Теоретические основы трехмерной графики

Теория построения трехмерных изображений и рендеринга.

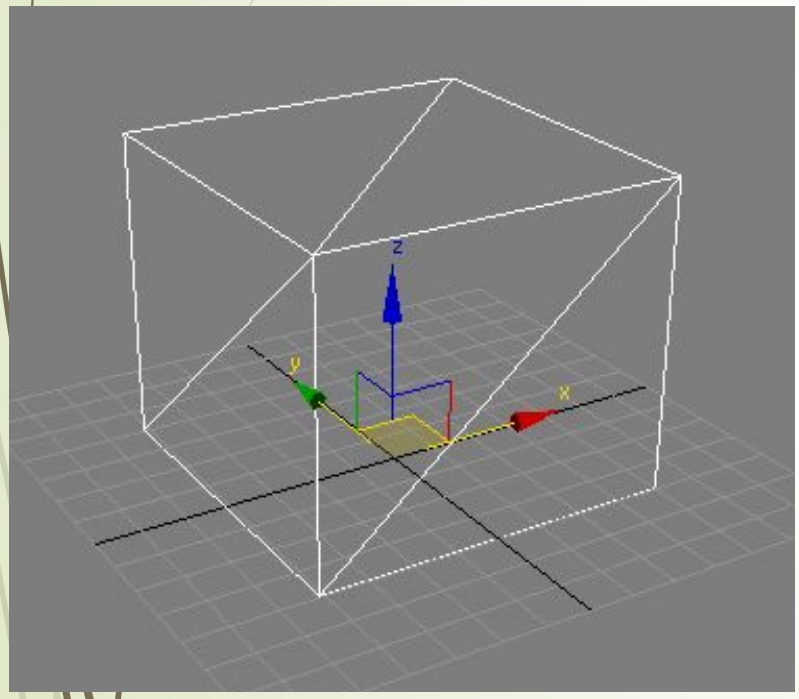
3D Graphics

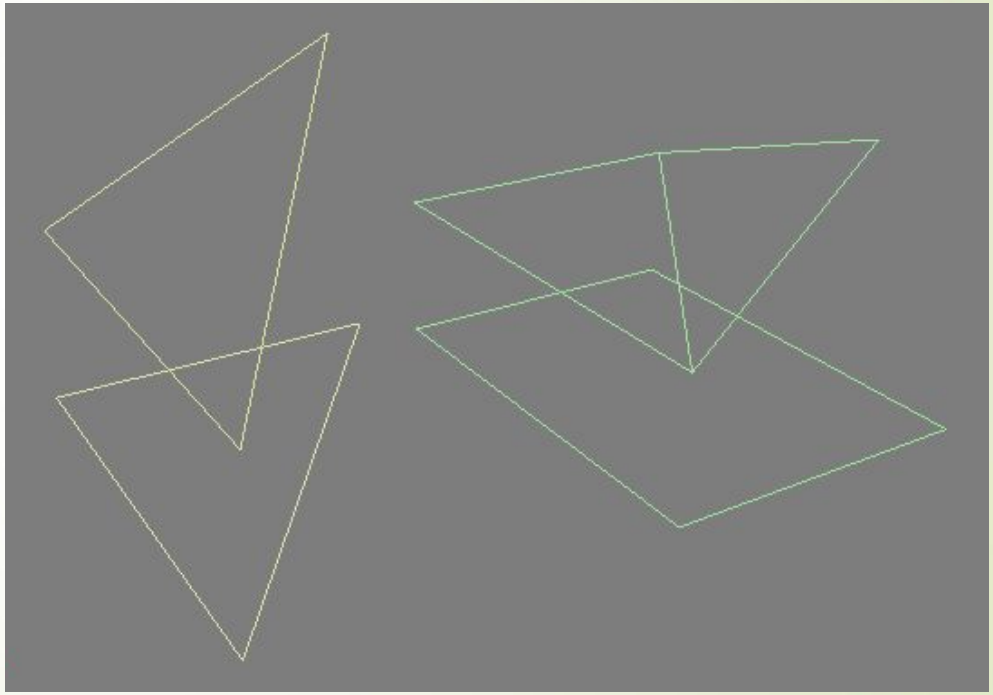
- **Трехмерная графика (3D Graphics)** – визуальное отображение трехмерной сцены или объекта. Для представления трехмерной графики на двумерном устройстве (дисплее) применяют **рендеринг**.

rendering

- **Рендеринг (rendering)** – процесс создания реалистичных изображений на экране, использующий математические модели и формулы для добавления цвета, тени и т. д.

Трехмерная модель

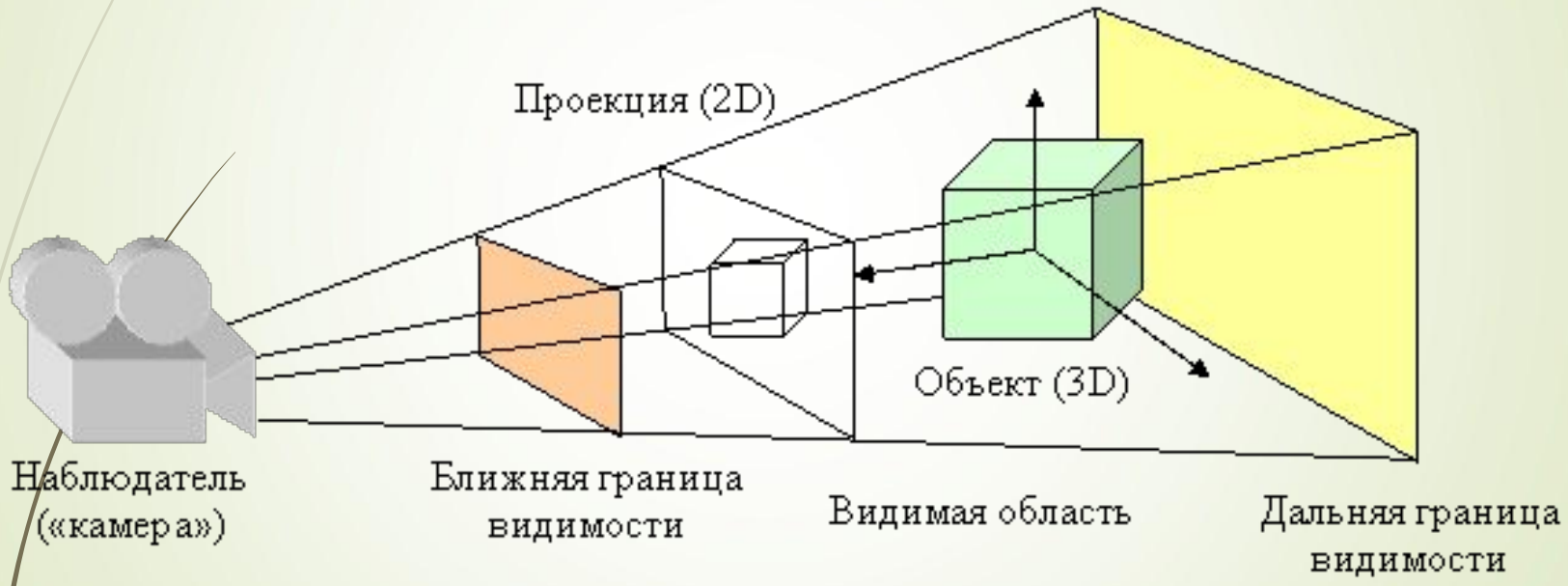




Из 3D в 2D

- Для отображения трехмерной модели на двухмерном экране монитора требуется произвести специальные математические преобразования трехмерных координат всех точек в двухмерные.

Камера

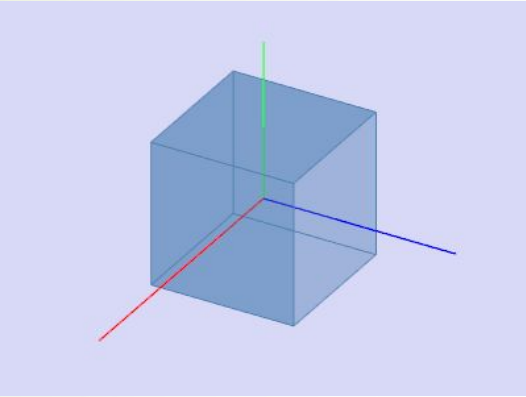

Проекция (2D)
Объект (3D)
Наблюдатель («камера»)
Ближняя граница видимости
Видимая область
Дальняя граница видимости

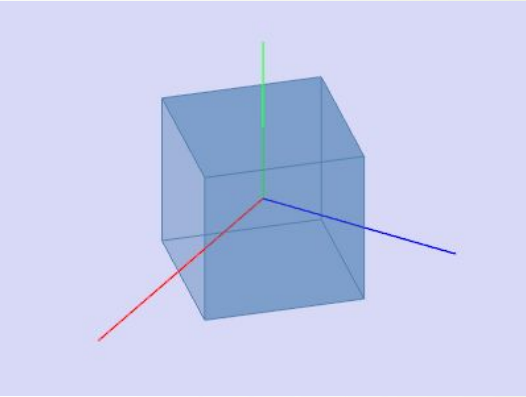
Аффинные преобразования

- При смене позиции наблюдателя требуется произвести преобразования координат, связанные с поворотом объекта, его масштабированием и учетом перспективы.

- Такие преобразования называются **аффинными преобразованиями**

Пример преобразования





- Аффинные преобразования применяются при повороте и трансформации объекта (масштабирование, скос).
- Допустим, что при повороте ординаты точек (y-координаты) не меняются. Значит, поворот происходит относительно оси y.

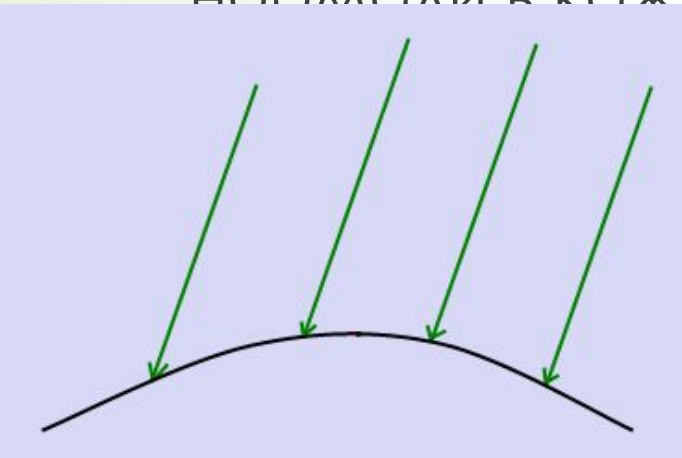
Освещение

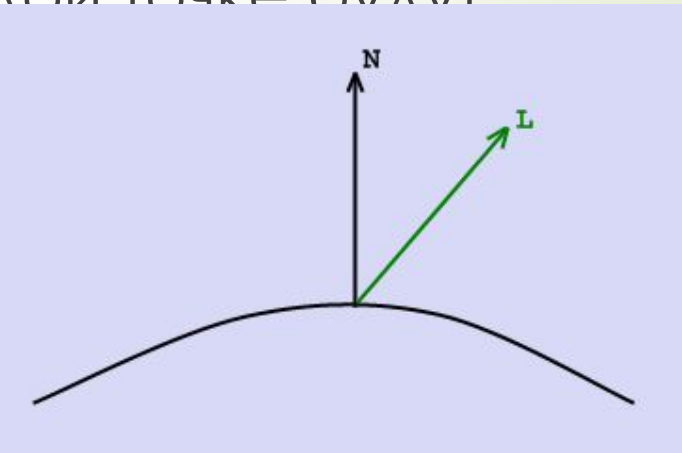
- Модель начинает выглядеть трехмерной только после моделирования освещения, а это напрямую связано с положением плоскостей (полигонов) модели, точнее, с направлением перпендикуляров к плоскостям – нормалей.

- Степень освещенности точки зависит от угла падения лучей света. Чем меньше угол между нормалью и лучом света, тем больше освещенность точки поверхности.

Нормали

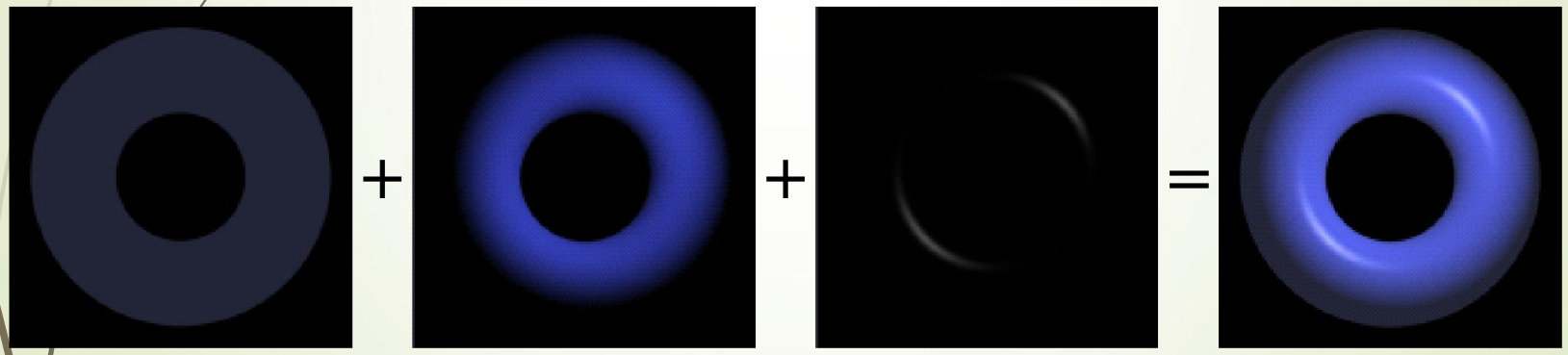
- Если взять ровную поверхность, то нормали в каждой ее точке будут одинаковыми и освещенность также будет одинаковой.
- А если поверхность неровная, то нормали в каждой точке будут





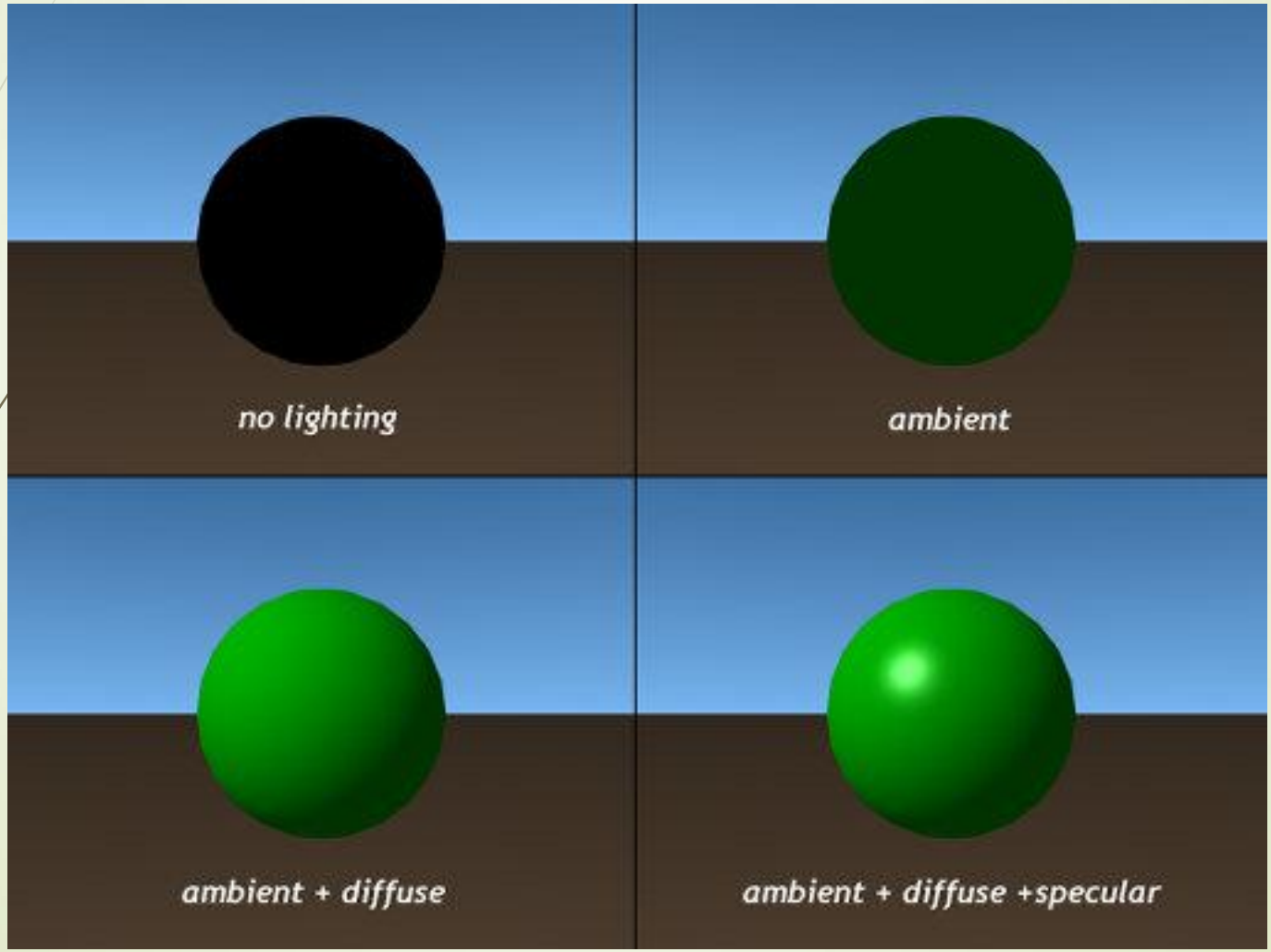
Модель освещенности Фонга

- Вклад в это значение от каждого источника света зависит от расстояния между источником света и точкой на поверхности.

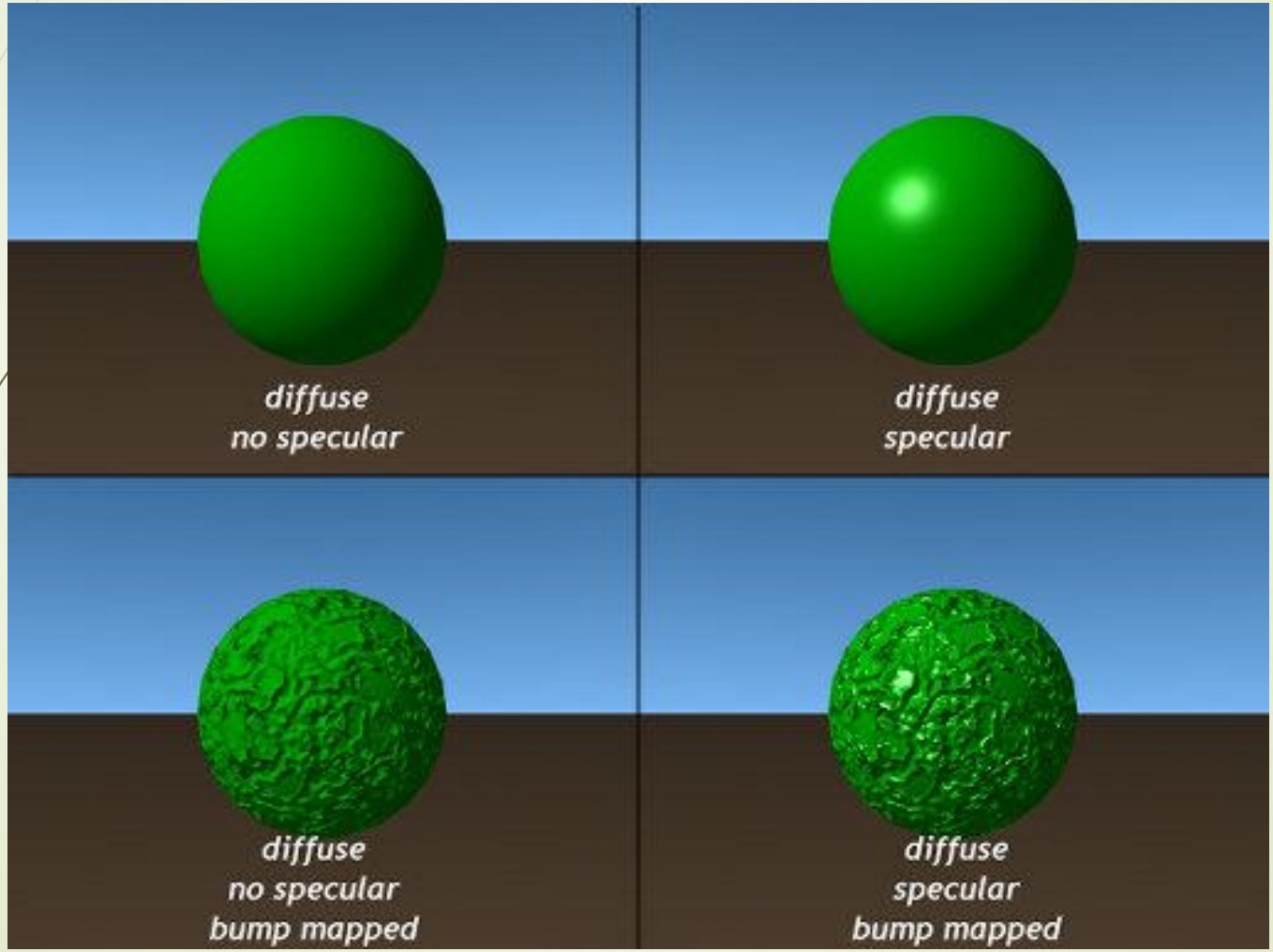


Фоновая составляющая | Рассеянная составляющая | Зеркальная составляющая | Суммарное освещение

Модель освещенности Фонга

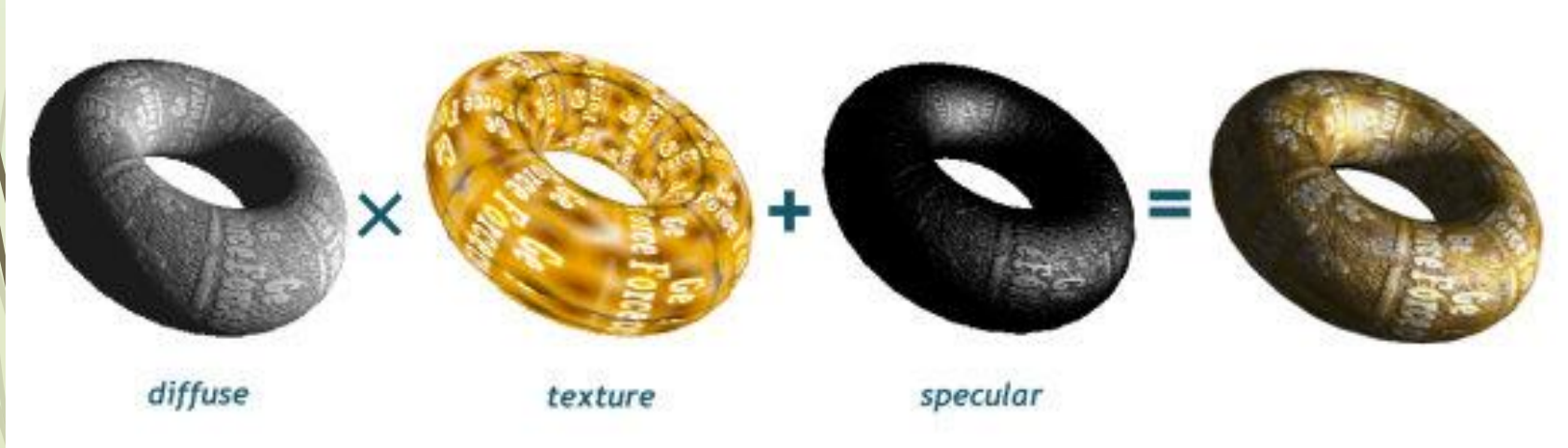


Модель Фонга-Блинна



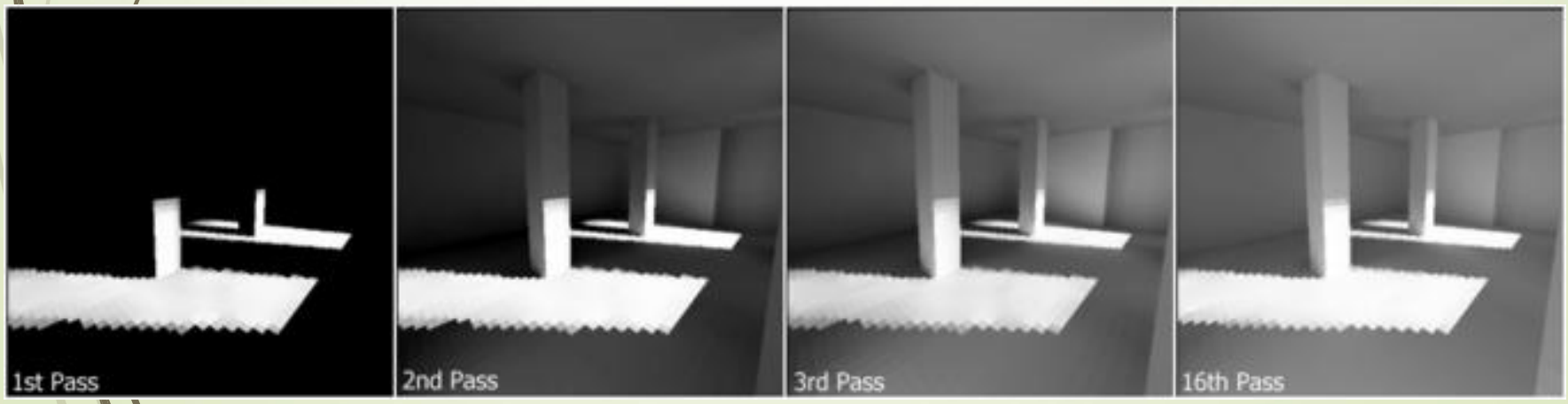
Дальнейшее развитие

- В дальнейшем, исследователи придумывали иные способы вычисления этой составляющей, более сложные (Blinn, Cook-Torrance, Ward), учитывающие распределение энергии света, его поглощение материалами

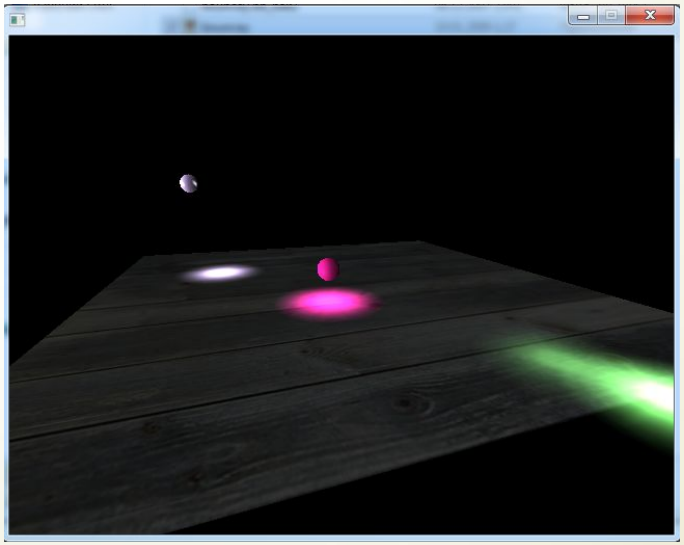


Игра света и тени

- С целью расчета теней и полутеней (диффузных теней) применяются различные алгоритмы – Ray tracing (трассировка лучей), Radiosity (излучение).
- На рисунке представлены результаты просчета теней и диффузной засветки при помощи Radiosity.



Первые уроки 3D программирование



Окно в 3d мир.

Камера
Свет

голова

Объекты
Действия с объектами

тело

окончание

ноги

Структура программы

голова

Graphics3D 640,480,16,1

Устанавливает 3D Графический способ.
Синтакс такой: Graphics3D ширина, высота, глубина цвета, режим
Ширина, высота – размер создаваемого экрана
Глубина – глубина цвета – 16, 24 или 32 бита на пиксель.
Режим:
0 – оконный во время разработки (если возможно), полноэкранный в .exe формате;
1 –полноэкранный
2 –оконный
3 –оконный, с возможностью изменения размеров окна.

голова

SetBuffer BackBuffer() – установить поток, буфер.

cam=CreateCamera() – это самая основная команда в 3D программах. Если не будет даже одной камеры то 3D программа работать не будет, камера-это тоже самое, что и наши глаза, через неё мы видим окружающий 3D мир.

Lit=CreateLight() – Это команда создает источник света. Можно изменять тип света, в скобки вписываем следующие значения:

1:направленный
2:точка
3: пятно

тело

cub=CreateCube() – Это мы создаем 3D объект куб.

Геометрические объекты:
CreateSphere () - сфера

CreateCylinder () - цилиндр

CreateCone() – конус

PositionEntity cub,0,0,5 – Эта команда помещает объект в позиции и в 3D месте.
ScaleEntity cub,1,0.3,1 – Изменяет размер объекта

RotateEntity cub,0,40,0 – Вращает объект так, чтобы это было при абсолютной ориентации.

тело

TurnEntity cub,0.6,0.2,0.3 – вращение
TurnEntity entity,pitch#, yaw#, roll#, [,global]

Tpitch# - X угол и эквивалентна наклону вперед / назад.

Yaw# - Y угол и эквивалентен вращению налево / вправо.

Roll# - Z угол и эквивалентен наклону налево / вправо.

UpdateWorld ()– UpdateWorld [anim_speed#]

anim_speed# (необязательный) – скорость для мультипликации.
Установлен по умолчанию в 1.

тело

Repeat

……..

Until KeyHit(1) – Между этими двумя командами будет происходить основной цикл игры. При нажатии кнопки ESC начинает читать следующую команду.

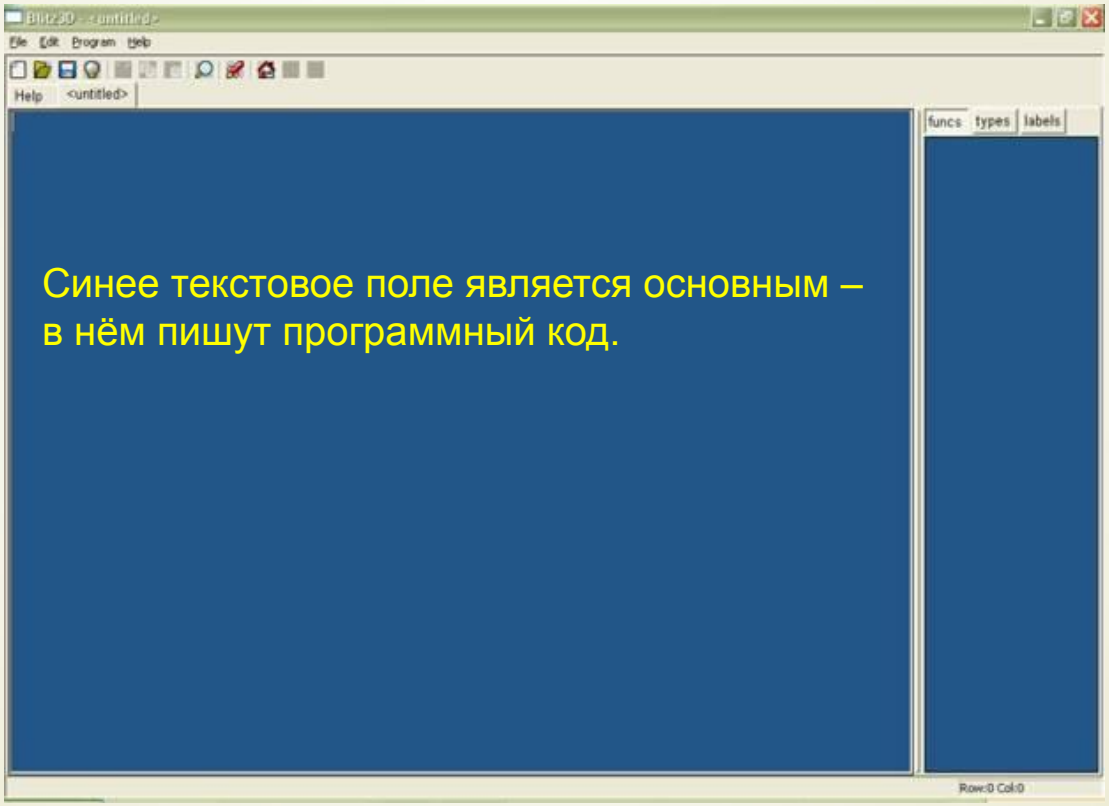
НОГИ

End – Завешает программу

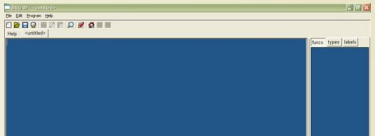
Создаём шаблон

Интерфейс программы.

Так выглядит окно Blitz3d:



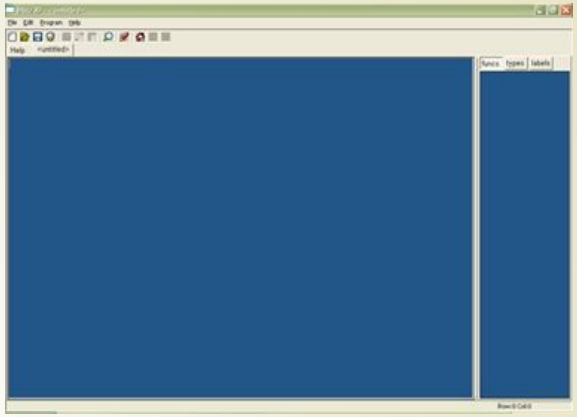
Синее текстовое поле является основным – в нём пишут программный код.



Кнопки панели инструментов:

- (New) – создание нового проекта
- (Open) – открытие существующего проекта
- (Save) – сохранение проекта
- (Close) – закрытие текущего проекта
- (Cut) – вырезать в буфер обмена
- (Copy) – копировать в буфер обмена
- (Paste) - вставить
- (Find) – найти текст
- (Run) – запуск проекта
- (Home) – перейти на страницу помощи
- (Back) – вернуться на предыдущую страницу в окне помощи
- (Forward) – вернуться на следующую страницу в окне помощи

Для закрытия\открытия панели инструментов: меню Edit -> ShowToolbars или Shift + Esc



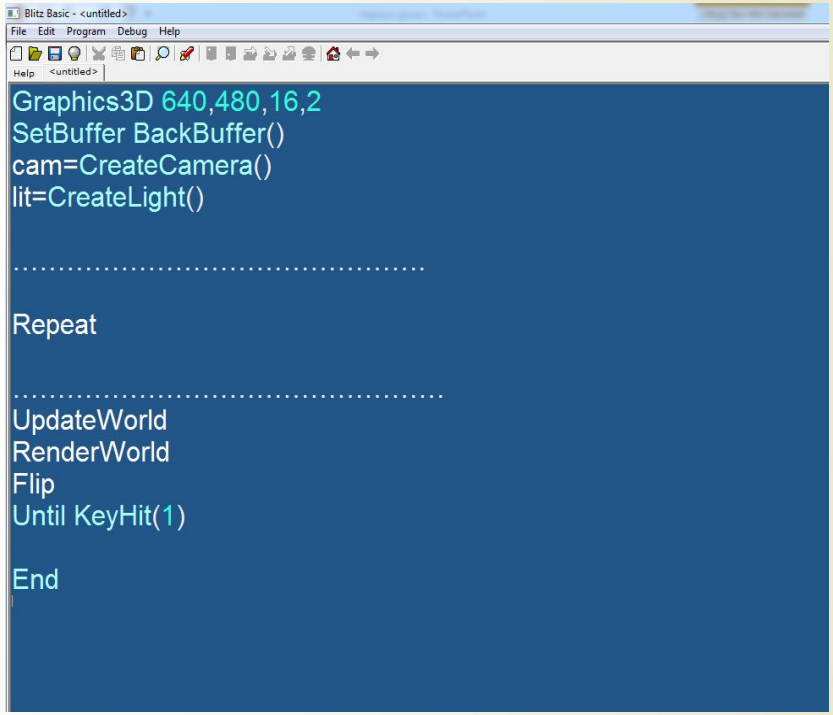
Запуск и отладка программы:

Для запуска программы нажмите F5 или

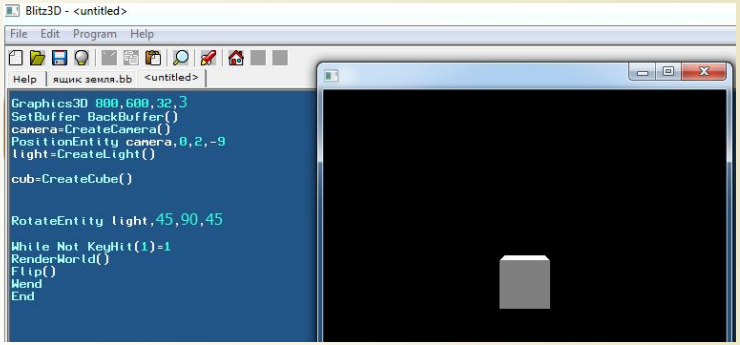
Для запуска предыдущей программы нажмите F6
Для проверки ошибок нажмите F7
Для создания .exe файла нажмите: меню Program -> Create Executable
Для включения\отключения отладочного окна: меню Program -> Debug Enabled

В правой части окна находится панель с тремя вкладками:
Funcs – отображает все функции, которые используются в проекте.
Types – отображает все типы, которые используются в проекте.
Label – отображает все метки.

```
Graphics3D 640,480,16,2
SetBuffer BackBuffer()
cam=CreateCamera()
lit=CreateLight()

……................................

Repeat

……..................................
UpdateWorld
RenderWorld
Flip
Until KeyHit(1)

End
```



Нарисуем КУБ




```
Graphics3D 800,600,32,2
SetBuffer BackBuffer()
camera=CreateCamera()

PositionEntity camera,0,2,-9

light=CreateLight()

cub=CreateCube()

RotateEntity light,45,90,45

While Not KeyHit(1)=1
RenderWorld()
Flip()
Wend
End
```

камера

Позиция камеры

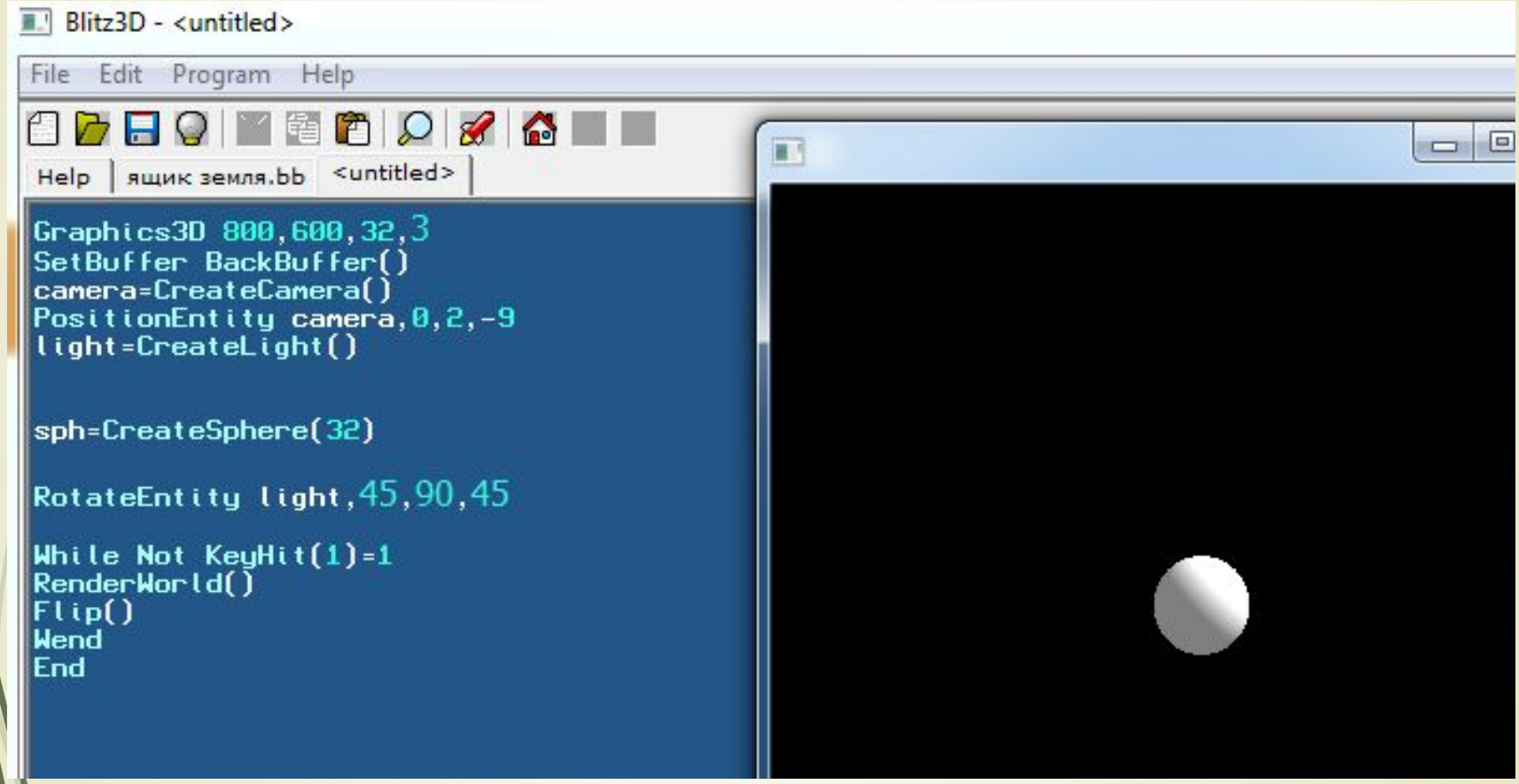
свет

Ставим куб

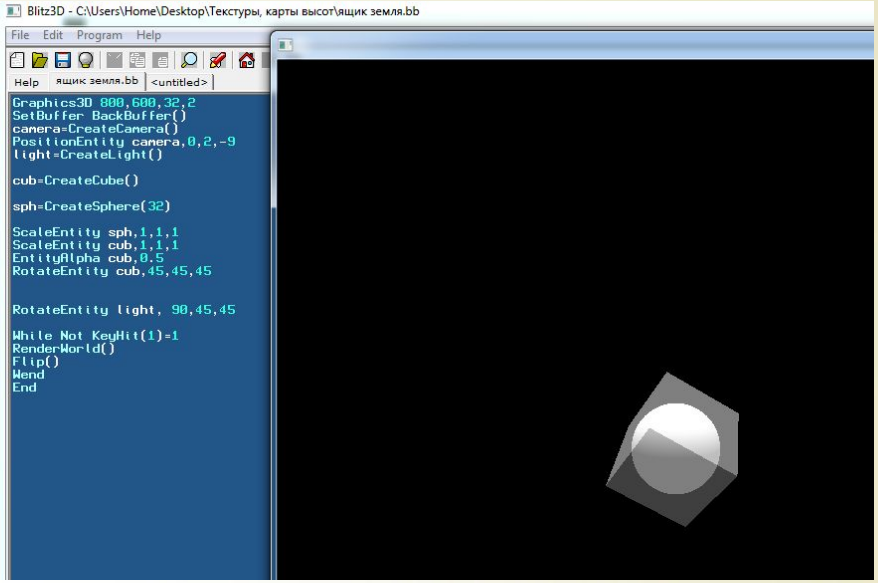
Поворот куба

Пока не нажата ESC

Поставим сферу.



```
Graphics3D 800,600,32,2
SetBuffer BackBuffer()
camera=CreateCamera()
PositionEntity camera,0,2,-9
light=CreateLight()

cub=CreateCube()

sph=CreateSphere(32)

ScaleEntity sph,1,1,1
ScaleEntity cub,1,1,1
EntityAlpha cub,0.5
RotateEntity cub,45,45,45

RotateEntity light, 90,45,45

While Not KeyHit(1)=1
RenderWorld()
Flip()
Wend
```

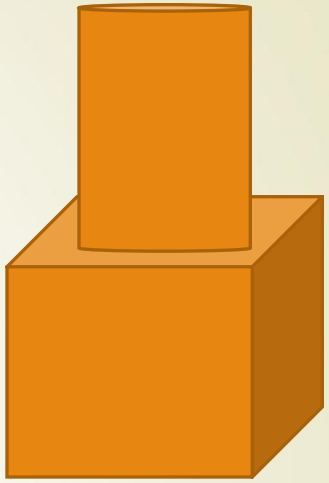


Размер объекта

Прозрачность объекта

ДОМА:

Поставить цилиндр на куб.



Используем:

CreateCylinder () - цилиндр

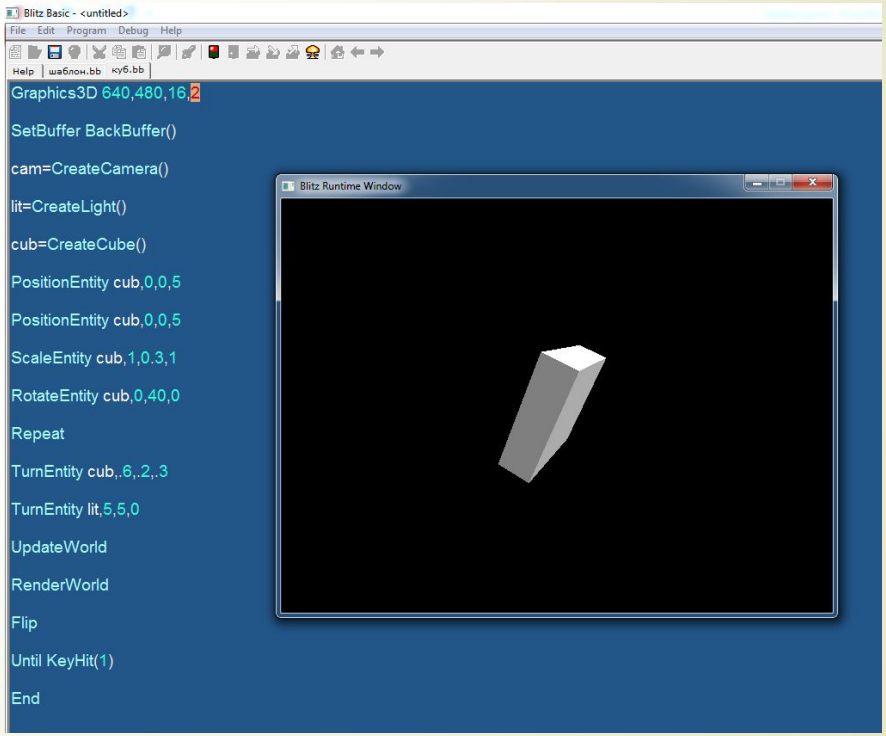
CreateCube() - куб

PositionEntity cub,0,0,5 – Эта команда помещает объект в позиции и в 3D месте.

ScaleEntity cub,1,0.3,1 – Изменяет размер объекта

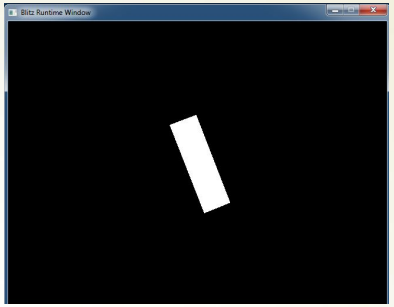
Поставим объект

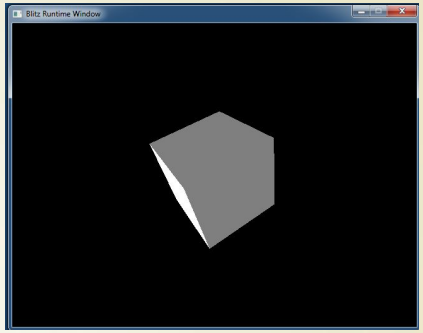
```
Graphics3D 640,480,16,1
SetBuffer BackBuffer()
cam=CreateCamera()
lit=CreateLight()
cub=CreateCube()
PositionEntity cub,0,0,5
ScaleEntity cub,1,0.3,1
RotateEntity cub,0,40,0
Repeat
TurnEntity cub,.6,.2,.3
TurnEntity lit,5,5,0
UpdateWorld
RenderWorld
Flip
Until KeyHit(1)
End
```

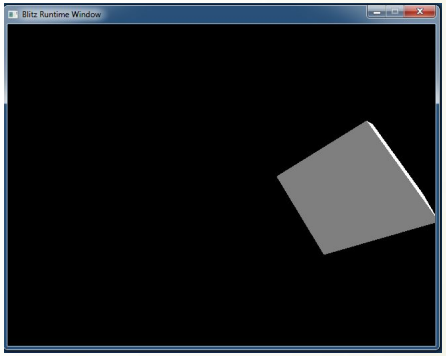


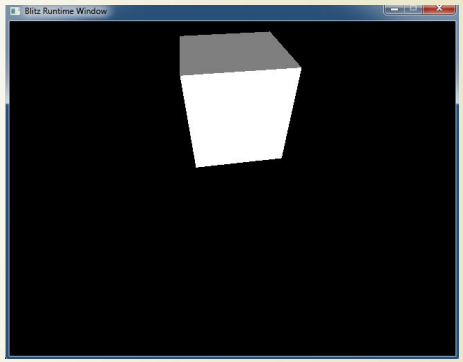
Практическая работа.

1. Изменить параметры света.
2. Изменить размер объекта.
3. Переместить объект вверх
4. Переместить объект вправо.
5. Изменить скорость вращения.

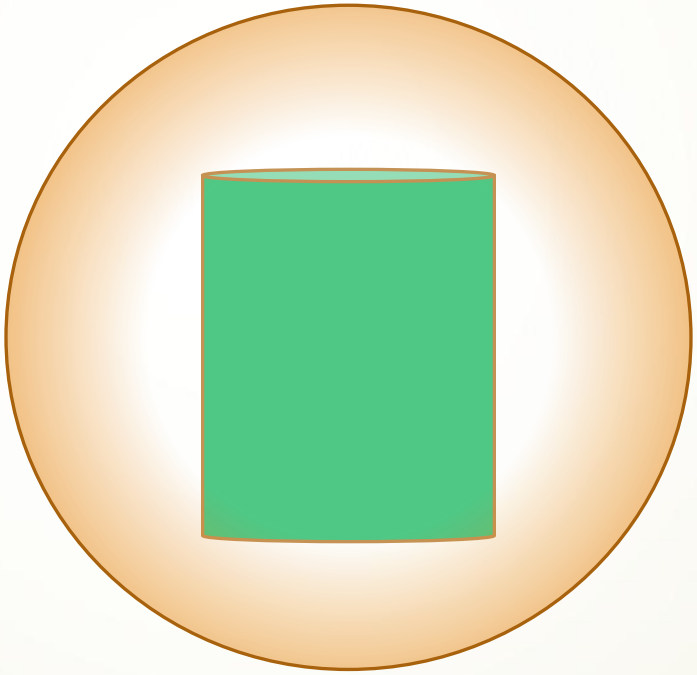








Дома: 1. Установить программу
2. Создать сферу и цилиндр.

Дома: 1. Установить программу

2. Создать полупрозрачную сферу и внутри цилиндр.