КТМУ ТИК
КОМПЬЮТЕРДИК ИНЖЕНЕРИЯ

КОМПЬЮТЕРДИК ГРАФИКАДА ЖАРЫКТАНДЫРУУНУ МОДЕЛДӨӨ
(МАГИСТРДИК ДИССЕРТАЦИЯ)

Махабат КУЛМУРЗАЕВА

БИШКЕК
2017

КЫРГЫЗ - ТҮРК “МАНАС” УНИВЕРСИТЕТИ

ТАБИГЫЙ ИЛИМДЕР ИНСТИТУТУ

КОМПЬЮТЕРДИК ИНЖЕНЕРИЯ БӨЛҮМҮ

КОМПЬЮТЕРДИК ГРАФИКАДА ЖАРЫКТАНДЫРУУНУ МОДЕЛДӨӨ

(МАГИСТРДИК ДИССЕРТАЦИЯ)

Махабат КУЛМУРЗАЕВА

БИШКЕК 2017

КЫРГЫЗ - ТҮРК “МАНАС” УНИВЕРСИТЕТИ

ТАБИГЫЙ ИЛИМДЕР ИНСТИТУТУ

КОМПЬЮТЕРДИК ИНЖЕНЕРИЯ БӨЛҮМҮ

КОМПЬЮТЕРДИК ГРАФИКАДА ЖАРЫКТАНДЫРУУНУ МОДЕЛДӨӨ

Даярдаган

Махабат Кулмурзаева

Илимий жетекчи

Доцент, ф. м. и .к Замиргүль Казакбаева

Магистрдик диссертация

Май 2017

БИШКЕК, КЫРГЫЗСТАН

ИЛИМИЙ ЭТИКАГА ШАЙКЕШТИГИ

Бул диссертацияда колдонулган бардык маалыматтар академиялык жана этикалык эрежелерге жооп берет. Ошондой эле, башка илимий булактардан алынган маалыматтарга “ АДАБИЯТ ” бөлүмүндө шилтемелер көрсөтүлдү.

Махабат Кулмурзаева

Колу

ЧЕЧИМ

ф. м. и. к., доцент Замиргүль КАЗАКБАЕВА жетекчилиги астында Махабат КУЛМУРЗАЕВА тарабынан даярдалган “Компьютердик графикада жарыктандырууну моделдөө” темасындагы диссертация, калыстар тарабынан Кыргыз Түрк "Манас" Университети, Табигый Илимдер Институту Компьютердик инженерия бөлүмүндө Магистрдик диссертация катары кабыл алынды.

19.06.2017

КАЛЫСТАР:

Илимий жетекчи	Доц. док. Замиргүль Казакбаева	……
Төрагасы	ф. м. и. к., доц. Баратбек Сабитов	……...
Мүчө	т. и. д., проф. Улан Бримкулов	……...
Мүчө	т. и. к. Чинара Жумабаева	…….....
Мүчө	доц. м. а., др. Рита Исмаилова	……...

19.06.2017

Доцент, докт. Дагыстан Шимшек

Институт мүдүрү

КИРИШ СӨЗ

Диссертацияны жазууда ар тараптуу илимий кеңештерин аябаган, ф. м. и. к., доцент Замиргүль КАЗАКБАЕВА эжейиме, терең ыраазычылык билдермекчимин.

Махабат Кулмурзаева

Бишкек, Май 2017

КОМПЬЮТЕРДИК ГРАФИКАДА ЖАРЫКТАНДЫРУУНУ МОДЕЛДӨӨ

МАХАБАТ КУЛМУРЗАЕВА

КЫРГЫЗ-ТҮРК МАНАС УНИВЕРСИТЕТИ, ТАБИГИЙ ИЛИМДЕР ИНСТИТУТУ

МАГИСТРДИК ДИССЕРТАЦИЯ, МАЙ 2017

ИЛИМИЙ ЖЕТЕКЧИ: ДОЦ. ДОК. ЗАМИРГҮЛЬ КАЗАКБАЕВА

КЫСКАЧА МАЗМУНУ

Компьютердик графика, эки жана үч өлчөмдүү моделдөө, ар кандай мүнөздөгү графикалык сүрөттөлүштөрдү компьютердик програм жана алгоритмдердин жардамы менен түзүү дегенди билдирет. Үч өлчөмдүү графиканы түзүүдөгү чоң маселе бул – сценаны чындыкка жакын кылып жарыктандыруу. Туура жарыктандыруу буюмдардын формасын жакшыраак берип тим болбостон, сценага жалпы маанай берет. Жарыкты моделдөө өтө оор, анткени ал – татаал система.

Жарыктандырууну туура моделдөө маселеси, сценанын жарыктандыруусун эсептөө үчүн, ар кандай алгоритмдердин иштелип чыгышына алып келди. Бирок бардык эле алгоритмдер бирдей жыйынтык бербейт. Кээ бир моделдер тез иштейт, кээ бирөөсү жакшы сүрөттөлүш берет, бирок татаал эсептөөлөрдү талап кылып, интерактивдүү тикемелер үчүн жарабайт. Ар бир учурда, түзүлө түрган виртуалдуу сцена канчалык татаал жана чындыкка жакын болсо, ошончо көп эсептөөлөр жүргүзүлүш керек, жана ал сүрөт экранга ошончолук жай чыгарылат.

Заманбап графикалык тиркемелерде үч өлчөмдүү сценаларды көрсөтүүнүн эки жолу колдонулат: глобалдуу жана локалдуу жарыктандыруу моделдери. Глобалдуу жарыктандыруунун бир нече артыкчылыктары бар: жогорку

деңгээлдеги чындыкка жакын сүрөт алуу, көлөкөлөрдү куруу үчүн кошумча алгоритмдер колдонулбайт. Локалдуу моделдер салыштырмалуу сапаты төмөн сүрөттөлүштү берет, бирок алар бат жана өндүрүмдүү болгондуктан интерактивдүү тиркемелерде ийгиликтүү колдонулуп келет. Локалдуу жарыктандыруу модели үч компоненттен турат: фондук жарык, диффуздук чагылуу жана спекулярдык (күзгүлүү) чагылуу. Компоненттер өз өзүнчө эсептелип кошулат.

Бул иште компьютердик графикадагы жарыктандыруу моделдеринин теоретикалык негиздери жана практикада ишке ашырылуусу каралды. Бул иште жети локалдуу жарыктандыруу модели каралды. Алар Ламберт, Wrap-around, Орен-Наяр, Миннеарт диффуздук чагылуу модели, Фонг, Блинн-Фонг, Вард спекулярдык чагылуу модели. Бардык моделдер үчүн фондук компонент бирдей. Ламберт, Wrap-around, Орен-Наяр жана Миннеарт моделдери жалгыз гана диффуздук чагылууну эсепейт, күзгүлүү чагылуу компоненти кошулбайт. Фонг, Блинн-Фонг, Вард моделдери күзгүлүү чагылууну моделдейт, ал эми диффуздук чагылуу Ламберт модели негизинде эсептелет.

Бул иш жыйынтыгында аталган моделдерди ишке ашырган реалдуу убакыттагы демонстрациялык графикалык тиркеме иштелип чыкты. Тиркемени колдонуп моделдер берген көрүнүштү көрсө болот, жана моделдер берген сүрөттөлүштөрдү салыштырса болот. Тиркеме Visual Studio 2017 (Community Edition) иштеп чыгуу чөйрөсүндө Vusial C++ тилинде, OpenGL графикалык китепканасын колдонуу менен ишке ашырылды. Жарыктандыруу моделдери OpenGL-дин шейдерлер тили – GLSL-де ишке ашырылды. Тиркеме ыңгайлуу интейфейске ээ жана жөнөкөй башкаруу элементтерин колдонот. Аппараттык чектөөлөрдөн улам моделдердин өндүрүмдүүлүгү салыштырылбады.

Үч өлчөмдүү компьютердик графика боюнча кыргыз тилинде адабият жокко эсе болгондуктан бул иш бул боштукту толтуруп, башка иштердин жазылуусуна өбөлгө болот деген ишенимдебиз.

Ачкыч сөздөр: Компьютердик графика, үч өлчөмдүү графика, жарыктандыруу, жарыктандыруу моделдери, үч өлчөмдүү моделдер, жарык булагы, виртуалдуу сцена, OpenGL, GLSL.

MODELLING LIGHTING IN COMPUTER GRAPHICS

MAKHABAT KULMURZAEVA

Kyrgyz Turkish Manas University Institute Of Science

MASTER THESIS, MAY 2017

SUPERVISOR: Associate Professor ZAMIRGUL KAZAKBAEVA

ABSTRACT

Computer graphics - an area of activity that includes the creation and editing of various images on the computer using special algorithms and software, as well as two-dimensional and three-dimensional modeling. The biggest problem with creating 3D graphics is the realistic lighting of the scene. Proper lighting significantly enhances the impression of the scene. It is not only allows you to better convey the shape of objects, but also creates a general mood in the scene. With the help of bright colors and an abundance of light, one can get a holiday effect, and the muffled light and shaded objects creates a feeling of tension and anxiety.

Light is a very complex system to model it perfectly. That's why we rarely see the computer generated 3D images that would be truly photorealistic. In all cases, the more complex and realistic the virtual scene you create, the more calculations one needs to make, and the slower it will be played on the screen. Over time, computer graphics are becoming more complex, thus creating images has become a separate craft. The search for a solution to the problem of synthesis of photorealistic images led to the development of various algorithms for calculating the lighting of the scene. These algorithms differ in the speed and quality of the synthesized image. In all cases, the more complex and realistic the virtual scene one creates, the more calculations he needs to make, and the slower it will be played on the screen. In reality, a falling beam of light undergoes a huge number of reflections and refractions, while in computer graphics, the number of drops and reflections of the beam is determined only by the hardware capabilities of the computer.

Lighting plays a vital role in the three-dimensional graphics. Without lighting, the objects of 3D graphics look flat and artificial. Lighting provides visibility of the objects of the scene, and also gives the whole scene a sense of bulk and reality.

In modern graphics applications, two approaches to modeling the lighting of three-dimensional scenes are used: global illumination methods and local lighting models. The first approach gives a high realism of the resulting image, and does not require additional algorithms for building shadows from objects. In contrast to the first method, the algorithms for calculating local illumination give a less realistic image, but are much more productive, which allows them to be successfully applied in applications where interactive interaction with a three-dimensional scene is required. The local model of illumination consists of three components: background, diffuse and specular.

In this paper, the lighting models in the three-dimensional graphics and their implementation are examined. Seven local lighting models are considered: Lambert, Wrap-around, Oren-Nayar, Minneart, Phong, Blinn-Fong, Ward. For all lighting models, the background component is the same. Models Lambert, Wrap-around, Oren-Nayar and Minneart simulate only diffuse reflection, the mirror component does not exist. Models Fong, Blinn-Fong, Ward differ in the computation of the mirror component, and the diffuse component is calculated by the Lambert model.

In the end, a graphical real-time application was developed demonstrating these models in practice, so that you can change and compare them. The application was written in Vusial C ++ in the Visual Studio 2017 (Community Edition) development environment using the OpenGL graphics library. The lighting models themselves were implemented in the OpenGL shader language – GLSL. Due to the hardware limitations, the performance of the lighting models was not evaluated.

Keywords: Computer graphics, 3D graphics, lighting, lighting model, light source, virtual scene, OpenGL, GLSL.

МОДЕЛИРОВАНИЕ ОСВЕЩЕНИЯ В КОМПЬЮТЕРНОЙ ГРАФИКЕ

МАХАБАТ КУЛМУРЗАЕВА

Кыргызско Турецкий Университет "Манас", Институт Науки

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ, МАЙ 2017

НАУЧНЫЙ РУКОВОДИТЕЛЬ: Доцент, к. ф.-м. н. Замиргуль КАЗАКБАЕВА

АННОТАЦИЯ

Компьютерная гра́фика — область деятельности включающая в себя создание и редактирование различных изображений на компьютере при помощи специальных алгоритмов и ПО, а также двумерное и трехмерное моделирование. Самая большая проблема при создании трехмерной графики – это реалистичное освещение сцены. Правильное освещение позволяет лучше передать форму предметов. Свет — это очень сложная система, чтобы смоделировать ее в совершенстве. Поиск решения проблемы синтеза фотореалистичных изображений привел к разработке различных алгоритмов расчета освещения сцены. Эти алгоритмы отличаются скоростью и качеством синтезируемого изображения. Во всех случаях, чем сложнее и реалистичнее создаваемая вами виртуальная сцена, тем больше вычислений вы должны произвести, и тем медленнее она будет воспроизводиться на экран.

В современных графических приложениях используются два подхода к моделированию освещения трехмерных сцен: методы глобального освещения и локальные модели освещения. Первый подход даёт высокую реалистичность получаемого изображения, и при этом не требуется дополнительных алгоритмов для построения теней от объектов. В отличие от первого метода, алгоритмы расчета локального освещения дают менее реалистичное изображение, но намного производительнее, что позволяет их с успехом применять в приложениях, где требуется интерактивность взаимодействия с трехмерной сценой. Локальная

модель освещения состоит из трех компонентов: фоновая, диффузная (рассеянная) и спекулярная (зеркальная).

В этой работе рассмотрены семь локальных моделей освещения: Ламберт, Wrap-around, Орен-Наяр, Миннеарт, Фонг, Блинн-Фонг, Вард. Для всех моделей освещения фоновая составляющая одинаковая. Модели Ламберт, Wrap-around, Орен-Наяр и Миннеарт моделируют только диффузное отражение, зеркальное составляющая отсутсутвует. Модели Фонг, Блинн-Фонг, Вард отличаются вычисленим зеркальной составляющей, а диффузный компонент вычисляется моделью Ламберта.

В конце работы было разработано графическое приложение реального времени демонстрирующая эти модели на практике, можно посмотреть изображения дающие эти модели и сравнивать их. Приложение было написано на Vusial C++ в среде разработки Visual Studio 2017 (Community Edition) с использованием графической библиотеки OpenGL. Сами модели освещния были реализованы на языке шейдеров OpenGL – GLSL. Из-за аппаратных ограничений не была произведена оценка производительности моделей освещения.

Ключевые слова: Компьютерная графика, трехмерная графика, освещение, модели освещения, источник света, виртуальная сцена, OpenGL, GLSL.

BILGISAYAR GRAFIKLERINDE AYDINLATMAYI MODELLEME

MAHABAT KULMURZAYEVA

KIRGIZISTAN TÜRKIYE MANAS ÜNIVERSITESI, FEN BILIMLERI ENSTITÜSÜ

YÜKSEK LİSANS, MAYIS 2017

DANIŞMAN: ZAMİRGUL KAZAKBAYEVA

GENİŞ ÖZET

Bilgisayar grafikleri, bilgisayarların ve özel bir grafik donanımı ve yazılımının yardımıyla bir bilgisayar tarafından görüntü verisinin temsilini kullanarak oluşturulmuş grafiklerdir. Üç boyutlu grafikler oluşturulmasında karşılaşan en büyük sorun - bu sahneye gerçek aydınlatma vermektir. Doğru aydınlatma sahne izlenimini arttırır. O nesnelerin şeklini daha iyi gösterir ve aynı zamanda sahnenin genel bir ruh halini yaratır. Parlak renkler ve ışık bolluğu tatil efekti verir ve bastırılmış ışık ile gölgeli nesneler gerginlik ve endishe hissi uyandırır.

Aydınlatma üç boyutlu grafiklerde önemli bir rol oynar. Üç boyutlu grafik nesneleri, aydınlatma olmadıgı zaman düz ve yapay görünür. Aydınlatma sahnedeki nesnelerin görünürlüğünü sağlar, hem de tüm sahneye üç boyutluluk ve gerçeklik duygusu verir.

Işığı mükemmel şekilde modellemek için o çok karmaşık bir sistemdir. Bundan dolayı bilgisayarda yapılan fotogerçekçi üç boyutlu görüntüler nadiren karşılaşır. Ne kadar karmaşık ve gerçekçi görüntü almak istenirse, o kadar çok hesaplamaları yapmak gerekir, demek ekrana da yavaş oynanacak.

Zamanla bilgisayar grafikleri daha karmaşık hale geliyor ve bu nedenle görüntülerin oluşturulması ayrı bir zanaat haline geldi. Fotogerçekçi görüntülerin sentezinin sorununa çözümü arama sahne aydınlatması için çeşitli algoritmaların geliştirilmesine yol açmıştır. Bu algoritmaların hızı ve sentezlenmiş görüntülerin kalitesi

farklıdır. Bazı algoritmalar hızlı çalışıyor, diğerleri yüksek kaliteli görüntülerini verir, ancak uzun sureli hesaplamalar gerektirir.

Aslında, ışık demeti yansıma ve kırılmaya çok sayıda rastlanır ve bilgisayar grafiklerinde ise, ışının yansıma sayısı sadece bilgisayarın donanım özelliklerine göre belirlenir. Doğal aydınlatma prensiplerinin yazılımda uygulanması oldukça karmaşıktır. Dolayısıyla, interaktiv uygulamada, küçük hesaplamalar ile gerçekçi bir sonuç verecek, çeşitli aydınlatma modelleri kullanılır. Gerçek fotogerçekçi görseller için yoğun algoritmaları kullanmak gerekir.

Üç boyutlu grafiği programlamak için güçlü grafik işlemci (GPU) gerekir. Çağdaş grafik işlemcinin tüm imkanlarını etkili kullanabilmek için gerekli olan grafik arayüzleri (API) kullanılmalı. Böyle API uygulama ve grafik işlemci arasında bir köprü görevi yapar. Şu anda gerçek zamanlı üç boyutlu grafiği programlamak için, yaygın kullanılmakta olan iki tür API vardır. Bunlar OpenGL ve Direct3D. OpenGL çapraz platform kütüphanesi ve Direct3D tek Windows platformunda kullanılabilir.

OpenGL kütüphanesi üç boyutlu grafiklerle çalışmak için en popüler programlama arayüzleri (API) biridir. OpenGL standartı yazılım geliştirme alanında önde gelen firmaları tarafından 1992 yılında onaylanmıştır. Onun temeli Silicon Graphics tarafından geliştirilen, IRIS GL kütüphanesi oldu.

OpenGL "Açık Grafik Kütüphanesi" olarak çevirir. Başka bir deyişle, OpenGL - yüzlerce fonksiyonları içeren bir spesifikasyondur. OpenGL grafik kütüphanesi bağımsız çapraz platform yazılım arayüzü belirler, onun yardımıyla programcı iki boyutlu ve üç boyutlu bilgisayar grafikleri kullanan uygulamaları geliştirebilir.

Farklı aydınlatma modellerini gerçekleştirmek için OpenGL Tarayıcı Dili – GLSL kullanmak gerekir. Tarayıcı dilinde yazılan programlar merkez işlemci tarafından değil, grafiksel işlemci tarafından çalıştırılır. Bu dilde yazılan programlar shader adlanır. Shaderleri kullanınıyla farklı aydınlatma modelleri gerçekleştirilebilir. Eski OpeanGL de shaderler yoktu, ve dolayısıyla programcılar grafiksel uygulamalarında sadece bir sabit aydınlatma modelini kullanmak zorunda kalırdı ve o yalnız modelin parametrelerini

değiştirebilirlerdi. Grafiksel işlemci hızı merkez işlemciye göre çok yüksek olduğu için bu programlar verileri hızlı ve parallel işletir.

Çağdaş grafiksel uygulamada üç boyutlu sahneyi göstermek için iki yöntem kullanır. Onlar global (evrensel) ve yerel aydınlatma modelleri. Evrensel modeller foto-gerçekçi görüntü verir, gölgeleri hesaplamak için ek algoritmalar gerekmiyor. Yerel aydınlatma modelleri ise (evrensel modellere göre) daha az foto-gerçekçi görüntü verir, fakat daha verimli. Bu nedenle gerçek zamanlı üç boyutlu grafiksel uygulamalarda yerel modelleri kullanılmakta.

Yerel aydınlatma modelleri üç temel bileşenden oluşur. Onlar ortam ışığı (ambient light), dağınık yansıma (diffuse reflection) ve düzgün yansıma (specular reflection) diye adlandırılır.

Ortam ışığı (ambient light) – sahneye başlangıç sabit aydınlatma verir. O ışık kaynağınan gelip binlerce yansımaya uğrayıp alana yayılan ışığı modelini yapar.

Dağınık yansıma (diffuse reflection) – belli bir yönden nesne yüzeyine gelip de her tarafa eşit şiddette yayılan ışığı modeler. Biz nesnelerin rengi olarak kabul ettiğimiz aslında beyaz ışıkla aydınlatılan nesneyin dağınık yansımasıdır.

Düzgün yansıma (specular reflection) – belli bir yönden nesne yuzeyine gelip de nesne yüzünen belli bir tarafa yansıyan ışığı modeller. Böyle yansımadan dolayı nesne yuzeyında parlayan beyaz noktalar ortaya çıkar.

Üç bileşen ayrı-ayrı hesaplanır ve toplanır. Böyle aydınlatma modeli yerel aydınlatma modellerinin temelini sağlar.

Bu çalışmada, 7 yerel aydınlatma modelleri ve onların matematiksel temelleri incelenmiştir. Onlar Lambert, Wrap-around, Oren-Nayar, Minnaert dağınık yansıma modelleri ve Phong, Blinn-Phong, Ward düzgün yansıma modelleri. Tüm modeller için ortam ışığı aynıdır. Lambert, Wrap-around, Oren-Nayar, Minnaert modelleri sadece dağınık yansımayı modeller, düzgün yansıma bileşeni yoktur. Phong, Blinn-Phong, Ward modellerinde dağınık yansıma bileşeni Lambert modeli ile hesaplanır, düzgün yansıma bileşeni ise farklı yöntemler ile bulunur.

Bu çalışma sonunda bu aydınlatma modellerini gösteren gerçek zamanlı grafiksel uygulama geliştirilmiştir. Uygumalada modeller seçebilir ve karşılaştırabilir. Uygulama OpenGL grafik kütüphanesini kullanarak Visual Studio 2017 (Community Edition) ortamında C ++ programlama dili ile yazılmıştır. Aydınlatma modelleri GLSL dilinde (OpenGL tarayıcı dili) uygulanmıştır.

Anahtar Kelimeler: Bilgisayar grafikleri, 3D bilgisayar grafikleri, aydınlatma, aydınlatma modelleri, OpenGL, GLSL.

МАЗМУНУ

КОМПЬЮТЕРДИК ГРАФИКАДА ЖАРЫКТАНДЫРУУНУ МОДЕЛДӨӨ

КЫСКАРТЫЛГАН СӨЗДӨР

Кыскартылышы	**Чечмелениши**
OpenGL	Open Graphics Library
GLSL	OpenGL Shading Language
GLM	OpenGL Mathematics
GPU	Graphics Processing Unit

СҮРӨТТӨРДҮН ТИЗМЕСИ

1. КОМПЬЮТЕРДИК ГРАФИКА. ЖАЛПЫ ТҮШҮНҮК

Компьютердик графика жөнөкөй сызык жана кесиндилерди сызуудан баштап, виртуалдуу чындык жана толук-метраждуу кинофильм түзүүгө чейинки жолду басып өттү [1]. *Компьютердик графика* (же машиналык графика) – бул компьютер жардамы менен модель жана алардын сүрөттөлүштөрүн түзүү, өзгөртүү, сактоо, иштетүүнү изилдөөчү дисциплина [2].

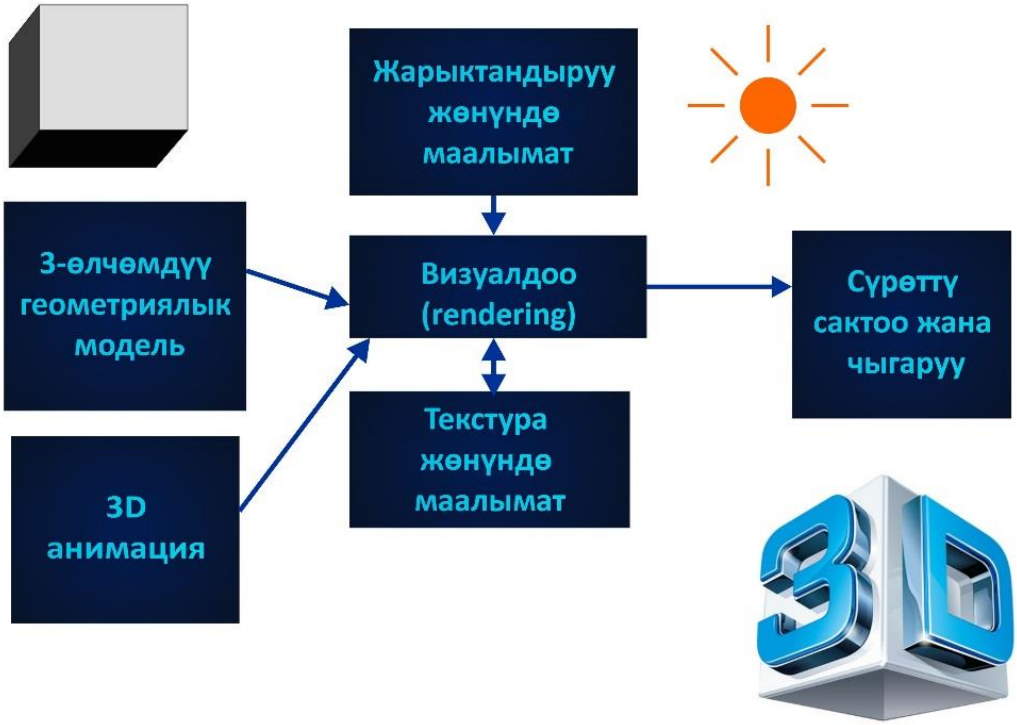
XXI-к. компьютердик графика каражаттары фотографиялык сүрөттөрдүн сапатынан кем эмес реалисттик сүрөттөрдү түзүүгө мүмкүндүк берет. Ар түрдүү жана ар багыттуу сүрөттөлүштөрдү алуу үчүн түрлүү аппараттык жана программдык жабдыктар түзүлгөн. Мисалы, жөнөкөй чийүү-сызуудан баштап, табигый объекттердин чындыкка жакын элестерине чейин. Компьютердик графика визуалдуу түрдө кабыл алуу жана маалыматты берүү үчүн дээрлик бардык илимий жана инженердик тармактарда колдонулат [3].

Учурда компьютердик графиканы колдонуу чөйрөсү өтө кеңири. Жаңы автомобилди куруу алгач эскиз түрүндө компьютерде жасалат. Медицинада компьютердик томографтар колдонулат. Архитектурада компьютердик графика методдоруна негизделген визуалдуу автоматташтырылган долборлоо тутумдары (CAD — Computer Aided Design) кеңири колдонулат. Химиктер берилиштерди визуалдуу көрсөтүү каражаттарын колдонуп, татаал белоктордун молекулаларын изилдешет. Кинематографияда компьютердик графиканы колдонуу абдан керектүү иш болуп калды. Математикада берилиштерди графикалык көрсөтүү каражаттарысыз фракталдар теориясынын өнүгүшү мүмкүн эмес эле. Заманбап иштетүү тутумдары графикалык режимде иштешет [4].

Заманбап компьютердик графика – бул абдан чоң жана татаал, көп кырдуу илимий-техникалык дисциплина. Бул дисциплинанын айрым бөлүктөрү дээрлик толугу менен изилденип бүтсө, айрым бөлүктөрү активдүү өнүгүүдө: растрдык сканерлөө, түс моделдөө, жарыктандыруу, текстуралоо, тунуктук жана жарым-тунуктук эффекттери [5].

1. 1. **Компьютер экранында үч өлчөмдүү сүрөттү түзүүнүн негизги этаптары**

Кандай гана программдык каражаттар колдонулбасын үч өлчөмдүү дүйнөнү түзүү төмөнкү этаптардан турат: моделдөө, материалдарды (текстура) коюу, жарыктандыруу, камераны орнотуу, анимация, визуалдоо. Бул процесс – графикалык процесс деп аталат (сүрөт 1.1). Ар бир этаптын кыскача мүнөздөмөсүн карап көрөлү [6-10].



Сүрөт 1.1. Графикалык процесс [35]

1. 1. 1. Моделдөө

Моделдөө – үч өлчөмдүү объектин формасын түзүү. Объекттерди көрсөтүү үчүн, эреже катары, көп бурчтуктар (көпчүлүк учурда үч бурчтуктар) колдонулат. Алар керектүү форманын катмарын түзгөндөй жайгаштырылат. Объект

түзүлгөндөн кийин, аны жылдыруу, айлантуу, масштабдоо, көчүрүү, чагылдырууга болот [6].

1. 1. 2. Материалдарды коюу

Чыныгы дүйнөдөгү бардык буюмдар материалдан (заттардан – пластмасса, жыгач, кирпич, мрамор ж. б.) турат. Материалдар объекттин сырткы көрүнүшүн, жарыктандырылуусун жана кээ бир физикалык параметрлерин аныктайт. Материал чагылган жарыктын көлөмүн аныктайт. Материалдар – бул объекттин түсү жана текстуралар [7, 10]. Текстура бул объект бетине капталуучу сүрөт.

1. 1. 3. Жарыктандыруу

Жарык мейкиндикте жайылат. Ал таркалат, чагылат, сынат, заттарды даана ачык же күңүрт, элес-булас көрсөтөт. 3D-технологияларында жарык ар кандай визуалдоо алгоритмдери менен эсептелет. Жарык сценага жалпы маанай берет. Мисалы күүгүмдөгү жарык сценага капалуу сезим берет. Туура жарыктандырылган сцена, жарыксыз сценага караганда жакшы көрүнөт [10].

1. 1. 4. Камера орнотуу

Сценаны көрсөтүү ыкмасын тандоо көрүүчү (байкоочу) үчүн өтө маанилүү. Колдонуучу сценаны виртуалдык камера аркылуу көрүү параметрлерин башкара алат. Виртуалдык камера үч өлчөмдүү объекттердин эки өлчөмдүү сүрөткө кандай чагылышын аныктайт. Камеранын негизги параметрлери: камера орду, багыттоо чекити, объективдин фокустук аралыгы. Сценанын көрүнүшү тандалган фокустук аралыкка жараша, атайын алгоритмдер жардамы аркылуу эсептелет [6, 7].

1. 1. 5. Анимация

Анимация — үч өлчөмдүү моделдөөнүн эң оор баскычы. Ал кадр (frame) деп аталган, өз өзүнчө сүрөттөрдөн турат. Кадрлар бири бирин белгилүү ылдамдыкта алмаштырып турушат. Жыйынтыкта персонаждар кыймылдагандай сезилет. Демек

анимацияда кадрлар бат алмашкандыктан, көрүүдө кыймыл иллюзиясы пайда болот. Ар бир кадр кандайдыр бир объекттин ар кандай позиция, айлануу бурчу, өлчөмдөгү сүрөтүн камтыйт. Экранга чыгаруу ылдамдыгы секундада канча кадр көрсөтүүлөрүнө (FPS — frames per second) жараша аныкталат. Кинематографияда ар секундада 24 кадр чыгарылат [8, 9].

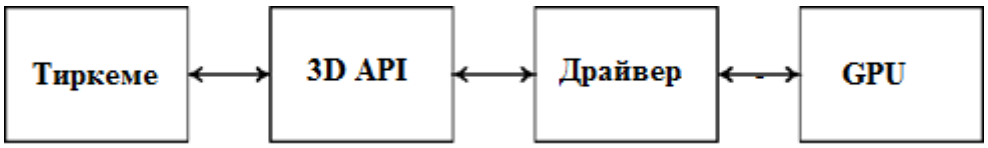
1. 1. 6. Визуалдоо (Рендеринг)

Модель түзүлүп, материалдар тандалып, жарык булактары жана камера орнотулгандан кийин, сүрөттү калыптоо – визуалдоо аткарылат. Үч өлчөмдүү компьютер графика дүйнөсүндө визуалдоо – бул сценаны растрдык сүрөткө өткөрүү, тактап айтканда ар бир пикселдин түс жана жарыктыгын эсептөө. Визуалдоогоо бир топ убакыт сарпталышы мүмкүн. Сарпталган убакыт сценанын татаалдыгы менен компьютер кубаттуулугунан көз каранды. Дал ушул этапта програм бардык көлөкө, блик жана чагылыштарды эсептеп сүрөткө түшүрөт [6].

1. 2. Компьютердик графиканы программдоо

Үч-өлчөмдүү графика менен иштөө үчүн күчтүү видеоадаптер (графикалык процессор) керек. Заманбап графикалык процессорлордун бардык ресурстарын эффективдүү колдонуу үчүн керектүү графикалык (API – Application Program Interface) интерфейсин колдонуу керек. Мындай API тиркеме жана графикалык процессор ортосунда байлаштырчу звено болот (сүрөт 1.2). Азыркы учурда үч-өлчөмдүү графиканы чыныгы убакта программдоо үчүн кеңири таралган эки интерфейс бар: OpenGL жана Direct3D [11].

Direct3D Microsoft компаниясынын продукту, аны жалгыз гана Windows платформасында колдонсо болот. Ал эми OpenGL кросс-платформалуу программдык интерфейс.



Сүрөт 1.2. Тиркеме менен графикалык процессордун өз ара аракеттешүүсүнүн жөнөкөйлөштүрүлгөн схемасы [11]

1. 2. 1. OpenGL

OpenGL китепканасы үч-өлчөмдүү графика менен иштөөгө арналган эң популярдуу программдык интерфейстин катарына кирет. OpenGL стандарты 1992-ж. программдык жабдыкты иштеп чыгуучу алдыңкы фирмалар тарабынан бекитилген. Анын негизин Silicon Graphics фирмасынын IRIS GL китепканасы түзгөн [12].

Open Graphics Library (OpenGL) – “ачык графикалык китепкана” дегенди билдирет. Башкача айтканда OpenGL – бул өзүнө жүздөгөн функцияларды камтыган спецификация. Ал программдоо тилинен көз карандысыз кросс-платформалуу программдык интерфейсти аныктайт. Анын жардамы менен программчы эки жана үч өлчөмдүү компьютердик графиканы колдонгон тиркеме түзөө алат [13].

Бул китепкана графикалык жабдууга көз карандысыз программдык интерфейс сунуштайт, б. а. аны ар кандай аппараттык платформаларда колдонууга болот. OpenGL командалары төмөнкү иш-аракеттерди аткарууга мүмкүнчүлүк берет [14]:

1. Графикалык примитивтерден (чекит, сызык, үч бурчтук) моделдерди түзүү.
2. Объекттерди үч өлчөмдүү мейкиндикте жайгаштыруу жана сценанын оңтойлуу каroo чекитин тандоо.
3. Бардык объекттердин түсүн эсептөө. Түс тиркеме тарабынан берилиши мүмкүн, жарыктандыруу шарттары же текстура менен аныкталышы мүмкүн, же бул үчөөнүн комбинациясы болушу мүмкүн.

4. Объекттердин математикалык сүрөттөлүшүн жана алардын түсү тууралу маалыматты пикселдерге өткөрүү. Бул процесс *растерлөө* деп аталат.

OpenGL компьютердик оюндар, автоматташтырылган долборлоо тутумдар (CAD), виртуалдуу чындыкты түзүүдө, илимий изилдөөлөрдөгү визуалдоодо колдонулат.

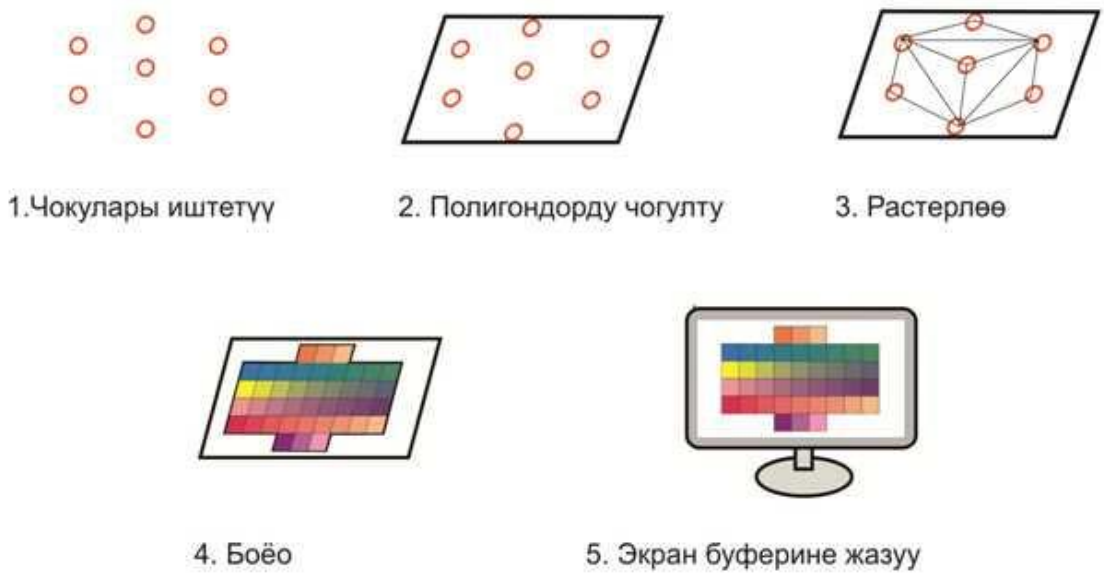
1. 2. 2. Шейдерлерди колдонуу

Бардык заманбап компьютердик графиканы (компьютердик оюн, илимий визуалдоо же атайын эффекттерди түзүүчү тутум ж. б.) шейдерлерсиз (shader) элестетүүгө мүмкүн эмес. Шейдерлер – бул атайын тилде жазылган, графикалык процессордо аткарылуучу чакан программдар. Шейдерлерди колдонуу областы өтө кеңири жана дагы деле көбөйүүдө. Бирок графикалык процессордо аткарылуучу шейдерлерди программдоо кадимки процессорлорду программдоодон айырмаланат. Анткени GPU үч өлчөмдүү объекттерди визуалдоого (рендеринг) багытталган, спецификалык архитектурага ээ процессор. Заманбап графикалык процессордун негизги концепциясы – программдалуучу графикалык конвейер (сүрөт 1.3) (programmable graphics pipeline) [15].

Эски OpenGL-де конвейердин бардык этаптары бекитилген (fixed) болчу. Мындай учурда ар кандай жарык моделдерин, түрдүү эффекттерди ишке ашырууга мүмкүн эмес. Жаңы графикалык процессорлордун пайда болушу менен визуалдоо конвейердин кээ бир этаптарын программдоого мүмкүнчүлүгүн берди. Бул учурда стандарттуу 1- жана 4-этаптарын вертекстик (чокулар) жана фрагменттик (пикселдик) шейдерлер алмаштырылып, графиканын сапаты жана ылдамдыгы жакшырат. Графиканы шейдерлерди колдонуп программдоо бир топ оор маселе [6].

GLSL – OpenGL-дин жогорку дээңгелдеги шейдердик тили. GLSL C, C++ жана Render Man Shading Language тилдерине өтө жакын. Бирок C++ тилине караганда түшүнүүнү татаалдаштырган, ката, ачык, так эместикке алып келе турган кээ бир элементтер жоюлган [11]. Ошол эле убакта GLSL, 3D-дүйнөнү түзүүдө

колдонууга ыңгайлуу, типтердин (вектор, матрица ж. б.) кеңири жыйындысын камтыйт. Бул тилде C++ тилинен өздөштүрүлгөн механизмдер да колдонулат: функцияларды аргументтердин тиби боюнча кайра жүктөө, өзгөрмөлөрдү колдонуудан мурун жарыялоо. GLSL тили циклдарды, шарттуу туюнтмаларды, функцияларды колдойт. Көптөгөн функциялар жыйындысы шейдердик алгоритмдерди түзүү үчүн көп мүмкүнчүлүктөрдү берет [16].

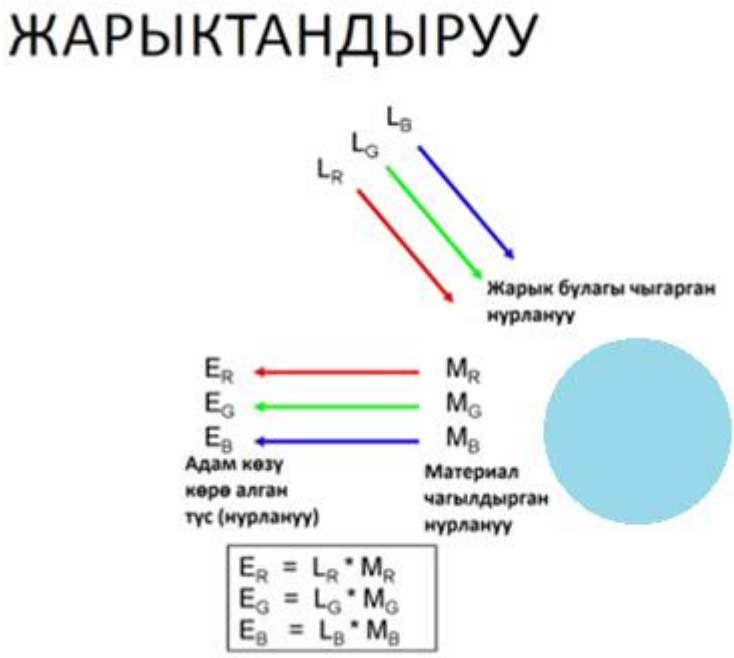


Сүрөт 1.3. OpenGL конвейери [6]

2. ЖАРЫКТАНДЫРУУ

2. 1. **Жарыктандырууну моделдөө**

Чыныгы жашоодо жарык объектке тийгенде, жарыктын кандайдыр бир бөлүгү ал объекттин бетинен чагылат (сүрөт 2.1). Чагылган жарык биздин көзүбүзгө жеткенде гана, биз объектти көрүп, анын түсүн айырмалай алабыз. Мисалы, ак кубик ак жарыкты чагылтат. Ал жарык көзүбүзгө жеткенде, биз кубикти ак түскө ээ деп кабыл алабыз [17].



Сүрөт 2.1. Жарыктын чагылуу модели [35]

Заманбап графикалык тиркемелерде үч өлчөмдүү сценаларды көрсөтүүнүн эки жолу колдонулат: глобалдуу жана локалдуу жарыктандыруу моделдери. Глобалдуу жарыктандыруунун бир нече артыкчылыктары бар: жогорку деңгээлдеги чындыкка жакын сүрөт алуу, көлөкөлөрдү куруу үчүн кошумча алгоритмдер колдонулбайт. Локалдуу моделдер салыштырмалуу сапаты төмөн

сүрөттөлүштү берет, бирок алар бат жана өндүрүмдүү болгондуктан интерактивдүү тиркемелерде ийгиликтүү колдонулуп келет [18]. OpenGL графикалык интерфейсинде локалдык жарыктандыруу методдорун гана ишке ашырса болот.

OpenGL жарыкты кызыл, жашыл, көк түстөрдөн курамасы катары эсептейт. Ошентип, жарык булагы өзү нурланткан кызыл, жашыл, көк жарыктын көлөмү менен мүнөздөлөт. Ал эми үстүртүлүктүн материалы өзү чагылткан жарыктын кызыл, жашыл, көк компоненттери менен мүнөздөлөт. OpenGL-де ишке ашырылуучу локалдык жарык моделдери чыныгы жарыкты аппроксимациялайт. Бирок алар жетишерлик жакшы иштейт жана бат эсептелет [19].

Сүрөт 2.2. Жарыктандыруу моделдөө [35]

2. 2. **Жарык булактары**

Энергия чыгарган, ар бир объект жарык булагы болуп эсептелет. Ар бир жарык булагы, сценадагы объекттердин жарыктандыруусуна, кандайдыр бир

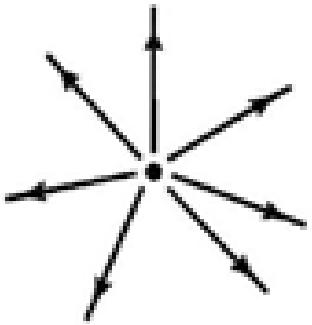
салым кошот. Жарык булактарын моделдөөдө, аларга ар кандай форма жана мүнөздөмөлөр берилиши мүмкүн (сүрөт 2.2). Жарык булагын аныкташ үчүн көптөгөн касиеттер колдонулат. Анын ордун, түсүн, багытын жана формасын берсе болот [20].

2. 2. 1. Чекиттик жарык булактары

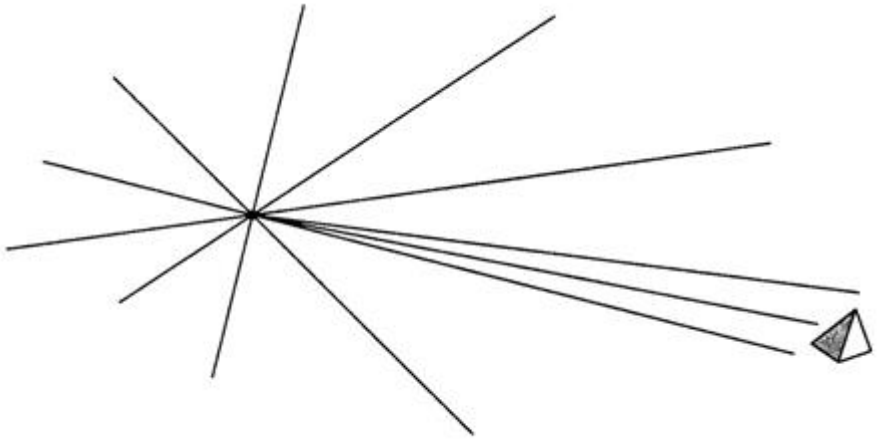
Чекиттик жарык булагынын нурлары жалгыз бир чекиттен тарайт. Мындай жарык булактарына лампа, фонарь, факел ж. б. у. с кирет. Мындай жарык булактары орду жана түсү менен берилет. Чекиттик жарык булагынын нурларынын багыты анын позициясы менен аныкталат (сүрөт 2. 3). Мындан улам сценанын ар кайсы бөлүктөрүндө жарык ар кандай багытка ээ болот [17].

2. 2. 2. Багытталган жарык булактары

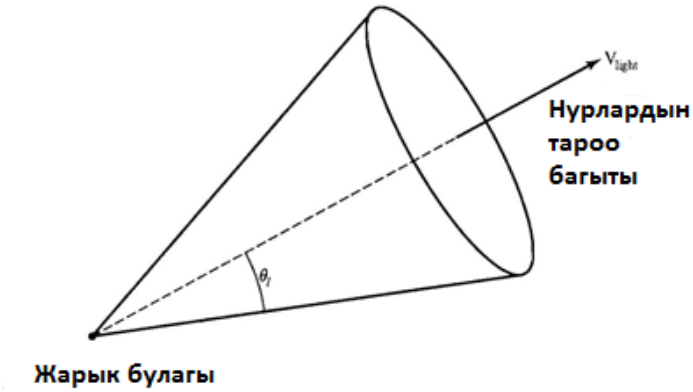
Эгерде жарык булагы сценадан өтө эле алыста жайгашса, анда жарык булагынын нурлары бардык чекиттер үчүн бирдей багытка ээ деп божомолдосо болот. Мындай жарык булактары багытталган деп аталат (сүрөт 2.4). Ошондой эле байкоочу өтө алыс жайгашса, кароо багыты бардык чекиттер үчүн бирдей деп эсептесек болот. Мындай болжолдор эсептөөлөрдү оңойлотот, ошондуктан башка жарык булактарына караганда, багытталган жарык булагын ишке ашырган код бат иштейт. Мындай жарык булактары күн сыяктуу жарык булактарын имитациялоо үчүн колдонулат [21].



Сүрөт 2.3. Чекиттик жарык булагы [20]



Сүрөт 2.4. Багытталган жарык булагы [20]



Сүрөт 2.5. Прожектор [20]

2. 2. 3. Прожекторлор

Прожекторлор жарыкты мейкиндикте конус сымал, чектелген бир аймакта гана жайылтат (сүрөт 2.5). Прожекторлор орду, багыты жана ушул багыттан (багыт векторунан) өлчөнгөн чектөө бурчу (θ) менен аныкталат. Эгерде объект

багытталган нурлардын тышында орун алса, анда мындай жарык булагы, анын жарыктандыруусуна таасир тийгизбейт. Прожекторду чекиттик жарык булагынан курса болот.

Чектөө бурчу жарык нурларынын энин жана бурч боюнча ургаалдуулуктун басаңдоо көрсөткүчүн аныктайт. Чектөө бурчун (θ) 0 ° - 90 ° аралыгында өзгөртүү менен прожектордун «конус» эффектин ала албыз [22].

2. 3. **Стандарттуу жарыктандыруу модели**

OpenGL шейдерлерин программдоо сценанын жарыктандыруусу үчүн дээрлик чексиз мүмкүнчүлүктөрдү берет. Эски OpenGL-де өзгөртүлгүс стационардык жарыктандыруу моделдери колдонулчу. Мындай учурда ар кандай жарык моделдерин, түрдүү эффекттерди ишке ашырууга мүмкүн эмес эле. Программдалуучу шейдерлер болсо жакшыраак сапат бере алышат, өзгөчө реализм жаатында. Бирок, ага карабастан стандарттуу жарыктандыруу моделин жакшы түшүнүү керек. Бул модель растерлөөгө негизделген жарыктандыруу ыкмаларынын негизин түзүп, жакшыртылган ыкмаларды түшүнүүгө өбөлгө болот. Классикалык модель өз өзүнчө эсептелген компоненттерден турат. Жалпы жарыктандырууну алуу үчүн ал бөлүктөр бириктирилет. Ал компоненттер: фондук, диффуздук жана күзгү сымал (күзгүлүү) [21].

2. 3. 1. Фондук жарык

Фондук (ambient) жарык – бул объекттер бетинен көп жолу чагылгандыктан, багытын аныктоого мүмкүн эмес болгон, жайылган жарык. Ал бардык тараптан келгендей сезилет. Мисалы, күндүз бөлмөнүн ичи көбүн эсе фондук жарык менен жарыктандырылат. Фондук жарык үстүртүлүккө тийгенде бардык тарапка бирдей жайылат [14].

Фондук жарыктандыруунун математикалык модели төмөнкү формула менен берилет (формула 2.1):

$$I_{ambient} = K_a I_a \quad (2.1)$$

бул жакта K_a – фондук жарыктандыруунун коэфициентти, I_a – фондук жарыктандыруунун ургалдуулугу [23-28].

2. 3. 2. Диффуздук жарык жана Ламберт модели

Диффуздук (diffuse – жайылган, чачылган) жарык белгилүү бир багыттан келип, объекттин бетинен бардык багытта чагылып жайылат. Бул чагылуу диффуздук чагылуу деп аталат. Биз нерселердин түсү катары кабыл алганыбыз чынында объектти ак түстөгү жарык (бардык түстөрдүн комбинациясы) менен жарыктандырылгандагы диффуздук чагылуунун түсү. Диффуздук чагылуу күңүрт, жылтырабаган, тунук эмес беттерге (кагаз, жыгач ж. б) таандык. Андай үстүртүлүктөр одуракай же бүртүктүү болушат. Мындай үстүртүлүктөр *идеалдуу диффуздук чагылдыргычтар* деп аталат. Ошондой эле алар *ламберттик чагылдыргычтар* деп да аталат, анткени үстүртүлүктүн ар бир чекитинен чагылган жарыктын энергиясы *Ламберттин косинустар мыйзамына баш иет* [20, 29, 30].

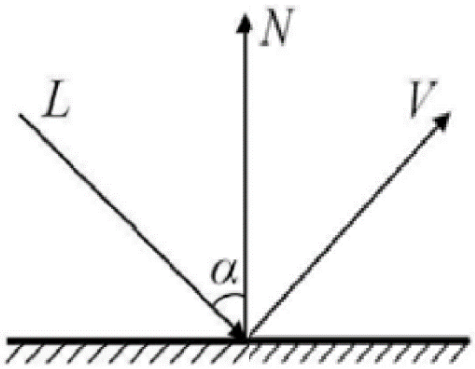
Ламберттин модели боюнча диффуздук чагылган жарык төмөнкү формула менен эсептелет (формула 2.2) [3, 4, 15, 18, 25-31]:

$$I_{lambert} = I_d K_d \cos \alpha \quad (2.2)$$

Бул жерде K_d – диффуздук чагылдыруу коэффициенти, I_d – диффуздук жарыктандыруунун ургалдуулугу, α – объекттин үстүртүлүгүнө багытталган бирдик нормаль ($\vec{N}$) жана жарык булагынан объектке багытталган бирдик векторлорунун ($\vec{L}$) ортосундагы бурч (сүрөт 2.6). $\vec{N}$ жана $\vec{L}$ векторлору бирдик векторлор болгондуктан, $\cos \alpha$ бул векторлордун скалярдык көбөйтүндүсүнө барабар (формула 2.3):

$$\cos\alpha = (\vec{N}, \vec{L}) \qquad (2.3)$$

Эсептөөлөр объекттин ар бир чокусу үчүн жүргүзүлгөндүктөн, бул модель объектти көлөкөлөп, ага көлөм берет. Фондук жарыктан айырмаланып, диффуздук жарык объектти бир түскө боёбойт, ошондуктан объекттин тигил же бул бөлүгүнө жарык тийип, башка жагы көлөкөдө калгандай сезилет [23]. Эгерде α бурчу 90° - тан чоң болсо (демек $cos\ \alpha < 0,0$) жарык булагынан келген нурлар объектке түз тийбейт [20]. Бул учурда фондук жарык жок болсо, объекттин ошол бети караңгыда калат.



Сүрөт 2.6. Жарыктандыруу векторлору [23]

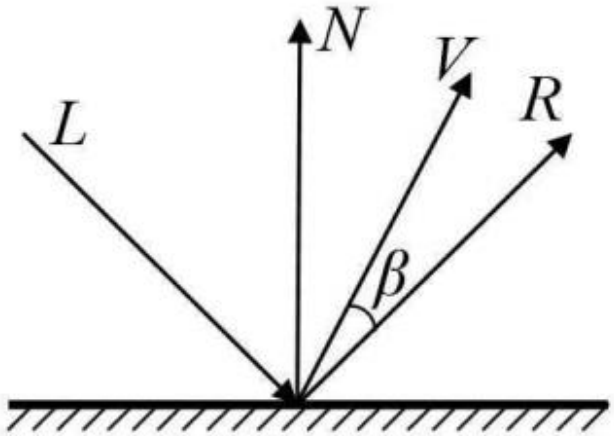
2. 3. 3. Күзгү сымал жарык жана Фонг модели

Жылтырак беттерден көрүнгөн жарык так же күзгү сымал чагылуу, толук (же дээрлик толук) чагылуудан пайда болот.

Чагылган же күзгүлүү (specular) жарык белгилүү бир багыттан келип, объекттин бетинен ошондой эле белгилүү бир багытта чагылат. Жогорку сапаттагы күзгү бетинен нурлар дээрлик 100% чагылат. Күн жарыгы тийген металл же пластмассанын дагы спекулярдык (күзгүлүү) чагылуу коэфициенти жогору. Ал эми

бор же килемдин (күзгүлүү) чагылуусу дээрлик жокко эсе. Спекулярдык (күзгүлүү) чагылууну жылтыроо (shininess) катары элестетсе болот [14].

Объект бетинде эч кандай одур, бодур жок болуп, текши, тегиз, жылмакай болсо, анда ал үстүртүлүк идеалдуу күзгүдөй үстүртүлүк болуп эсептелет. Мындай беттин өз түсү байкалбайт. Жарык энергиясы чагылган нурдун сызыгы боюнча гана чагылат. Бул сызык тышында эч кандай жайылуу байкалбайт [27].



Сүрөт 2.7. Кароо вектору [23]

Фонг модели боюнча чагылышуу (спекулярдык) жарык модели төмөнкү формула менен эсептелет (формула 2.4) [3, 4, 15, 18, 25-32]:

$$I_{phong\ specular} = I_s K_s \cos^{n}\beta \qquad (2.4)$$

Бул жерде K_s – спекулярдык жарыктандыруунун коэфициенти, I_s – спекулярдык жарыктандыруунун ургалдуулугу, β – байкоочуга багытталган бирдик вектору ($\vec{V}$) жана чагылуушу бирдик векторунун ($\vec{R}$) ортосундагы бурч (сүрөт 2.7), n – жылтыроо коэффициенти, бул канчалык көп болсо, бликтин аянты ошончолук аз болот. $\vec{V}$ жана $\vec{R}$ векторлору бирдик векторлор болгондуктан, $cos\,\beta$ бул векторлордун скалярдык көбөйтүндүсүнө барабар (формула 2.5):

$$cos\,\beta\ = (\vec{V} \cdot \vec{R}) \tag{2.5}$$

β бурчу 0°- 90° диапазон аралыгында болот, ошондуктан $cos\ \beta$ [0-1,0] аралыгында өзгөрөт [20, 23, 31, 32].

Күзгү сымал чагылуунун көрсөткүчү n 1-200 аралыгында өзгөрүп, үстүртүлүктүн түрүнө жараша алынат. Идеалдуу чагылдыргыч үчүн n чексиз чоң сан [28]. Аябай жылтырак беттерди моделдөөдө n өтө чоң маанилерге (100 же андан жогору) ээ болот, ал эми тунук эмес беттер үчүн өтө кичине маанилерге (1-ге чейин) ээ [20].

2. 3. 4. Жалпы модель

Табиятта идеалдуу күзгүдөй же идеалдуу диффуздук үстүртүлүк жок болгондуктан, компьютердик графикада объекттерди сүрөттөөдө күзгүлүү жана диффуздук чагылуу конкреттүү материалдын түрүнө мүнөздүү пропорцияда айкалыштырылып моделденет. Бул учурда толук чагылган жарык күзгүлүү жана диффуздук компоненттерден суммасы катары эсептелет (формула 2.6) [15, 27, 30]:

$$I = I_d\, K_d \cos\alpha + I_s\, K_s \cos^{\,n}\beta \tag{2.6}$$

Жарык түс тийбеген бөлүктөр караңгыда калбаш үчүн фондук жарыкты кошолу (формула 2.7):

$$I = C\,(K_a \cdot I_a + I_d\, K_d \cos\alpha + I_s\, K_s \cos^{\,n}\beta) \tag{2.7}$$

Бул жерде С объекттин түсү. Бул формула менен берилген жарыктандыруу модели толук Фонг модели же жөн эле Фонг модели деп аталат.

2. 4. **Локалдуу жарыктандыруу моделдери**

2. 4. 1. Wrap-around жарыктандыруу модели

Wrap-around модели Ламберт моделинин модификациясы. Бул модель f – жылыш коэфициеттинин жардамы менен жарыктандыруу областын кеңейтет (формула 2.8) [31]:

$$I_{Wrap\text{-}around} = k_d\, I_d \cdot \frac{(N,\ V)+f}{1+f} \tag{2.8}$$

2. 4. 2. Орен-Наяр жарыктандыруу модели

Орен-Найяр чагылуу модели, Майкл Орен жана Шри К. Найар тарабынан иштелип чыккан. Бул модель бодуракай беттер үчүн диффуздук чагылууну моделдейт. Ламберт модели бетон, шыбак, кум сыяктуу беттер үчүн диффуздук компоненти туура эмес эсептейт, анткени ал беттин бодуракайлыгын эске албайт. Ал эми Орен-Найяр чагылуу модели одуракай беттерди ар кандай жантайма кырлардын жыйындысы катары кабыл алат. Орен-Найяр чагылуу модели бетон, шыбак, кум сыяктуу материалдарды тагыраак моделдейт [33].

Бул модель боюнча жарыктандыруу төмөнкү формула менен берилет (формула 2.9) [31, 33]:

$$I_{oren} = max\left(0, \left(\vec{N}\cdot\vec{L}\right)\right)(A + B\, max\ (0, cos(\varphi_V - \varphi_L))\, sin(\alpha)\, tg(\beta)) \tag{2.9}$$

Параметрлерин төмөнкү формулалар менен аныктаса болот (формула 2.10, формула 2.11, формула 2.12, формула 2.13):

$$A = 1 - 0.5\frac{\sigma^2}{\sigma^2 - 0.33} \tag{2.10}$$

$$B = 0.45\,\frac{\sigma^2}{\sigma^2 + 0.99} \tag{2.11}$$

$$\alpha = \min(\theta_L, \theta_V) \tag{2.12}$$

$$\beta = \min(\theta_L, \theta_V) \tag{2.13}$$

φ_V – V векторунун чекиттин тегиздигине түшүрүлгөн проекциясы, φ_L – $\vec{L}$ векторунун чекиттин тегиздигине түшүрүлгөн проекциясы, σ параметри беттин тегиз эместигин аныктайт, ал канча чоң болсо бет ошончолук тегиз эмес болот.

2. 4. 3. Миннеарт жарыктандыруу модели

Бул модель планеталардын жарыктандыруусун моделдөө үчүн сунушталган эле, ошондой эле бул модель кээ бир кездеме түрлөрүн моделдөө үчүн да жарайт. Бул моделде диффуздук чагылуу төмөнкү формула менен аныкталат (формула 2.14) [31]:

$$I_{minnaert} = (n, l)^{1+k}(n, v)^{1-k} \tag{2.14}$$

Бул жерде – k эмпирикалык параметр.

2. 4. 4. Блинн-Фонг жарыктандыруу модели

Джим Блинн Фонг спекулярдык чагылуусун эсептөөнүн альтернативдүү ыкмасын сунуш кылган, анын ыкмасында көп эсептөөлөрдү талап кылган чагылуу векторунун ордуна аралык вектор колдонулат (формула 2.15) [15, 34]:

$$\vec{H} = \frac{\vec{L} + \vec{V}}{\left|\vec{L} + \vec{V}\right|} \tag{2.15}$$

Жалпы формула төмөндө берилди (формула 2.16):

$$I_{\text{blinn specular}} = k_S \, I\,(\vec{N}, \vec{H})^{specPower} \tag{2.16}$$

2. 4. 5. Ward изотроптук модели

Спекулярдык жарыкты берген дагы бир жөнөкөй модель – Вард модели. Бул моделде бликтик жарык төмөнкү формула менен аныкталат (формула 2.17) [31]:

$$I_{ward\ specular} = exp^{-k\frac{1-(h,n)^2}{(h,n)^2}} \tag{2.17}$$

Бул жерде k бодурдук коэфициентти.

3. МАТЕРИАЛ ЖАНА МЕТОДДОР

3. 1. **Материалдар**

Microsoft Visual Studio Community — Microsoft компаниясынын продуктусу. Өз ичине программдык жабдыкты иштеп чыгуучу айкалышкан чөйрөнү жана бир катар башка аспаптарды камтыйт. Бул продукттар консольдук тиркемелерди да, графикалык интерфейстүү тиркемелерди да иштеп чыгууга мүмкүндүк берет. Visual Studio 2017 2016-жылдын 17 - ноябрында чыгарылган.

Microsoft Visual C++ (MSVC) — C++ тилинде тиркемелерди иштеп чыккан айкалышкан чөйрө. Microsoft фирмасы тарабынан иштелип чыккан жана Microsoft Visual Studio комплектинин бир бөлүгү катары орнотулган.

C++ тили – типтелген статистикалык компиляцияланууучу программалоо тили. Процедуралык программалоону, абстракциялык берилиштерди, типтерди (объектер), виртуалдык функцияларды, объектке багытталган программалоону, жалпыланган программалоону, контейнерлер жана алгоритмдерди, жогорку деңгэлдеги жана төмөнкү деңгээлдеги тилдерди айкалыштырып өз ичине камтыйт. Эң популярдуу программалоо тилдеринен бири болуп, C++ программдык жабдыктарды иштеп чыгууда колдонулат. Операционндук сисемаларды, компьютерик оюндарды, компьютердин жабдууларына драйверлерди түзүү областарында эч бири менен алмаштырылгыз түрдө колдонулат. C++ тилинин ар түрдүү платформалар үчүн акылуу жана акысыз көптөгөн реализациясы бар. Мисалы: x86 платформасы үчүн - GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder жана башкалар. Бул тил “Java” жана “C#” сыяктуу башка программалоо тилдеринин өнүгүүсүнө чоң салымын кошкон.

OpenGL – эки өлчөмдүү жана үч өлчөмдүү графиканы колдонгон тиркемелерди жаратуу үчүн программалоо тилинен, платформадан көз карандысыз программдык интерфейс.

GLSL – OpenGL-дин шейдерлерди программдоо тили.

Кошумча китепканалар

GLEW (OpenGL Extension Wrangler Library) – OpenGL-дин кеңейтилиштери менен оңой иштөө үчүн колдонулат. Бул өзгөчө Windows үчүн програм жазуу үчүн актуалдуу, анткени Visual Studio сунуштаган бөрк жана китепкана файлдарынын версиясы – OpenGL 1.1

FREEGLUT – OpenGL китепканасында жок функцияларды (терезе түзүү, окуяларды иштетүү ж. б. у. с.) камсыздайт.

SOIL – сүрөттөрдү (текстура үчүн) жүктөө үчүн колдонулат.

GLM (OpenGL Mathematics) – математикалык операцияларды аткаруу үчүн, C++ тилинде атайын функцияларды жана класстарды сунуштаган жардамчы китепкана. GLM китепканасы GLSL спецификациясынын негизинде ишке ашырылган.

ASSIMP – үч өлчөмдүү моделдөө программдарында (3D Max, Blender ж. б.) түзүлгөн эки, үч өлчөмдүү моделдерди жүктөө үчүн колдонулат.

GLUI – коодонуучунун графикалык интерфейс (GUI) китепканасы.

3. 2. **Методдор**

C++ тилинде OpenGL китепканасынын жардамы менен графикалык тиркемелер түзүлдү, жарыктандыруу моделдери GLSL тилинде ишке ашырылды. Үч өлчөмдүү моделдер ачык интернет буклактарынан алынды.

4. ЖЫЙЫНТЫКТАР

Бул иште жети локалдык жарыктандыруу модели каралды: Ламберт, Wrap-around, Орен-Наяр, Миннеарт, Фонг, Блинн-Фонг, Вард изотроптук модели. Жарык булагы катары – чекиттик жарык булагы алынды.

4. 1. **Фондук жарыктандыруу**

Эң жөнөкөй жарыктандыруу модели – *фондук* (тегиз) жарыктандыруу. Фондук жарыктандыруу бардык сцена үчүн баштапкы туруктуу жарыктандырууну камсыздайт. Бул эң жөнөкөй жана бат иштечү, бирок көрүнүп тургандай эң аз чындыкка жакын жыйынтык берүүчү модель (сүрөт 4.1). Фондук жарыкты эсептөө үчүн фондук жарыктын ургалдуулугу (*ambientLight*) жана материалдын фондук жарыктандыруу коэфициенти (*Ka*) гана керек. Фондук жарык бардык жарыктандыруу моделдеринде бирдей эсептелет.

Бул моделди ишке ашырган чокулар жана фрагменттик (пиксельдик) шейдерлер коду төмөндө берилди:

```
// Жалгыз фондук жарыктын чокулук шейдери
#version 130
in vec3 position;              //чекит ( чоку ) координаттары ( өздүк координаттар системинде )
in vec2 texCoord;              //текстура координаттары
out vec2 TexCoord;             //текстура координаттарын фрагменттик шейдерге жиберүү
uniform mat4 model;            //модель матрицасы
uniform mat4 view;             //кароо матрицасы
uniform mat4 projection;               //проекция матрицасы
void main()
{
        TexCoord = texCoord;
        gl_Position = projection * view * model * vec4 ( position, 1.0f );
}
```

```
////////////////////////////////////////////////////////////////////////////
// Жалгыз фондук жарыктын фрагменттик шейдери
#version 130
in vec2 TexCoord;
out vec4 color;
uniform sampler2D diffuseTexture;          // Текстура
uniform vec3 Ka;                           // Фондук жарыктандыруу коэффициенти
void main ()
{
        vec3 ambient = Ka * ambientLight;          // фондук жарык
        color = texture ( diffuseTexture, TexCoord );
        color = color * vec4 ( ambient );          //Пиксельдин түсү
}
```

4. 2. **Ламберт модели**

Ламберт модели боюнча диффуздук чагылууну эсептөө үчүн диффуздук жарыктандыруунун ургалдуулугу (*diffuseLight*), диффуздук чагылдыруу коэффициенти (*Kd*), чекиттин нормаль вектору (*n*) жана жарык булагына багытталган вектору (*l*) керектелет. Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына багытталган вектор пикселдик шейдерде эсептелет.

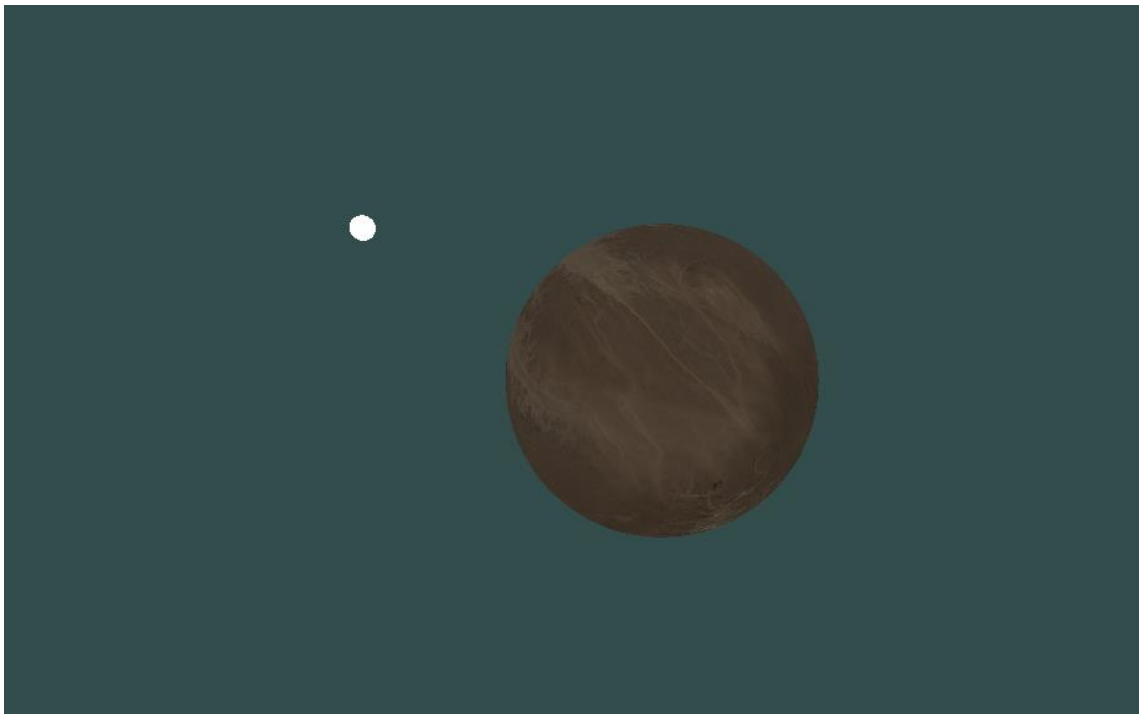
Бул моделди ишке ашырган чокулар жана фрагменттик (пиксельдик) шейдерлери төмөндө берилди:

```
// Ламберт чокулар шейдери
#version 130
in vec3 position;          //чекит (чоку) координаттары (өздүк координаттар системинде)
in vec2 texCoord;          //текстура координаттары
in vec3 normal;            //чекиттин нормалы
```

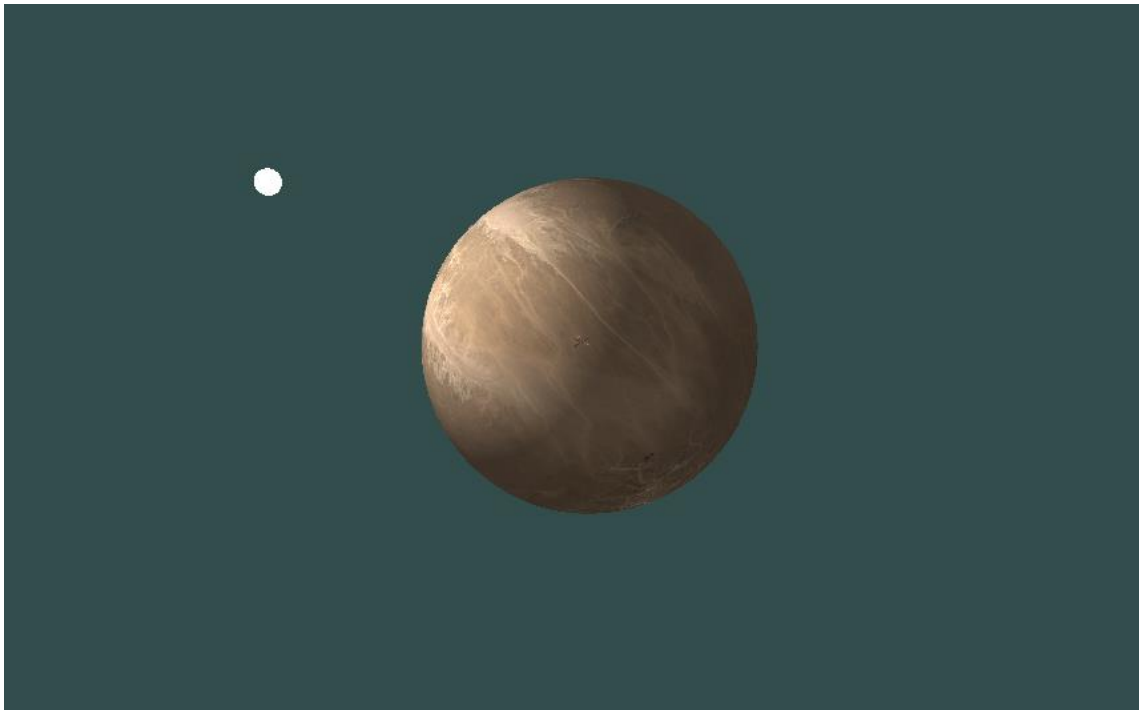
```
out vec3 p;                    //чекит (чоку) координаттары (дүйнө координаттар системинде)
out vec3 Normal;               //нормальды фрагменттик шейдерге жиберүү
out vec2 TexCoord;             //текстура координаттарын фрагменттик шейдерге жиберүү
uniform mat4 model;            //модель матрицасы
uniform mat4 view;             //кароо матрицасы
uniform mat4 projection;               //проекция матрицасы
uniform mat4 normalMatrix;             //нормаль матрицасы
void main ()
{
        p = vec3 ( model * vec4 ( position, 1.0f ));     // чекитти трансформациялоо
        Normal = mat3 ( normalMatrix ) * normal;     // нормальды трансформациялоо
        TexCoord = texCoord;
        gl_Position = projection * view * model * vec4 ( position, 1.0f );
}

//////////////////////////////////////////////////
// Ламберт фрагменттик шейдери
#version 130
in vec3 Normal;
in vec3 p;
in vec2 TexCoord;
out vec4 color;
uniform vec3 lightPos;                  // Жарык булагынын позициясы
uniform sampler2D diffuseTexture;       // Текстура
uniform vec3 diffuseLight;              // Диффуздук жарык
uniform vec3 ambientLight;              // Фондук жарык
uniform vec3 Ka;                        //Фондук жарыктандыруу коэфициенти
uniform vec3 Kd;                        //Диффуздук жарыктандыруу коэфициенти
void main ()
{
```

```
        vec3 ambient = Ka * ambientLight;           // фондук жарык
        vec3 n = normalize ( Normal );              // нормаль векторун нормалдаштыруу
        vec3 l = normalize ( lightPos - p );        // Жарык булагына багытталган бирдик вектор
        float diff = max ( dot ( n, l ), 0.0 );
        vec3 diffuse = diffuseLight * ( diff * Kd );    // Диффуздук чагылуу
        color = texture ( diffuseTexture, TexCoord );
        color = color * vec4( ambient + diffuse, 1.0 )          //Пиксельдин түсү
}
```

Бул модель берген көрүнүш сүрөт 4.2-де берилди.



Сүрөт 4.1. Фондук жарыктандыруу



Сүрөт 4.2. Ламберт жарыктандыруу модели

4. 3. **Wrap-around жарыктандыруу модели**

Wrap-around модели Ламберт моделинин модификациясы. Бул модель боюнча диффуздук чагылууну эсептөө үчүн диффуздук жарыктандыруунун ургалдуулугу (*diffuseLight*), диффуздук чагылдыруу коэффициенти (*Kd*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектору (*l*) жана жылыш коэффициенти (*f*) керектелет. Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына багытталган вектор пикселдик шейдерде эсептелет. Бул моделдин фрагменттик шейдери төмөндө берилди:

```
// Wrap-around фрагменттик шейдери
#version 130
in vec3 p;
in vec3 Normal;
```

```
in vec2 TexCoord;
out vec4 color;

uniform vec3 lightPos;
uniform sampler2D diffuseTexture;
uniform vec3 ambientLight;
uniform vec3 diffuseLight;
uniform vec3 Ka;
uniform vec3 Kd;
float factor = 0.5;

void main ()
{
        vec3 ambient = Ka * ambientLight;
        vec3 normal   = normalize ( Normal );
        vec3 l = normalize ( lightPos - p );
        float diff = max ( dot ( normal, l ) + factor, 0.0 ) / ( 1.0 + factor );

        vec3 diffuse = diffuseLight * (diff * Kd);
        color = texture(diffuseTexture, TexCoord);
        color = color * vec4( ambient + diffuse, 1.0);
}
```

Бул модель берген көрүнүш сүрөт 4.3-де берилди.

4. 4. **Орен-Наяр жарыктандыруу модели**

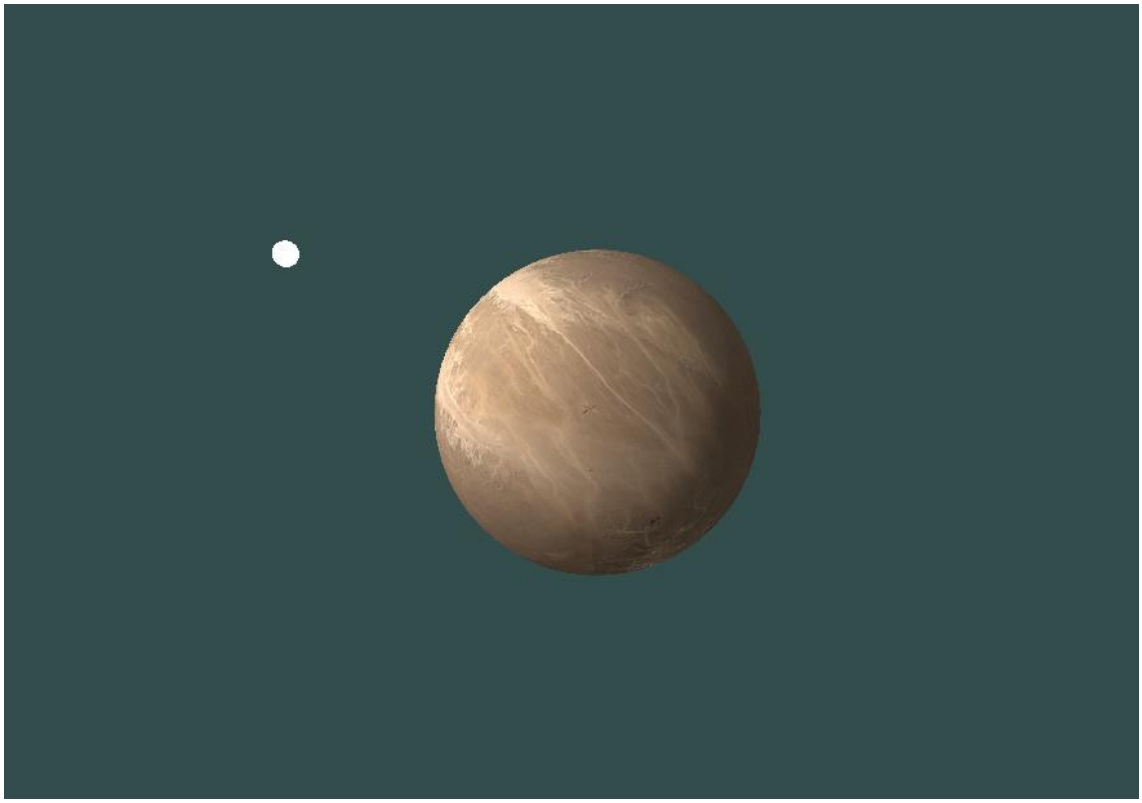
Бул модель бетон, кум сыяктуу материалдардын диффуздук чагылуусун Ламберт моделине караганда тагыраак моделдейт. Бул модель боюнча диффуздук чагылууну эсептөө үчүн диффуздук жарыктандыруунун ургалдуулугу (*diffuseLight*), диффуздук чагылдыруу коэффициенти (*Kd*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектор (*l*), кароо вектору (*v*) жана беттин

бодурдугун (тегиз эместигин) (*roughness*) аныктаган коэффициенти керектелет. Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына жана камерагага багытталган векторлор пикселдик шейдерде эсептелет. Бул моделдин пиксельдик щейдери төмөндө берилди:

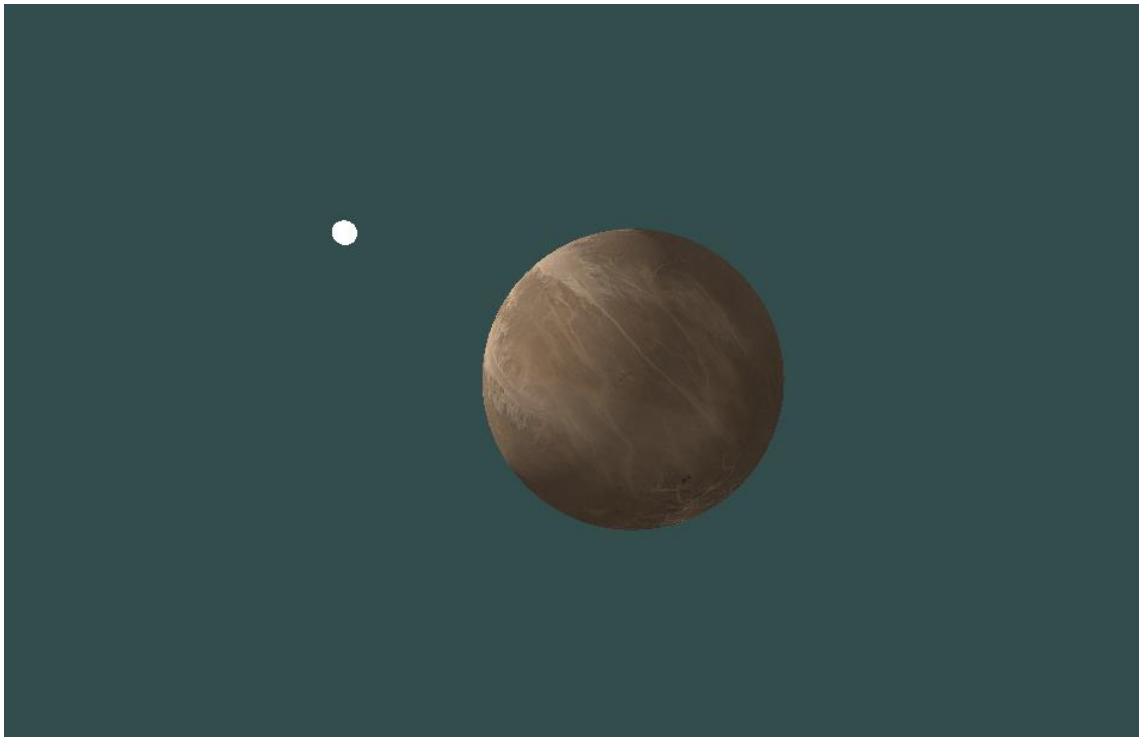
```
// Орен-Наяр фрагменттик шейдери
#version 130
in vec3 Normal;
in vec3 p;
in vec2 TexCoord;
out vec4 color;
uniform vec3 lightPos;                    // Жарык булагынын позициясы
uniform vec3 viewPos;                     // Байкоочунун (камеранын) позициясы
uniform sampler2D diffuseTexture;         // Текстура
uniform vec3 diffuseLight;          // Диффуздук жарык
uniform vec3 ambientLight;          // Фондук жарык
uniform vec3 Ka;                    //Фондук жарыктандыруу коэфициенти
uniform vec3 Kd;                    // Диффуздук жарыктандыруу коэфициенти
const float roughness = 10.0;        //беттин тегиз эместигин аныктайт
void main ()
{
        vec3 n = normalize ( Normal );
        vec3 l = normalize ( lightPos - p );
        vec3 v = normalize ( viewPos - p );
        float roughness2 = roughness * roughness;
        vec3 ambient = Ka * ambientLight;
        float A = 1.0 - ( 0.5 * roughness2 ) / ( roughness2 + 0.33 );
        float B = ( 0.45 * roughness2 ) / ( roughness2 + 0.09 );
        float VdotN = dot ( v, n );
        float LdotN = dot ( l, n );
```

```
        float diff = max ( 0.0, LdotN );
        float angleViewNormal  = acos ( VdotN );
        float angleLightNormal = acos ( LdotN );
        float angleDiff = max( 0.0, dot ( normalize ( v - n * VdotN ), normalize ( l - n * LdotN )));
        float alpha = max ( angleViewNormal, angleLightNormal );
        float beta  = min ( angleViewNormal, angleLightNormal );
        vec3 diffuse = Kd * diffuseLight * diff * ( A + B * angleDiff * sin ( alpha ) * tan ( beta ));
        color = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0 );
}
```

Бул модель берген көрүнүш сүрөт 4.4-де берилди.



Сүрөт 4.3. Wrap-around модели



Сүрөт 4.4. Орен-Наяр модели

4. 5. **Миннеарт жарыктандыруу модели**

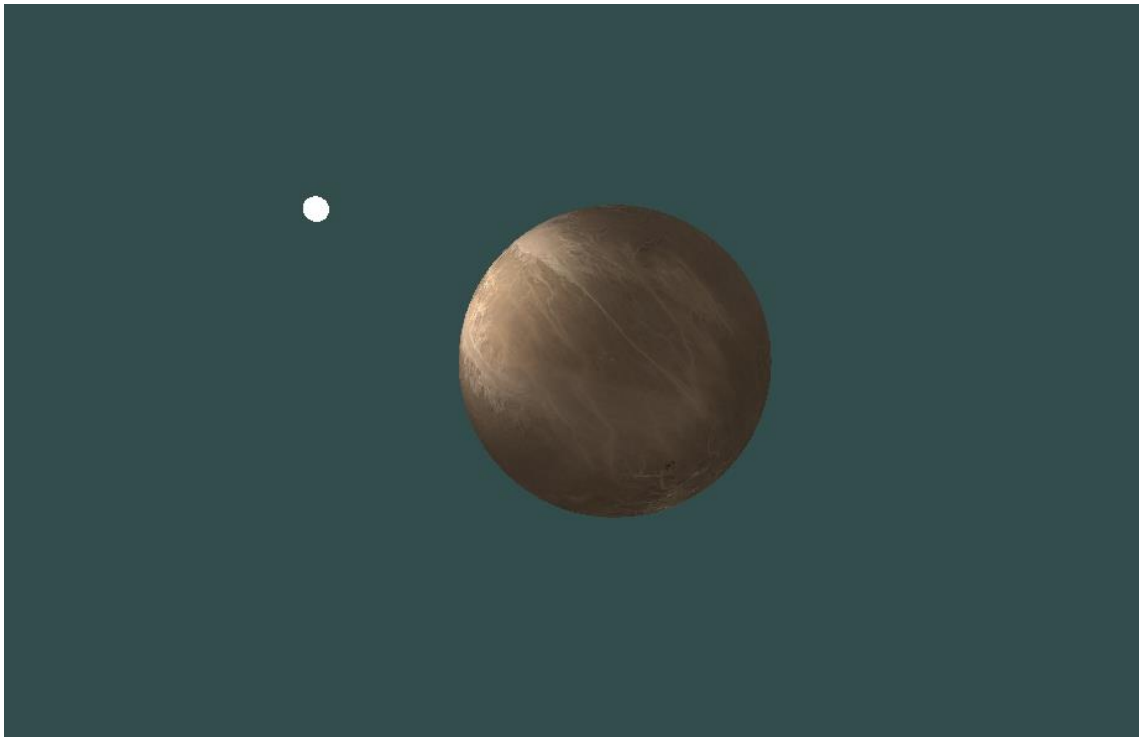
Бул модель боюнча диффуздук чагылууну эсептөө үчүн диффуздук жарыктандыруунун ургалдуулугу (*diffuseLight*), диффуздук чагылдыруу коэффициенти (*Kd*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектор (*l*), кароо вектору (*v*). Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына жана камерага багытталган векторлор пикселдик шейдерде эсептелет.

Бул моделдин пиксельдик щейдери төмөндө берилди:

```
//Minnaert фрагменттик шейдери
#version 130
in vec3 Normal;
```

```
in vec3 p;
in vec2 TexCoord;
out vec4 color;
uniform vec3 lightPos;
uniform vec3 viewPos;
uniform sampler2D diffuseTexture;
uniform vec3 diffuseLight;
uniform vec3 ambientLight;
uniform vec3 Ka;
uniform vec3 Kd;
void main ()
{
	vec3 ambient = Ka * ambientLight;
	vec3 n = normalize ( Normal );
	vec3 l = normalize ( lightPos - p );
	vec3 v = normalize ( viewPos - p );
	vec3 diffuse = diffuseLight * Kd;

	const float k = 0.8;
	float d1 = pow ( max ( dot ( n, l ), 0.0 ), 1.0 + k );
	float d2 = pow ( 1.0 - dot ( n, v ), 1.0 - k );
	diffuse = diffuse * d1 * d2;

	color = texture (diffuseTexture, TexCoord );
	color = color * vec4 ( ambient + diffuse, 1.0);
}
```

Бул модель берген көрүнүш сүрөт 4.5-де берилди.



Сүрөт 4.5. Миннеарт жарыктандыруу модели

4. 6. **Фонг жарыктандыруу модели**

Бул модель материал бетиндеги жылтыроону моделдейт. Бул модель боюнча күзгүлүү чагылууну эсептөө үчүн күзгүлүү жарыктандыруунун ургалдуулугу (*specularLight*), спекулярдык чагылдыруу коэффициенти (*Ks*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектор (*l*), кароо вектору (*v*) жана жылтыроо коэффициенти (*specPower*) керектелет. Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына жана камерага багытталган векторлор пикселдик шейдерде эсептелет. Ал эми диффуздук чагылуу компоненти Ламберт модели боюнча эсептелет.

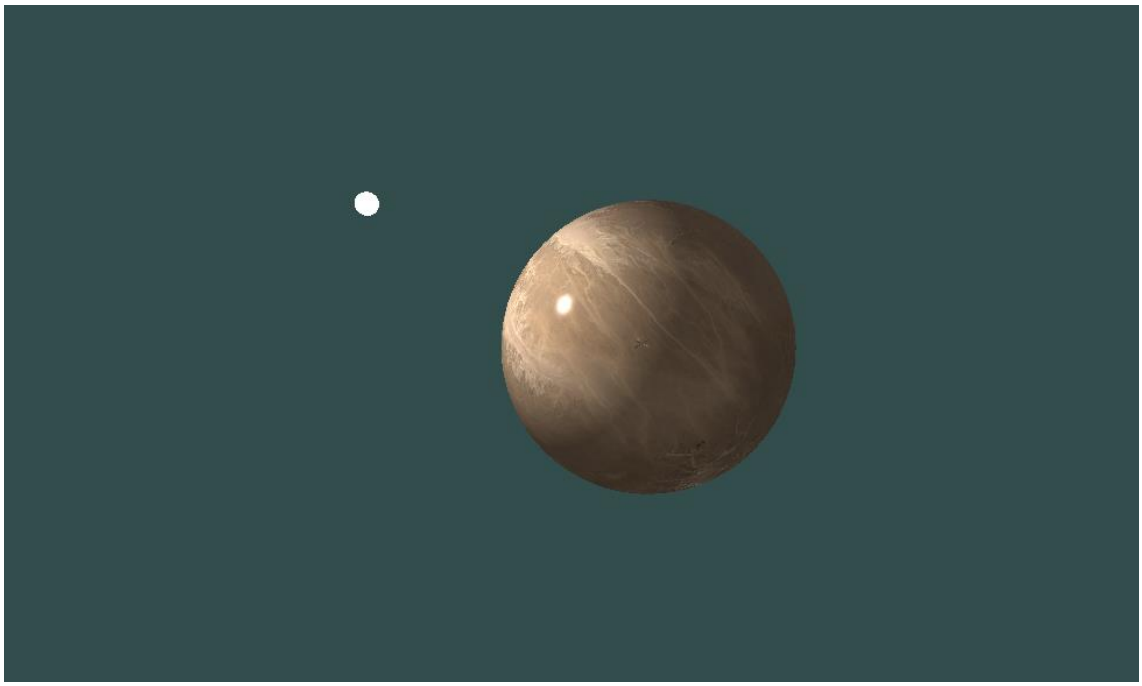
Бул моделдин пиксельдик щейдери төмөндө берилди:

```
// Фонг фрагменттик шейдери
#version 130
```

```
in vec3 Normal;
in vec3 p;
in vec2 TexCoord;
out vec4 color;

uniform vec3 lightPos;
uniform vec3 viewPos;
uniform sampler2D diffuseTexture;
uniform sampler2D specularTexture;
uniform vec3 ambientLight;
uniform vec3 diffuseLight;
uniform vec3 specularLight;
uniform vec3 Ka;
uniform vec3 Kd;
uniform vec3 Ks;
uniform float specPower;              //жылтыроо коэффициенти

void main (void)
{
        vec3 ambient = Ka * ambientLight;
        vec3 n = normalize ( Normal );
        vec3 l = normalize ( lightPos - p );
        vec3 v = normalize ( viewPos - p );
        float diff = max ( dot ( n, l ), 0.0 );
        vec3 diffuse = diffuseLight * ( diff * Kd );

        vec3 r = reflect ( -l, n );       // чагылуу вектору
        float spec = pow ( max ( dot ( v, r ), 0.0 ), specPower );
        vec3 specular = Ks * spec * specularLight;
```

```
    color = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0 ) +
    texture ( specularTexture, TexCoord ) * vec4 ( specular, 1.0 );
}
```

Бул модель берген көрүнүш сүрөт 4.6-да берилди.



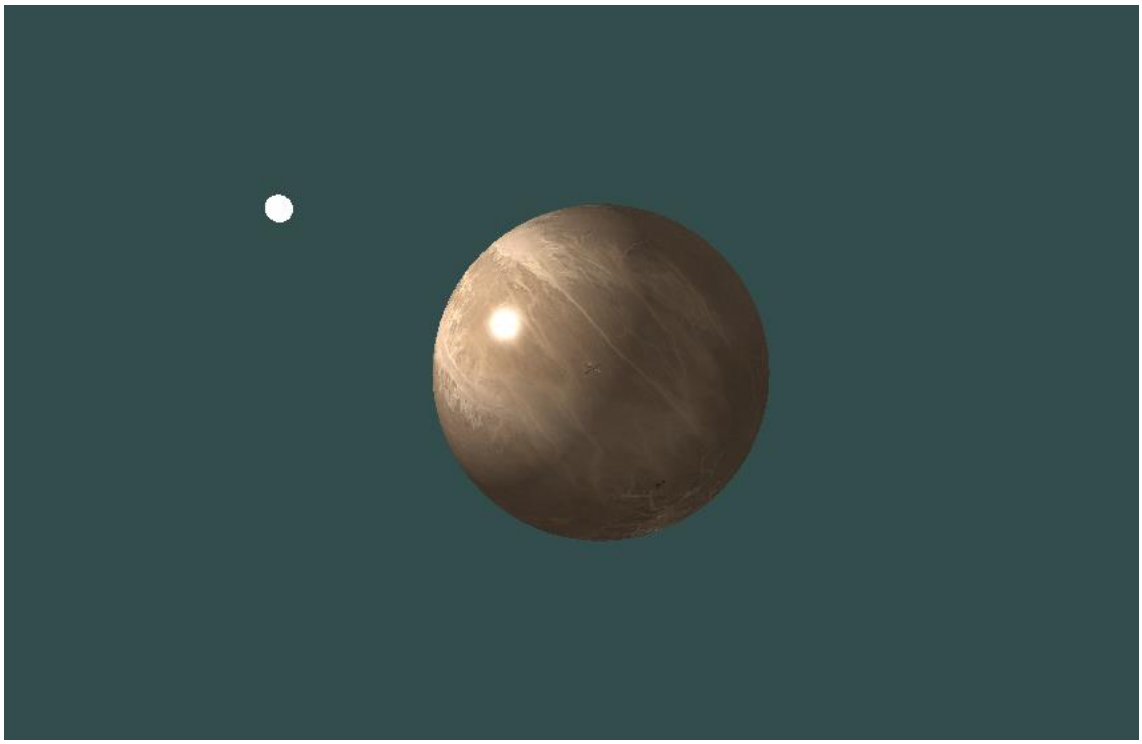
Сүрөт 4.6. Фонг жарыктандыруу модели

4. 7. **Блинн-Фонг жарыктандыруу модели**

Бул модель дагы материал бетиндеги жылтыроону моделдейт. Фонг моделинен айырмаланып, эсептөөдө чагылуу векторунун ордуна аралык вектору колдонулат. Бул модель боюнча күзгүлүү чагылууну эсептөө үчүн күзгүлүү жарыктандыруунун ургалдуулугу (*specularLight*), спекулярдык чагылдыруу коэффициенти (*Ks*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектор (*l*), кароо вектору (*v*) жана жылтыроо коэффициенти (*specPower*) керектелет.

Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына жана камерага багытталган векторлор пикселдик шейдерде эсептелет. Ал эми диффуздук чагылуу компоненти Ламберт модели боюнча эсептелет.

Бул модель берген көрүнүш сүрөт 4.7-де берилди.



Сүрөт 4.7. Блинн-Фонг модели

Бул моделдин пиксельдик щейдери төмөндө берилди:

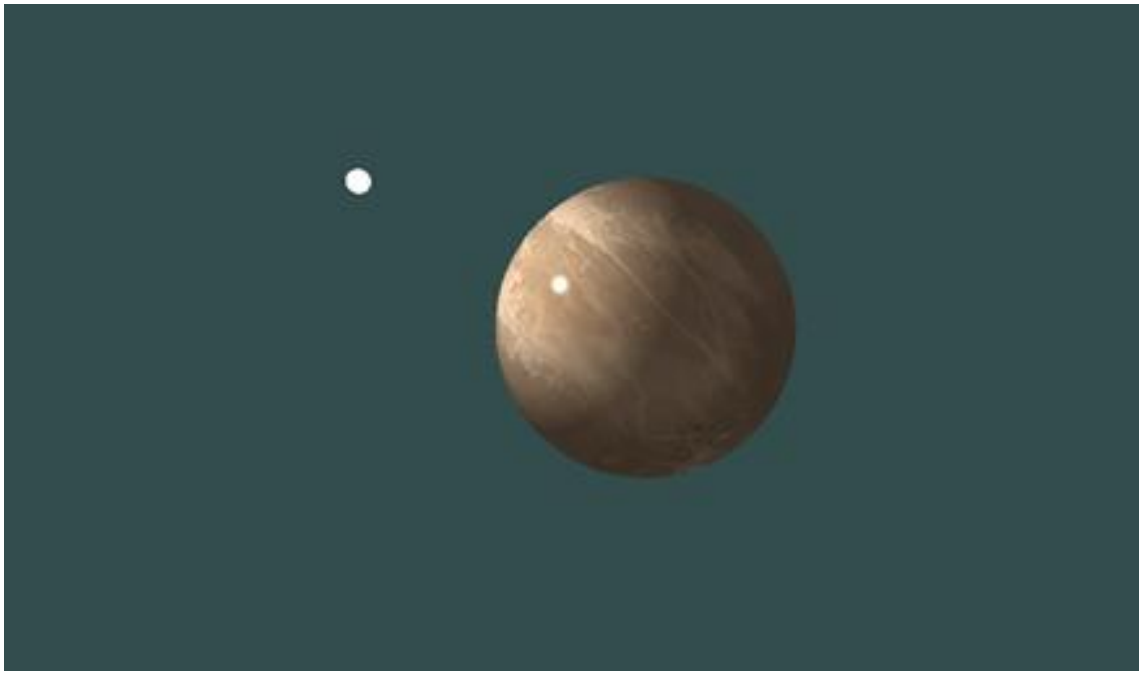
```
//Блин-Фонг фрагменттик шейдери
#version 130
in vec3 Normal;
in vec3 p;
```

```
in vec2 TexCoord;
out vec4 color;
uniform vec3 lightPos;
uniform vec3 viewPos;
uniform sampler2D diffuseTexture;
uniform sampler2D specularTexture;
uniform vec3 ambientLight;
uniform vec3 diffuseLight;
uniform vec3 specularLight;
uniform vec3 Ka;
uniform vec3 Kd;
uniform vec3 Ks;
uniform float specPower;              //жылтыроо коэффициенти

void main ()
{
        vec3 ambient = Ka * ambientLight;
        vec3 n = normalize ( Normal );
        vec3 l = normalize ( lightPos - p );
        vec3 v = normalize ( viewPos - p );
        float diff = max ( dot ( n, l ), 0.0 );
        vec3 diffuse = diffuseLight * ( diff * Kd );

        vec3 h = normalize ( l + v );            // аралык вектору
        float spec = pow ( max ( dot ( n, h ), 0.0 ), specPower );
        vec3 specular = Ks * spec * specularLight;

        color = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0 ) +
        texture ( specularTexture, TexCoord ) * vec4 ( specular, 1.0 );
}
```

4. 8. **Ward изотроптук модели**

Бул модель дагы материал бетиндеги жылтыроону моделдей. Блинн-Фонг модели сыяктуу бул моделде да аралык вектору колдонулат, бирок даражага чыгаруу функциясынын ордуна экспонента колдонулат. Бул модель боюнча күзгүлүү чагылууну эсептөө үчүн күзгүлүү жарыктандыруунун ургалдуулугу (*specularLight*), спекулярдык чагылдыруу коэффициенти (*Ks*), чекиттин нормаль вектору (*n*), жарык булагына багытталган вектор (*l*), кароо вектору (*v*) жана бодурдук коэффициенти (*k)* керектелет. Фрагменттин позициясы менен нормаль интерполяцияланат. Жарык булагына жана камерага багытталган векторлор пикселдик шейдерде эсептелет. Ал эми диффуздук чагылуу компоненти Ламберт модели боюнча эсептелет.

Бул модель берген көрүнүш сүрөт 4.8-де берилди.



Сүрөт 4.8. Ward изотроптук модели

Бул моделдин пиксельдик щейдери төмөндө берилди:

```
//Ward фрагменттик шейдери
#version 130
in vec3 Normal;
in vec3 p;
in vec2 TexCoord;
out vec4 color;

uniform vec3 lightPos;
uniform vec3 viewPos;
uniform sampler2D diffuseTexture;
uniform sampler2D specularTexture;
uniform vec3 ambientLight;
uniform vec3 diffuseLight;
uniform vec3 specularLight;
uniform vec3 Ka;
uniform vec3 Kd;
uniform vec3 Ks;
uniform float k;                //беттин тегиз эместигин аныктайт
void main (void)
{
        vec3 ambient = Ka * ambientLight;

        vec3 n = normalize ( Normal );
        vec3 l = normalize ( lightPos - p );
        vec3 v = normalize ( viewPos - p );
        float diff = max ( dot ( n, l ), 0.0 );
```

```
    vec3 diffuse = diffuseLight * ( diff * Kd );

    vec3 h = normalize ( l + v );              // аралык вектору
    float NdotH = dot( h, n );
    float spec = NdotH * NdotH;
    vec3 specular = Ks * specularLight * exp ( - k * (1.0 - spec) / spec );

    color = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0 ) +
    texture ( specularTexture, TexCoord ) * vec4 ( specular, 1.0 );
}
```

4. 9. **Финалдык проект**

Финалдык тиркеме жогоруда каралган жарыктандыруу моделдерин демонстрациялоого багытталган. Бул тиркеме моделдерди реалдуу убакыт компьютер графикасында ишке ашырат.

Тиркемеде төмөнкү жарыктандыруу моделдери ишке ашырылган:

1. Ламберт диффуздук модели (Lambert)
2. Wrap-around диффуздук модели (Wrap-around)
3. Орен-Наяр диффуздук модели (Oren-Nayar)
4. Миннеарт модели (Minnaert)
5. Фонг спекулярдык модели (Phong)
6. Блинн спекулярдык модели (Blinn-Phong)
7. Вард модели (Ward) (изотроптук)

Тиркеме ыңгайлуу интейфейске ээ жана жөнөкөй башкаруу элементтерин колдонот.

Системага коюлган талаптар

Тиркеме туура иштөө үчүн компьютер төмөнкү минималдуу талаптарга жооп бериш керек:

Иштетүү системасы: Windows 7 жана жогору

Процессор: Intel Pentium (R) же окшош

Оперативдик эс: 2 Gb

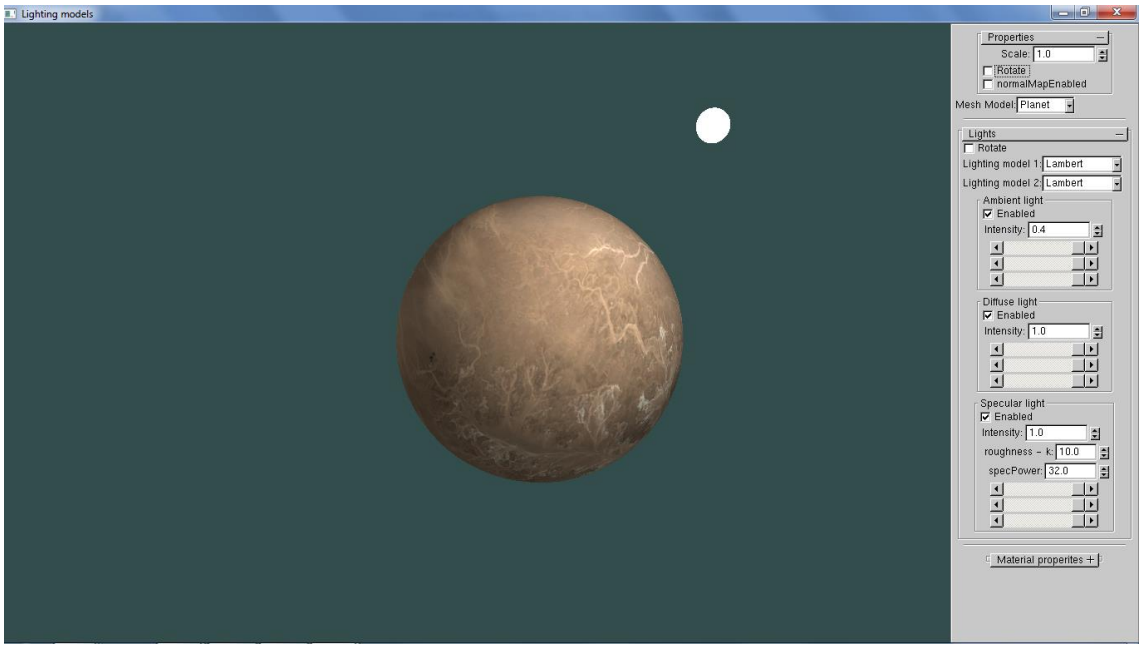
Видеокарта: OpenGL 3.0 же андан жогорку версиясын колдоо керек

Монитор: 1024*768 чечилүүгө ээ жана жогору

4. 9. 1. Тиркеменин интерфейси

Негизги терезе

Негизги папкадан тиркемени (Lighting_models.exe файлын) иштетиңиз, төмөнкү терезе ачылат (сүрөт 4.9):

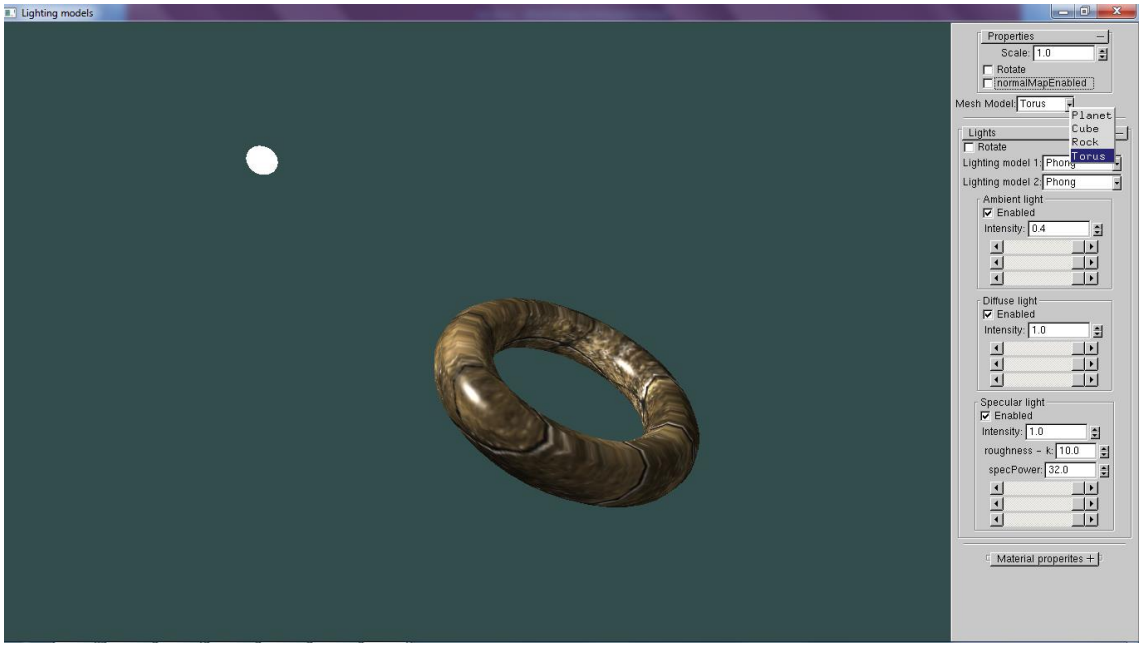


Сүрөт 4.9. Тиркеменин негизги терезеси

Тиркеменин оң жагында негизги башкаруу элементтери жайгашкан. Түшүүчү тизмеден керектүү жарыктандыруу модель түрүн жана үч өлчөмдүү модельди тандоого болот.

Объекттерди тандоо.

Биринчи менюдан объекттерди тандасаңыз болот, жеткиликтүү объекттер: “planet” (планета), “cube” (куб), “rock” (аска ташы), “torus” (торус) (сүрөт 4.10).



Сүрөт 4.10. Объекттерди тандоо

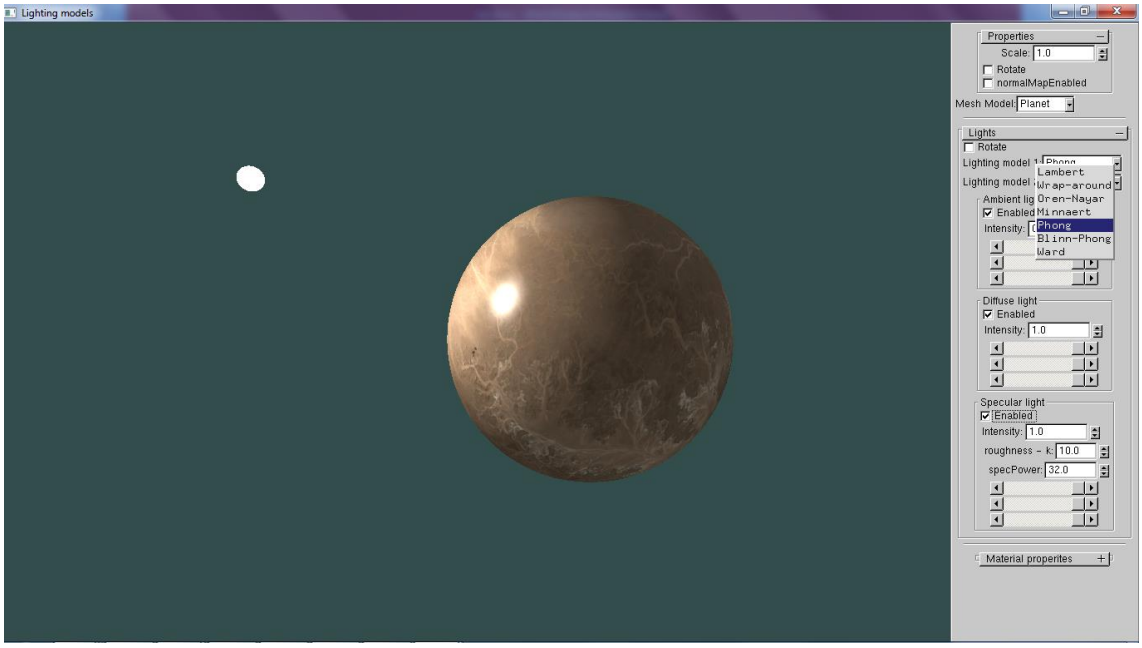
Ошондой эле объектти чоңойтсо жана айлантса болот.

Жарыктандыруу моделдерин тандоо

Экинчи менюдан жарыктандыруу моделдерин тандасаңыз болот (сүрөт 4.11). Ошондой эле жарык компоненттеринин (фондук – ambient, диффуздук –

diffuse, спекулярдык – specular) түсүн, ургалдуулугун (intensity), жарык булагынын позициясын өзгөртүп, жарыктандыруу кандай өзгөрөрүн байкаса болот.

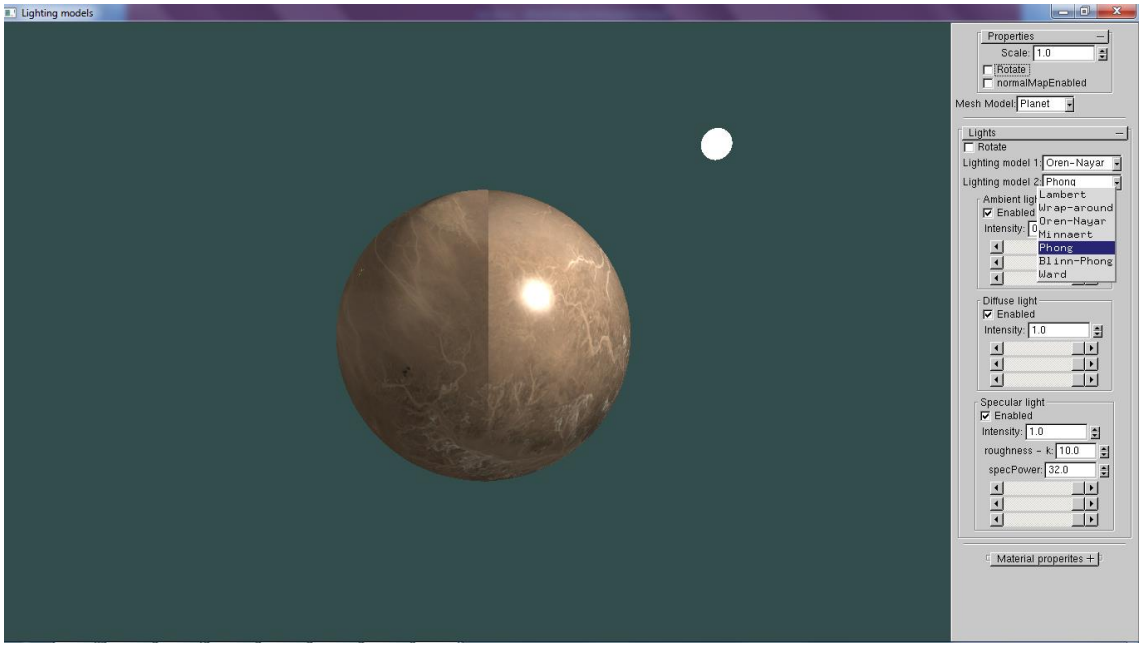
Андан тышкары эки жарыктандыруу моделдерин салыштырса болот. Тиркеменин сол жак жарымында бир жарыктандыруу модели, оң жагында башка жарыктандыруу моделдери ишке ашыралат (сүрөт 4.12).



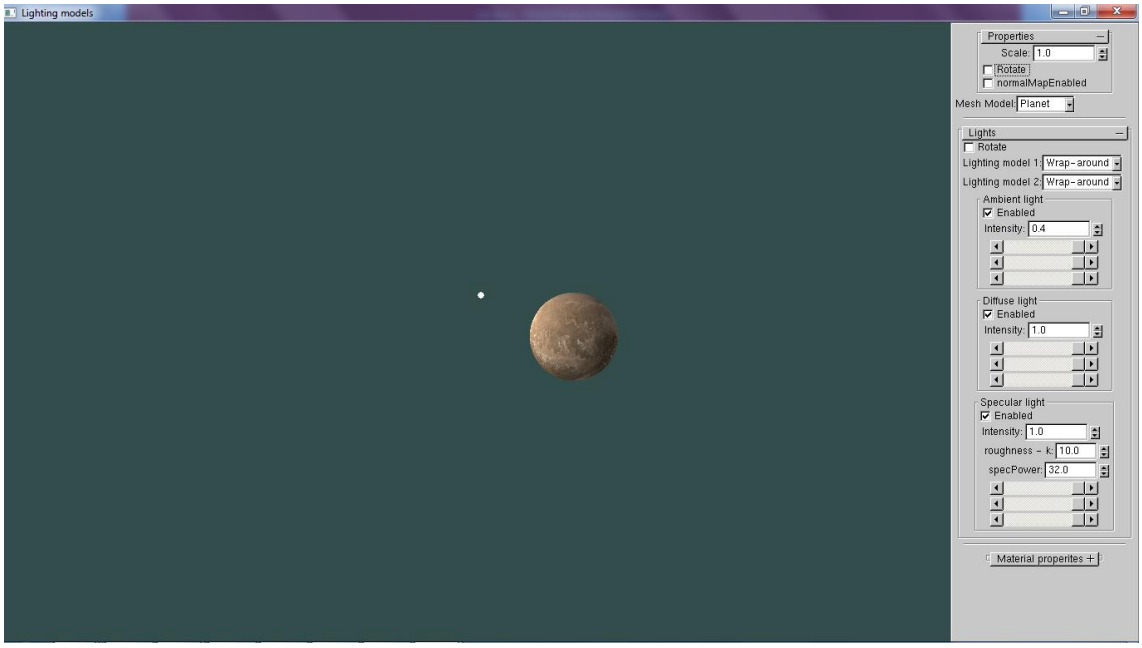
Сүрөт 4.11. Жарыктандыруу моделдерин тандоо

4. 9. 2. Сценаны башкаруу

Тиркемеде сценаны жакындатуу жана алыстатууга болот (сүрөт 4.13). Сценаны жакындатыш үчүн өйдө-төмөн (↑, ↓) жебе баскычтары колдонулат. Сценаны оңго же солго жылдырыш үчүн оңго-солго (→ , ←) жебе баскычтары колдонулат.



Сүрөт 4.12. Жарыктандыруу моделдерин салыштыруу



Сүрөт 4.13. Сценаны алыстатуу

Сценаны горизанталдык тегиздикте айландырыш үчүн чычкандын сол баскычын басып, коё бербей оңго же солго жылдыруу керек. Ал эми сценаны вертикалдык тегиздикте жылдыруу үчүн чычкандын оң баскычын басып, коё бербей жогору же төмөн жылдыруу керек.

4. 9. 3. Финалдык проекттин шейдерлери

Финалдык проектте жогоруда каралган бардык жарыктандыруу моделдеринин алгоритмдери бир шейдерде ишке ашырылды. Жарык моделдерин салыштыруу үчүн эки жарыктандыруу модели (*light_model1* жана *light_model2*) колдонулат. Андан тышкары жарык булагы (сфера) үчүн өзүнчө шейдерлер колдонулду.

Финалдык проекттин чокулар жана фрагменттик шейдери төмөндө берилди:

```
// Финалдык проект чокулар шейдери
#version 130
in vec3 position;          //чекит (чоку) координаттары (өздүк координаттар системинде)
in vec2 texCoord;          //текстура координаттары
in vec3 normal;            //чекиттин нормалы

out vec3 p;                //чекит (чоку) координаттары (дүйнө координаттар системинде)
out vec3 Normal;           //нормальды фрагменттик шейдерге жиберүү
out vec2 TexCoord;         //текстура координаттарын фрагменттик шейдерге жиберүү

uniform mat4 model;              //модель матрицасы
uniform mat4 view;               //кароо матрицасы
uniform mat4 projection;         //проекция матрицасы
uniform mat4 normalMatrix;       //нормаль матрицасы
```

```
void main()
{
	p = vec3(model * vec4 (position,1.0f));		// чекитти трансформациялоо
	Normal = mat3(normalMatrix) * normal;	// нормальды трансформациялоо
	TexCoord = texCoord;
	gl_Position = projection * view * model * vec4 (position, 1.0f);
}

//////////////////////////////////////////////////////////////////
//Финалдык проект фрагменттик шейдери
#version 130
in vec3 Normal;
in vec3 p;
in vec2 TexCoord;
out vec4 color;

uniform vec3 lightPos;
uniform vec3 viewPos;

uniform sampler2D diffuseTexture;
uniform sampler2D specularTexture;
uniform sampler2D normalTexture;
uniform vec3 ambientLight;
uniform vec3 diffuseLight;
uniform vec3 specularLight;
uniform vec3 Ka;
uniform vec3 Kd;
uniform vec3 Ks;
uniform float specPower;
```

```
uniform float k;

uniform int light_model1;
uniform int light_model2;
uniform int WIDTH;

vec3 Lambert ( vec3 n,vec3 l )
{
	float diff = max ( dot ( n, l ), 0.0 );
	vec3 diffuse = diffuseLight * ( diff * Kd);// Диффуздук чагылуу
	return diffuse;
}

vec3 Wrap_around ( vec3 n, vec3 l )
{
	float factor = 0.5;
	float diff = max ( dot ( n, l ) + factor, 0.0 ) / ( 1.0 + factor );
	vec3 diffuse = diffuseLight * ( diff * Kd);
	return diffuse;
}

vec3 Oren_Nayar ( vec3 n,vec3 l, vec3 v )
{
	const float roughness = 10.0;			//беттин тегиз эместигин аныктайт
	float roughness2 = roughness * roughness;
	float A = 1.0 - (0.5 * roughness2) / (roughness2 + 0.33);
	float B = ( 0.45 * roughness2) / (roughness2 + 0.09);
	float VdotN = dot(v, n);
	float LdotN = dot(l, n);
	float diff = max(0.0, LdotN);
```

```
        float angleViewNormal  = acos(VdotN);
        float angleLightNormal = acos(LdotN);
        float angleDiff = max ( 0.0, dot (normalize ( v - n * VdotN ), normalize ( l - n * LdotN)));
        float alpha = max (angleViewNormal, angleLightNormal);
        float beta  = min (angleViewNormal, angleLightNormal);
        vec3 diffuse = Kd * diffuseLight  * diff * (A + B * angleDiff * sin(alpha) * tan(beta));
        return diffuse;
}

vec3 Minnaert ( vec3 n,vec3 l, vec3 v )
{
        vec3 diffuse = diffuseLight * Kd;
        const float k = 0.8;
        float d1 = pow ( max ( dot ( n, l ), 0.0 ), 1.0 + k );
         float d2 = pow ( 1.0 - dot ( n, v ), 1.0 - k );
        diffuse = diffuse * d1 * d2;
        return diffuse;
}

vec3 Phong ( vec3 n,vec3 l, vec3 v )
{
        vec3 r = reflect ( -l, n );  // чагылуу вектору
        float spec = pow ( max ( dot ( v, r ), 0.0 ), specPower );
        vec3 specular =  Ks * spec * specularLight;
        return specular;
}

vec3 Blinn_Phong ( vec3 n,vec3 l, vec3 v )
{
        vec3 h = normalize ( l + v );          // аралык вектору
```

```
        float spec = pow ( max ( dot ( n, h ), 0.0 ), specPower );
        vec3 specular =  Ks * spec * specularLight;
        return specular;
}

vec3 Ward(vec3 n,vec3 l, vec3 v)
{
        vec3 h = normalize ( l + v );          // аралык вектору
        float      NdotH = dot( h, n );
        float      spec  = NdotH * NdotH;
        vec3 specular = Ks * specularLight * exp ( - k * (1.0 - spec) / spec );
        return specular;
}

void main (void)
{
        vec3 ambient = Ka * ambientLight;                    // фондук жарык
        vec3 n = normalize(Normal);         // нормаль векторун нормалдаштыруу
        vec3 l = normalize ( lightPos - p ); // Жарык булагына багытталган бирдик вектор
        vec3 v = normalize ( viewPos - p );            // Камерага багытталган бирдик вектор
        vec3 diffuse = vec3 (0.0);
        vec3 specular = vec3 (0.0);

        switch (light_model1)
        {
                case 0:  //Lambert
                {
                        diffuse = Lambert (n,l);
                        break;
                }
```

```
case 1:  //Wrap_around
{
        diffuse = Wrap_around (n,l);
        break;
}
case 2:  //Oren-Nayar
{
        diffuse = Oren_Nayar (n,l,v);
        break;
}
case 3:  //Minnaert
{
        diffuse = Minnaert (n,l,v);
        break;
}
case 4:  //Phong
{
        diffuse = Lambert(n,l);
        specular = Phong (n,l,v);
        break;
}
case 5:  //Blinn-Phong
{
        diffuse = Lambert(n,l);
        specular = Blinn_Phong (n,l,v);
        break;
}
case 6:  //Ward
{
        diffuse = Lambert(n,l);
```

```
                specular = Ward (n,l,v);
                break;
        }
        default:
        {
                break;
        }
}

vec4 color1 = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0 )
+ texture ( specularTexture, TexCoord ) * vec4( specular, 1.0);

diffuse = vec3 (0.0);
specular = vec3 (0.0);

switch (light_model2)
{
        case 0:
        {
                diffuse = Lambert ( n, l );
                break;
        }
        case 1:
        {
                diffuse = Wrap_around ( n, l );
                break;
        }
        case 2:
        {
                diffuse = Oren_Nayar ( n, l, v );
```

```
                break;
            }
            case 3:
            {
                diffuse = Minnaert ( n, l, v );
                break;
            }
            case 4:
            {
                diffuse = Lambert ( n, l );
                specular = Phong ( n, l, v );
                break;
            }
            case 5:
            {
                diffuse = Lambert ( n, l );
                specular = Blinn_Phong ( n, l, v );
                break;
            }
            case 6:
            {
                diffuse = Lambert ( n, l );
                specular = Ward (n, l, v );
                break;
            }
            default:
            {
                break;
            }
}
```

```
        vec4 color2 = texture ( diffuseTexture, TexCoord ) * vec4 ( ambient + diffuse, 1.0)
        + texture ( specularTexture, TexCoord ) * vec4 ( specular, 1.0 );
        if ( gl_FragCoord.x < WIDTH / 2 – 100 )
                color = color1;
        else
                color = color2;
}
```

Жарык булагы үчүн колдонулган чокулар дана фрагменттик шейдерлер төмөндө берилди:

```
// Жарык булагынын чокулар шейдери
#version 130
in vec3 position;
uniform mat4 model;                     //модель матрицасы
uniform mat4 view;                      //кароо матрицасы
uniform mat4 projection;  //проекция матрицасы
void main()
{
        gl_Position = projection * view * model * vec4 (position, 1.0f);
}
////////////////////////////////////////////////////////////////////////////////
// Жарык булагынын фрагменттик  шейдери
#version 130
out vec4 color;
uniform vec3 lightColor;
void main()
{
        color = vec4 ( lightColor, 1.0 );
}
```

5. ТАЛКУУЛАР ЖАНА АЛДЫДАГЫ ИЗИЛДӨӨЛӨР

Жарыктандырууну моделдөө компьютердик графикадагы чоң бир тармак, аны ишке ашыруунун көптөгөн ыкмалары бар. Бул иште каралган методдор жөнөкөй жана бат иштечү ыкмаларга кирет. Бул ыкмалардын жалпы кемчилиги көлөкөнү эсептөөнү камтышбайт. Ал үчүн башка кошумча ыкмаларды колдонуу керек. Дагы башка бир кемчилиги бардык үстүртүлүктөр үчүн жарай беришпейт, бир үстүртүлүктөрдүн чагылуусун жакшы моделдесе башка үстүртүлүктөрдүн чагылуусун туура эмес моделдейт. Бирок бул иште каралган жарыктандыруу моделдери стандарттуу моделдерге кирет, жана интерактивдүү тиркемелерде канааттанардык сапат жана ылдамдык беришет.

Бул иште жети локалдык жарыктандыруу модели каралды: Ламберт, Wrap-around, Орен-Наяр, Миннеарт диффуздук жарыктандыруу моделдери жана Фонг, Блинн-Фонг, Вард спекулярдык жарыктандыруу моделдери.

Диффуздук жарыктандыруу үчүн Ламберт модели эң көп колдонулат. Анын модификациясы Wrap-around модели жарыктандыруу областын кеңейтүү үчүн колдонулат. Ламберт модели жылтырабаган, тунук эмес, текши, жылмакай күңүрт материалдарды жакшы моделдейт. Ал эми Орен-Наяр модели күңүрт бодуракай, жылмакай эмес, текши эмес материалдардын диффуздук чагылуусун жакшы моделдейт. Ал эми Миннеарт жарыктандыруу модели планеталардын жарыктандырылуусун жакшы моделдейт.

Спекулярдык жарыктандыруу моделдери материалдардын күзгү сымал чагылуусун моделдешет. Фонг, Блинн-Фонг, Вард толук жарыктандыруу моделдеринде диффуздук чагылуу Ламберт методу менен эсептелет. Спекулярдык компоненти эсептөөдө эле айырма бар. Бул спекулярдык моделдер дээрлик окшош спекулярдык чагылуу берет. Бирок бирдей жылтыроо коэфициентти үчүн бликтердин көлөмү ар кандай болот. Ошондой эле Блинн-Фонг жана Вард моделдери эсептөөлөрдү ылдамдатыш үчүн көп эсептөөлөрдү талап кылган чагылуу векторун колдонбой, аралык векторун колдонушат. Ошондуктан Фонг моделине караганда Блинн-Фонг модели тиркемелерде көп колдонулат.

Бул моделдер оюндарда, графиканы колдонгон демонстрациялык программдар жана башка интерактивдүү тиркемелерде, компьютер графикасын окуутуда колдонулушу мүмкүн.

Бул иште каралган жарыктандыруу моделдери жакшырак көрүнүш берүү үчүн кошумча методдорду колдонуу керек (мисалы, normal mapping). Бирок кошумча методдор жарыктандырууну эсептөөнү эмес, чекит нормальдарын эсептөөнү алмаштырышат. Ошентип максатка жараша кошумча методдорду колдонуп же колдонбой интерактивдүү графикалык тиркемелерди түзүүгө болот. Интерактивдүү графикалык тиркемелерде жарыктандырууну жакшыртуу үчүн бир нече кошумча методдор колдонулат. Аларды изилдөө өзүнчө башка бир тема.

Бул иште каралган жарыктандыруу моделдери локалдык моделдерге кирет, андан тышкары глобалдык жарыктандыруу моделдери бар. Глобалдык жарыктандыруунун локалдык жарыктандырууга караганда бир нече артыкчылыктары бар: чыныгы фото-реалдуу сүрөттөлүштөрдү берет, көлөкөлөрдү эсептөө үчүн кошумча метод талап кылбайт. Ошол эле учурда алардын кемчиликтери бар: бул алгоритмдер көп татаал эсептөөлөрдү талап кылышат, графикалык процессордун мүмкүнчүлүктөрүн колдоно алышпайт. Анткени графикалык процессорлор рендеринг (визуалдоо) үчүн растеризация методун ишке ашырышат. Ал эми глобалдык жарыктандыруу методдору башка ыкмаларды колдонот. Азыркы учурду глобалдык жарыктандыруу моделдери интерактивдүү эмес тиркемелерде, фото-реалдуу сүрөттөлүштөрдү алуу үчүн колдонулат. Сүрөттөлүш канча татаал жана сапаттуу болгонуна жараша, ошончолук көп эсептөөлөр жүргүзүлөт жана убакыт көп сарпталат.

Бирок кубаттту аппараттык жабдык, атайын программдык жабдык жана параллельдүү эсептөөлөрдүү колдонуу менен глобалдык жарыктандырууну графикалык процессорлордо ишке ашырса болот. Тилеке каршы мындай жол менен да бул моделдер интерактивдүү тиркемелерде жакшы ылдамдыкка жете элек. Анкени бул ыкма жаңыдан эле изилденип келе жатат.

Келечекте локалдык жарыктандыруу методдорун жакшыртуу жана глобалдуу жарыктандыруу алгоритмерин изилдөөнү каалайбыз.

АДАБИЯТ

[1] В. И. Мураховский, "Компьютерная графика", Москва: АСТ-ПРЕСС СКД, 2002.

[2] М. А. Кудрина, К. Е. Климентьев, “Компьютерная графика”, Самара: Издательство СГАУ, 2013.

[3] К. В. Постнов, "Компьютерная графика", Москва, 2009.

[4] Т. О. Перемитина, "Компьютерная графика : учебное пособие", Томск: Эль Контент, 2012.

[5] А. И. Куликов, Т. Э. Овчинникова, "Алгоритмические основы современной компьютерной графики", Москва: Национальный открытый университет "ИНТУИТ", 2016.

[6] Л. А. Залогова, «Технологии трёхмерной компьютерной графики в высшей школе,» *Информатизация образования и науки,* т. 15, № 3, pp. 43-55, 2012.

[7] Ю. М. Баяковский, А. В. Игнатенко, “Начальный курс OpenGL”, Москва: Планета знаний, 2007.

[8] В. Т. Тозик, А. В. Меженин, К. А. Звягин, "3ds Max. Трехмерное моделирование и анимация на примерах", Санкт-Петербург: БХВ-Петербург, 2008.

[9] А. С. Стиренко, "3ds Max 2009/3ds Max Design 2009. Самоучитель", Москва: ДМК Пресс, 2008.

[10] А. В. Харьковский, "3ds Max 2013. Лучший самоучитель", Москва: Астрель, 2013.

[11] А. Боресков, "Расширения OpenGL", Сант-Петербург: БХВ-Петербург, 2005.

[12] А. А. Богуславский, С. М. Соколов, "Основы программирования на языке Си++", том II, Коломна: КГПИ, 2002.

[13] И. Е. Жигалов, И. А. Новиков, "Программирование компьютерной графики : учеб. пособие", Владимир : Изд-во ВлГУ, 2014.

[14] М. Ву, Т .Девис, Дж. Нейдер, Д. Шрайнер, “OpenGL. Руководство по програмированию”, 4- ред., Сант-Петербург: Питер, 2006.

[15] А. Боресков, "Разработка и отладка шейдеров", Сант-Петербург: БХВ-Петербург, 2006.

[16] Р. Д. Рост, "OpenGL. Трехмерная графика и язык программирования шейдеров. Для профессионалов", Сант-Петербург: Питер, 2005.

[17] К. Мацуда, Р. Ли, “WebGL: программирование трехмерной графики”, ДМК Пресс, 2015.

[18] В. В. Хлебников, А. А. Юров, «Моделирование реалистичных изображений объектов, используя различные алгоритмы расчета освещенности,» *Вестник ТГУ,* т. 15, № 2, pp. 732-735, 2010.

[19] Е. А. Снижко, Н. А. Флерова, А. В. Воронцов,, “Программирование компьютерной графики с использованием библиотеки OpenGl. Лабораторный практикум", Санкт-Петербург, 2005.

[20] Х. Дональд, М. П. Бейкер, "Компьютерная графика и стандарт OpenGL", Вильямс, 2005.

[21] D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane, “OpenGL programming guide”, 8th ed., Addison Wesley, 2013.

[22] V. S. Gordon, J. Clevenger, "Computer Graphics Programming in OpenGL with Java".

[23] А. В. Кравцов, «Обзор математических моделей освещения при построении трехмерных сцен,» *Известия ТулГУ, Технические науки,* № 3, 2010.

[24] M. Bailey, S. Cunningham, “Graphics Shaders Theory and Practice”, 2nd ed., CRC Press, 2012.

[25] И. Н. Егорова, М. А. Потемин, «Технология синтеза фотореалистичных изображений,» *Восточно-Европейский журнал передовых технологий,* т. 24, № 6/2, pp. 66-70, 2006.

[26] S. Parminder, "OpenGL ES 3.0 Cookbook", Packt Publishing, 2015.

[27] В. Порев, “Компьютерная графика”, Сант-Петербург: БХВ-Петербург, 2002.

[28] Дж. Фоли, А. ван Дэм, "Основы интерактивной машинной графики. В 2-х книгах", том II, Москва: Мир, 1985.

[29] E. Angel, D. Shreıner, “Interactive Computer Graphics. A top-down approach wıth shader-based opengl”, 6th ed., AddisonWesley, 2012.

[30] Е. А. Снижко, “Компьютерная геометрия и графика”, Санкт-Петербург, 2005.

[31] A. Boreskov, E. Shikin, “Computer Graphics. From pixels to programmable graphics hardware”, CRC Press, 2014.

[32] B. Phong, "Illumination for computer generated pictures," *Communications of ACM,* vol. 18, no. 6, 1975.

[33] S. K. Nayar, M. Oren, "Generalization of the Lambertian Model and Implications for Machine Vision," *International Journal on Computer Vision,* vol. 14, no. 3, pp. 227-251, 1995.

[34] J. Blinn, "Models of Light Reflection for Computer Synthesiezed Pictures," *Computer Graphics (SIGGRAPH 77 Proceedings),* vol. 11, no. 2, pp. 192-198, 1977.

[35] Mike Bailey; Steve Cunningham, "Introduction to computer graphics," in *SIGGRAPH Asia 2011 Courses,* Hong Kong, 2011.

ЖЕКЕ МААЛЫМАТ

Аты-жөнү	**Махабат Кулмурзаева**
Жарандыгы	Кыргыз
Туулган күнү жана жери	26.06.1990, Кыргызстан
Үй-бүлөлүк абалы	Никесиз
Тел:	+996 773 11 10 58
Fax	…….
E-mail	min_kiyal@mail.ru

БИЛИМ ДЕҢГЭЭЛИ

Даража	**Окуу жай**	**Бүткөн жылы**
Магистратура	Кыргыз-Түрк Манас Унив.	2017
Бакалавр	Кыргыз-Түрк Манас Унив.	2013
Мектеп	Н. Жүндүбаева атындагы орто-мектеп	2008

ИШ ТАЖРЫЙБАСЫ

Жыл	**Иш жай**	**Милдети**
……	…….	……..
…….	……	……..

ЧЕТ ТИЛДЕРИ

Орусча
Түркчө
Англисче

МАКАЛАЛАРЫ

1. Казакбаева З. М., Кулмурзаева М. И., ” Преподавание трехмерной компьютерной графики с использованием информационных технологий”, **Известия КГТУ им. И. Раззакова,** 2016