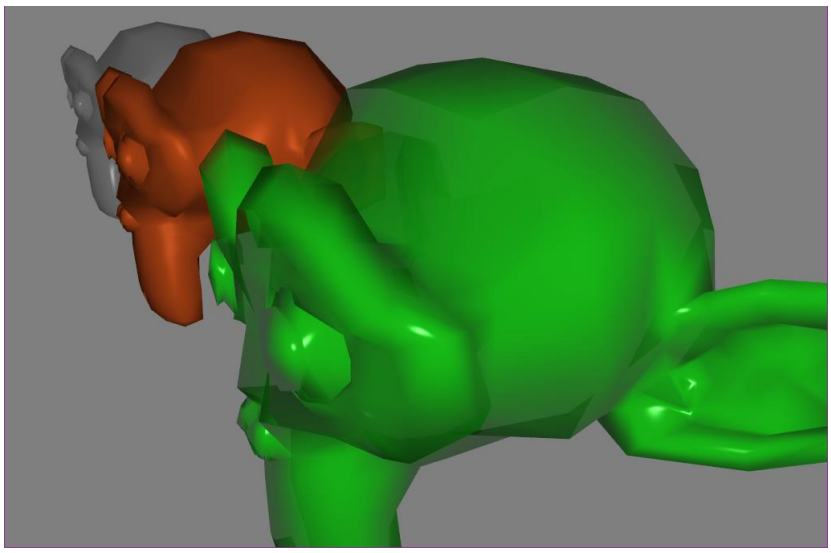
Τα χαρακτηριστικά του υλικού για το μοντέλο (ambient, diffuse, specular και shininess) υλοποιήθηκαν ως uniform μεταβλητές στον fragment shader (συγκεκριμένα, στην υλοποίηση Phong Shading). Παράλληλα, ορίζεται η δομή Material και ένα vector που περιέχει 3 διαφορετικά υλικά (chrome, copper και emerald). Στο mainLoop, για κάθε υλικό ενημερώνονται τα αντίστοιχα uniforms και η ίδια η Suzanne σχεδιάζεται πολλαπλές φορές, μετατοπισμένη κατάλληλα κατά μήκος του άξονα x, έτσι ώστε να εμφανίζονται πολλαπλά instances του ίδιου αντικειμένου με διαφορετικά υλικά.

```
struct Material
{
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
};
```

Ιδιότητες των υλικών που επέλεξα

Chrome	ambient	0.25 0.25 0.25 1
	diffuse	0.4 0.4 0.4 1
	specular	0.774597 0.774597 0.774597 1
	shininess	76.8
Copper	ambient	0.19125 0.0735 0.0225 1
	diffuse	0.7038 0.27048 0.0828 1
	specular	0.256777 0.137622 0.086014 1
	shininess	12.8
Emerald	ambient	0.0215 0.1745 0.0215 0.55
	diffuse	0.07568 0.61424 0.07568 0.55
	specular	0.633 0.727811 0.633 0.55
	shininess	76.8





HOMEWORK 5

Make light properties as uniform variables and use the keyboard keys to adjust them (light position, color and power).

Απάντηση:

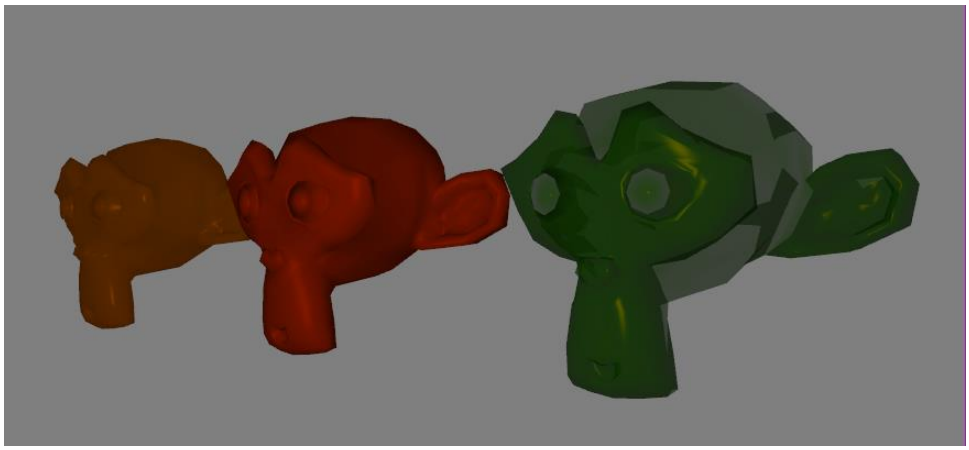
Στο παρόν ερώτημα, υλοποίησα την παραμετροποίηση των ιδιοτήτων του φωτός μέσω uniform μεταβλητών, ώστε να ρυθμίζονται από το πληκτρολόγιο. Στο lab.cpp ορίζω global μεταβλητές για τη θέση, το χρώμα και την ισχύ του φωτός (lightPos, lightColor, lightPower), βρίσκω τις θέσεις των uniforms light_position_worldspace, lightColor και light_power από τα shader και τις ενημερώνω σε κάθε frame πριν σχεδιαστούν τα μοντέλα.

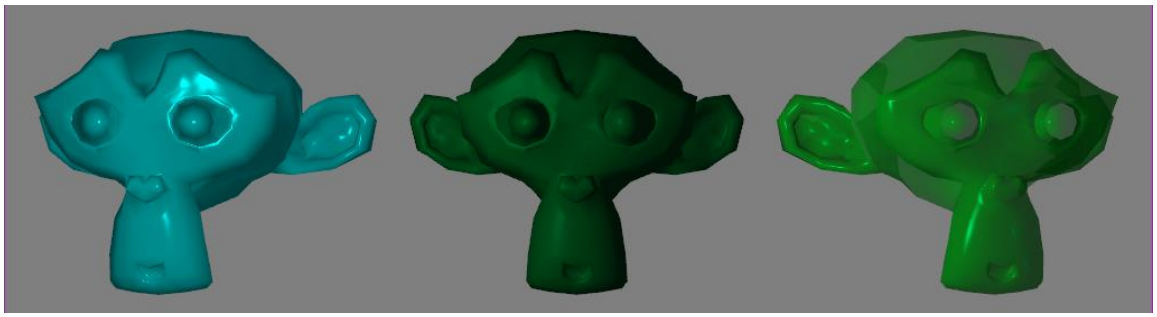
Στη συνάρτηση pollKeyboard αντιστοιχίζω πλήκτρα για την μετακίνηση της πηγής φωτός στους 3 άξονες (U/O, J/L, I/K), για αύξηση/μείωση των συνιστωσών του χρώματος (R/T, G/H, B/N) με χρήση clamp στο [0, 1], καθώς και για έλεγχο της ισχύος με τα πλήκτρα +/- μέσα σε προκαθορισμένο εύρος [0, 50].

Στον fragment shader πρόσθεσα τα uniforms lightColor και light_power, χρησιμοποίησα το lightColor για τον καθορισμό των La, Ld, Ls και αντικατέστησα τη σταθερή τιμή του light_power στον υπολογισμό της απόσβεσης (που υλοποιήσαμε κατά την διάρκεια του εργαστηρίου) με το αντίστοιχο uniform.

Παράδειγμα κατά την εκτέλεση:

https://youtu.be/HrzeHCDGSpU





HOMEWORK 6A

Display three objects each being shaded by a different algorithm (flat, Gouraud, Phong) at the same time. You will have to switch between three shader programs and get the uniform locations for each one.

Απάντηση:

Αρχικά, δημιουργήθηκε η βοηθητική δομή ShaderInfo για την οργάνωση του Program ID και των Uniform locations κάθε shader. Ένα vector<ShaderInfo> αρχικοποιείται στην createContext() περιλαμβάνοντας τους 3 ζητούμενους αλγορίθμους shading. Στην mainLoop, γίνεται εναλλαγή μεταξύ των αποθηκευμένων shaders, ώστε να σχεδιαστούν 3 διαφορετικά instances της Suzanne, εφαρμόζοντας στο καθένα την αντίστοιχη μέθοδο shading!

Παράδειγμα κατά την εκτέλεση:

https://youtu.be/NAN0ZgM145w

Στις εικόνες που ακολουθούν, η πηγή φωτός βρίσκεται ακριβώς πίσω από κάθε ένα κρανίο της Suzanne:

Phong **Gouraud** **Flat**