**Лабораторная работа №2.**

**«Полигональные сетки, работа с векторами».**

1. Задание

Создать программу, рисующую сцену средствами OpenGL или DirectX.

Сцена должна содержать "карту высот", небольшую плоскую горизонтальную площадку, экструзивную поверхность, поверхность вращения, геодезический купол, несколько кубов или циллиндров, любые полупрозрачные объекты, спрайты, источники света.

Для всех объектов сцены должны быть заданы нормали и текстурные координаты, если это возможно.

Объекты должны быть освещены и покрыты текстурой.

Наблюдатель имеет возможность свободно перемещаться по сцене.

Вывод производится плавно, без иcпользования 100% CPU.

Наблюдатель имеет возможность свободно перемещаться по сцене.

Имеется возможность включить (и выключить) проверку столкновений с объектами сцены.

Дополнительные задания

Учесть инерцию, гравитацию и трение для наблюдателя.

Реализовать скольжение вдоль стен и вертикальных объектов для наблюдателя.

1. Теория

OpenGL (Open Graphics Library — открытая графическая библиотека) — спецификация, определяющая независимый от языка программирования кросс-платформенный программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

OpenGL -- это конечный автомат, который можно переводить в различные состояния. У различных состояний существуют свои параметры. Например, текущий цвет является параметром состояния. Его можно изменять и после каждого изменения все объекты будут отображаться именно этим, измененным цветом. Другие параметры состояния отвечают за текущее модельно-видовое преобразование, тип линий, многоугольников, методы отображения многоугольников, режим упаковки пикселов и т.д. Многие параметры состояния можно включать и выключать при помощи glEnable/glDisable. Каждый параметр состояния имеет значение по умолчанию и его в любой момент можно запросить у системы.

В общем смысле все, что делает OpenGl можно представить в виде следующего конвейера:



Векторные данные (вершины, линии, многоугольники) подвергаются конвейерной обработке рядом блоков, включая блоки оценки и операции с вершинами, а растровые данные (пиксели, изображения и битовые образы) обрабатываются отдельно.

1. Алгоритм

*Основная идея*: Создание движущихся корабликов по волнам с переливающимся бликом от солнца.

Для создания окна и реакции на нажатие клавиш используется следующий набор инструкций.

def main():

glutInit()

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA|GLUT\_DEPTH)

glutInitWindowSize(800,600)

glutCreateWindow('sea')

glutDisplayFunc(display)

glutKeyboardFunc(kbd\_down)

glutKeyboardUpFunc(kbd\_up)

glutPassiveMotionFunc(mouse)

while(looping):

glutMainLoopEvent()

glutPostRedisplay()

Реализованы следующие примитивы:

1. "карта высот" (волны)

Стандартными средствами OpenGl устанавливаем отраженный цвет материала (голубой) и коэффициент отражения (максимальный == 128).

C помощью функции drawSurface() была создана карта 40х40 (BORDERxBORDER) с изменением координаты z по закону FSurf(x,y) = sin(0.3\*x)+sin(0.7\*y)\*sin(0.5\*y). С помощью glCallList рисуем только видимую часть видимую часть поверхности. Отрисовка происходит с помощью массива вершин. Благодаря массиву данных увеличивается производительность. т.к. число вызовов становится минимальным (по сравнению с методами glBegin()/glEnd()).

Фунции NormSurface() и TextureSurface() по сформированным точкам поверхности высчитывают и записывают в массив координаты нормалей и текстурные координаты.

glVertexPointer(), glNormalPointer(), glTexCoordPointer() -- определяются пространственные координаты, координаты нормали и текстурные координаты. glDrawArrays(GL\_QUADS,0,4\*BORDER\*BORDER) -- построение примитива с помощью GL\_QUADS.

1. плоская площадка (берег)

Берег рисуется при помощи функции drawCoast(), в которой устанавливается текстура. Отраженный цвет материала устанавливаем в коричневый.

Затем, рисуем видимую часть куба с помощью функции drawCube(), а далее с помощью модельно-видовых преобразований (поворота и растяжения) переделываем куб в прямоугольник.

1. поверхность вращения (икосаэдр -- генератор кораблей, вращающийся шар -- солнце)

В программе икосаэдр играет роль генератора кораблей и отрисовывается с помощью функции drawGenerator(). С помощью библиотеки GLUT, входящей в состав OpenGl рисуем икосаэдр: glutSolidIcosahedron() и отменяем наложение текстуры. Наш икосаэдр вращается благодаря методу glRotatef(alphaIco,0,0,1), где alphaIco -- угол вращения.

Солнце -- шар, который вращается вокруг моря. При вращении изменяется блик на волнах.

Отрисовка солнца происходит в функции drawSun(). Шар рисуется методом из библиотеки GLU: gluSphere(quadratic,0.5,16,16). На шар накладывается текстура методом квадратирования gluNewQuadric(). Отраженный цвет материала -- желтый с максимальным коэффициентом отражения. От солнца создаем еще один источник света GL\_LIGHT1, который отвечает за желтые блики. Благодаря методам glPushMatrix()/glPopMatrix() новый источник света движется вместе с солнцем.

1. плоский диск (НЛО, захватывающее корабли)

Реализуется с помощью метода drawNLO(). Плоский диск рисуется методом gluDisk(quadratic,1.0,3.0,32,32) из стандартной библиотеки GLU, с квадратированием для наложения текстуры на объект GLU. НЛО вращается вокруг своей оси и перемещается в зависимости от глобальной переменной stateBorder. stateBorder отвечает за достижения кораблей-спрайтов до берега. Если корабль дошел до берега, то параметр stateBorder == True.

1. спрайты (облака, деревья, 3 типа кораблей (различаются по характеру движения -- sin, cos, tg, и по текстурам))

Реализуются классом Sprite с методами draw и move. draw() отвечает за установку спрайта к наблюдателю. Метод move() отвечает за движение объектов-корабликов по волнам. Если они сталкиваются с аркой, то координата движения смещается на одну из функций F: sin, cos, tg. Если корабль дошел до берега, то он автоматически удаляется из списка спрайтов.

Метод createSprites() добавляет в глобальный список всех спрайтов по очереди: сначала облака(20 шт) в небе, затем деревья(20 шт) на берегу, потом корабли (30шт) 3х типов: гоночные (двигаются по tg), грузовые (двигаются по sin) и пассажирские (по cos). Создание всех спрайтов происходит в самый первый раз и всякий раз, когда все корабли достигают берега.

1. источники света (общий фоновый источник, доп источник от солнца, создающий блики на воде)

Общий фоновый свет устанавливается функцией setupLight(), в которой устанавливаем общий диффузный и фоновый цвет -- голубым.

Источник от солнца устанавливается функцией sunLight() и отвечает за блики на воде, двигается вместе с солнцем.

1. покрытие текстур (солнце, берег)

Сначала загружается само изображение и возвращается его ID функцией loadImage(). setTexture() связывает название текстуры с его ID из функции loadImage().

1. экструзивная поверхность (арка, которые должны огибать кораблики)

Экструзия была создана путем выдавливания двумерной формы в пространстве. Арка была построена путем разбития невыпуклого полигона на совокупность выпуклых. Затем каждый выпуклый полигон был экструдирован. Точки были составлены "от руки". Во время составления точек нужно аккуратно следить за последовательностью перечисления, иначе стандартные средства OpenGl не смогут определить заднюю поверхность и нормали к лицевой поверхности.

Арка была построена функцией drawArch(). Экструзия заключалась в изменении координаты х на величину step = 0.4. Вывод объекта производился при помощи массива вершин.

Присутствуют:

1) свободное перемещение по сцене

2) отрисовка только видимой части у "карты высот" и куба (берега)

3) проверка столкновений с объектами сцены включена: столкновение с аркой, икосаэдром, берегом и границей сцены функциями checkArch(sx,sy), checkIco(sx,sy), checkSurface(cx,cy,cz) и checkBorder(sx,sy,sz) соответственно.

Дополнительно:

1) Гравитация возвращает высоту на z = 3.0

2) Инерция двигает нас ближе к координатам (0,0) при столкновении с аркой

3) Кораблики генерируются рандомно, автоматически

4) Скольжение вдоль границы карты присутствует

1. Выводы

OpenGL (Open Graphics Library) — открытая графическая библиотека OpenGL.

Графическая библиотека OpenGL, как утвержденный индустриальный стандарт, разработана и утверждена в 1992 году девятью ведущими IT-фирмами: Digital Equipment Corp., Evans & Sutherland, Hewlett-Packard Co., IBM Corp., Intel Corp., Intergraph Corp., Silicon Graphics Corp, Inc., Sun Microsystems, Inc., Microsoft Corp. В основе стандарта лежит библиотека IRIS GL, разработанная Silicon Graphics. OpenGL широко используется при разработке компьютерных игр. У OpenGL много достоинств:

Стабильность: OpenGL устоявшийся стандарт. Все изменения, вносимые в него, анонсируются и реализуются так, чтобы уже существующее ПО не сбоило на новых графических картах.

Надежность: Все приложения, использующие OpenGL, гарантируют одинаковый визуальный результат, независимо от оборудования и операционной системы.

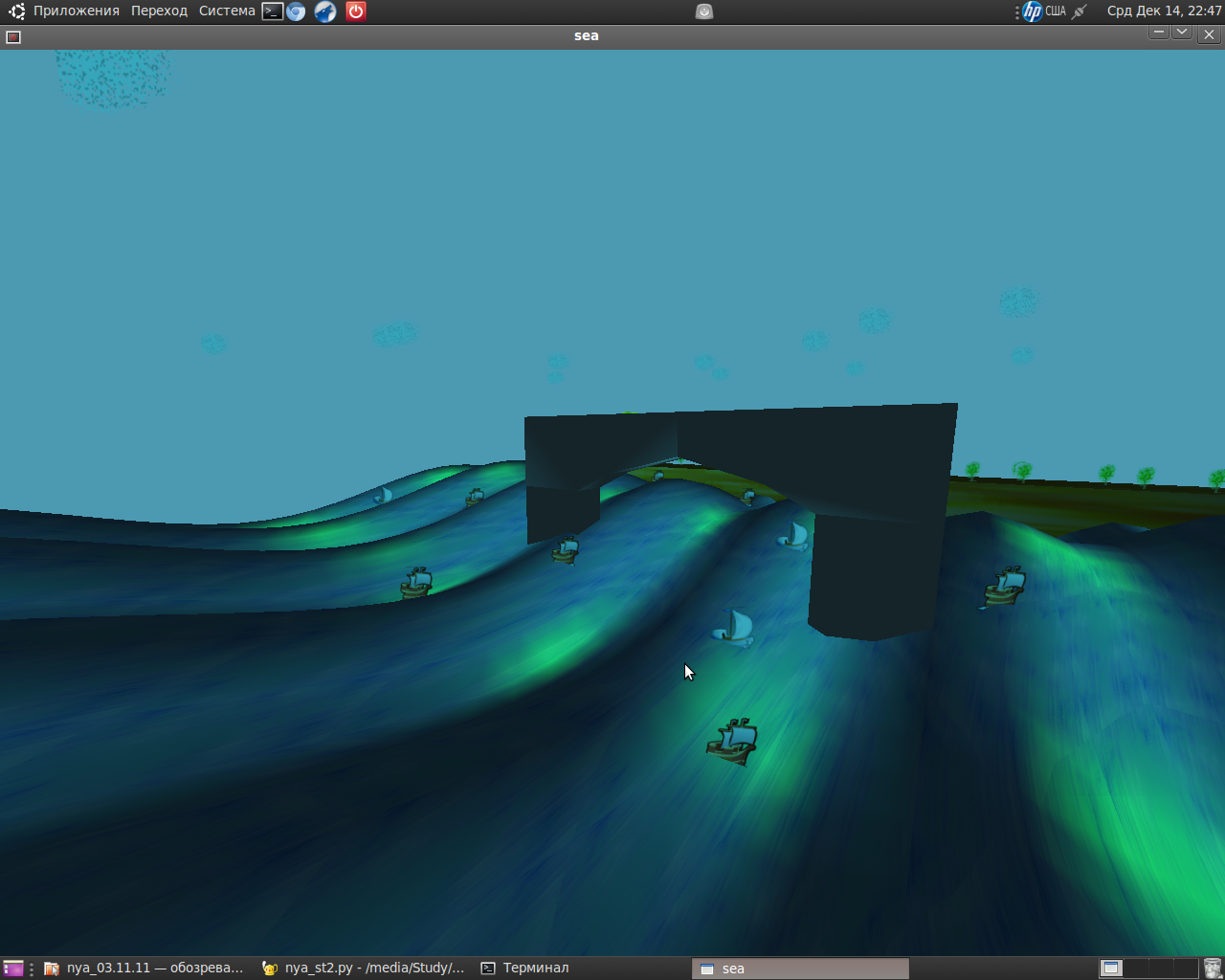
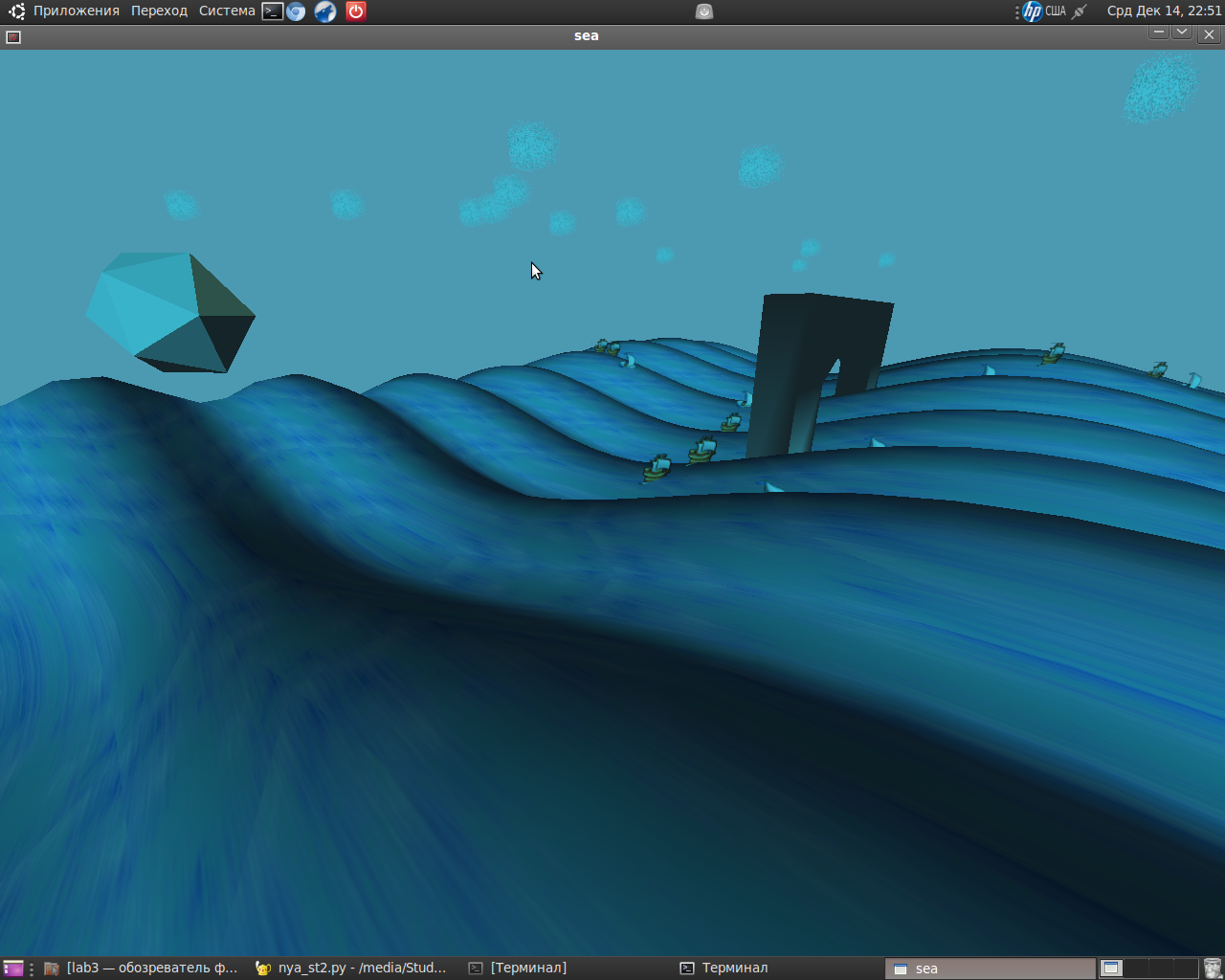
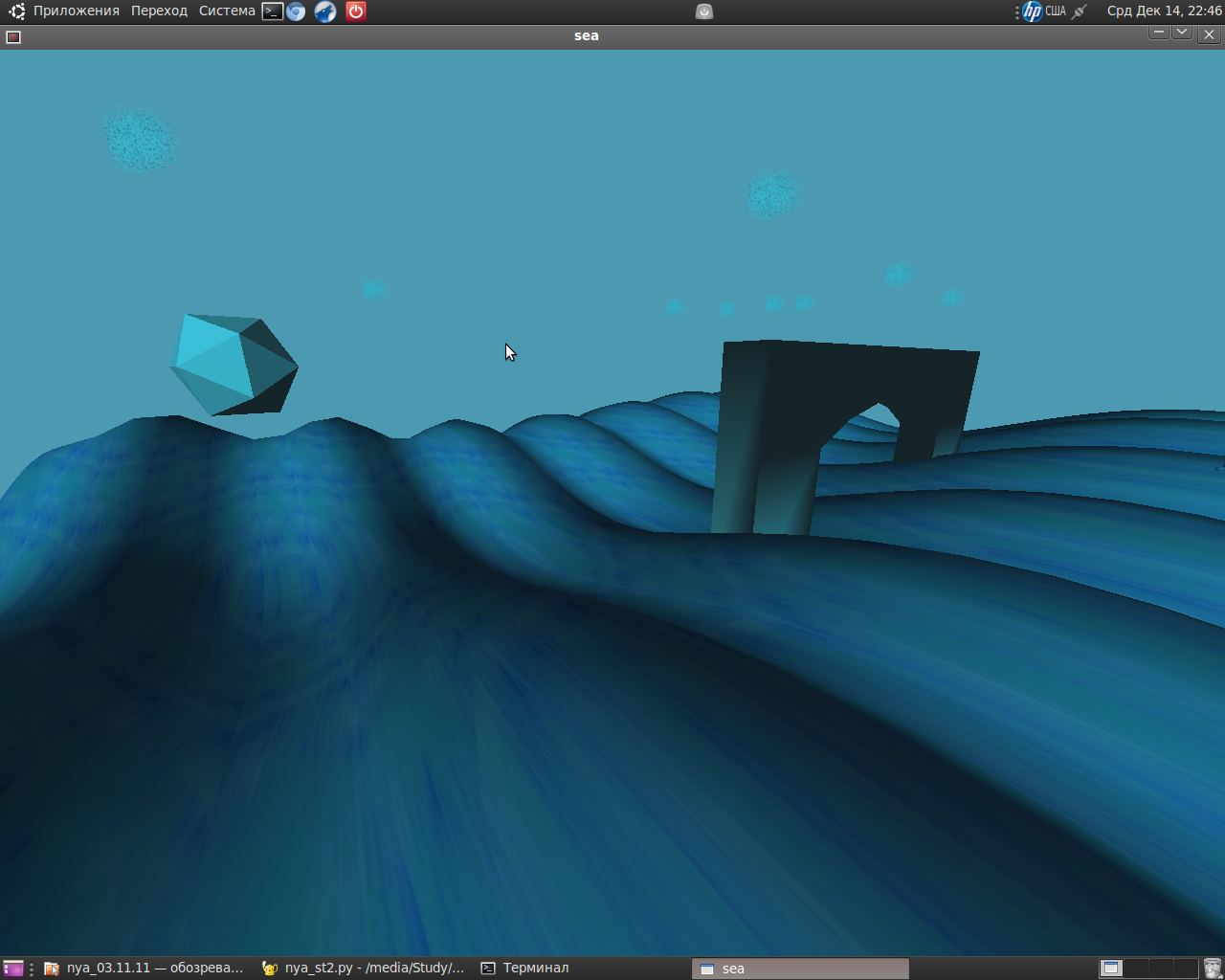
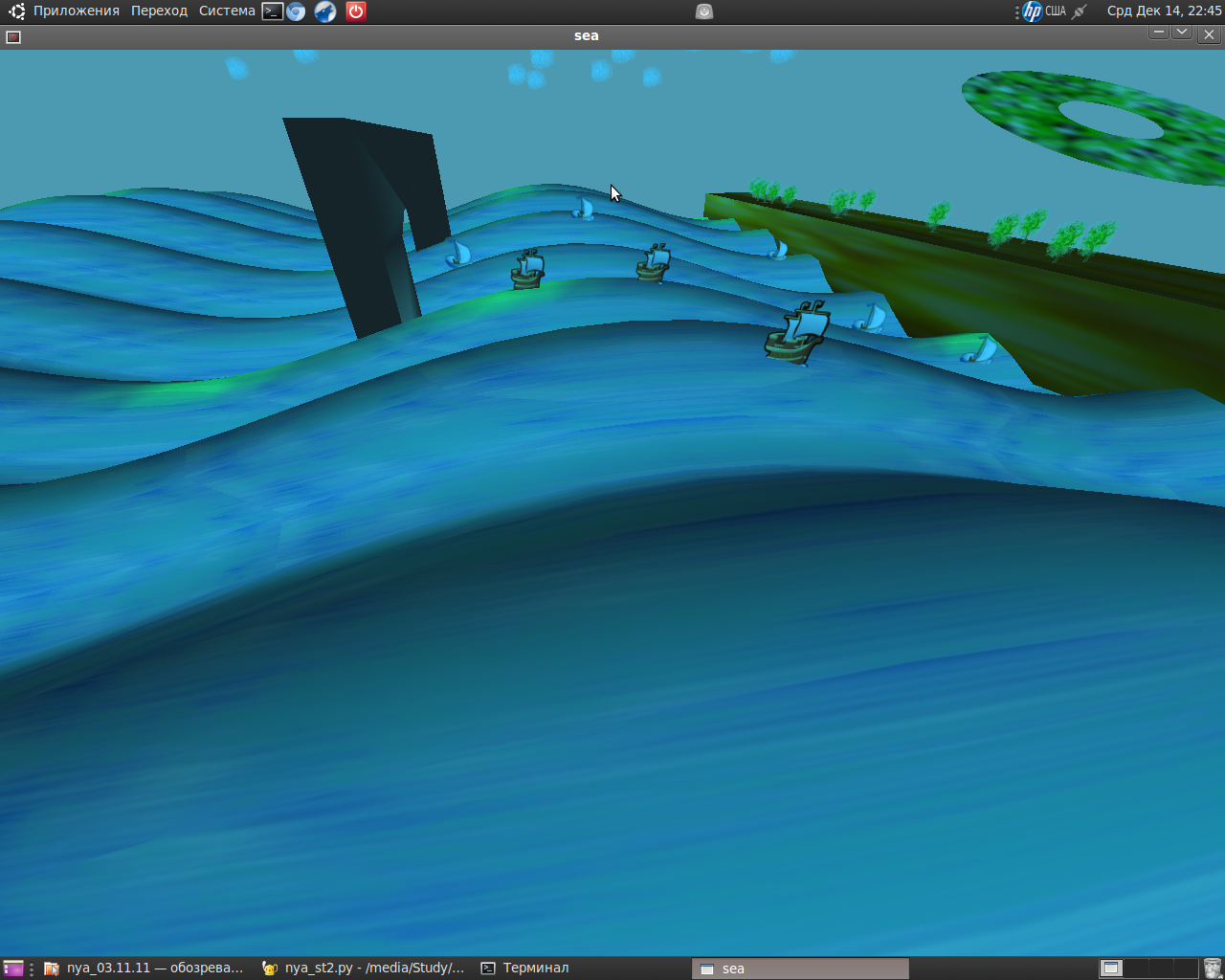
Переносимость: Приложения, использующие OpenGL, могут запускаться на различных архитектурах и под различными операционными системами (естественно, при условии перекомпиляции приложения, то есть OpenGL обеспечивает переносимость на уровне исходных кодов).

Главная особенность OpenGL — его клиент-серверная архитектура, что позволяет, теоретически, поместить клиент (приложение, использующие OpenGL) и сервер (исполнительная часть OpenGL) на разные машины.

OpenGL развивается с помощью механизма «расширений» — специальных модификаций базовой версии API OpenGL, которые добавляют новые возможности и/или расширяют старые. Когда накапливается множество расширений, консорциум OpenGL выпускает спецификацию новой версии OpenGL.

1. Литература
2. А.В. Боресков Графика трехмерной компьютерной игры на основе OpenGL
3. Ф.Хилл OpenGL. Программирование компьютерной графики.
4. OpenGL Programming Guide - The Redbook
5. NeHe

**Screen:**

****