OpenGL. Занятие 2. Примитивы рисования

Как уже было сказано на предыдущей лекции, для рисования в фигур в OpenGL нужно поместить в трехмерное пространство некоторое количество вершин (Vertex) и указать, каким способом их можно соединить. В коде программы это оформляется следующим образом:

Сначала командой `glBegin(СПОСОБ_СОЕДИНЕНИЯ)` Вы указываете, что начиная с этого момента Вы будете перечислять вершины, которые нужно будет соединить между собой способ, заданным при помощи константы-параметра, указанного в скобках команды `glBegin`.

После этого Вы перечисляете вершины (командами `glVertex3d(x, y, z)`).

В завершении, командой `glEnd()` Вы указываете, что закончили перечисление вершин, соединяемых этим способом (обычно это заканчивает фигуру или часть фигуры).

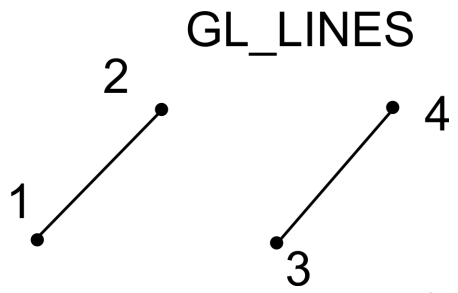
Перед любой командой `glVertex3d()` можно командой `glColor3f(r, g, b)` задать цвет вершин, которые будут перечислены далее. При этом аргументы `r`, `g`, `b` задают яркость, соответственно, красной, зеленой и синей компонент цвета в цветовой модели RGB и могут принимать значения от 0 до 1 каждая.

На предыдущем занятии мы уже рассматривали такой пример, рисуя четырехугольники. Напомню:

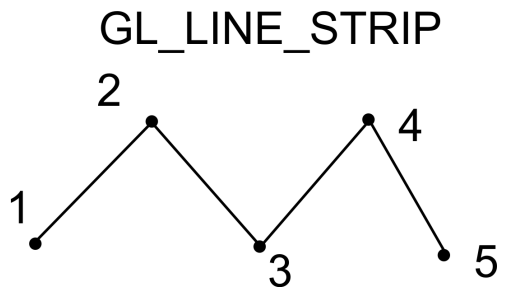
```
glBegin(GL_QUADS)
glColor3f(1, 0, 0)
glVertex3d( 0.5,  0.5, 0)
glVertex3d(-0.5,  0.5, 0)
glVertex3d(-0.5, -0.5, 0)
glVertex3d( 0.5, -0.5, 0)
glEnd()
```

Этот фрагмент программы рисует прямоугольник красного цвета по центру экрана. Если окно при этом будет квадратным, то мы увидим квадрат.

Рассмотрим подробно различные способы соединения вершин (можно считать это различными способами рисования фигур или *примитивами рисования*). На рисунках цифрами показаны номера вершин в порядке их перечисления между командами `glBegin()` и `glEnd()`.

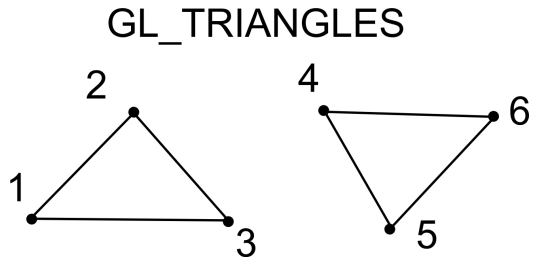


Это рисование прямых линий (правильнее, конечно, сказать — отрезков). Каждые две вершины соединяются прямой линией. Если указать нечетное количество вершин, последняя вершина игнорируется.

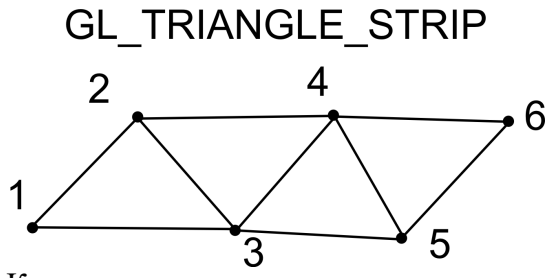


Так как часто требуется рисовать соединенные друг с другом фигуры, для убыстрения работы видеокарты (и уменьшения почти вдвое необходимости пересылать в видеокарту координаты вершин), для рисования ломаной линии используют отдельный примитив. То есть, вместо того, чтобы каждый отрезок ломаной передавать отдельно, парой вершин, можно использовать этот примитив.

Сначала строится прямая линия между первыми двумя вершинами. А потом строится прямая линия между последующей вершиной и предыдущей (то есть, между двумя соседними). Таким образом, количество прямых отрезков ломаной будет на 1 меньше количества указанных вершин.

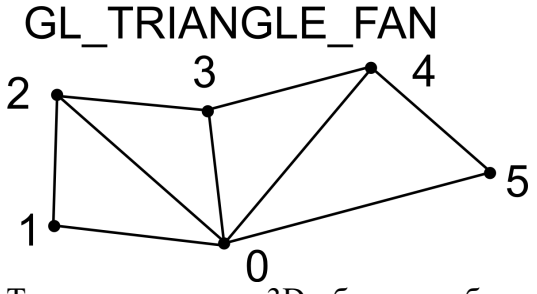


Рисование отдельных треугольников. По каждой тройке вершин строится отдельный треугольник. Если указать количество вершин, не кратное трем, последние ("лишние") вершины игнорируются.



Как и в случае с прямыми линиями, этот вид соединения вершин используется, когда нужно соединять соседние треугольники и хочется сэкономить на быстродействии за счет сокращения указания "двойных" координат вершин.

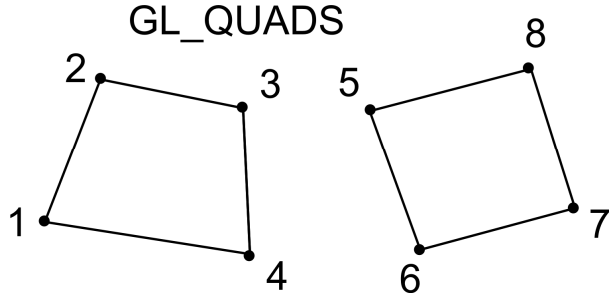
По первым трем вершинам строится треугольник. А потом берется две последние (по поярдку перечисления) вершины и к ним добавляется еще одна (следующая). ПО этим трем вершинам также строится треугольник. И так далее, для всех оставшихся вершин. Таким образом, количество треугольников будет на 2 меньше количества указанных вершин. И результирующая фигура будет несколько похожа на ленту (STRIP).



Так как рисование 3D-объектов обычно осуществляется при помощи треугольников, для треугольников имеется еще один способ соединения вершин. Так как в этом способа первая указанная вершина имеет отдельное, очень важное значение, я специально указал ее номер не 1, а 0.

По первым трем вершинам (с номерами 0, 1 и 2), как и ранее, строится треугольник. А последующие треугольники строятся так: начальная вершина (номер 0), последняя вершина только что нарисованного треугольника, и следующая вершина. То есть, треугольников получается столько же, сколько и в `GL_TRIANGLE_STRIP`, но порядок соединения вершин совсем другой. В результате получается что-то похожее на веер (FAN).

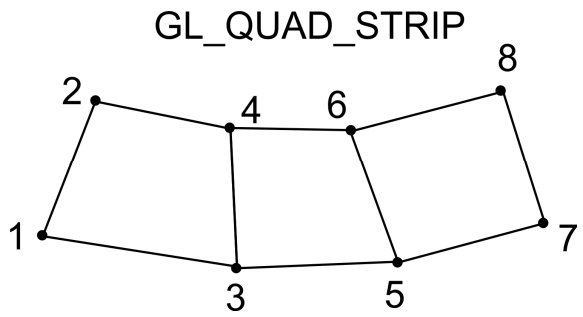
Заметьте, это практически единственный из примитивов рисования, который требует закрывания (`glEnd()`) и нового открывания (`glBegin()`) при рисовании нового объекта, построенного таким же образом.



Этот способ соединения вершин мы уже использовали. По каждой четверке вершин строится отдельный четырехугольник. Если указать количество вершин, не кратное четырем, то "лишние" вершины игнорируются. Важно помнить два условия, которые должен соблюдать программист:

1. Все 4 вершины каждого четырехугольника должны лежать в одной плоскости.

2. Порядок обхода вершин должен быть по часовой стрелке или против часовой стрелки (а не буквой Z).



Вероятно, Вы и сами по названию понимаете, что этот режим служит для соединения четырехугольников "в ленту", для убыстрения рисования. По первым четырем вершинам строится четырехугольник. А потом берется последние две его вершины и следующие две вершины. И по ним тоже строится четырехугольник. И так для каждых следующих двух. Если указать нечетное количество вершин, последняя игнорируется.

Несмотря на то, что для этого способа соединения вершин действует те же два правила, что и для `GL_QUADS`, на практике можно обычно указывать вершины так, как показано на рисунке (нечетные с одной сторону, четные — с другой) и "лента" нарисуется правильно.

В OpenGL имеется еще три примитива рисования: `GL_POINTS`, `GL_LINE_LOOP`, `GL_POLYGON`. Если интересно, можете изучить их самостоятельно.

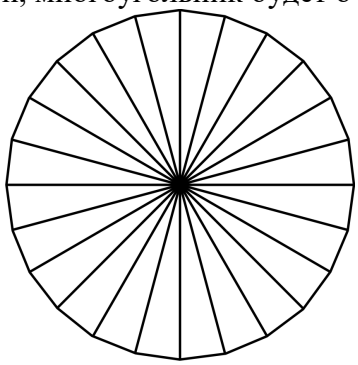
Рисование круга из треугольников

Рассмотрим применение указанных примитивов для рисования более сложных фигур.

Начнем с рисования круга.

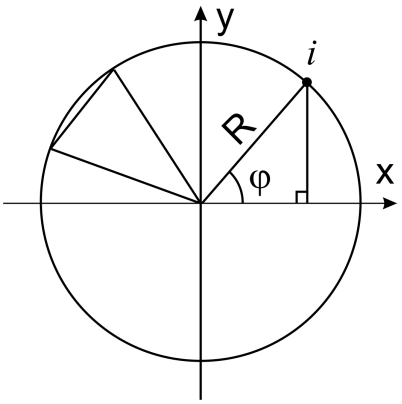
В отличие от многих других графических библиотек, рисование круга не является графическим примитивом OpenGL. Его нужно рисовать, используя базовые примитивы.

Мы нарисуем круг как закрашенный многоугольник. Если взять достаточно большое количество сторон, многоугольник будет очень похож на круг.



Вероятно, Вы заметили, что получаемая фигура будет очень похожа на "веер". То есть, будет рисовать ее при помощи `GL_TRIANGLE_FAN`. С указанием местоположения первой вершины проблем нет. Расположим ее, например (для простоты) в точке (0, 0). Вернее сказать, в точке (0, 0, 0). Но так как мы собираемся нарисовать пока что плоскую фигуры (круг), будем считать, что она находится в плоскости "экрана" и что все ее Z-координаты равны нулю. А вот как указать, в каких координатах находятся остальные вершины?

Воспользуемся тем фактом, что все остальные вершины многоугольника/круга находятся на одинаковом расстоянии от точки (0, 0). Будем в цикле перебираться все эти вершины. Наша ближайшая задача — понять, как, зная номер вершины в цикле получить ее "угол отклонения" от оси OX (то есть, ее полярный угол в полярных координатах).



То есть, будем в цикле менять номер вершины от 0 до (например), 20-ти. Тогда можно будет составить пропорцию:

Номеру вершины *i* поставим в соответствие угол φ.

А номеру вершины 20 (полный круг) поставим в соответствие угол 2π (полный круг).

Получаем пропорцию:

i — φ

20 — 2π

Отсюда выразим угол φ: $\varphi = \frac{2\pi i}{20}$.

Декартовы координаты вершин найдем "из прямоугольного треугольника":

$x = R \cos \varphi$

$y = R \sin \varphi$

Подставив в эти выражения угол φ получим требуемые формулы для вычисления координат вершин треугольников.

Запишем все вместе в виде программы на Питоне:

```
glBegin(GL_TRIANGLE_FAN)
glColor3f(1, 0, 0)
glVertex3d(0, 0, 0)
for i in range(21):
    glVertex3d(R * cos(2*pi*i/20), R * sin(2*pi*i/20), 0)
glEnd()
```

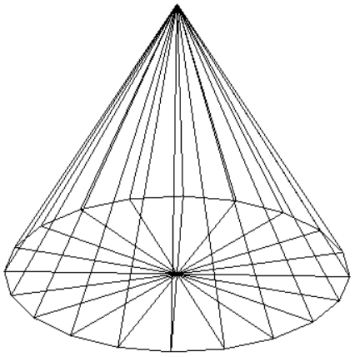
В цикле используем `range(21)`, чтобы последние значение счетчика цикла было бы 20. Если использовать вместо этого число 20, вместо круга получится "PacMan" — последний треугольник не будет нарисован (вершина (R, 0) должна быть указана и как одна из вершин первого треугольника, и как последняя вершина последнего треугольника).

Задания

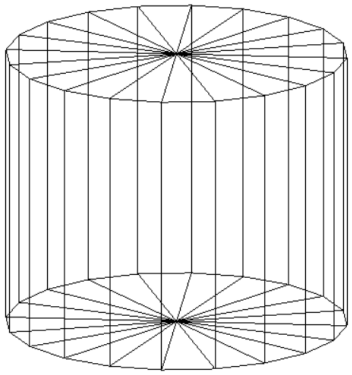
Используя полученные знания, нарисуйте фигуры, указанные ниже.

Все фигуры рисовать с центром в начале координат и размером 1. Рекомендую рисовать каждый отдельный элемент фигуры своим цветом. Например, у конуса — два элемента (основание и боковая поверхность), а у цилиндра — три (два основания и боковая поверхность).

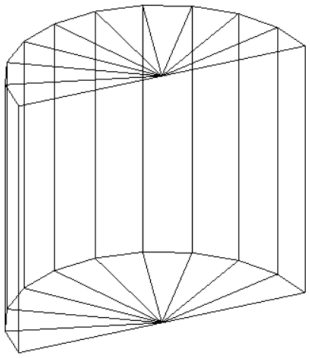
1) Конус



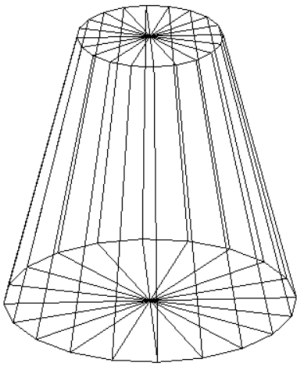
2) Цилиндр



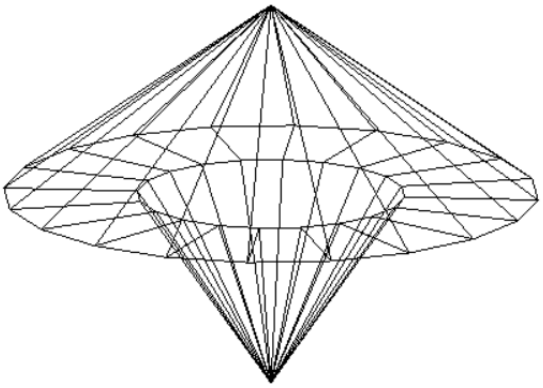
3) Полуцилиндр



4) Усеченный конус



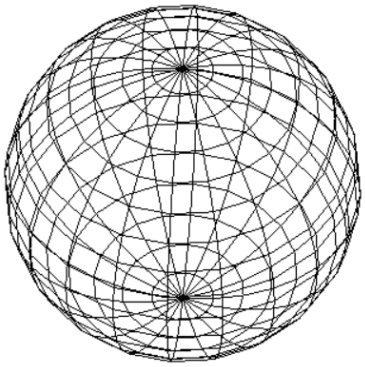
5) "Волчок"



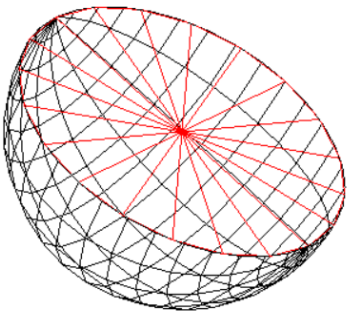
Подсказки:

1. Конус. Нарисуйте круг и еще один круг, но "поднимите" его начальную точку. Обратите внимание на то, что нужно нарисовать конус, а не "кулек". То есть, у конуса должна быть крышка (основание).
2. Цилиндр. Основания — два круга. Боковую поверхность рекомендую рисовать через QUAD_STRIP.
3. Полуцилиндр — полцилиндра. Все циклы сократите вдвое. Не забудьте нарисовать прямую боковую поверхность.
4. Усеченный конус — тот же цилиндр, но с разными радиусами оснований.
5. "Волчок" — два "кулька" и что-то вроде "боковой поверхности усеченного конуса".

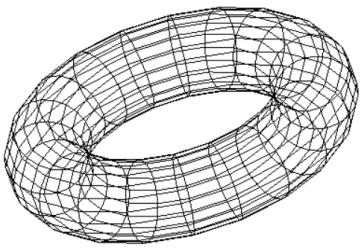
6) Сфера



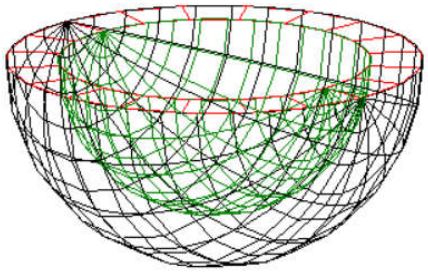
7) Полусфера



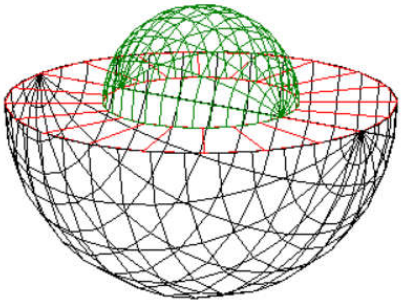
8) Тор



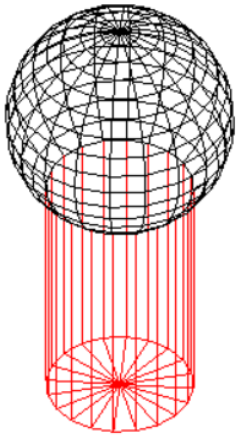
9) "Чаша"



10) "Мороженное"



11) "Жезл"



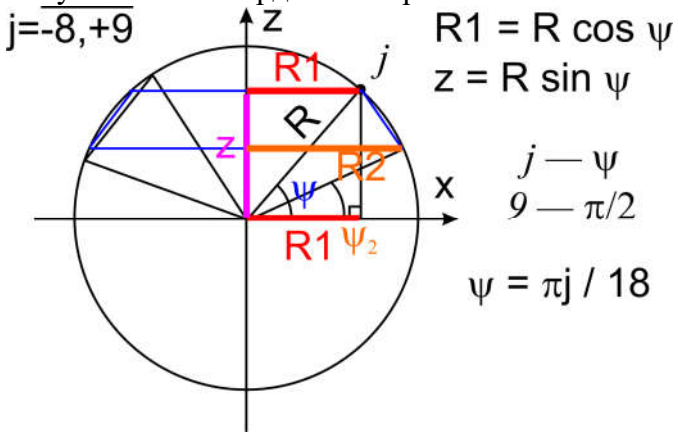
Для рисования этих фигур нужно использовать уже два вложенных цикла.

Во всех этих фигурах внутри фигуры никаких линий быть не должно. То есть, фигура должна быть задана только внешними поверхностями.

Сфера строится таким способом — "установкой друг на друга" боковых поверхностей усеченного конуса. Это хорошо бы себе представлять как глобус с его параллелями и меридианами.

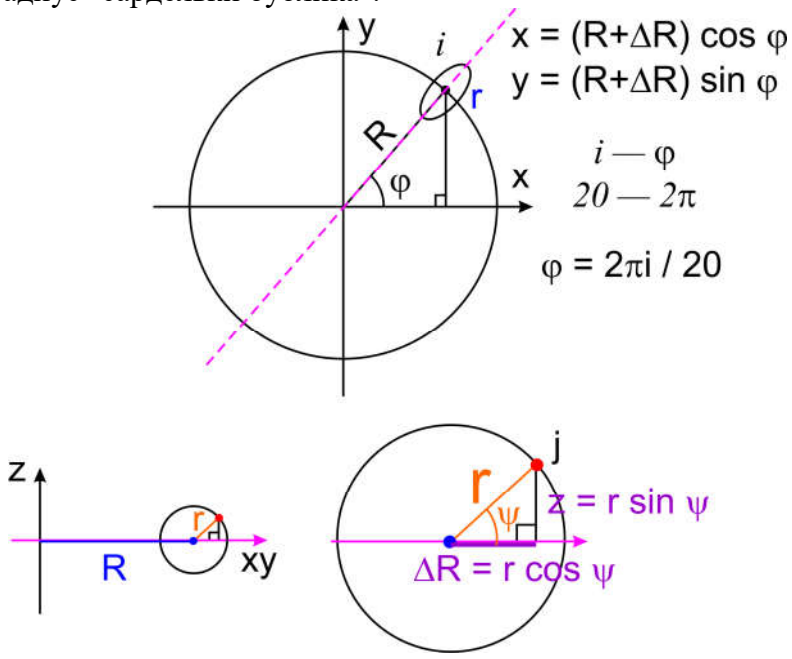
Нужно использовать примерно те же идеи, что и при рисовании круга, но еще учесть, что во внешнем цикле для каждого "усеченного конуса" нужно вычислить радиусы оснований и местоположение вдоль оси Z.

Вот рисунок-подсказка для рисования сферы. Это проекция на плоскость XZ (сбоку). Синим обозначена рисуемая на шаге цикла номер *j* боковая поверхность конуса. R1 и R2 — радиусы оснований этого конуса. Z — z-координата вершин.



Рекомендую использовать внешний цикл от –8 до 9 (или от –9 до 8), каждый шаг цикла задает одно "кольцо-параллель" глобуса с "высотой" 10 градусов. Угол ψ — угол наклона к оси OX. Внутренний цикл, как и ранее, будет перебирать номера вершин в плоскости XY (и для них будет та же зависимость угла φ от счетчика цикла *i*). Только на каждом шаге внутреннего цикла нужно будет задать не одну, а две вершины — для разных значений R1 и R2 и с разными значениями Z.

Рисование тора происходит по похожим принципам. Считаем, что тор рисуется радиусами *R* и *r*.
R — радиус окружности, проходящей через центр "сардельки бублика".
r — радиус "сардельки бублика".



Верхний рисунок показывает проекцию на плоскость XY, в которой лежит "плоскость бублика" (если бублик разрезать напополам "вдоль", то разрез будет находиться в плоскости XY).

Нижний рисунок показывает проекцию бублика в плоскости "ZR". То есть, вертикально, и сквозь ту точку на бублике, на которой располагаются вершины одного из оснований "скособоченного цилиндра" (из которых строится бублик).

Внешний цикл перебирает номер "скособоченного цилиндра", основания которого "смотрят" на "ось тора". По этому номеру вычисляем угол φ. По нему вычисляем положение центра этого "скособоченного цилиндра" на воображаемой окружности, проходящей по центру "сардельки бублика".

Во внутреннем цикле перебираются точки на маленькой окружности, составляющей основание "скособоченного цилиндра".