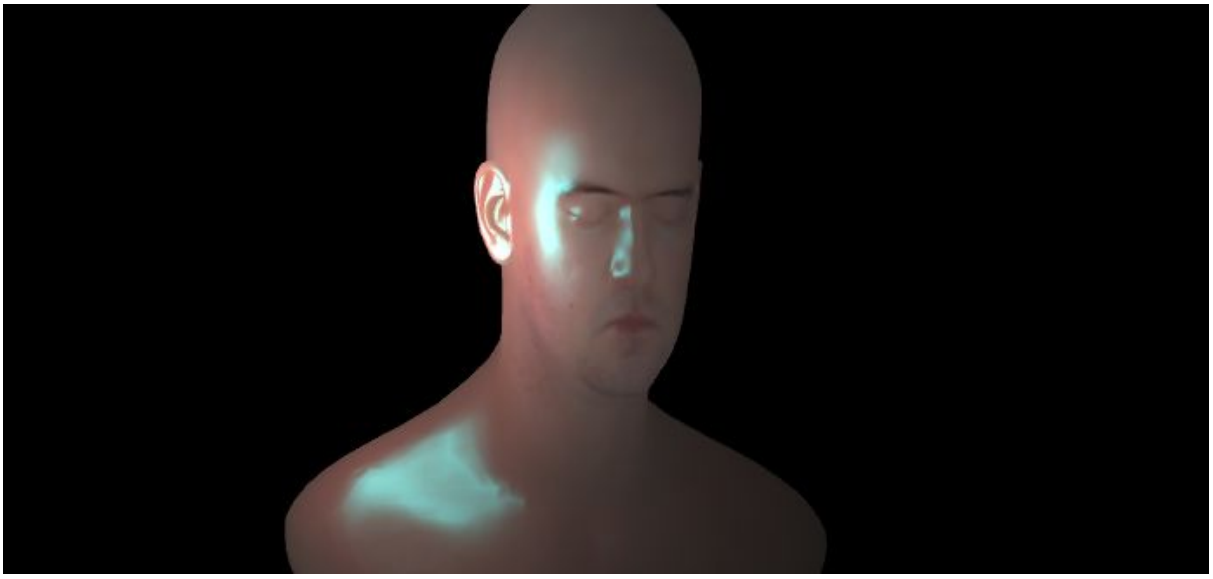
GRÀFICS I VISUALITZACIÓ DE DADES

PRÀCTICA 2



Albert Morales, Sergi Martorell i Arnau Vancells
13/05/2018

Introducció

En aquesta segona pràctica de Gràfics i Visualització de Dades hem començat a utilitzar OpenGL, per tal d'aprofitar l'execució en paral·lel que ens ofereixen les targetes gràfiques, per tal de representar objectes en una escena.

Per a fer això, programarem els shaders de la gràfica(vertex i fragment shader), per tal de representar els colors dels pixels de l'escena com vulguem.

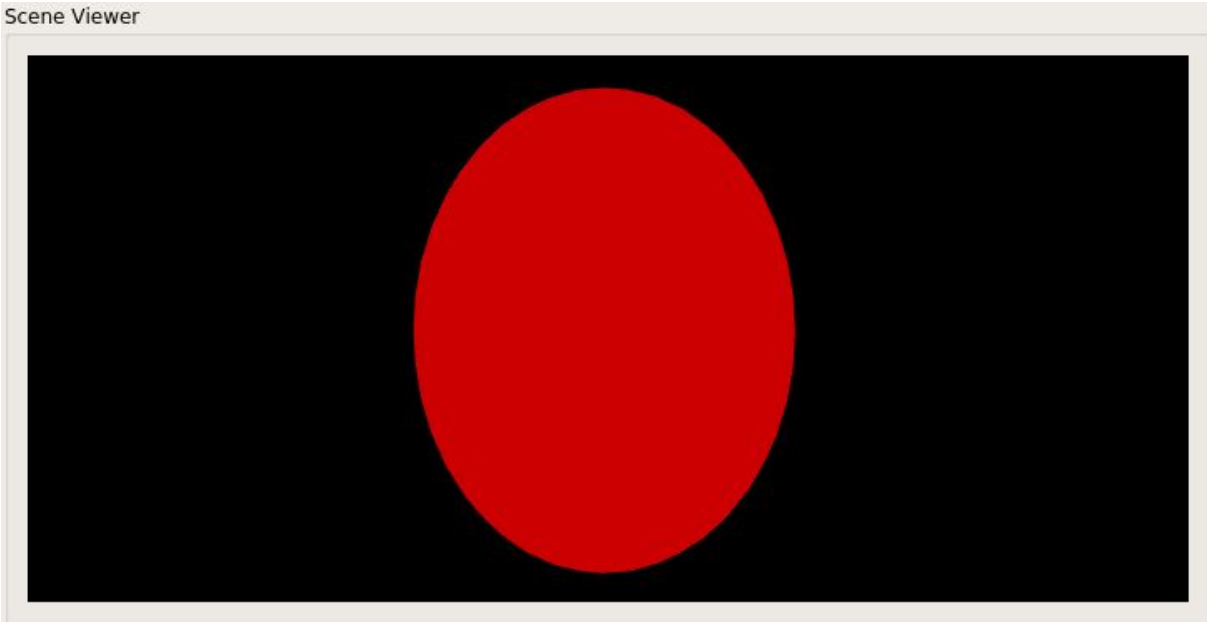
Fase 1

Per començar, se'ns demana que creem una nova classe Material, dins el projecte proporcionat pels professors de l'assignatura.

Per a crear aquesta classe Material hem re-aprofitat codi de la pràctica 1. Aquesta classe defineix les propietats òptiques d'un objecte, igual que a la primera pràctica, però ara no diferenciarem entre diferents tipus de materials.

Un cop programat el material, enviant les seves propietats a la gràfica, podem pintar un objecte a partir dels valors del material:

Imatge d'una esfera amb el color difús del material



Un cop ja tenim les propietats dels objectes programades, podem procedir a afegir efectes d'il·luminació.

Per a fer això hem d'afegir una classe Light, definir les seves propietats, i implementar el pas de CPU a GPU de Light, tal com hem fet amb la classe Material.

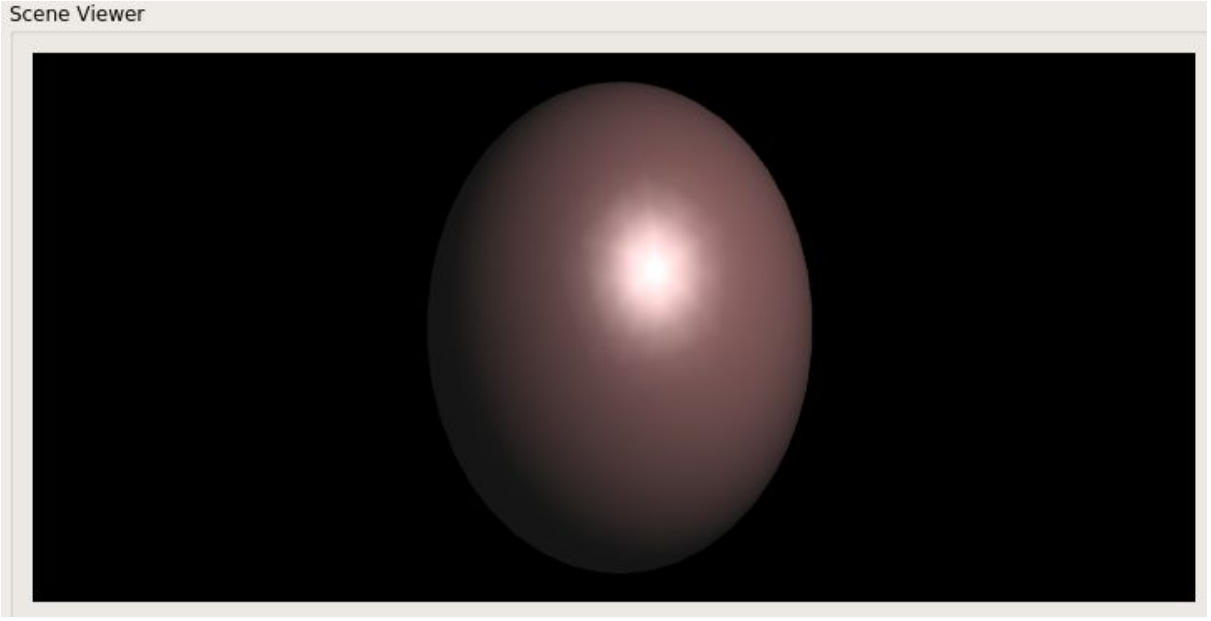
Hem de tenir en compte que podem tenir més d'una llum, per això, les dades enviades a GPU funcionen com un vector, i caldrà tenir-ho en compte a l'hora de llegir les dades als shaders.

Implementat Light, ja podem programar els shaders. Per a implementar els shaders hem aprofitat codi de la pràctica 1, per tal d'utilitzar blinn-phong per a fer la projecció de les llums.

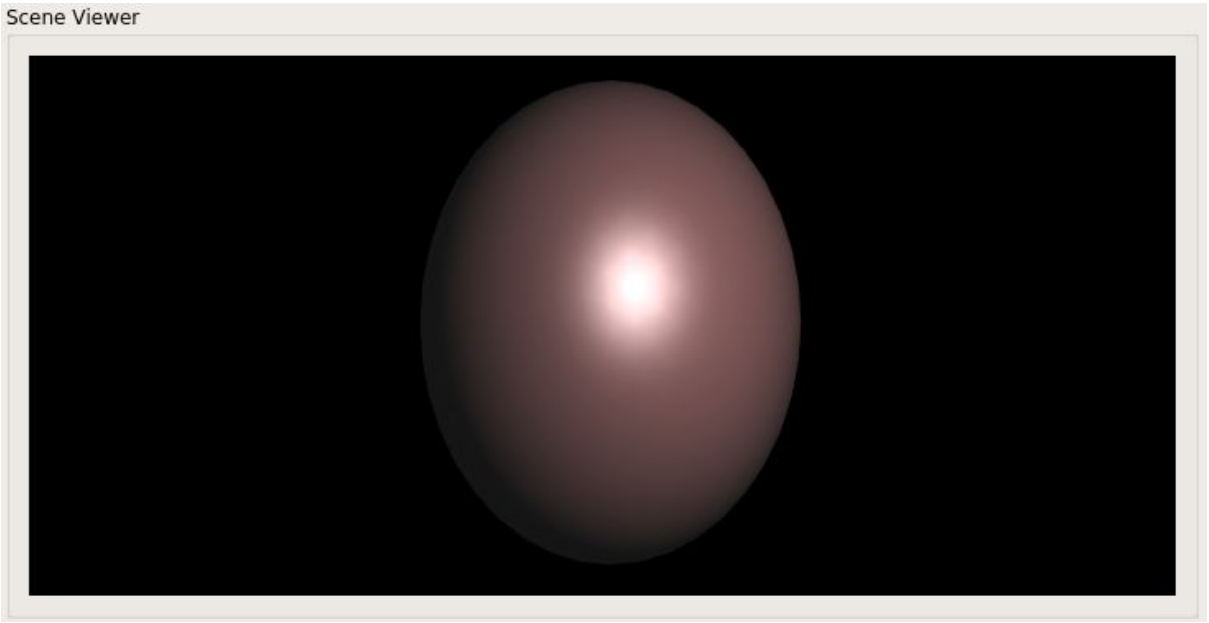
Per a implementar el shader Gouraud, farem el càlcul del color al vèrtex shader, ja que calcularem el color de cada vertex, i llavors deixarem que la gràfica n'interpoli els valors dels colors dels vertexs.

En canvi, pel shader Phong, el càlcul del color es realitzarà a nivell de pixel, per això estarà programat al fragment shader, ja que volem que es calculi el color per a cada pixel del nostre viewport.

Esfera (amb les propietats de l'exemple del campus) fent servir Gouraud shader

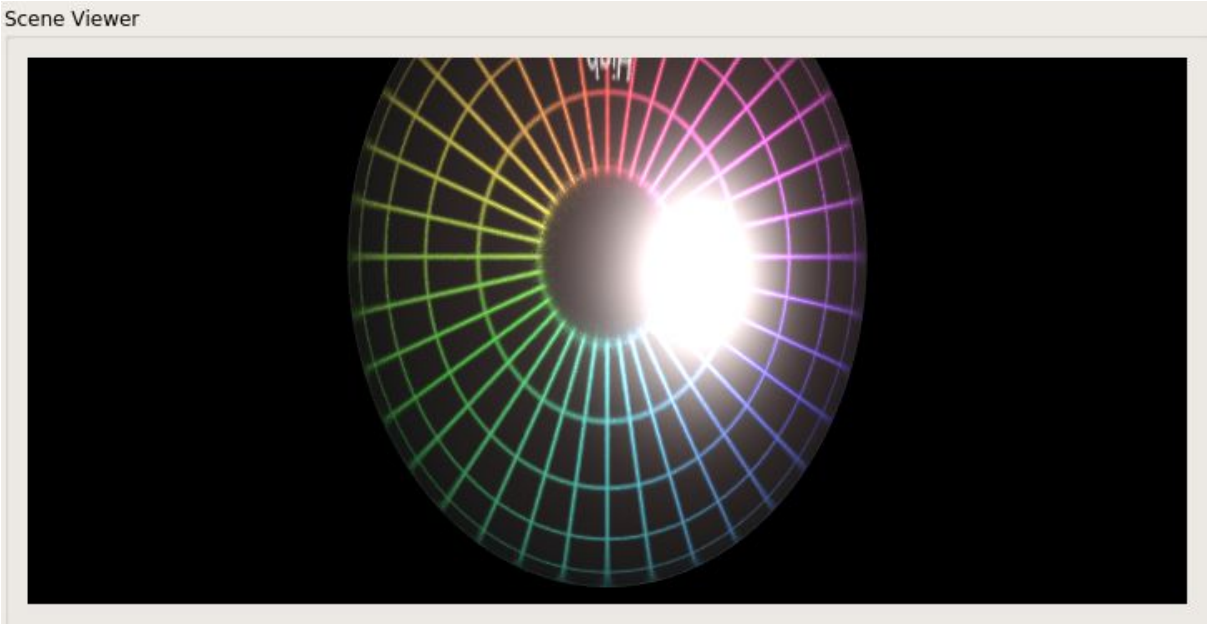


Esfera (amb les propietats de l'exemple del campus) fent servir Phong shader

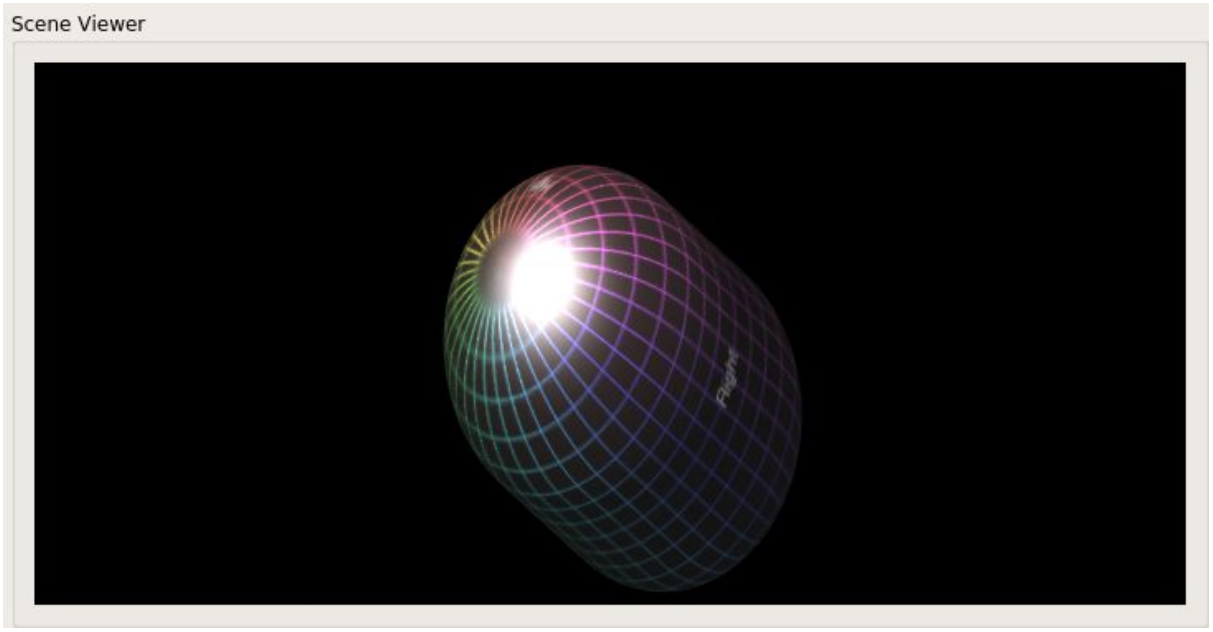


Tenint els shaders implementats, podem afegir l'ús de textures en els objectes.
Per a incloure textures utilitzarem mapping directe, fent servir les posicions a la textura carregat des del fitxer .obj.
Per a aplicar la textura ho farem fent servir una ponderació del 75% de color de textura, i 25% del color del material. Per a implementar-ho hem utilitzat com a base el phong shader, i l'hem modificat per a que llegeixi les cordenades de textura i ho apliqui als objectes.

Càpsula (amb les propietats de l'exemple del campus) fent servir PhongTex shader



Càpsula fent servir PhongTex shader, amb l'imatge moguda

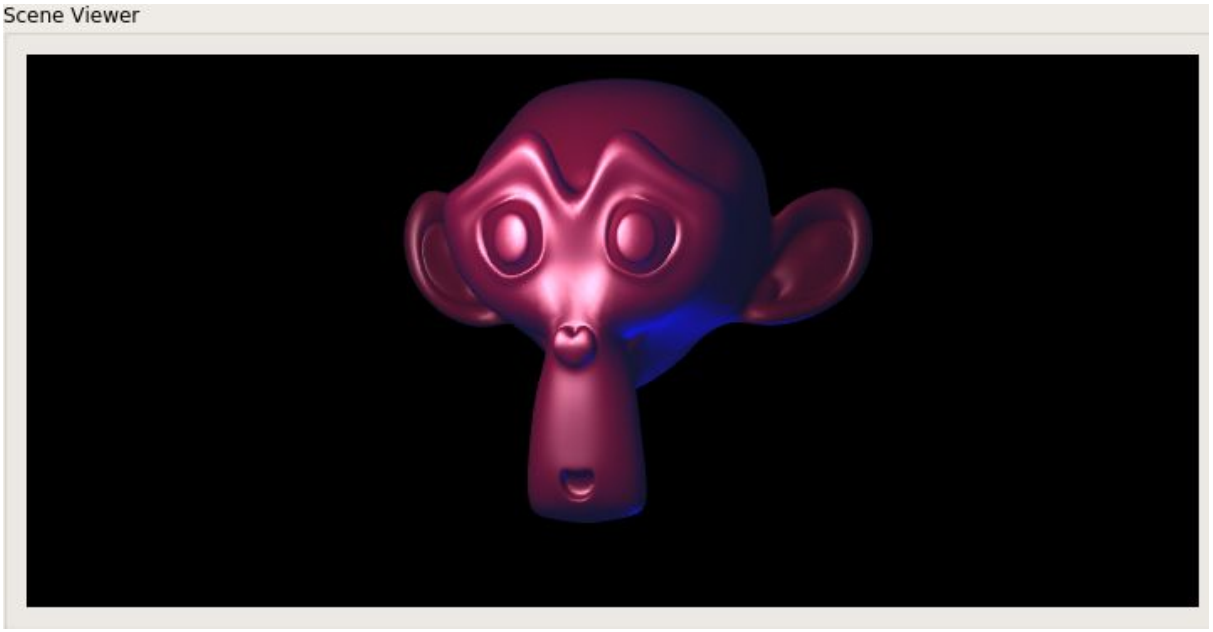


Parts Opcionals

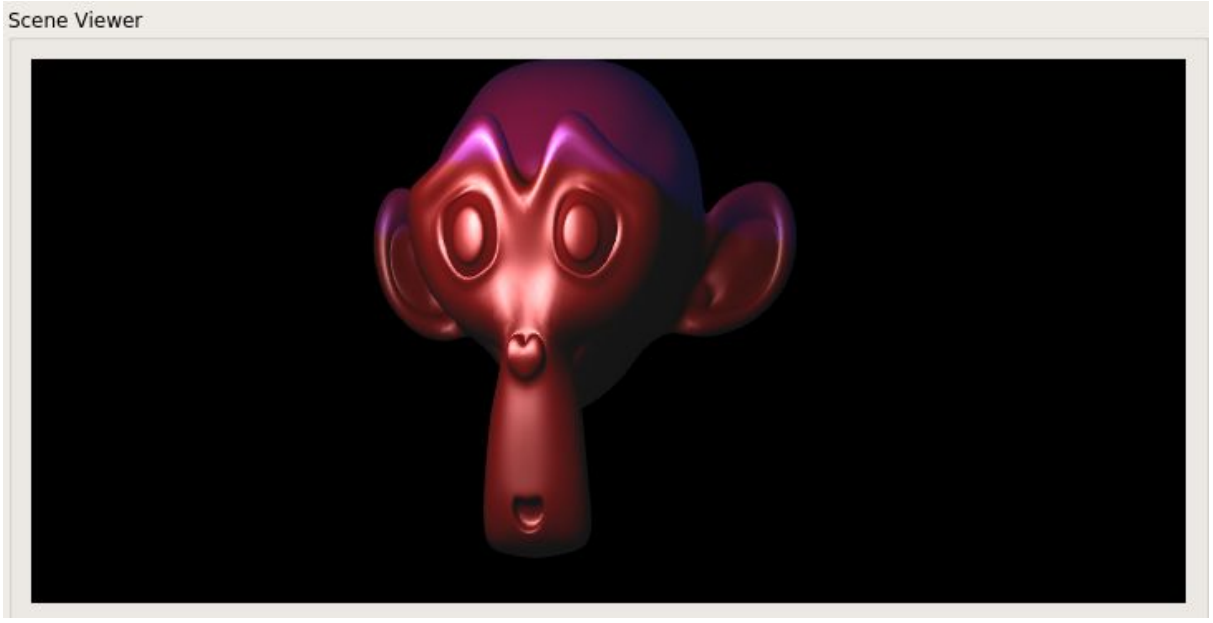
Spotlight

Hem modificat els diferents shaders, per a que es tingui en compte el tipus de llum que hi ha a l'escena. Com a extra, hem implementat la llum Spotlight, afegint a light els paràmetres necessàris(angle i direcció del con). Aquests valors no genereran problemes si es tracta d'un altre tipus de llum.

MonkeyTex amb una llum Puntual i una llum direccional de color blau



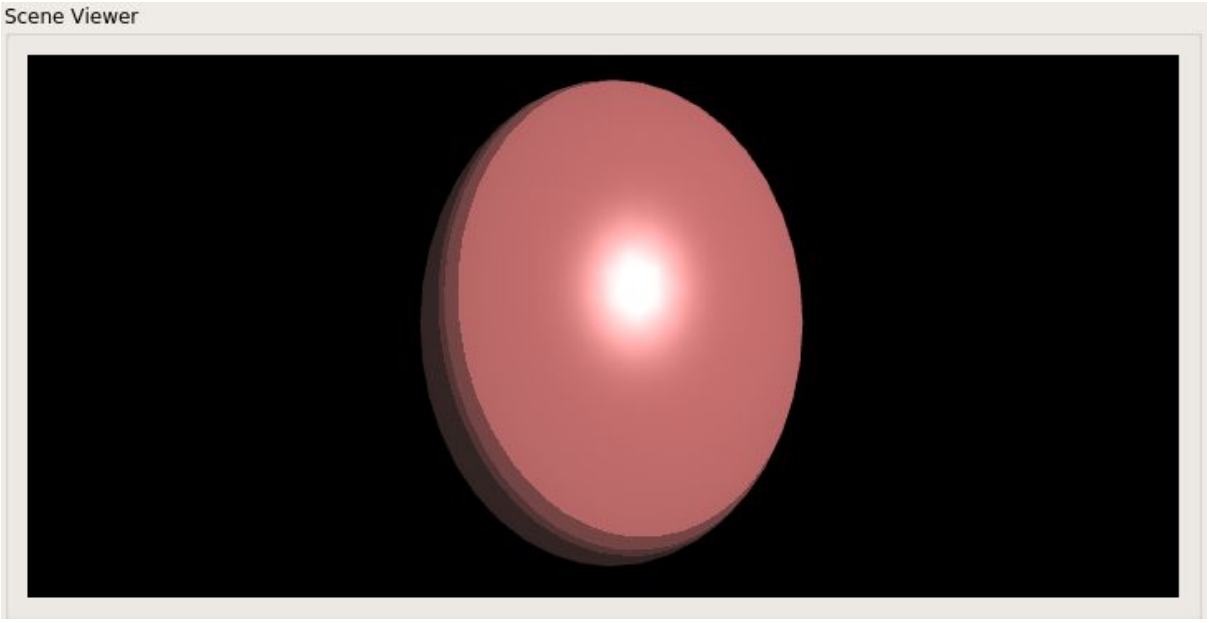
MonkeyTex amb una llum Puntual i una Spotlight de color blau



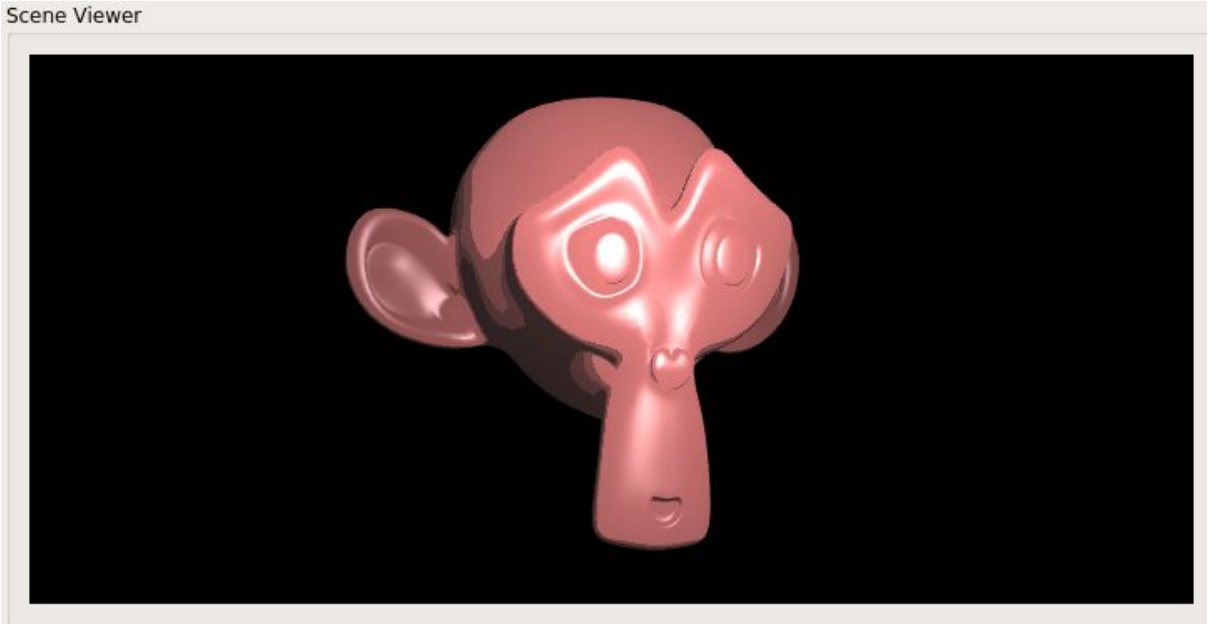
Toon Shading

Seguint les instruccions de la pràctica, i de les diapositives de teoria, hem fet un nou shader, a partir del phong shader, que implementa un shading discret segons el producte escalar del vector de la llum direccional amb la normal al punt.
Hem seleccionat una gamma de colors vermella per a definir el color dels diferents intervals(els mateixos colors que hi ha a les diapositives de teoria).

Esfera (amb les propietats de l'exemple del campus) fent servir Toon shader



MonkeyTex fent servir Toon shader



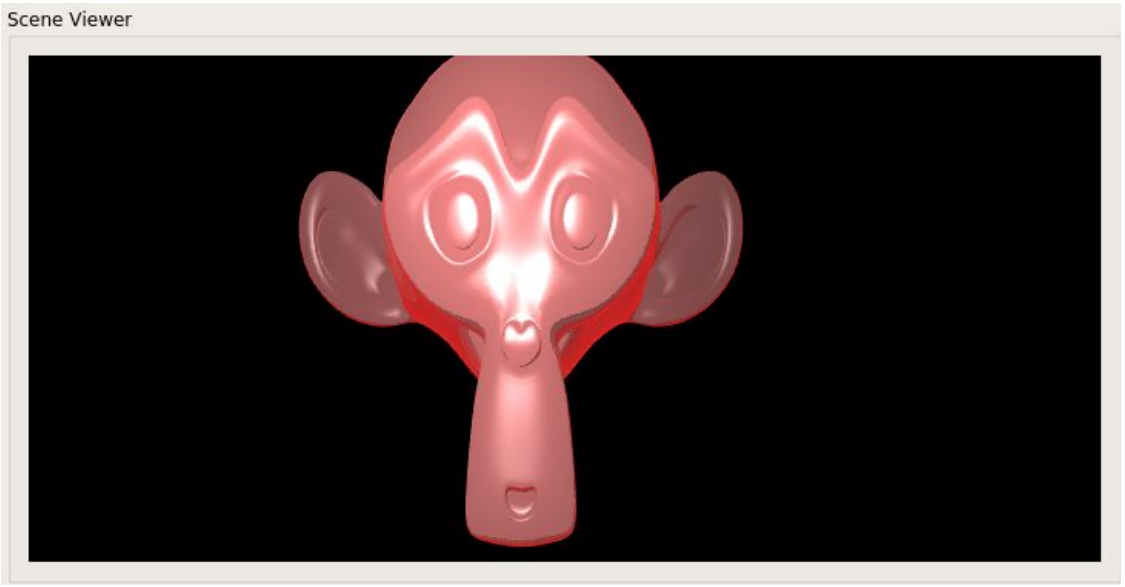
Èmfasis de siluetes

Al toon shader li podem afegir èmfasis a les siluetes dels objectes, això ho fem calculant el color a partir de l'angle entre el vector de visió i la normal dels pixels.

Hem vist que aplicant directament com ho diu a la pràctica, els pixels que tenen un angle pràcticament a 0 entre la normal i el punt de visió, es queden molt enfosquits.

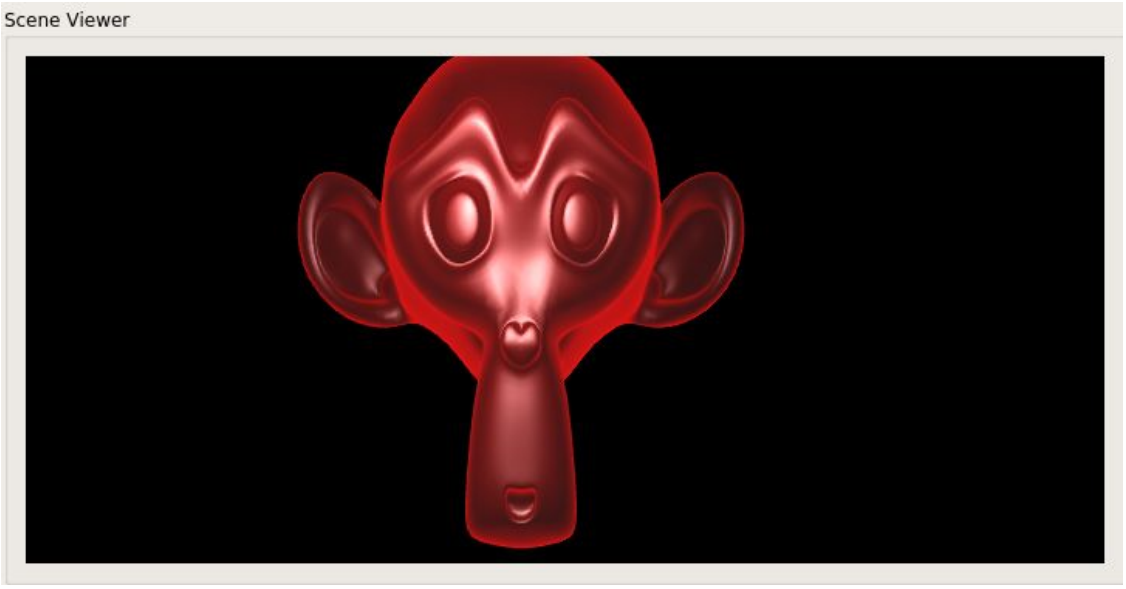
Per això prenem com a color final, el màxim entre el color calculat per a la silueta, i el color del shading original. D'aquesta forma, s'augmentarà el color dels contorns, però a la resta de llocs es mantindrà igual.

MonkeyTex fent servir Toon shader amb èmfasis de siluetes

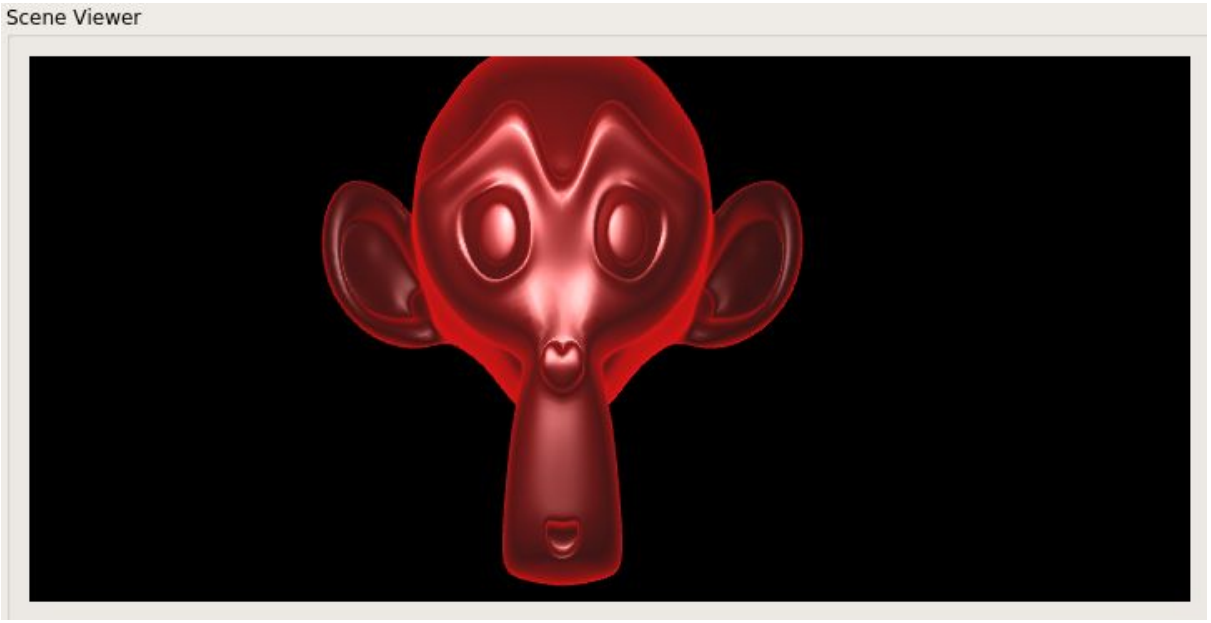


Hem afegit també, tant al phong com al gouraud shader, l'èmfasis de silueta(en el codi hem deixat el codi comentat, ja que aquesta part és opcional).

MonkeyTex fent servir Gouraud shader amb èmfasis de siluetes



MonkeyTex fent servir Phong shader amb èmfasis de siluetes



Mapping indirecte

Quan carreguem un objecte que no té les coordenades de textura al fitxer object, podem utilitzar mapping indirecte. Per això hem modificat la classe object, per a que carregui les coordenades de textura amb un valor de (-1.0, -1.0), de manera que al shader de textura, si llegim aquest valor, podem generar coordenades de textura fent servir mapping indirecte(el valor de les coordenades de textura va entre 0 i 1, de manera que utilitzar -1 no ens causarà problemes).
Per calcular les coordenades hem fet servir les fòrmules proporcionades a la pràctica.

Esfera fent servir PhongTex shader amb mapping indirecte