Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»

Кафедра общих и естественно-научных дисциплин ИСТР
Кафедра автоматизированных систем управления

“УТВЕРЖДАЮ”
ДИРЕКТОР ИСТР
д.соц.н. Осьмук Л. А.
“___ ”_____ _____ г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Графические системы в сфере социальной реабилитации

Образовательная программа: 09.03.01 Информатика и вычислительная техника, профиль: Автоматизированные системы обработки информации и управления в социальной сфере

Курс: 2, семестр : 3

Институт социальных технологий и реабилитации,

		Семестр
№	**Вид деятельности**	**3**
1	Всего зачетных единиц (кредитов)	3
2	Всего часов	108
3	Всего занятий в контактной форме, час.	63
4	Лекции, час.	18
5	Практические занятия, час.	0
6	Лабораторные занятия, час.	36
7	из них в активной и интерактивной форме, час.	8
8	Аттестация, час.	2
9	Консультации, час.	7
10	Самостоятельная работа, час.	45
11	Виды самостоятельной работы (курсовой проект, курсовая работа, РГЗ, подготовка к контрольной работе)	РГЗ
12	Вид аттестации	ДЗ

Новосибирск 2017

Рабочая программа составлена на основании федерального государственного образовательного стандарта высшего образования по направлению (специальности): 09.03.01 Информатика и вычислительная техника

ФГОС введен в действие приказом №5 от 12.01.2016 г. , дата утверждения: 09.02.2016 г.

Место дисциплины в структуре учебного плана: Блок 1, вариативная

Рабочая программа разработана на основе компетентностной модели выпускника по направлению (специальности): 09.03.01 Информатика и вычислительная техника

Рабочая программа обсуждена на заседании кафедры

ЕНД ИСТР, протокол заседания кафедры №_____ от 20.06.2017
АСУ, протокол заседания кафедры №7 от 20.06.2017

Утверждена на совете института социальных технологий и реабилитации, протокол № 6 от 21.06.2017

Программу разработал:

преподаватель высшей квалификационной категории, Бертик А. А.

Заведующий кафедрой:

доцент, к.п.н. Рощенко О. Е.
профессор, д.т.н. Гриф М. Г.

Ответственный за образовательную программу:

заведующий кафедрой Гриф М. Г.

1. Внешние требования

Таблица 1.1

***Компетенция ФГОС*: ОПК.1 способность инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем; *в части следующих результатов обучения*:**
з1. знать методы и средства компьютерной графики и геометрического моделирования
***Компетенция ФГОС*: ОПК.2 способность осваивать методики использования программных средств для решения практических задач; *в части следующих результатов обучения*:**
з7. знать универсальность математических методов в познании окружающего мира
у8. уметь применять основные методы математического аппарата в математических моделях объектов и процессов
***Компетенция НГТУ*: ПК.9.В/ПК готовность к разработке моделей компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина"; *в части следующих результатов обучения*:**
з6. знать особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья
у5. уметь использовать современные информационные технологии и инструментальные средства для создания интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья

2. Требования НГТУ к результатам освоения дисциплины

Таблица 2.1

Результаты изучения дисциплины по уровням освоения (иметь представление, знать, уметь, владеть)	**Формы организации занятий**
Графические системы в сфере социальной реабилитации	
ОПК.2.з7 знать универсальность математических методов в познании окружающего мира	
1.о теоретических основах построения современных графических пакетов	Лекции; Лабораторные работы; Самостоятельная работа
2.о особенностях реализации современных графических пакетов;	Лекции; Лабораторные работы; Самостоятельная работа
3.о средствах доступа к услугам современных графических пакетов	Лекции; Лабораторные работы; Самостоятельная работа
4.о перспективах развития современных графических пакетов	Лекции; Лабораторные работы; Самостоятельная работа
5.методы реализации основных алгоритмов функционирования современных графических пакетов;	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.1.з1 знать методы и средства компьютерной графики и геометрического моделирования	
6.методы доступа к программным средствам, предоставляемых современными графическими пакетами для решения задач прикладного программирования.	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.2.з7 знать универсальность математических методов в познании окружающего мира	
7.разрабатывать новые компоненты графических пакетов, создавать прикладные программы с применением пакета OpenGL.	Лекции; Лабораторные работы; Самостоятельная работа
8.разработки интерактивных графичесих программ с использованием современных графических систем типа OpenGL и DirectX	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.2.у8 уметь применять основные методы математического аппарата в математических моделях объектов и процессов	
9.уметь применять основные методы математического аппарата в математических моделях объектов и процессов	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.2.з7 знать универсальность математических методов в познании окружающего мира	

10.знает методы и средства компьютерной графики и геометрического моделирования	Лекции; Лабораторные работы; Самостоятельная работа
ПК.9.В/ПК.з6 знать особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	
11.умеет анализировать и применять на практике правовые нормы, связанные с особенностями жизни и труда лиц с ограниченными возможностями здоровья, в том числе в сфере разработки и создания информационных систем	Лекции; Лабораторные работы; Самостоятельная работа
12.знает особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	Лекции; Лабораторные работы; Самостоятельная работа
13.умеет использовать современные информационные технологии и инструментальные средства для создания интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.2.з7 знать универсальность математических методов в познании окружающего мира	
14.методы и средства компьютерной графики и геометрического моделирования	Лекции; Лабораторные работы; Самостоятельная работа
15.универсальность математических методов в познании окружающего мира	Лекции; Лабораторные работы; Самостоятельная работа
ОПК.2.у8 уметь применять основные методы математического аппарата в математических моделях объектов и процессов	
16.применять основные методы математического аппарата в математических моделях объектов и процессов	Лекции; Лабораторные работы; Самостоятельная работа
ПК.9.В/ПК.з6 знать особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	
17.особенности пользовательских интерфейсов ИС	Лекции; Лабораторные работы; Самостоятельная работа
ПК.9.В/ПК.у5 уметь использовать современные информационные технологии и инструментальные средства для создания интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	
18.уметь использовать современные ИТ и инструментальные средства с учетом ограничения возможностей здоровья	Лекции; Лабораторные работы; Самостоятельная работа

3. Содержание и структура учебной дисциплины

Таблица 3.1

Темы лекций	Активные формы, час.	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 3				
Дидактическая единица: Введение в программирование с использование библиотеки OpenGL				
1. Определение, основные задачи. Сферы применения компьютерной графики. Классификация применений компьютерной графики. Введение в программирование с использование библиотеки OpenGL. Возможности OpenGL. Архитектура, конвейер. Сравнение с Direct3D. Синтаксис команд. Примитивы, атрибуты OpenGL. Определение объектов сцены. Процесс визуализации	0	1	1, 2, 3, 4	изучение лекции

1. Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой.	0	1	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	изучение лекции
Дидактическая единица: Реалистичная визуализация				
3. Реалистичная визуализация граней. Уровни реалистичности. Модели освещения. Закон Ламберта. Модель Фонга. Закраска Гуро и Фонга. Использование источников света в OpenGL. Текстурирование. Отображение текстуры. Типы текстур. Bump mapping.	0	1	1, 15, 16	изучение лекции
Дидактическая единица: Системы координат. Преобразования плоскости и пространства				
4. Преобразования координат и объектов. Системы координат. Преобразования вида. Однородные координаты. Геометрический смысл однородных координат. Аффинные преобразования. Использование аффинных преобразований в OpenGL. Проективные преобразования. Типы проекций. Перспективная проекция. Точка схода. Отсечение. Преобразование в экранные координаты.	0	1	1, 15, 16, 2	изучение лекции
Дидактическая единица: Реалистичная визуализация				
5. Удаление невидимых линий и поверхностей. Классификация методов. Лицевые и нелицевые грани. Свойства (не)лицевых граней. Алгоритм художника. Метод двоичного разбиения пространства. BSP-деревья. Буфер глубины. Поддержка в OpenGL.	0	1	1, 15, 16, 2	изучение лекции

6. Трассировка лучей. Прямая трассировка лучей. Обратная трассировка лучей. Алгоритм обратной трассировки лучей. Пересечение луча с объектом. Оптимизация трассировки лучей. Экстенты. Модель освещения Кука-Торренса. Скелет трассировщика лучей. Скелет процедуры нахождения цвета пиксела. Добавление новых примитивов. Реализация отражения, преломления лучей, теней.	0	1	1, 2, 3, 4, 5, 6	изучение лекции
7. Создание кривых и поверхностей. Представление кривых. Кривые Безье, их свойства. Алгоритм де Кастельо. B-сплайны, рациональные сплайны и NURBS-кривые. Моделирование криволинейных поверхностей. Поддержка в OpenGL.	0	2	1, 2, 3, 4, 5, 6	изучение лекции
8. Теория цвета. Восприятие цвета. Цветовые пространства. Квантование цвета. Псевдотонирование. Фильтры. Шумоподавление. Метрики качества. Смешение цветов. Прозрачность. Поддержка в OpenGL.	0	1	1, 2, 3, 4, 5, 6	изучение лекции
Дидактическая единица: Основы теории фракталов				
9. Основы теории фракталов.	0	1	4, 5, 6, 7, 8	изучение лекции
10. Фрактальная графика. Кодирование изображений с помощью простых преобразований. Фрактальное сжатие изображений. IFS-фракталы. Декодирование сжатых изображений.	0	1	4, 5, 6, 7, 8	изучение лекции
Дидактическая единица: Реалистичная визуализация				
11. Мировая и локальная система координат. Роль локальных координат в компьютерной графике. Координатный репер, его матрица. Однородные координаты точки. Координаты вектора. Связь мировых и локальных координат. Вращение объектов и сцены с помощью мыши	0	1	1, 2, 3, 4, 5	изучение лекции
12. Сплайны в компьютерной графике. Построение и редактирование объектов с помощью сплайнов. Алгебраическая форма кубического сплайна кривой.	0	1	1, 3, 4	изучение лекции

Дидактическая единица: Полигональное моделирование в компьютерной графике				
13. О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике.	0	3	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	изучение лекции
Дидактическая единица: Реалистичная визуализация				
14. Неявное моделирование в компьютерной графике. Понятие о поверхности, заданной неявно. Примеры поверхностей - сфера, тор, цилиндр. Метабол. Моделирование капли с помощью двух метаболов. Примитивы. Использование примитивов при моделировании сложных поверхностей. Скелетное моделирование. Полигонизация кривой, заданной неявно. Полигонизация поверхности методом марширующих кубов. Метод марширующих треугольников.	0	1	4, 5, 6, 7, 8	изучение лекции
15. Задача локализации точки. Принадлежность точки простому многоугольнику. Принадлежность точки выпуклому многоугольнику. Выпуклый многогранник. Структура данных для работы с многогранником. Выпуклая оболочка множества точек на плоскости. Алгоритмы построения выпуклых оболочек. Оценка их эффективности. Построение выпуклых оболочек в пространстве. Объединение выпуклых оболочек. Пересечение выпуклых оболочек. Триангуляция конечного числа точек на плоскости. Триангуляция Делоне. Построение триангуляции Делоне. Удаление невидимых поверхностей. Алгоритм сортировки по глубине. Геометрический поиск в планарном подразбиении. Метод детализации триангуляции. Задача о принадлежности точки выпуклому многограннику.	0	1	8	изучение лекции

Таблица 3.2

Темы лабораторных работ	Активные формы, час.	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 3				
Дидактическая единица: Полигональное моделирование в компьютерной графике				
1. Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL.	4	9	1, 10, 11, 12, 13, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	Создание простейшего проиложения для рисования простейших геометрических объектов (примитивов) в OpenGL при помощи библиотеки GLUT.
2. Ввод и взаимодействие с пользователем и анимация в OpenGL.	4	9	1, 10, 11, 12, 13, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	Создания интерактивного приложения, реализующего взаимодействие с пользователем и анимацию средствами OpenGL.
3. Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры.	0	9	1, 17, 18, 5, 6, 7, 9	Создание приложения, создающего модель разноцветного куба и плоских проекций этого трехмерного объекта в OpenGL. Использование функций для задания положения и ориентации камеры и наложения текстуры.
4. Использование источников света в OpenGL и свойств материала. Кривые и поверхности в OpenGL	0	9	11, 12, 13, 14, 15, 16, 17, 18, 9	Создания приложения, использующего источники света и свойства материала средствами OpenGL. Изучение средств вывода кривых и поверхностей в OpenGL

Таблица 3.3

Темы для самостоятельного изучения	Активные формы, час.	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 3				
Дидактическая единица: Неявное моделирование в компьютерной графике				
1. выполнение ргз	0	10	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	работа с литературой и интернетом
Дидактическая единица: Элементы вычислительной геометрии				
2. подготовка к итоговой аттестации	0	5	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	подготовка к итоговой аттестации

Таблица 3.4

	Активные формы, час.	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 3				
Дидактическая единица: Сплайны кривых и поверхностей				
0.	0	0		

4. Самостоятельная работа обучающегося

№	Виды самостоятельной работы	Ссылки на результаты обучения	Часы на выполнение	Часы на консультации
Семестр: 3				
1	РГЗ	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	15	5
Расчетно графическое задание: алгоритмы компьютерной графики.: Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043 Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.			
2	Подготовка к занятиям	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	5	0
Подготовка к занятиям: Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043 Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.			
3	Дополнительная учебная деятельность	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	5	0

Дополнительная учебная деятельность: Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043 Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.

4	Подготовка к аттестации	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	5	2

Подготовка к аттестации: Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043 Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.

5	Самостоятельное изучение теоретического материала	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 2, 3, 4, 5, 6, 7, 8, 9	15	0

Студент изучает темы, приведенные в таблице 3.3 : Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043 Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.

5. Технология обучения

Для организации и контроля самостоятельной работы обучающихся, а также проведения консультаций применяются информационно-коммуникационные технологии (табл. 5.1).

Таблица 5.1

Деятельность	Информационно-коммуникационные технологии
Информирование	e-mail; Личный типовой сайт; Портал НГТУ; Социальные сети; Среда электронного обучения НГТУ; Чат; ЭБС
Консультирование	e-mail; Личный типовой сайт; Портал НГТУ; Социальные сети; Среда электронного обучения НГТУ; Чат; ЭБС
Контроль	e-mail; Личный типовой сайт; Портал НГТУ; Социальные сети; Среда электронного обучения НГТУ; Чат; ЭБС
Размещение учебных материалов	e-mail; Личный типовой сайт; Портал НГТУ; Социальные сети; Среда электронного обучения НГТУ; Чат; ЭБС

Таблица 5.2

Активные и интерактивные формы проведения занятий

№	Наименование активных форм
1	Проблемный метод
Краткое описание применения:	

6. Правила аттестации обучающихся по учебной дисциплине

Для аттестации обучающихся по дисциплине используется балльно-рейтинговая система (БРС), позволяющая выставлять оценки по традиционной шкале и 15-уровневой ECTS. Краткая информация о БРС приведена в табл. 6.1.

Таблица 6.1

Оцениваемые виды деятельности обучающихся	Мин. балл	Максимальный балл
Семестр: 3		
Подготовка к занятиям:	5	10
Контролирующие материалы приводятся в "Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043"		
Дополнительная учебная деятельность:	1	5
Контролирующие материалы приводятся в "Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана."		
Самостоятельное изучение теоретического материала:	5	25
Контролирующие материалы приводятся в "Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана."		
Лекция:	1	5
Контролирующие материалы приводятся в "Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана."		
Лабораторная:	10	15
Контролирующие материалы приводятся в "Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана."		
РГЗ:	15	20

Контролирующие материалы приводятся в "Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана."		
Зачет:	10	30
Контролирующие материалы (список вопросов) приводятся в "Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана."		

В таблице 6.2 представлено соответствие форм контроля заявляемым требованиям к результатам освоения дисциплины.

Таблица 6.2

Коды компетенций ФГОС	Результаты обучения	Формы контроля	
		Защита РГЗ	Зачет
ОПК.1	з1. знать методы и средства компьютерной графики и геометрического моделирования	+	+
ОПК.2	з7. знать универсальность математических методов в познании окружающего мира	+	+
	у8. уметь применять основные методы математического аппарата в математических моделях объектов и процессов	+	+
	ПК.9.В/ПК з6. знать особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья		+
	ПК.9.В/ПК у5. уметь использовать современные информационные технологии и инструментальные средства для создания интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья		+

Фонд оценочных средств по дисциплине представлен в приложении № 1 к рабочей программе.

7. Литература

Основная литература

1. Веретельникова Е. Л. Графические системы [Электронный ресурс] : конспект лекций / Е. Л. Веретельникова ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222407. - Загл. с экрана.

2. Дегтярев В. М. Инженерная и компьютерная графика : [учебник] / В. М. Дегтярев, В. П. Затыльникова. - М., 2010. - 238, [1] с. : ил., табл.

3. Гринько М. Е. Компьютерная графика : учебное пособие / М. Е. Гринько [и др.] ; Новосиб. гос. техн. ун-т. - Новосибирск, 2009. - 286, [1] с. : ил.. - Режим доступа: http://www.ciu.nstu.ru/fulltext/textbooks/2009/grinko.pdf

Интернет-ресурсы

1. ЭБС НГТУ : http://elibrary.nstu.ru/

2. ЭБС «Издательство Лань» : https://e.lanbook.com/

3. ЭБС IPRbooks : http://www.iprbookshop.ru/

4. ЭБС "Znanium.com" : http://znanium.com/

5. :

8. Методическое и программное обеспечение

8.1 Методическое обеспечение

1. Графические системы : методические указания к лабораторным работам для очной и заочной форм обучения АВТФ, направления 230100 "Информатика и вычислительная техника" / Новосиб. гос. техн. ун-т ; [сост.: В. В. Ландовский, Е. Н. Павенко]. - Новосибирск, 2015. - 54, [2] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000215043

2. Гужов В. И. Компьютерная графика (8 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=301. - Загл. с экрана.

3. Гужов В. И. Компьютерная графика (9 семестр) [Электронный ресурс] : электронный учебно-методический комплекс / В. И. Гужов ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://courses.edu.nstu.ru/index.php?show=155&curs=302. - Загл. с экрана.

4. Трошина Г. В. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2012]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000175987. - Загл. с экрана.

5. Задорожный А. Г. Компьютерная графика [Электронный ресурс] : электронный учебно-методический комплекс / А. Г. Задорожный ; Новосиб. гос. техн. ун-т. - Новосибирск, [2015]. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000222401. - Загл. с экрана.

8.2 Специализированное программное обеспечение

1 Microsoft Windows

2 Microsoft Office

9. Материально-техническое обеспечение

Специальное оборудование

№	Наименование	Назначение
1	Комплект звукоусиливающий (Класс речевой)	Для слабослышащих студентов
2	Доска интерактивная	Для учебных презентаций

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»

Кафедра автоматизированных систем управления
Кафедра общих и естественно-научных дисциплин истр

“УТВЕРЖДАЮ”
ДИРЕКТОР ИСТР
д.соц.н., профессор Л.А. Осьмук
“___”_____ _____ г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

УЧЕБНОЙ ДИСЦИПЛИНЫ

Графические системы в сфере социальной реабилитации
Образовательная программа: 09.03.01 Информатика и вычислительная техника, профиль: Автоматизированные системы обработки информации и управления в социальной сфере

Квалификация бакалавр

Новосибирск 2017

1. **Обобщенная структура фонда оценочных средств учебной дисциплины**

Обобщенная структура фонда оценочных средств по **дисциплине** Графические системы в сфере социальной реабилитации приведена в Таблице.

Таблица

Формируемые компетенции	**Показатели сформированности компетенций (знания, умения, навыки)**	**Темы**	**Этапы оценки компетенций**	
			Мероприятия текущего контроля (курсовой проект, РГЗ(Р) и др.)	**Промежуточная аттестация (экзамен, зачет)**
ОПК.1 способность инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	з1. знать методы и средства компьютерной графики и геометрического моделирования	.1 Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL. Дидактическая единица: Введение в программирование с использование библиотеки OpenGL .1 Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой. .1 выполнение ргз .2 Ввод и взаимодействие с пользователем и анимация в OpenGL. .2 подготовка к итоговой аттестации .3 Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры. .6 Трассировка лучей. Прямая трассировка лучей. Обратная трассировка лучей. Алгоритм обратной трассировки лучей. Пересечение луча с объектом. Оптимизация трассировки лучей. Экстенты. Модель освещения Кука-Торренса. Скелет трассировщика лучей. Скелет процедуры нахождения цвета пиксела. Добавление новых примитивов. Реализация отражения, преломления лучей, теней. .7 Создание кривых и поверхностей.	РГЗ, задачи 1-8	Зачет, вопросы 1-15

		Представление кривых. Кривые Безье, их свойства. Алгоритм де Кастельо. B-сплайны, рациональные сплайны и NURBS-кривые. Моделирование криволинейных поверхностей. Поддержка в OpenGL. .8 Теория цвета. Восприятие цвета. Цветовые пространства. Квантование цвета. Псевдотонирование. Фильтры. Шумоподавление. Метрики качества. Смешение цветов. Прозрачность. Поддержка в OpenGL. .9 Основы теории фракталов. .10 Фрактальная графика. Кодирование изображений с помощью простых преобразований. Фрактальное сжатие изображений. IFS-фракталы. Декодирование сжатых изображений. .13 О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике. .14 Неявное моделирование в компьютерной графике. Понятие о поверхности, заданной неявно. Примеры поверхностей - сфера, тор, цилиндр. Метабол. Моделирование капли с помощью двух метаболов. Примитивы. Использование примитивов при моделировании сложных поверхностей. Скелетное моделирование. Полигонизация кривой, заданной неявно. Полигонизация поверхности методом марширующих кубов. Метод марширующих треугольников.		
ОПК.2 способность осваивать методики использования программных средств для решения практических задач	з7. знать универсальность математических методов в познании окружающего мира	.1 Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL. .1 Определение, основные задачи. Сферы применения компьютерной графики. Классификация применений компьютерной графики. Введение в	РГЗ, задачи 1-8	Зачет, вопросы 1-15

		программирование с использование библиотеки OpenGL. Возможности OpenGL. Архитектура, конвейер. Сравнение с Direct3D. Синтаксис команд. Примитивы, атрибуты OpenGL. Определение объектов сцены. Процесс визуализации .1 выполнение ргз Дидактическая единица: Введение в программирование с использование библиотеки OpenGL .1 Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой. .2 подготовка к итоговой аттестации .2 Ввод и взаимодействие с пользователем и анимация в OpenGL. .3 Реалистичная визуализация граней. Уровни реалистичности. Модели освещения. Закон Ламберта. Модель Фонга. Закраска Гуро и Фонга. Использование источников света в OpenGL. Текстурирование. Отображение текстуры. Типы текстур. Bump mapping. .3 Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры. .4 Использование источников света в OpenGL и свойств материала. Кривые и поверхности в OpenGL .4 Преобразования координат и объектов. Системы координат. Преобразования вида. Однородные координаты. Геометрический смысл однородных координат. Аффинные преобразования. Использование аффинных преобразований в OpenGL. Проективные преобразования. Типы проекций. Перспективная проекция. Точка схода. Отсечение. Преобразование в экранные		

		координаты. .5 Удаление невидимых линий и поверхностей. Классификация методов. Лицевые и нелицевые грани. Свойства (не)лицевых граней. Алгоритм художника. Метод двоичного разбиения пространства. BSP-деревья. Буфер глубины. Поддержка в OpenGL. .6 Трассировка лучей. Прямая трассировка лучей. Обратная трассировка лучей. Алгоритм обратной трассировки лучей. Пересечение луча с объектом. Оптимизация трассировки лучей. Экстенты. Модель освещения Кука-Торренса. Скелет трассировщика лучей. Скелет процедуры нахождения цвета пиксела. Добавление новых примитивов. Реализация отражения, преломления лучей, теней. .7 Создание кривых и поверхностей. Представление кривых. Кривые Безье, их свойства. Алгоритм де Кастельо. B-сплайны, рациональные сплайны и NURBS-кривые. Моделирование криволинейных поверхностей. Поддержка в OpenGL. .8 Теория цвета. Восприятие цвета. Цветовые пространства. Квантование цвета. Псевдотонирование. Фильтры. Шумоподавление. Метрики качества. Смешение цветов. Прозрачность. Поддержка в OpenGL. .9 Основы теории фракталов. .10 Фрактальная графика. Кодирование изображений с помощью простых преобразований. Фрактальное сжатие изображений. IFS-фракталы. Декодирование сжатых изображений. .11 Мировая и локальная система координат. Роль локальных координат в компьютерной графике. Координатный репер, его матрица. Однородные координаты точки. Координаты вектора. Связь мировых и локальных координат. Вращение объектов и сцены с помощью мыши .12 Сплайны в компьютерной графике. Построение и редактирование объектов с помощью сплайнов. Алгебраическая форма кубического сплайна кривой. .13 О полигональном моделировании в		

		компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике. .14 Неявное моделирование в компьютерной графике. Понятие о поверхности, заданной неявно. Примеры поверхностей - сфера, тор, цилиндр. Метабол. Моделирование капли с помощью двух метаболов. Примитивы. Использование примитивов при моделировании сложных поверхностей. Скелетное моделирование. Полигонизация кривой, заданной неявно. Полигонизация поверхности методом марширующих кубов. Метод марширующих треугольников. .15 Задача локализации точки. Принадлежность точки простому многоугольнику. Принадлежность точки выпуклому многоугольнику. Выпуклый многогранник. Структура данных для работы с многогранником. Выпуклая оболочка множества точек на плоскости. Алгоритмы построения выпуклых оболочек. Оценка их эффективности. Построение выпуклых оболочек в пространстве. Объединение выпуклых оболочек. Пересечение выпуклых оболочек. Триангуляция конечного числа точек на плоскости. Триангуляция Делоне. Построение триангуляции Делоне. Удаление невидимых поверхностей. Алгоритм сортировки по глубине. Геометрический поиск в планарном подразбиении. Метод детализации триангуляции. Задача о принадлежности точки выпуклому многограннику.		
ОПК.2	у8. уметь применять основные методы математического аппарата в математических моделях объектов и	.1 Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL. Дидактическая	РГЗ, задачи 1-8	Зачет, вопросы 1-15

	процессов	единица: Введение в программирование с использование библиотеки OpenGL .1 Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой. .1 выполнение ргз .2 Ввод и взаимодействие с пользователем и анимация в OpenGL. .2 подготовка к итоговой аттестации .3 Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры. .4 Использование источников света в OpenGL и свойств материала. Кривые и поверхности в OpenGL .13 О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике.		
ПК.9.В/ПК готовность к разработке моделей компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина"	з6. знать особенности интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	Дидактическая единица: Введение в программирование с использование библиотеки OpenGL .1 Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение.		Зачет, вопросы 1-15

		Заливка области с затравкой. .1 Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL. .1 выполнение ргз .2 Ввод и взаимодействие с пользователем и анимация в OpenGL. .2 подготовка к итоговой аттестации .3 Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры. .4 Использование источников света в OpenGL и свойств материала. Кривые и поверхности в OpenGL .13 О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике.		
ПК.9.В/ПК	у5. уметь использовать современные информационные технологии и инструментальные средства для создания интерфейсов информационных систем для пользователей с ограниченными возможностями здоровья	.1 выполнение ргз .1 Введение в OpenGL. Рисование простейших геометрических объектов. Работа с OpenGL при помощи библиотеки GLUT.Примитивы OpenGL. Дидактическая единица: Введение в программирование с использование библиотеки OpenGL .1 Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой. .2 подготовка к итоговой аттестации .2 Ввод и взаимодействие с пользователем и анимация в OpenGL. .3 Модель разноцветного куба. Способы получения плоских проекций		Зачет, вопросы 1-15

		трехмерных объектов в OpenGL. Задание положения и ориентации камеры. Наложение текстуры. .4 Использование источников света в OpenGL и свойств материала. Кривые и поверхности в OpenGL .13 О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность. Основные инструменты редактирования полигональной поверхности. Об оптимизации полигональных поверхностей в компьютерной графике.		

2. Методика оценки этапов формирования компетенций в рамках дисциплины.

Промежуточная аттестация по **дисциплине** проводится в 3 семестре - в форме дифференцированного зачета, который направлен на оценку сформированности компетенций ОПК.1, ОПК.2, ПК.9.В/ПК.

Зачет проводится в устной (письменной) форме, по билетам (тестам).

Кроме того, сформированность компетенций проверяется при проведении мероприятий текущего контроля, указанных в таблице раздела 1.

В 3 семестре обязательным этапом текущей аттестации является расчетно-графическое задание (работа) (РГЗ(Р)). Требования к выполнению РГЗ(Р), состав и правила оценки сформулированы в паспорте РГЗ(Р).

Общие правила выставления оценки по дисциплине определяются балльно-рейтинговой системой, приведенной в рабочей программе учебной дисциплины.

На основании приведенных далее критериев можно сделать общий вывод о сформированности компетенций ОПК.1, ОПК.2, ПК.9.В/ПК, за которые отвечает дисциплина, на разных уровнях.

Общая характеристика уровней освоения компетенций.

Ниже порогового. Уровень выполнения работ не отвечает большинству основных требований, теоретическое содержание курса освоено частично, пробелы могут носить существенный характер, необходимые практические навыки работы с освоенным материалом сформированы не достаточно, большинство предусмотренных программой обучения учебных заданий не выполнены или выполнены с существенными ошибками.

Пороговый. Уровень выполнения работ отвечает большинству основных требований, теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые виды заданий выполнены с ошибками.

Базовый. Уровень выполнения работ отвечает всем основным требованиям, теоретическое

содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые из выполненных заданий, возможно, содержат ошибки.

Продвинутый. Уровень выполнения работ отвечает всем требованиям, теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному.

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Кафедра автоматизированных систем управления
Кафедра общих и естественно-научных дисциплин истр

Паспорт зачета

по дисциплине «Графические системы в сфере социальной реабилитации», 3 семестр

1. Методика оценки

Зачет проводится в устной (письменной) форме, по билетам (тестам). Билет формируется по следующему правилу: первый вопрос выбирается из диапазона четных вопросов , второй вопрос из диапазона нечетных вопросов (список вопросов приведен ниже).

Форма билета для зачета

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Факультет ИСТР

Билет № _____
к зачету по дисциплине «Графические системы в сфере социальной реабилитации»

1. Вопрос 1
2. Вопрос 2.

Утверждаю: зав. кафедрой ____ _____ должность, ФИО
(подпись)
(дата)

2. Критерии оценки

- Ответ на билет для зачета считается **неудовлетворительным**, если студент при ответе на вопросы не дает определений основных понятий, не способен показать причинно-следственные связи явлений, при решении задачи допускает принципиальные ошибки, оценка составляет ____ *баллов.*
- Ответ на билет для зачета засчитывается на **пороговом** уровне, если студент при ответе на вопросы дает определение основных понятий, может показать причинно-следственные связи явлений, при решении задачи допускает непринципиальные ошибки, например, вычислительные,

оценка составляет ____ *баллов.*

- Ответ на билет для зачета билет засчитывается на **базовом** уровне, если студент при ответе на вопросы формулирует основные понятия, законы, дает характеристику процессов, явлений, проводит анализ причин, условий, может представить качественные характеристики процессов, не допускает ошибок при решении задачи, оценка составляет ____ *баллов.*
- Ответ на билет для зачета билет засчитывается на **продвинутом** уровне, если студент при ответе на вопросы проводит сравнительный анализ подходов, проводит комплексный анализ, выявляет проблемы, предлагает механизмы решения, способен представить количественные характеристики определенных процессов, приводит конкретные примеры из практики, не допускает ошибок и способен обосновать выбор метода решения задачи, оценка составляет ____ *баллов.*

3. Шкала оценки

Зачет считается сданным, если сумма баллов по всем заданиям билета оставляет не менее 51 балла (из 100 возможных).

В общей оценке по дисциплине баллы за зачет учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

4. **Вопросы к** зачету **по дисциплине «**Графические системы в сфере социальной реабилитации**»**

1. Определение, основные задачи. Сферы применения компьютерной графики. Классификация применений компьютерной графики. Введение в программирование с использование библиотеки OpenGL. Возможности OpenGL. Архитектура, конвейер. Сравнение с Direct3D. Синтаксис команд. Примитивы, атрибуты OpenGL. Определение объектов сцены. Процесс визуализации

2. Алгоритмы растровой графики. Преобразование отрезков из векторной формы в растровую. Простейший пошаговый алгоритм. Алгоритм Брезенхема для отрезков прямых. Выравнивание отрезков. Линии постоянной яркости. Растровая развертка окружностей. Восьмисторонняя симметрия. Алгоритм Брезенхема. Заполнение многоугольников. Построчное заполнение. Заливка области с затравкой.

3. Реалистичная визуализация граней. Уровни реалистичности. Модели освещения. Закон Ламберта. Модель Фонга. Закраска Гуро и Фонга. Использование источников света в OpenGL. Текстурирование. Отображение текстуры. Типы текстур. Bump mapping.

4. Преобразования координат и объектов. Системы координат. Преобразования вида. Однородные координаты. Геометрический смысл однородных координат. Аффинные преобразования. Использование аффинных преобразований в OpenGL. Проективные преобразования. Типы проекций. Перспективная проекция. Точка схода. Отсечение. Преобразование в экранные координаты.

5. Удаление невидимых линий и поверхностей. Классификация методов. Лицевые и нелицевые грани. Свойства (не)лицевых граней. Алгоритм художника. Метод двоичного разбиения пространства. BSP-деревья. Буфер глубины. Поддержка в OpenGL.

6. Трассировка лучей. Прямая трассировка лучей. Обратная трассировка лучей. Алгоритм обратной трассировки лучей. Пересечение луча с объектом. Оптимизация трассировки лучей. Экстенты. Модель освещения Кука-Торренса. Скелет трассировщика лучей. Скелет процедуры нахождения цвета пиксела. Добавление новых примитивов. Реализация отражения, преломления лучей, теней.

7. Создание кривых и поверхностей. Представление кривых. Кривые Безье, их свойства. Алгоритм де Кастельо. B-сплайны, рациональные сплайны и NURBS-кривые. Моделирование криволинейных поверхностей. Поддержка в OpenGL.

8. Теория цвета. Восприятие цвета. Цветовые пространства. Квантование цвета. Псевдотонирование. Фильтры. Шумоподавление. Метрики качества. Смешение цветов. Прозрачность. Поддержка в OpenGL.

9. Основы теории фракталов. Принцип обратной связи; основные типы процессов обратной связи; побочный эффект малых возмущений; устойчивость вычислений.

10. Динамические процессы. Фрактальная графика. Кодирование изображений с помощью простых преобразований. Фрактальное сжатие изображений. IFS-фракталы. Декодирование сжатых изображений.

11. Мировая и локальная система координат. Роль локальных координат в компьютерной графике. Координатный репер, его матрица. Однородные координаты точки. Координаты вектора. Связь мировых и локальных координат.

12. Сплайны в компьютерной графике. Построение и редактирование объектов с помощью сплайнов.

13. О полигональном моделировании в компьютерной графике. Аппроксимация кривой ломаной. Длина кривой. Об оптимальной аппроксимации кривой ломаной. Полигональная поверхность.

14. Неявное моделирование в компьютерной графике. Понятие о поверхности, заданной неявно. Примеры поверхностей - сфера, тор, цилиндр. Метабол. Моделирование капли с помощью двух метаболов. Примитивы. Использование примитивов при моделировании сложных поверхностей. Скелетное моделирование. Полигонизация кривой, заданной неявно. Полигонизация поверхности методом марширующих кубов. Метод марширующих треугольников.

15. Задача локализации точки. Принадлежность точки простому многоугольнику. Принадлежность точки выпуклому многоугольнику. Выпуклый многогранник.

Дополнительные вопросы:

1. Назначение пакета OpenGl.
2. Назначение и использование библиотеки GLUT.
3. Синтаксис команд OpenGL.
4. Создание приложения с использованием OpenGL в среде Visual C++.
5. Режимы отображения информации в окне.
6. Определение примитива. Примитивы OpenGL.
7. Рисование точек и линий средствами OpenGL.
8. Рисование треугольников и многоугольников средствами OpenGL.
9. Полосы из треугольников и прямоугольников.
10. Назначение анимации. Методы создания анимации средствами OpenGL.

11. Диалоговые средства ввода. Реализация диалоговых средств ввода в библиотеке GLUT.
12. Видовое преобразование. Функции OpenGL, обеспечивающие видовое преобразование.
13. Матричное представление геометрических преобразований в однородных координатах.
14. Графический конвейер OpenGl.
15. Виды проецирования. Параллельные и перспективные проекции.
16. Способы получения триметрической, диметрической и изометрической проекции.
17. Перспективные проеции. Точки схода.
18. Вывод графических изображений на экран средствами OpenGL.
19. Назначение текстуры. Наложение текстуры на объект средствами OpenGL.
20. Форматы графических файлов.
21. Организация видеопамяти персональных компьютеров в режимах SVGA.
22. Организация видеопамяти персональных компьютеров в режимах VGA 12h, 13h.
23. Цветовые модели, используемые в компьютерной графике. RGB – модель.
24. Методы закрашивания граней объекта Метод Гуро. Метод Фонга.
25. Способы задания свойств источников света в OpenGL.
26. Способы задания свойств материала в OpenGL.
27. Математические основы использования бета-сплайнов в компьютерной графике.
28. Интерполяция и аппроксимация с использованием сплайнов.
29. Кривые и поверхности Безье. Математические основы.
30. Построение кривых и поверхностей Безье средствами OpenGL.
31. Программистская модель интерактивной машинной графики.
32. Классы логических устройств ввода. Селектор.
33. Реализация логического устройства ввода типа «селектор» средствами OpenGL.

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Кафедра автоматизированных систем управления
Кафедра общих и естественно-научных дисциплин истр

Паспорт
расчетно-графического задания (работы)

по дисциплине «Графические системы в сфере социальной реабилитации», 3 семестр

1. **Методика оценки**

В рамках расчетно-графического задания (работы) по дисциплине студенты должны разработать алгоритмы и программы в соответствие с заданием.

Обязательные структурные части РГЗ.

Оцениваемые позиции:

2. **Критерии оценки**

- Работа считается **не выполненной,** если выполнены не все части РГЗ(Р), отсутствует анализ объекта, алгоритмы и программы не разработаны или не соответствуют современным требованиям, оценка составляет 2 баллов.
- Работа считается выполненной **на пороговом** уровне, если части РГЗ(Р) выполнены формально: алгоритмы и программы разработаны с ошибками, оценка составляет 3 баллов.
- Работа считается выполненной **на базовом** уровне, если анализ объекта выполнен в полном объеме, алгоритмы и программы разработаны, алгоритмы разработаны ,но не оптимизированы, оценка составляет 4 баллов.
- Работа считается выполненной **на продвинутом** уровне, если анализ объекта выполнен в полном объеме, алгоритмы и программы разработаны и оптимизированы, оценка составляет 5 баллов.

3. **Шкала оценки**

В общей оценке по дисциплине баллы за РГЗ(Р) учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

4. **Примерный перечень задач РГЗ(Р)**

ЗАДАЧА № 1

ВВЕДЕНИЕ В OPENGL. РИСОВАНИЕ ПРОСТЕЙШИХ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ.
РАБОТА С OPENGL ПРИ ПОМОЩИ БИБЛИОТЕКИ GLUT

Цель работы

Изучение основ использования OpenGL и GLUT. Создание простейшего приложения с использованием OpenGL.

1. ЧТО ТАКОЕ GLUT?

OpenGL является мультиплатформенной библиотекой, т.е. программы, написанные с помощью OpenGL, можно легко переносить на различные операционные системы, при этом получая один и тот же визуальный результат. Единственное, что плохо – это то, что для конкретной операционной системы необходимо по-своему производить настройку OpenGL. То есть, допустим, Вы написали OpenGL программу под Windows и захотели перенести её в Linux, код OpenGL должен перенестись без проблем, но операции с окнами, интерфейс управления, операции с устройствами ввода/вывода нужно заново переписать уже под другую операционную систему – Linux. К счастью, существует специальная мультиплатформенная библиотека, позволяющая решить вышеописанные проблемы. И называется эта библиотека – GLUT.

2. ОСНОВНЫЕ ШАГИ ДЛЯ ПОСТРОЕНИЯ МИНИМАЛЬНОЙ ПРОГРАММЫ

Прежде всего необходимо научиться самому основному – это созданию окна, в котором можно будет рисовать с помощью OpenGL. Минимальная программа, которая создает окно и что-нибудь рисует там, состоит из следующих шагов:

1. Инициализация GLUT
2. Установка параметров окна.
3. Создание окна.
4. Установка функций, отвечающих за рисование в окне и изменение формы окна.
5. Вход в главный цикл GLUT.

Рассмотрим все 5 пунктов поподробнее.

1. Инициализация GLUT производится командой:

*void glutInit(int *argcp, char **argv);*

Первый параметр представляет собой указатель на количество аргументов в командной строке, а второй – указатель на массив аргументов. Обычно эти значения берутся из главной функции программы: int main(int argc, char *argv[]).

2. Установка параметров окна содержит в себе несколько этапов. Прежде всего, необходимо указать размеры окна:

void glutInitWindowSize(int width, int height);

Первый параметр width – ширина окна в пикселях, второй height – высота окна в пикселях. Если эту команду опустить, то GLUT сам установит размеры окна по умолчанию, обычно это 300x300.

Далее можно задать положение создаваемого окна относительно верхнего левого угла экрана. Делается это командой:

void glutInitWindowPosition(int x, int y);

Необходимо также установить для окна режим отображения информации, т. е. установить для окна такие параметры, как: используемая цветовая модель, количество различных буферов, и т.д. Для этого в GLUT существует команда:

void glutInitDisplayMode(unsigned int mode);

У команды имеется единственный параметр, который может быть представлен одной из следующих констант или комбинацией этих констант с помощью побитового ИЛИ.

Константа	Значение
GLUT_RGB	Для отображения графической информации используются 3 компоненты цвета RGB
GLUT_RGBA	То же что и RGB, но используется также 4 компонента ALPHA (прозрачность)
GLUT_INDEX	Цвет задается не с помощью RGB компонентов, а с помощью палитры. Используется для старых дисплеев, где количество цветов, например, 256
GLUT_SINGLE	Вывод в окно осуществляется с использованием 1 буфера. Обычно используется для статического вывода информации
GLUT_DOUBLE	Вывод в окно осуществляется с использованием 2 буферов. Применяется для анимации, чтобы исключить эффект мерцания
GLUT_ACCUM	Использовать также буфер накопления (Accumulation Buffer). Этот буфер применяется для создания специальных эффектов, например отражения и тени
GLUT_ALPHA	Использовать буфер ALPHA. Этот буфер, как уже говорилось, используется для задания 4-го компонента цвета – ALPHA. Обычно применяется для таких эффектов, как прозрачность объектов и антиалиасинг
GLUT_DEPTH	Создать буфер глубины. Этот буфер используется для отсечения невидимых линий в 3D пространстве при выводе на плоский экран монитора
GLUT_STENCIL	Буфер трафарета используется для таких эффектов, как вырезание части фигуры, делая

	этот кусок прозрачным. Например, наложив прямоугольный трафарет на стену дома, Вы получите окно, через которое можно увидеть, что находится внутри дома
GLUT_STEREO	Этот флаг используется для создания стереоизображений. Используется редко, так как для просмотра такого изображения нужна специальная аппаратура

Вот пример использования этой команды: void glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);

3. Создание окна. После того как окно установлено, необходимо его создать.

*int glutCreateWindow(const char *title);*

Эта команда создаёт окно с заголовком, который Вы укажете в качестве параметра, и возвращает HANDLER окна в виде числа int. Этот HANDLER обычно используется для последующих операций над этим окном, таких как изменение параметров окна и закрытие окна.

4. Установка функций, отвечающих за рисование в окне и изменение формы окна.

После того как окно, в которое будет выводиться графическая информация, подготовлено и создано, необходимо связать с ним процедуры, которые будут отвечать за вывод графической информации, следить за размерами окна, следить за нажатиями на клавиши и т.д. Самая первая и самая необходимая функция, которую мы рассмотрим, отвечает за рисование. Именно она всегда будет вызываться операционной системой, чтобы нарисовать (перерисовать) содержимое окна. Итак, задаётся эта функция командой:

*void glutDisplayFunc(void (*func)(void));*

Единственный параметр этой функции – это указатель на функцию, которая будет отвечать за рисование в окне. Например, чтобы функция void Draw(void), определенная в вашей программе, отвечала за рисование в окне, надо присоединить ее к GLUT следующим образом: glutDisplayFunc(Draw);

И ещё одна функция, которая является важной – это функция, которая отслеживает изменения окна. Как только у окна изменились размеры, необходимо перестроить вывод графической информации уже в новое окно с другими размерами. Если этого не сделать, то, например, увеличив размеры окна, вывод информации будет производиться в старую область окна, с меньшими размерами. Определить функцию, отвечающую за изменение размеров окна, нужно следующей командой:

*void glutReshapeFunc(void (*func)(int width, int height));*

Единственный параметр – это указатель на функцию, отвечающую за изменение размеров окна, которая как видно должна принимать два параметра width и height, соответственно ширина и высота нового (измененного) окна.

5. Вход в главный цикл GLUT.

Ну и последнее, что необходимо сделать, чтобы запустить программу – это войти в так называемый главный цикл GLUT. Этот цикл запускает на выполнение так называемое сердце GLUT, которое обеспечивает взаимосвязь между операционной системой и теми функциями, которые отвечают за окно, получают информацию от устройств ввода/вывода. Для того чтобы перейти в главный цикл GLUT, надо выполнить единственную команду:

void glutMainLoop(void);

Вот в принципе основные шаги для самой простейшей GLUT программы. Ниже приведена простейшая программа с использованием средств библиотеки GLUT, которая рисует в окне изображение сферы.

Команда *glFlush()* гарантирует, что команда рисования будет выполнена немедленно, а не сохранена в буфере.

И ещё, прежде чем использовать функции из GLUT, необходимо к программе подключить заголовок: #include <GL/glut.h>.

3. СОЗДАНИЕ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ, ИСПОЛЬЗУЮЩЕГО OPENGL

Самым простым объектом, с помощью которого можно увидеть всю мощь OpenGL, является сфера. Давайте попытаемся ее изобразить. Для этого надо создать новый проект в VisualC++, выполнить следующие действия:

1. Запустите MSVisualC++6.0.
2. Щелкните меню File->New->Win32 Console Application.
3. Выберите каталог и имя проекта, впишите – sphere, щелкните OK.
4. Выберите An Empty Project, щелкните Finish.
5. Создайте в проекте файл sphere.c.
6. Щелкните Build->Set Active Configuration и установите тип проекта sphere – Win32 Release
7. Далее, щелкайте Project->Settings->Link->Object/library modules и добавьте туда opengl32.lib, glu32.lib и gllut32a.lib.
8. Теперь откомпилируйте и запустите Вашу программу. Меню Build->Execute Sphere.exe

Текст программы Sphere.exe:

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#include <GL/glut.h>
void resize(int width,int height)
{
}
void  display(void)
{
          glColor3d(1,1,0);
          glutSolidSphere(1.0, 25, 25);
         glFlush();
}
void init(void)
{
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   glOrtho(-5.0,5.0,-5.0,5.0,2.0,12.0);
  gluLookAt( 0,0,5, 0,1,0, 0,1,0 );
   glMatrixMode( GL_MODELVIEW);
}
int main(int argc,char ** argv)
{
 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
 glutInitWindowPosition(50,10);
 glutInitWindowSize(400,400);
 glutCreateWindow(«Hello»);
 glutReshapeFunc(resize);
 init();
  glutDisplayFunc(display);
 glutMainLoop();
 return 0;
 }
```

4. СИНТАКСИС КОМАНД OPENGL

Командами в OpenGl называются функции или процедуры. Для их описания используется следующий синтаксис:

type glCommand_name[1 2 3 4][b s i f d ub us ui][v](type1 arg1,-,typeN argN)

gl – это имя библиотеки, в которой описана эта функция: для базовых функций OpenGL, функций из библиотек GLU, GLUT, GLAUX это gl, glu, glut, aux соответственно.

Command_name имя команды

[1 2 3 4]число аргументов команды

[b s i f d ub us ui]тип аргумента:

Доступны следующие типы:

b – GLbyte байт; s – GLshort короткое целое; i – GLint целое; f – GLfloat дробное; d – GLdouble дробное с двойной точностью; ub – GLubyte беззнаковый байт; us – GLushort беззнаковое короткое целое; ui – GLuint беззнаковое целое; v – массив из n параметров указанного типа;

[v] – наличие этого символа показывает, что в качестве параметров функции используется указатель на массив значений.

Символы в квадратных скобках в некоторых названиях не используются. Например, команда glVertex2i() описана как базовая в библиотеке OpenGL, и использует в качестве параметров два целых числа, а команда glColor3fv() использует в качестве параметра указатель на массив из трех вещественных чисел.

Ниже приведены примеры использования команд OpenGL с различными типами аргументов.

```
double array[] = {0.5, 0.75, 0.3, 0.7};
```

```
...
glColor3dv(array);
glColor3ub(200,100,0);   // приводится к 200/256, 100/256, 0/256
glColor3d(0.25,0.25,0);  // темно-желтый
glColot3ub(0,100,0);     // темно-зеленый
glColor3ub(0,0,255);     // синий
```

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Ввести и отладить программу, приведенную в п.3.

2.Заменить функцию glutSolidSphere на функцию, из указанных ниже с соответсвующими параметрами. Значения параметров устанавливайте порядка единицы – 0,5…1,7. Если Вы укажете слишком маленький размер, фигуру будет плохо видно; если слишком большой, то она получится урезанной. Это связано с тем, что ее край как бы вылезет из монитора.

```
glutSolidCube(width) // куб
glutSolidTorus(r,R) // тор
glutSolidCylinder(r,height) // цилиндр
glutSolidCone(r,height) // конус
glutSolidIcosahedron(width) // многогранники
glutSolidOctahedron(width)
glutSolidTetrahedron(width)
glutSolidDodecahedron(width)
glutSolidTeapot(width) // рисует чайник
```

3. С помощью вышеуказанных функций нарисовать проволочные фигуры, заменив и имена функций Solid на Wire.

Контрольные вопросы

1. Назначение пакета OpenGl.
2. Назначение и использование библиотеки GLUT.
3. Синтаксис команд OpenGL.
4. Создание приложения с использованием OpenGL в среде Visual C++.
5. Режимы отображения информации в окне.

ЗАДАЧА № 2

ПРИМИТИВЫ OPENGL

Цель работы

Изучение команд OpenGL для рисования графических примитивов.

1. ПРИМИТИВЫ OPENGL

Точки, линии, треугольники, четырехугольники, многоугольники – простые объекты, из которых состоят любые сложные фигуры. В предыдущей лабораторной работе мы рисовали сферу, конус и тор. OpenGL непосредственно не поддерживает функций для создания таких сложных объектов, т.е. таких функций нет в opengl32.dll. Эти функции есть в библиотеке утилит glu32.dll, и устроены они следующим образом. Для того чтобы нарисовать сферу, функция glutSolidSphere использует функции из библиотеки glu32.dll, а те, в свою очередь, используют базовую библиотеку opengl32.dll и из линий или многоугольников строят сферу. Примитивы создаются следующим образом:

```
glBegin(GLenum mode);     // указываем, что будем рисовать
 glVertex[2 3 4][s i f d](...); // первая вершина
                               // тут остальные вершины
 glVertex[2 3 4][s i f d](...); // последняя вершина
glEnd();                       // закончили рисовать примитив
```

Сначала Вы говорите, что будете рисовать – glBegin с соответсвующим параметром. Возможные значения mode перечислены ниже в табл. 2.1. Далее Вы указываете вершины, определяющие объекты указанного типа. Обычно, Вы будете задавать вершину одним из четырех способов.

```
glVertex2d(x,y);    // две переменных типа double
glVertex3d(x,y,z);  // три переменных типа double
glVertex2dv(array); // массив из двух переменных типа double
glVertex3dv(array);  // массив из трех переменных типа double
```

И, наконец, Вы вызваете glEnd(), чтобы указать, что Вы закончили рисовать объекты типа, указанного в glBegin(). Далее мы подробно разберем создание всех примитивов.

Таблица 2.1

Значение mode	Описание
GL_POINTS	Каждый вызов glVertex задает отдельную точку
GL_LINES	Каждая пара вершин задает отрезок
GL_LINE_STRIP	Рисуется ломаная
GL_LINE_LOOP	Рисуется ломаная, причем ее последняя точка соединяется с первой
GL_TRIANGLES	Каждые три вызова glVertex задают треугольник
GL_TRIANGLE_STRIP	Рисуются треугольники с общей стороной
GL_TRIANGLE_FAN	То же самое, но по другому правилу соединяются вершины
GL_QUADS	Каждые четыре вызова glVertex задают четырехугольник
GL_QUAD_STRIP	Четырехугольники с общей стороной
GL_POLYGON	Многоугольник

2. ТОЧКИ

Вы можете нарисовать столько точек, сколько вам нужно. Вызывая glVertex3d, вы устанавливаете новую точку. Размер точки можно устанавливать с помощью функции:
void glPointSize(GLfloat size);

Режим сглаживания можно устанавливать вызовом функции
glEnable(GL_POINT_SMOOTH);

Отключается, соответственно, вызовом glDisable() с этим параметром. Последние функции – glPointSize и glEnable/glDisable надо вызывать вне glBegin/glEnd, иначе они будут проигнорированы.

Примеры рисования точек:

```
// рисуем точки
 glPointSize(2);
 glBegin(GL_POINTS);
  glColor3d(1,0,0);
  glVertex3d(-4.5,4,0); // первая точка
  glColor3d(0,1,0);
  glVertex3d(-4,4,0);   // вторая точка
  glColor3d(0,0,1);     // третья
  glVertex3d(-3.5,4,0);
 glEnd();

 glPointSize(5);
 glBegin(GL_POINTS);
  glColor3d(1,0,0);
  glVertex3d(-2,4,0); // первая точка
  glColor3d(0,1,0);
  glVertex3d(-1,4,0);   // вторая точка
  glColor3d(0,0,1);     // третья точка
  glVertex3d(0,4,0);
 glEnd();
```

3. ЛИНИИ

Для линий Вы также можете изменять ширину, цвет, размер, сглаживание. Если вы зададите разные цвета для начала и конца линии, то ее цвет будет переливающимся. OpenGL по умолчанию делает интерполяцию. Так же Вы можете рисовать прерывистые линии, делается это путем наложения маски при помощи следующей функции:
void glLineStipple(GLint factor, GLushort pattern);

Второй параметр задает саму маску. Например, если его значение равно 255(0x00FF), то, чтобы вычислить задаваемую маску, воспользуемся калькулятором. В двоичном виде это число выглядит так: 0000000011111111, т.е. всего 16 бит. Старшие восемь установлены в ноль, значит, тут линии не будет. Младшие установлены в единицу, тут будет рисоваться линия. Первый

параметр определяет, сколько раз повторяется каждый бит. Скажем, если его установить равным 2, то накладываемая маска будет выглядить так:
0000000000000000111111111111111

Далее приведен исходный текст с комментариями

```
glLineWidth(1);       // ширину линии  устанавливаем 1
glBegin(GL_LINES);
  glColor3d(1,0,0);     // красный цвет
 glVertex3d(-4.5,3,0); // первая линия
 glVertex3d(-3,3,0);
 glColor3d(0,1,0);     // зеленый
 glVertex3d(-3,3.3,0); // вторая линия
 glVertex3d(-4,3.4,0);
glEnd();
 glLineWidth(3);      // ширина 3
glBegin(GL_LINE_STRIP); // см. ниже
 glColor3d(1,0,0);
 glVertex3d(-2.7,3,0);
 glVertex3d(-1,3,0);
 glColor3d(0,1,0);
 glVertex3d(-1.5,3.3,0);
 glColor3d(0,0,1);
 glVertex3d(-1,3.5,0);
glEnd();
 glLineWidth(5);
glEnable(GL_LINE_SMOOTH);
glEnable(GL_LINE_STIPPLE); // разрешаем рисовать  прерывистую линию
glLineStipple(2,58360);    // устанавливаем маску
glBegin(GL_LINE_LOOP);
 glColor3d(1,0,0);
 glVertex3d(1,3,0);
 glVertex3d(4,3,0);
 glColor3d(0,1,0);
 glVertex3d(3,2.7,0);
 glColor3d(0,0,1);
 glVertex3d(2.5,3.7,0);
  glEnd();
glDisable(GL_LINE_SMOOTH);
glDisable(GL_LINE_STIPPLE);
```

4. ТРЕУГОЛЬНИКИ

Для треугольника можно задавать те же параметры, что и для линии, плюс есть еще одна функция glPolygonMode. Она устанавливает опции для отрисовки многоугольника. Первый параметр может принимать значения – GL_FRONT, GL_BACK и GL_FRONT_AND_BACK. Второй параметр указывает, как будет рисоваться многоугольник. Он принимает значения: GL_POINT(рисуются только точки), GL_LINE(рисуем линии) и GL_FILL(рисуем заполненный многоугольник). Первый параметр указывает: к лицевой, тыльной или же к обеим сторонам применяется опция, заданная вторым параметром. Треугольники можно рисовать, передав GL_TRIANGLE_STRIP или GL_TRIANGLE_FAN в glBegin. В первом случае, первая, вторая и третья вершины задают первый треугольник. Вторая, третья и четвертая вершины – второй треугольник. Третья, четвертая и пятая вершина – третий треугольник и т.д. Вершины n, n+1 и n+2 определят n-й треугольник. Во втором случае, первая, вторая и третья вершины задают первый треугольник. Первая, третья и четвертая вершины задают второй треугольник и т.д. Вершины 1, n+1, n+2 определяют n-й треугольник. Далее следует пример с комментариями.

```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // см. выше
 glBegin(GL_TRIANGLES);
  glColor3d(1,0,0);      // рисуем треугольник
  glVertex3d(-4,2,0);
  glVertex3d(-3,2.9,0);
  glVertex3d(-2,2,0);
 glEnd();
glLineWidth(2);
 glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); //рисуем  проволочные треугольники
glBegin(GL_TRIANGLE_STRIP); // обратите внимание на порядок вершин
  glColor3d(0,1,0);
```

```
 glVertex3d(1,2,0);
 glVertex3d(0,2.9,0);
 glVertex3d(-1,2,0);
 glVertex3d(0,1.1,0);
glEnd();
glEnable(GL_LINE_STIPPLE);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
glBegin(GL_TRIANGLE_FAN);
 glColor3d(0,0,1);
 glVertex3d(4,2,0);
 glVertex3d(2.6,2.8,0);
 glVertex3d(2,2,0);
 glVertex3d(3,1.1,0);
glEnd();
glDisable(GL_LINE_STIPPLE);
```

5. ЧЕТЫРЕХУГОЛЬНИКИ

Четырехугольники рисуются вызовом функции glBegin с параметром GL_QUADS или GL_QUAD_STRIP. Для первого случая каждые четыре вершины определяют свой четырехугольник. Во втором случае рисуются связанные четырехугольники. Первая, вторая, третья и четвертая вершины определяют первый четырехугольник. Третья, четвертая, пятая и шестая вершины – второй четырехугольник и т.д. (2n-1), 2n, (2n+1) и (2n+2) вершины задают n-й четырехугольник. Многоугольники задаются вызовом glBegin с параметром GL_POLYGON. Все вершины определяют один многоугольник. Для многоугольников можно задавать стили при помощи вышеописанной функции glPolygonMode, толщину линии, толщину точек и цвет.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изобразить точки, линии, треугольники, многоугольники в одном окне.
2. Построить изображение фигуры, заданной преподавателем.

Контрольные вопросы

1. Определение примитива. Примитивы OpenGL.
2. Рисование точек и линий средствами OpenGL.
3. Рисование треугольников и многоугольников средствами OpenGL.
4. Полосы из треугольников и прямоугольников.

ЗАДАЧА № 3

ВВОД И ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ И АНИМАЦИЯ

Цель работы

Изучение способов организации интерактивного взвамодействия с пользователем в программах, использующих OpenGL.

1. ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ В OPENGL

Функции библиотеки GLUT реализуют так называемый событийно-управляемый механизм. Это означает, что есть некоторый внутренний цикл, который запускается после соответствующей инициализации и обрабатывает одно за другим все события, объявленные во время инициализации. К событиям относятся: щелчок мыши, закрытие окна, изменение свойств окна, передвижение курсора, нажатие клавиши, и «пустое» (idle) событие, когда ничего не происходит. Для проведения периодической проверки совершения того или иного события надо зарегистрировать функцию, которая будет его обрабатывать. Для этого используются функции вида:

```
void glutDisplayFunc (void (*func) (void))
void glutReshapeFunc (void (*func) (int width, int height))
void glutMouseFunc (void (*func) (int button, int state, int x, int y))
void glutIdleFunc (void (*func) (void))
void glutKeyboardFunc(void (*func)(unsigned int key, int x, int y)
void glutMotionFunc(void (*func)(int x, int y));
```

*void glutKeyboardFunc(void (*func)(unsigned int key, int x, int y);*

Определяет функцию (func), которая вызывается, когда нажата клавиша на клавиатуре. Возвращаемые параметры:

key – сгенерированный клавиатурой ASCII код;

x,y – координаты положения мыши в координатах отображаемого окна, в момент, когда была нажата кнопка на клавиатуре.

*void glutMouseFunc(void (*func)(int button, int state, int x, int y));*

Определяет функцию (func) которая вызывается, когда кнопка мыши нажата или отпущена. Возвращаемый функцией параметр button может принимать значения GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, или GLUT_RIGHT_BUTTON. Значение параметра state есть GLUT_UP или GLUT_DOWN в зависимости от того, была ли кнопка мыши нажата или отпущена, x and y параметры указывают на координаты в текущем окне, где находилась мышь в момент нажатия или отпускания кнопки.

*void glutMotionFunc(void (*func)(int x, int y));*

Определяет функцию (func) которая вызывается, когда указатель мыши перемещается в пределах окна при нажатой одной или более кнопке, x and y параметры указывают на координаты в текущем окне, где находилась мышь в момент начала события *void glutPostRedisplay(void);*

Отмечает текущее окно как требующее перерисовки. На следующем шаге работы программы будет вызвана функция, зарегистрированная в glutDisplayFunc().

*void glutIdleFunc (void (*func) (void));*

glutIdleFunc() задает функцию, которая будет вызываться каждый раз, когда нет событий от пользователя.

2. ВИДОВОЕ ПРЕОБРАЗОВАНИЕ

К видовым преобразованиям будем относить перенос, поворот и изменение масштаба вдоль координатных осей. Для проведения этих операций достаточно умножить на соответствующую матрицу каждую вершину объекта и получить измененные координаты этой вершины:

$$(x\text{-}, y\text{-}, z\text{-}, 1)^T = M * (x, y, z, 1)^T,$$

где M – матрица видового преобразования. Матрица M может быть создана с помощью следующих команд:

void glTranslate[f d](GLtype x, GLtype y, GLtype z)

void glRotate[f d](GLtype angle, GLtype x, GLtype y, GLtype z)

void glScale[f d](GLtype x, GLtype y, GLtype z)

glTranlsate..() производит перенос объекта, прибавляя к координатам его вершин значения своих параметров.

glRotate..() производит поворот объекта против часовой стрелки на угол angle (измеряется в градусах) вокруг вектора (x,y,z).

glScale..() производит масштабирование объекта (сжатие или растяжение), домножая соответствующие координаты его вершин на значения своих параметров.

Все эти преобразования применимы к примитивам, описания которых будут находиться ниже в программе. В случае, если надо, например, повернуть один объект сцены, а другой оставить неподвижным, удобно сначала сохранить текущую видовую матрицу в стеке командой glPushMatrix(), затем вызвать glRotate..() с нужными параметрами, описать примитивы, из которых состоит этот объект, а затем восстановить текущую матрицу командой glPopMatrix().

3. АНИМАЦИЯ

Если содержимое буфера кадра изменяется в процессе регенерации изображения, то зритель может увидеть совершенно нежелательные эффекты, например, дерганье изменяющейся картинки.

Эту проблему можно решить с использованием двойной буферизации – стандартной технологии организации компьютерной анимации. В этом случае в нашем распоряжении имеется два буфера кадра, которые принято называть рабочим и фоновым. Рабочий буфер – это тот, из которого выполняется регенерация изображения на экране, а в фоновом буфере изображение формируется программой. По командам из прикладной программы можно переключать функции буферов: сделать рабочим тот, который ранее был фоновым, а фоновым – тот, который ранее был рабочим. Механизм двойной буферизации устанавливается в процессе инициализации аргументом функции glutInitDisplayMode(). Вместо константы GLUT_SINGLE нужно задать константу GLUT_DOUBLE. Переключение буферов выполняется функцией glutSwapBuffers(). Все операторы формирования изображения включатся в функцию display(), но при использовании двойной буферизации в этой функции сначала нужно очистить рабочий буфер, вызвав команду glClear(), а последним оператором вызвать функцию переключения буферов glutSwapBuffers().

Программа рисования вращающегося квадрата

```
#include <GL/glut.h>
#include <stdlib.h>
static GLfloat spin = 0.0;
void spinDisplay(void)
```

```
{
   spin = spin + 2.0;
  if (spin > 360.0)
     spin = spin – 360.0;
  glutPostRedisplay();
}
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);
  glPushMatrix();
  glRotatef(spin, 0.0, 0.0, 1.0);
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_POLYGON);
        glVertex2f(-25.0,-25.0);
        glVertex2f(25.0,-25.0);
        glVertex2f(25.0,25.0);
        glVertex2f(-25.0,25.0);
        glEnd( );
  glPopMatrix();
  glutSwapBuffers();
}
void init(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glShadeModel (GL_FLAT);
}
void reshape(int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize (250, 250);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutIdleFunc(spinDisplay);
  glutMainLoop();
  return 0;   /* ANSI C requires main to return int. */
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Ввести и отладить программу рисования вращающегося квадрата.
2. Изменить программу п.1 таким образом, чтобы она управлялась нажатием клавиш на клавиатуре – при нажатии клавиши «x», квадрат вращается, при нажатии клавиши «X»- вращение прекращается, при нажатии клавиши «Esc» – программа завершает свою работу.
3. Изменить программу п.1 таким образом, чтобы она управлялась нажатием клавиши мыши – при нажатии левой клавиши–квадрат вращается, при нажатии правой клавиши–вращение прекращается.
4. Нарисовать любую фигуру в одном из углов экрана. Обеспечить перемещение этой фигуры по экрану при нажатой и удерживаемой клавише мыши.

Контрольные вопросы

1. Назначение анимации. Методы создания анимации средствами OpenGL.

2. Диалоговые средства ввода. Реализация диалоговых средств ввода в библиотеке GLUT.
3. Видовое преобразование. Функции OpenGL, обеспечивающие видовое преобразование.
4. Матричное представление геометрических преобразований в однородных координатах.

ЗАДАЧА № 4

Модель разноцветного куба. Способы получения плоских проекций трехмерных объектов. Задание положения и ориентации камеры.

Цель работы

Изучение методов создания трехмерных объектов средствами OpenGL. Исследование свойств полоских проекций трехмерных объектов.

1. РИСОВАНИЕ ТРЕХМЕРНОГО КУБА

Куб следует рассматривать как шесть многоугольников, которые определяют его грани. Массив вершин куба может быть представлен в следующем виде:

```
GLfloat vertices[][3]={{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
{1.0,1.0,-1.0},{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},
{1.0,-1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};
```

Для определения граней куба можно использовать список точек–элементов массива вершин. Например, одна грань куба в тексте программы определяется следующим образом:

```
glBegin(GL_POLYGON);
        glVertex3fv(vertices[0]);
        glVertex3fv(vertices[3]);
        glVertex3fv(vertices[2]);
        glVertex3fv(vertices[1]);
glEnd();
```

Другие пять граней определяются аналогично. При определении трехмерных многогранников порядок перечисления вершин имеет большое значение. Следует учитывать, что многоугольник имеет две стороны – внутреннюю и внешнюю. Будем называть грань внешней, если при взгляде с внешней стороны объекта на эту грань ее вершины «обходятся» против часовой стрелки. Этот метод известен как «правило правой руки», поскольку, если расположить четыре согнутых пальца правой руки вдоль направления обхода контура, большой палец будет указывать наружную сторону грани.

Список вершин можно использовать и для хранения информации, необходимой для раскрашивания куба. С вершинами в данном примере будут ассоциироваться чистые цвета вершин цветового куба (черный, белый, красный, зеленый, синий, голубой, фиолетовый, желтый):

```
GLfloat colors[][3]={{0.0,0.0,0.0},{1.0,0.0,0.0},
{1.0,1.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0},
{1.0,0.0,1.0},{1.0,1.0,1.0},{0.0,1.0,1.0}};
```

Для управления режимом интерполяции цветов используется команда void **glShadeModel**(GLenummode), вызов которой с параметром GL_SMOOTH включает интерполяцию (установка по умолчанию), а с GL_FLAT отключает.

Функция quad() вычерчивает четырехугольник, заданный точками в списке вершин, а функция colorcube() задает шесть граней таким образом, чтобы все они были внешними.

```
GLfloat vertices[][3]={{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
{1.0,1.0,-1.0},{-1.0,1.0,-1.0},{-1.0,-1.0,1.0},
{1.0,-1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0}};
GLfloat colors[][3]={{0.0,0.0,0.0},{1.0,0.0,0.0},
{1.0,1.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0},
{1.0,0.0,1.0},{1.0,1.0,1.0},{0.0,1.0,1.0}};
void polygon(int a, int b, int c, int d)
{
        glBegin(GL_POLYGON);
        glColor3fv(colors[a]);
        glVertex3fv(vertices[a]);
        glColor3fv(colors[b]);
        glVertex3fv(vertices[b]);
        glColor3fv(colors[c]);
        glVertex3fv(vertices[c]);
        glColor3fv(colors[d]);
        glVertex3fv(vertices[d]);
```

```
        glEnd();
}
void colorcube()
{
        polygon(0,3,2,1);
        polygon(2,3,7,6);
        polygon(0,4,7,3);
        polygon(1,2,6,5);
        polygon(4,5,6,7);
        polygon(0,1,5,4);
}
```

2. ПРОЕКТИВНЫЕ ПРЕОБРАЗОВАНИЯ В OPENGL

В составе OpenGL имеются две функции для задания перспективных проекций и одна для задания параллельных проекций. Каждая из функций определяет зону видимости – пирамиду или параллелепипед. Объекты, не попадающие в эту зону, отсекаются и не включаются в отображаемую сцену.

3. ПЕРСПЕКТИВНЫЕ ПРЕОБРАЗОВАНИЯ В OPENGL

Параметры пирамиды видимости задаются функцией **glFrustum()**, смысл аргументов которой поясняет рис. 4.1.

*void **glFrustum**(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);*

Значения аргументов near и far, задающих положение передней и задней отсекающих плоскостей, должны быть положительными и отсчитываться от центра проецирования вдоль оси проецирования.

Поскольку матрица проецирования умножается на текущую матрицу, сначала нужно задать режим работы с этой матрицей. Типичная последовательность операций представлена ниже.

```
glMatrixMode(GL PROJECTION);
```

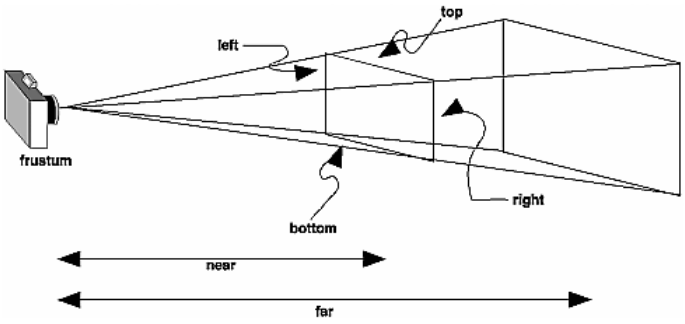

Рис. 4.1

```
glLoadIdentity();
glFrustum(xmin, xmax, ymin, ymax, near, far);
```

Во многих приложениях предпочтительнее задавать не линейные параметры, характеризующие положение углов усеченной пирамиды видимости, а угол и поле зрения. Однако если картинная плоскость является прямоугольником, а не квадратом, то нужно задавать пару углов зрения: один в вертикальной плоскости, другой- в горизонтальной (рис. 4.2).

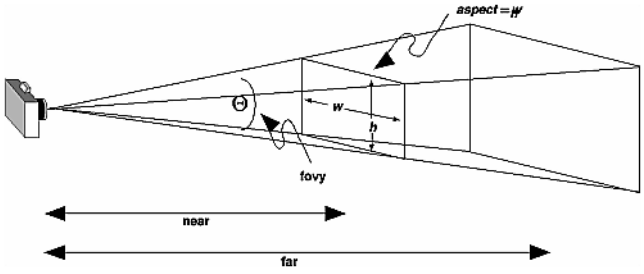

Рис. 4.2

*void **gluPerspective**(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far);*

Аргументы этой функции имеют следующий смысл:

- fovy – угол зрения в вертикальной плоскости;

- aspect – отношение ширины окна картинной плоскости к его высоте;
- near и far – расстояние от центра проецирования до передней и задней отсекающих плоскостей.

4. ПАРАЛЛЕЛЬНОЕ ПРОЕЦИРОВАНИЕ В OPENGL

В составе OpenGL имеется только одна функция для задания параметров параллельного проецирования, которая формирует ортогональную проекцию. Зона видимости при этом превращается в параллелепипед (рис. 4.3.).

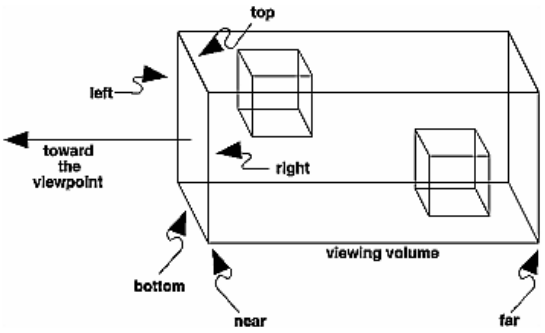

Рис. 4.3

*void **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);*

Аргументы вызова имеют тот же геометрический смысл, что и одноименные аргументы функции glFrustum().

5. ЗАДАНИЕ ПОЛОЖЕНИЯ И ОРИЕНТАЦИИ КАМЕРЫ

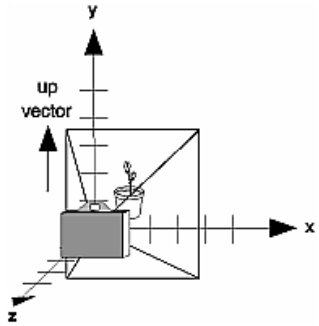

Рис. 4.4

В составе OpenGL имеется функция gluLookAt(), которая позволяет задать положение и ориентацию камеры (рис. 4.4).

*void **gluLookAt**(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);*

Аргументы функции имеют следующий вид:

- eyex, eyey, eyez – координаты точки наблюдения;
- centerx, centery, centerz – координаты контрольной точки объекта, указывающей центр сцены;
- upx, upy, upz- компоненты точки, которая задает положительное направления оси Y сцены.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Составить программу рисования куба.
2. Получить перспективную и параллельную проекцию куба.
3. Организовать перемещение камеры вокруг куба, изменяя координаты точки наблюдения – eyex, eyey, eyez. Для перемещения камеры использовать клавиатуру.

Контрольные вопросы

1. Графический конвейер OpenGl.
2. Виды проецирования. Параллельные и перспективные проекции.
3. Способы получения триметрической, диметрической и изометрической проекции.
4. Перспективные проеции. Точки схода.

ЗАДАЧА № 5

РАБОТА С ИЗОБРАЖЕНИЕМ. НАЛОЖЕНИЕ ТЕКСТУРЫ

Цель работы

Изучение методов работы с растровыми примитивами OpenGl и методов наложения текстур.

1. РАБОТА С ИЗОБРАЖЕНИЕМ

Существует множество графических форматов – bmp, pcx, gif, jpeg и прочие. OpenGL напрямую не поддерживает ни один из них. В OpenGL нет функций чтения/записи графических файлов, но поддерживается работа с массивами пикселей. Вы загружаете графический файл, используя библиотеки других фирм, в память и работаете с ними средствами OpenGL. В массиве данные о пикселах могут располагаться разными способами: RGB, BGR, RGBA; могут присутствовать не все компоненты; каждый элемент массива может занимать один байт, два, четыре или восемь; выравнивание может быть по байту, слову или двойному слову. В общем, форматов расположения данных о графическом изображении в памяти очень много.

Наиболее часто применяется формат, в котором информация о каждом пикселе хранится в формате RGB и занимает три байта, выравнивание по байту. В Auxiliary Library есть функция auxDIBImageLoad(LPCSTR), которая загружает в память bmp-файл и возвращает указатель на структуру:

```
typedef struct _AUX_RGBImageRec {
  GLint sizeX, sizeY;
  unsigned char *data;
} AUX_RGBImageRec;
```

В OpenGL имеются функции для вывода массива пикселей на экран(glDrawPixels), копирования(glCopyPixels), масштабирования(gluScaleImage). Здесь мы рассмотрим только glDrawPixels. Все остальные функции работы с изображениями устроены похожим образом. Для того чтобы отобразить графический файл в окне OpenGL, вы должны загрузить его в память, указать выравнивание, установить точку, с которой начинается вывод изображения, и вывести его на экран. Создайте новый проект. Объявите глобальную переменную –

```
AUX_RGBImageRec *image
```

Перед вызовом функции glutMainLoop() в функции main вставьте строку:

```
image = auxDIBImageLoad(«photo.bmp»);
```

Выравнивание устанавливается вызывом функции glPixelStorei с параметром GL_UNPACK_ALIGNMENT и вторым параметром – целым числом, которое указывает выравнивание.

void glPixelStore{if}(GLenum pname, TYPEparam);

Функция установки способа хранения пикселов для выполнения следующих функций: glDrawPixels(), glReadPixels(), glBitmap(), glPolygonStipple(), glTexImage1D(), glTexImage2D(), glTexSubImage1D(), glTexSubImage2D(), and glGetTexImage().

Изображения выводятся прямо на экран. Поэтому все происходит в двухмерных координатах. Позиция, с которой начинается вывод изображения, указывается при помощи функции glRasterPos2d(x,y).

```
void glRasterPos{234}{sifd}(TYPE x, TYPE y, TYPE z, TYPE w);
void glRasterPos{234}{sifd}v(TYPE *coords);
```

Функция установки текущей позиции растра: x, y, z, w – однородные координаты, определяющие позицию растра. Если используется функция glRasterPos2*(), то z подразумевается равным 0, а w равным 1.

Также Вы можете установить размер пикселя, вызвав функцию glPixelZoom. Первый параметр этой функции – ширина, второй – высота пикселя. Вызов этой функции с аргументами (1,1), что соответствует нормальному пикселю. Замените (1,1) на (3,2) и Вы увидите, как картинка растянется в три раза по горизонтали и в два раза по вертикали. Это случилось потому, что теперь каждый пиксель изображения соответствует прямоугольнику 3x2 в окне.

```
void glPixelZoom(GLfloat zoomx, GLfloat zoomy);
```

Функция устанавливает фактор увеличения/уменьшения при выполнении операции записи пикселя в буфер кадра (glDrawPixels() или glCopyPixels()) по x и y координатам. По умолчанию zoomx и zoomy равны 1.0. Если оба значения равны 2, то каждый пиксель исходного изображения выводится в виде 4 пикселей в буфере кадра. Отрицательное значение фактора выполняет зеркальное отображение изображения относительно текущей позиции растра.

И, наконец, вывод осуществляет функция glDrawPixels. Первые два параметра – это ширина и высота. Далее, вы указываете формат, в котором хранится информация в памяти, и тип элементов массива. Последним указывается массив данных.

```
void glDrawPixels(GLsizei width, GLsizei height, GLenum format,
GLenum type, const GLvoid *pixels);
```

Функция выводит изображение прямоугольного массива пикселей размером width на height. Прямоугольник выводится, начиная с левого нижнего угла изображения в текущей позиции растра, format и type параметры могут принимать одно из значений, приведенных в Руководстве программиста по OpenGL. Массив pixels содержит данные о пикселях, которые должны быть выведены.

В функцию display вставьте следующий код:

```
glRasterPos2d(-4.5,-3);                 // нижний левый угол
glPixelZoom(1,1);
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);        // выравнивание
glDrawPixels(image->sizeX, image->sizeY, // ширина и высота
            GL_RGB, GL_UNSIGNED_BYTE,      // формат и тип
                image->data);     // сами данные
```

Также в OpenGL имеется функция glBitmap для отображения битовых массивов. Битовый массив – это последовательность байт, которые кодируют картинку из двух цветов.

2. НАЛОЖЕНИЕ ТЕКСТУРЫ

Одного вывода изображений недостаточно для создания полноценных трехмерных сцен. Часто возникает потребность накладывать изображение на трехмерные объекты и поворачивать/сдвигать их. Для этих целей существуют текстуры. Также текстуры помогут вам покрыть весь объект в виде мозаики. Скажем, когда у вас имеется кирпичная стена, то вам не надо загружать изображение с кучей кирпичей. Достаточно загрузить один кирпич и указать, что эту текстуру нужно размножить по всей плоскости.

Для того чтобы наложить текстуру на объект, вы должны:

1. Загрузить графический файл в память.
2. Создать имя-идентификатор текстуры.
3. Сделать его активным.
4. Создать саму текстуру в памяти.
5. Установить параметры текстуры.
6. Установить параметры взаимодействия текстуры с объектом.
7. Связать координаты текстуры с объектом.

1. Загрузка графического файла выполняется в соответствии с п. 1 лабораторной работы.

2. Создать имя-идентификатор текстуры.

При использовании в сцене нескольких текстур в OpenGL применяется подход, напоминающий создание списков изображений. Вначале с помощью команды

```
void glGenTextures(GLsizei n, GLuint*textures)
```

надо создать n идентификаторов для используемых текстур, которые будут записаны в массив textures.

Для единственной текстуры в приложении можно использовать следующую последовательность операторов:

```
static GLuint texName;
…
glGenTextures(1, &texName);
```

3. Сделать имя текстуры активным.

Перед началом определения свойств очередной текстуры следует вызвать команду

```
void glBindTexture(GLenum target, GLuint texture),
```

где target может принимать значения GL_TEXTURE_1D или GL_TEXTURE_2D, а параметр texture должен быть равен идентификатору той текстуры, к которой будут относиться последующие команды. Для того чтобы в процессе рисования сделать текущей текстуру с некоторым идентификатором, достаточно опять вызвать команду glBindTexture() с соответствующим значением target и texture. Таким образом, команда glBindTexture() включает режим создания текстуры с идентификатором texture, если такая текстура еще не создана, либо режим ее использования, т. е. делает эту текстуру текущей. Например,

```
glBindTexture(GL_TEXTURE_2D, texName);
```

4. Создание текстуры в памяти.

Теперь создать саму текстуру в памяти. Массив байт в структуре AUX_RGBImageRec не является еще текстурой, потому что у текстуры много различных параметров. Создав текстуру, мы наделим ее определенными свойствами. Среди параметров текстуры вы указываете уровень детализации, способ масшабирования и связывания текстуры с объектом. Уровень детализации нужен для наложения текстуры на меньшие объекты, т.е. когда площадь на экране меньше размеров изображения. Нулевой уровень детализации соответствует исходному изображению размером $2^n x 2^m$, первый уровень – 2^n-1x2^m-1, k-й уровень – 2^n-kx2^m-k. Число уровней соответствует min(n,m). Для создания текстуры имеется две функции glTexImage[1/2]D и gluBuild[1/2]DMipmaps.

```
glTexImage2D(                gluBuild2DMipmaps(
  GLenum target,               GLenum target,
  GLint lavel,                 GLint components,
  GLint components,            GLsizei width,
  GLsizei width,               GLsizei height,
  GLsizei height,              GLenum format,
  GLint border,                GLenum type,
  GLenum format,               const GLvoid* pixels)
  GLenum type,
```

const GLvoid* pixels)

Основное различие в том, что первая функция создает текстуру одного определенного уровня детализации и воспринимает только изображения, размер которых кратен степени двойки. Вторая функция более гибкая. Она генерирует текстуры всех уровней детализации. Также эта функция не требует, чтобы размер изображения был кратен степени двойки. Она сама сожмет/растянет изображение подходящим образом, хотя возможно окажется, что и не вполне подходящим. Я воспользуюсь функцией glTexImage2D. Первый параметр этой функции должен быть GL_TEXTURE_2D. Второй – уровень детализации. Нам нужно исходное изображение, поэтому уровень детализации – ноль. Третий параметр указывает количество компонентов цвета. У нас изображение хранится в формате RGB. Поэтому значение этого параметра равно трем. Четвертый и пятый параметры – ширина и высота изображения. Шестой – ширина границы; у нас гарницы не будет, поэтому значение этого параметра – ноль. Далее, седьмой параметр – формат хранения пикселей в массиве – GL_RGB и тип – GL_UNSIGNED_BYTE. И, наконец, восьмой параметр – указатель на массив данных. Еще вы должны вызвать функцию glPixelStorei и задать, что выравнивание в массиве данных идет по байту.

Добавьте следующий код в функцию main.

```
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
glTexImage2D(GL_TEXTURE_2D, 0, 3,
        photo_image->sizeX,
        photo_image->sizeY,
        0, GL_RGB, GL_UNSIGNED_BYTE,
        photo_image->data);
```

Аналогичный результат можно получить, вставив вызов gluBuild2DMipmaps с параметрами, указанными ниже.

```
gluBuild2DMipmaps(GL_TEXTURE_2D, 3,
            photo_image->sizeX,
            photo_image->sizeY,
            GL_RGB, GL_UNSIGNED_BYTE,
            photo_image->data);
```

5. Установить параметры текстуры.

Теперь нужно установить параметры текстуры. Для этого служит функция
glTexParameter[if](GLenum target, GLenum pname, GLenum param)

Первый параметр принимает значение GL_TEXTURE_1D или GL_TEXTURE_2D. Второй – pname – определяет параметр текстуры, который вы будете изменять. И третий параметр – это устанавливаемое значение. Если вы воспользовались gluBuild2DMipmaps вместо glTexImage2D, то вам не надо устанавливать следующие параметры, так как уже сформированы текстуры всех уровней детализации, и OpenGL сможет подобрать текстуру нужного уровня, если площадь объекта не совпадает с площадью текстуры. В противном случае, Вы должны добавить следующие строки:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
```

Вы указали, что для уменьшения и увеличения текстуры используется алгоритм GL_NEAREST. Это означает, что цветом пикселя объекта, на который накладывается текстура, становится цвет ближайшего пикселя элемента текстуры. Вместо GL_NEAREST можно указать GL_LINEAR, т.е. цвет элемента объекта будет вычисляться как среднее арифметическое четырех элементов текстуры. Имеются еще четыре алгоритма вычисления цвета элемента объекта. Их можно устанавливать, когда вы создали текстуру со всеми уровнями детализации, т.к. применяют алгоритмы GL_NEAREST и GL_LINEAR к одному или двум ближайшим уровням детализации.

6. Установка параметров взаимодействия текстуры с объектом.

Вы можете установить взаимодействие текстуры с объектом. Тут имеются два режима при использовании трех компонентов цвета. Первый режим, установленный по умолчанию, – когда у вас учитывается цвет объекта и цвет текстуры. Результирующий цвет получается перемножением компонентов цвета текстуры на компоненты цвета объекта. Скажем, если цвет текстуры – (r,g,b), а цвет объекта, на который она накладывается, – (r0,g0,b0), то результирующим цветом будет – (r*r0,g*g0,b*b0). В случае, если цвет объекта черный – (0,0,0), то вы не увидите на нем текстуру, так как она вся будет черной. Второй режим взаимодействия, когда цвет объекта не учитывается. Результирующим цветом будет цвет текстуры. Эти параметры можно установить следующим образом.

```
glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE)
glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)
```

По умолчанию используется режим GL_MODULATE. На этом заканчивается создание текстуры.

6. Связь координат текстуры с объектом.

Осталось связать координаты текстуры с координатами объекта.

void glTexCoord{1234}{sifd}(TYPEcoords);
void glTexCoord{1234}{sifd}v(TYPE *coords);

Функция устанавливает соответствие текущих координат текстуры (s, t, r, q) с координатами вершины примитива OpenGL, следующей за этой командой. При исползовании функции glTexCoord2*() небходимо определить только координаты s и t.

Например, чтобы связать координаты текстуры с вершинами прямоугольника, можно использовать следующую последовательность команд в функции display:

```
glEnable(GL_TEXTURE_2D);
glColor3d(1,1,1);
glBindTexture(GL_TEXTURE_2D, space_tex );
 glBegin(GL_QUADS);
    glTexCoord2d(0,0); glVertex3d(-5,-5, -0.1);
    glTexCoord2d(0,1); glVertex3d(-5, 5, -0.1);
    glTexCoord2d(1,1); glVertex3d( 5, 5, -0.1);
    glTexCoord2d(1,0); glVertex3d( 5,-5, -0.1);
 glEnd();
glDisable(GL_TEXTURE_2D);
```

glTexCoord2d сопоставляет координаты текстуры вершинам четырехугольника. Нижний левый угол текстуры имеет координаты (0,0), а верхний правый – (1,1).

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить средствами OpenGL изображение .bmp- файла, заданного преподавателем.
2. Наложить изображение этого файла на все грани куба из лабораторной работы № 4.

Контрольные вопросы

1. Вывод графических изображений на экран средствами OpenGL.
2. Назначение текстуры. Наложение текстуры на объект средствами OpenGL.
3. Форматы графических файлов.
4. Организация видеопамяти персональных компьютеров в режимах SVGA.
5. Организация видеопамяти персональных компьютеров в режимах VGA 12h, 13h.

ЗАДАЧА № 6

ИСПОЛЬЗОВАНИЕ ИСТОЧНИКОВ СВЕТА В OPENGL И СВОЙСТВ МАТЕРИАЛА

Цель работы

Изучение способов включения источников света в сцену и методов учета свойств материала в OpenGL

1. ОПИСАНИЕ ИСТОЧНИКОВ СВЕТА В OPENGL

В системе OpenGl поддерживаются источники света четырех типов: фонового освещения (ambient lighting), точечные источники (point sources), прожекторы (spotlights), удаленные источники света (distant light). В одной программе может использоваться до восьми источников света. Каждый источник света имеет свой набор параметров, в том числе программный код включения/выключения. Параметры, описывающие источник света, соответствуют параметрам модели Фонга. Для установки векторных параметров используется функция glLightfv(), которая имеет следующий формат обращения:

glLightfv(source, parameter, pointer_to_array);

Существует четыре векторных параметра, которые определяют положение и направление лучей источника и цветовой состав его составляющих – фоновой, диффузионной и зеркальной.

Для установки скалярных параметров в OpenGL служит функция glLightf():

glLightf(source, parameter, value);

Пусть, например, требуется включить в сцену источник GL_LIGHT0, который должен находиться в точке (1.0, 2.0, 3.0). Положение источника сохраняется в программе в виде точки в однородных координатах:

GLfloat light0_pos[]={1.0, 2.0, 3.0, 1.0};

Если четвертый компонент этой точки равен нулю, то точечный источник превращается в удаленный, для которого существенно только направление лучей:

GLfloat light0_dir[]={1.0, 2.0, 3.0, 0.0};

Далее определяется цветовой состав фоновой, диффузионной и зеркальной составляющих источника. Если в рассматриваемом примере источник имеет белую зеркальную составляющую, а фоновая и диффузионная составляющие должны быть красными, то фрагмент программы, формирующий источник, выглядит следующим образом:

```
GLfloat diffise0[]= {1.0, 0.0, 0.0, 1.0};
GLfloat ambient0[]={1.0, 0.0, 0.0, 1.0};
GLfloat specular0[]={1.0, 1.0, 1.0, 1.0};
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
```

Функция glEnable() вызывается дважды: сначала для включения режима анализа освещения, а затем для включения в сцену конкретного источника.

В сцену можно включить и глобальное фоновое освещение, которое не связано ни с каким отдельным источником освещения. Если, например, требуется слабо подсветить все объекты сцены белым цветом, в программу следует включит такой фрагмент кода:

```
GLfloat global_ambient[]={0.1, 0.1, 0.1, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
```

В модели освещения член, учитывающий расстояние до источника, имеет вид:

f(d)= 1/(a+ b*d+ c*d^2)

и постоянную, линейную и квадратичную составляющие. Соответствующие коэффициенты для каждого источника задаются индивидуально с помощью функции установки скалярных параметров, например:

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENATION, a);
```

Для преобразование точечного источника в прожектор нужно задать направление луча прожектора (GL_SPOT_DIRECTION), показатель функции распределения интенсивности (GL_SPOT_EXPONENT) и угол рассеяния луча (GL_SPOT_CUTTOF). Эти параметры устанавливаются с помощью функций glLightf() и glLightfv().

Параметры, устанавливаемые для источников света по умолчанию приведены в таблице 6.1.

Таблица 6.1

Параметры, устанавливаемые для источников света по умолчанию

Имя параметра	Значение	Содержание
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x, y, z, w*) position of light
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x, y, z*) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cutoff angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

2. СПЕЦИФИКАЦИЯ МАТЕРИАЛОВ В OPENGL

В OpenGL свойства материалов соответствуют поддерживаемым параметрам источников света и модели отражения Фонга. Программист имеет возможность связывать разные материалы с внутренней и внешней сторонами одной и той же поверхности. Все параметры, обрабатываемые в модели отражения, задаются вызовом двух функций:

```
glMaterialfv(face, type, pointer_to_array);
glMaterilf(face, type, value);
```

Для определения коэффициентов отражения для фоновой, диффузионной и зеркальной составляющих (k_a, k_d, k_s) по каждому из первичных цветов в программу нужно включить

определение трех массивов:

```
GLfloat ambient[]= {0.2, 0.2, 0.2, 1.0};
GLfloat diffise[]= {1.0, 0.8, 0.0, 1.0};
GLfloat specular[]= {1.0, 1.0, 1.0, 1.0};
```

Первый задает небольшое значение коэффициента отражения фоновой составляющей, причем коэффициент одинаков для всех первичных цветов, что эквивалентно отражению белого цвета. Для диффузной составляющей набор коэффициентов по отдельным цветам задает в результате отражение желтого цвета, а для зеркальной составляющей коэффициенты отражения по всем первичным цветам опять одинаковы. Если внешние и внутренние стороны поверхностей имеют одинаковые параметры материала, то при вызове функции glMaterialfv() ей в качестве параметра передается константа GL_FRONT_AND_BACK:

```
glMaterialfv(G_FRONT_AND_BACK, GL_AMBIENT, ambient);
glMaterialfv(G_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
glMaterialfv(G_FRONT_AND_BACK, GL_SPECULAR, specular);
```

Если параметры для зеркальной и диффузионной составляющих одинаковы, то можно задавать их одним вызовом функции glMaterialfv(), передав ей в качестве параметра type константу GL_DIFFUSE_AND_SPECULAR. При индивидуальном определении параметров материалов для внутренней и внешней стороны в качестве аргумента face используются соответственно константы GL_FRONT и GL_BACK.

Коэффициент резкости бликов – показатель степени зеркального отражения в модели Фонга – задается вызовом функции glMaterialfv(), которой в качестве параметра type передается константа GL_SHININESS:

```
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, 100);
```

Свойства материала являются параметрами режима – их текущие значения ассоциируются со всеми объектами, задаваемыми в программе, до тех пор, пока не будут изменены с помощью функций glMaterilfv() или glMaterialf().

В системе OpenGL можно включить в сцену излучающую поверхность, которая сама по себе является источником света. Со всей такой поверхностью ассоциируется постоянный свет, который задается так же, как и другие свойства материала. Например, для придания такой поверхности сине-зеленого (бирюзового) цвета нужно включить в программу следующий фрагмент:

```
GLfloat emission[]={0.0, 0.3, 0.3, 1.0};
glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, emission);
```

Параметры, устанавливаемые для свойств материалов по умолчанию, приведены в табл. 6.2.

Таблица 6.2

Значения по умолчанию для параметра glMaterial*()

Имя параметра	Значение	Содержание
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive color of material
GL_COLOR_INDEXES	(0,1,1)	ambient, diffuse, and specular color indices

3. ПРИМЕР ПРОГРАММЫ, ИСПОЛЬЗУЮЩЕЙ ИСТОЧНИК СВЕТА И СВОЙСТВА МАТЕРИАЛА

```
#include <GL/glut.h>
#include <stdlib.h>
/*  Initialize material property, light source, lighting model,
 *  and depth buffer.
 */
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
```

```
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_COLOR_MATERIAL);
}
void display(void)
{
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glColor3f(1.0,0.0,0.0);
  glutSolidSphere (1.0, 20, 16);
  glFlush ();
}
void reshape (int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
      1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
  else
    glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
      1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
  }
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

4. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Отладить программу, приведенную в п.3.
2. Отключить все источники света, кроме глобального фонового освещения.
3. Добавить точечный источник света.
4. Превратить точечный источник света в прожектор.
5. Изменить свойства материала в соответствии с вариантом, заданным преподавателем

Контрольные вопросы

1. Цветовые модели, используемые в компьютерной графике. RGB – модель.
2. Методы закрашивания граней объекта Метод Гуро. Метод Фонга.
3. Способы задания свойств источников света в OpenGL.
4. Способы задания свойств материала в OpenGL.

ЗАДАЧА № 7

КРИВЫЕ И ПОВЕРХНОСТИ В OPENGL

Цель работы

Изучение средств OpenGL для изображения кривых и поверхностей

1. КРИВЫЕ БЕЗЬЕ

Кривая Безье задается векторной функцией одной переменной

$$C(u) = [\, X(u), Y(u), Z(u)],$$

где **u** изменяется в некоторой области, например, [0.0, 1.0].

Фрагмент поверхности Безье задается векторной функцией двух переменных

$$S(u, v) = [\, X(u, v), Y(u, v), Z(u, v)\,].$$

Для каждого значения **u** и **v** формула C( ) или S( ) вычисляет точку на кривой (поверхности). При использовании Безье-вычисления сначала выбирают функцию C( ) или S( ), включают ее (Безье-вычислитель), а затем используют команду glEvalCoord1( ) или glEvalCoord2( ) вместо команды glVertex*( ). В этом случае вершина кривой или поверхности может использоваться точно так же, как и любая другая вершина, например, для формирования точки или линии. Кроме того, другие команды автоматически генерируют серии вершин, образующих пространство регулярной однородной сетки по оси **u** (или по осям **u** и **v**).

Если P_i представляет набор контрольных точек, то уравнение $C(u) = \sum_{i=0}^{u} B_i^n(u) \times P_i$ – представляет собой кривую Безье при изменении **u** от 0.0 до 1.0, где $B_i^n(u)$ – многочлен Бернштейна степени n, который задается следующим уравнением

$$B_i^n(u) = (_i^n) \times u^i \times (1-u)^{n-i}$$

В составе OpenGL имеются средства поддержки работы с кривыми и поверхностями Безье – Безье-вычислитель, которые позволяют вычислять значения полиномов Безье любого порядка. Безье вычислитель можно использовать для работы с полиномами от одной, двух, трех и четырех переменных.

Функция обработки полинома одной переменной настраивается в процессе инициализации OpenGL – программы посредством вызова функции

glMap1f(type, u_min, u_max, stride, order, point array)

Аргумент type задает тип объекта, который будет представлен полиномом Безье. Можно назначить в качестве значения этого аргумента константы, задающие трех- и четырехмерные геометрические точки, цвет в формате RGBA, нормали, индексированные цвета и координаты текстур (от одно- до четырехмерных).

Указатель на массив опорных точек полинома передается функции через аргумент point_array. Аргументы u_min, u_max определяют область существования параметра полинома. Аргумент stride представляет собой количество значений параметра между сегментами кривой. Значение аргумента order должно быть равно количеству опорных точек. Для формирования кубической трехмерной кривой в форме B-сплайна, определенной на интервале (0,1), функции glMap1f() следует передать такой набор аргументов:

```
point data[]= {…};
glMap1f(GL_MAP_VERTEX_3, 0.0, 1.0, 3, 4, data);
```

После настройки функция активизируется посредством вызова:

```
glEnable(type);
```

Если функция расчета активизирована, то можно получить от нее значения полинома, вызвав функцию:

```
glEvalCoord1f(u);
```

Таким образом, обращение к glEvalCoord1f() может заменить обращение к функциям glVertex(), glColor(), glNormal(). Пусть, например, функция расчета настроена на формирование кривой Безье на интервале (0,10) по некоторому массиву опорных точек. Набор из 100 точек кривой, равноотстоящих на этом интервале можно получить с помощью такого фрагмента программы:

```
glBegin(GL_LINE_STRIP)
for(i=0; i<100; i++) glEvalCoord1f( (float)i/100.);
glEnd();
```

Если значения параметра u распределены равномерно, то для вычисления точек на кривой следует использовать функции glMapGrid1f() и glEvalMesh1(), например:

```
glMapGrid1f(100, 0.0, 10.0);
glEvalMesh1(GL_LINE, 0, 100);
```

После вызова glMapGrid1f() устанавливается равномерная сетка в 100 отсчетов, а после вызова

функции glEvalMesh1() будет сформирована кривая.

Пример программы вычисления и рисования полинома Безье.

```
#include <GL/glut.h>
#include <stdlib.h>
GLfloat ctrlpoints[4][3] = {
          { -4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},
          {2.0, -4.0, 0.0}, {4.0, 4.0, 0.0}};
void init(void)
{
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glShadeModel(GL_FLAT);
  glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);
  glEnable(GL_MAP1_VERTEX_3);
}
void display(void)
{
  int i;
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_LINE_STRIP);
    for (i = 0; i <= 30; i++)
      glEvalCoord1f((GLfloat) i/30.0);
  glEnd();
  /* The following code displays the control points as dots. */
  glPointSize(5.0);
  glColor3f(1.0, 1.0, 0.0);
  glBegin(GL_POINTS);
    for (i = 0; i < 4; i++)
      glVertex3fv(&ctrlpoints[i][0]);
  glEnd();
  glFlush();
}
void reshape(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho(-5.0, 5.0, -5.0*(GLfloat)h/(GLfloat)w,
            5.0*(GLfloat)h/(GLfloat)w, -5.0, 5.0);
  else
    glOrtho(-5.0*(GLfloat)w/(GLfloat)h,
            5.0*(GLfloat)w/(GLfloat)h, -5.0, 5.0, -5.0, 5.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
  }
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc (keyboard);
  glutMainLoop();
```

```
  return 0;
}
```

2. ПОВЕРХНОСТИ БЕЗЬЕ

Математически фрагмент поверхности Безье задается уравнением

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}B_i^n(u)B_j^m(v)P_{ij},$$

где P_{ij} – представляет собой множество $m\times n$ контрольных точек, а функции B_i – те же самые многочлены Бернштейна, что и для одного измерения. Значения P_{ij} могут представлять вершины, нормали, цвета или текстурные координаты.

Поверхности Безье формируются в OpenGL примерно по той же методике, что и кривые, только роль функции инициализации играет не glMap1*(), а glMap2*(), а для считывания результатов следует обращаться к функции glEvalCoord2*() вместо glEvalCoord1*(). В обеих функциях нужно специфицировать данные, относящиеся к двум независимым параметрам u и v. Например, функция glMap2f() имеет такой формат вызова:

glMap2f(type, u_min, u_max, u_stride, u_order, v_min, v_max, v_stride, v_order, point_array);

Настройка функции вычисления на работу с бикубической поверхностью Безье, определенной на области (0,1)x(0,1), выполняется таким вызовом glMap2f():

glMap2f(GL_MAP_VERTEX_3, 0.0, 1.0, 3, 4, 0.0, 1.0, 12, 4, data);

Для обоих независимых переменных нужно задать порядок полинома (аргументы u_order и v_order) и количество значений параметра между сегментами (аргументы u_stride и v_stride), что обеспечивает дополнительную гибкость при формировании поверхности. Обратите внимание на то, что значение v_stride для второго параметра равно 12, поскольку в массиве опорных точек data данные хранятся по строкам. Поэтому для перехода к следующему элементу этой же строки нужно «перешагнуть» три числа в формате float, а для перехода к следующему элементу в этом же столбце нужно «перешагнуть» через 3*4=12 чисел в формате float. Способ вызова программы расчета зависит от того, какой результат мы хотим получить, – вывести на экран сеть или сформировать многоугольники для последующего раскрашивания. Если ставится задача сформировать на экране сеть, то соответствующий фрагмент программы должен выглядеть примерно так:

```
    for(j=0; j<100; j++)
{
    glBegin(GL_LINE_STRIP);
            for(i=0; i<100; i++)
            glEvalCoord2f((float)i/100.0, (float)j/100.0);
    glEnd();
    glBegin(GL_LINE_STRIP);
            for(i=0; i<100; i++)
            glEvalCoord2f((float)j/100.0, (float)i/100.0);
    glEnd();
    }
```

Если же желательно сформировать множество многоугольников, то фрагмент должен выглядеть так:

```
    for(j=0; j<99; j++)
    {
            glBegin(GL_QUAD_STRIP)
            for(i=0; i<=100; i++)
            {
            glEvalCoord2f( (float)i/100.0, (float)j/100.0);
            glEvalCoord2f( (float)(i+1)/100.0, (float)j/100.0);
            }
            glEnd();
    }
```

Для работы на равномерной сетке следует использовать функции glMapGrid2*() и glEvalMesh2(). Тогда в самое начало программы, в ту часть, которая отвечает за инициализацию, нужно включить такой фрагмент:

glMapGrid2f(100, 0.0, 1.0, 100, 0.0, 1.0);

В функции отображения display() нужно вызвать glEvalMesh2():

glEvalMesh2(GL_FILL, 0, 100, 0, 100);

Для работы алгоритмов тонирования (закрашивания) сформированной поверхности при настройке режима учета освещения нужно дополнительно вызвать функцию glEnable(), передав ей в качестве аргумента константу GL_AUTO_NORMAL:

glEnable(GL_AUTO_NORMAL);

Это позволит OpenGL автоматически вычислять вектор нормали к каждому участку формируемой поверхности и использовать этот вектор при закрашивании участков этой поверхности.

Пример программы аппроксимации с помощью поверхности Безье функции z=sin(x+y):

```
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
GLfloat ctrlpoints[6][6][3];
void initlights(void)
{
  GLfloat ambient[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat position[] = {0.0, 0.0, 2.0, 1.0};
  GLfloat mat_diffuse[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat mat_shininess[] = {50.0};
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
  glLightfv(GL_LIGHT0, GL_POSITION, position);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
}
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glPushMatrix();
  glRotatef(85.0, 1.0, 0.0, 0.0);
        glEvalMesh2(GL_FILL, 0, 10, 0, 10);
  glPopMatrix();
  glFlush();
}
void init(void)
{
        float x,y;
        int i,j;
for (i=0; i<6;i++)
{
        x=3.1415/5.0*(float)i;
        for(j=0; j<6; j++)
        {
                y=3.1415/5.0*(float)j;
                ctrlpoints[i][j][0]=x;
                ctrlpoints[i][j][1]=y;
                ctrlpoints[i][j][2]=sin(x+y);
        }
}
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glEnable(GL_DEPTH_TEST);
  glMap2f(GL_MAP2_VERTEX_3, 0, 4, 3, 4,
        0, 4, 18, 4, &ctrlpoints[0][0][0]);
  glEnable(GL_MAP2_VERTEX_3);
  glEnable(GL_AUTO_NORMAL);
 glMapGrid2f(10, 0.0, 4.0, 10, 0.0, 4.0);
  initlights();       /* for lighted version only */
}
void reshape(int w, int h)
{
  glViewport(0, 0, 400, 400);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
      glOrtho(-1.0,2.0, -1.0, 2.0, -2.0, 2.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
```

```
}
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
  }
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow(argv[0]);
  init();
  glutReshapeFunc(reshape);
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Отладить и запустить программу рисования кривой Безье, приведенную в тексте лабораторной работы.
2. Изобразить кривую Безье на основе данных, заданных преподавателем.
3. Отладить и запустить программу рисования поверхности Безье, приведенную в тексте лабораторной работы.
4. Изобразить поверхность Безье на основе данных, заданных преподавателем.

Контрольные вопросы

1. Математические основы использования бета-сплайнов в компьютерной графике.
2. Интерполяция и аппроксимация с использованием сплайнов.
3. Кривые и поверхности Безье. Математические основы.
4. Построение кривых и поверхностей Безье средствами OpenGL.

ЗАДАЧА № 8

РЕАЛИЗАЦИЯ ВЫБОРА ОБЪЕКТОВ В ИНТЕРАКТИВНОЙ ГРАФИЧЕСКОЙ ПРОГРАММЕ

Цель работы

Изучение механизма выбора OpenGL – средства, реализующего функции логического устройства типа «селектор»

1. ВЫБОР И ОБРАТНАЯ СВЯЗЬ

Некоторые графические прикладные программы просто рисуют статическое изображение двух и трехмерных объектов. Другие приложения позволяют пользователю идентифицировать объект на экране, а затем перемещать, изменять, удалять или еще как-то управлять этими объектами. Графическая система OpenGL предназначена для поддержки таких интерактивных приложений. Для помощи в решении проблемы выбора объекта сцены для перемещения, вращения и других преобразований OpenGL использует механизм выбора, который автоматически сообщает, какие объекты нарисованы внутри указанной области окна.

Обычно OpenGL работает в режиме выбора. Другой режим, который может использоваться для выбора объектов – обратная связь. В режиме обратной связи вы используете ваши графические аппаратные средства и OpenGL для выполнения обычных вычислений сцены, однако вместо использования вычисленных результатов для рисования изображения на экране OpenGL возвращает вам информацию о рисунке.

И в режиме выбора и в режиме обратной связи информация о рисунке возвращается приложению вместо того, чтобы направляться в буфер кадра, как это делается в режиме исполнения. Таким образом, экран остается неизменным пока OpenGL находится в режиме выбора или обратной связи. В лабораторной работе рассматривается только один из указанных механизмов создания интерактивных приложений – режим выбора.

2. ВЫБОР

Как правило, когда Вы планируете использование механизма выбора OpenGL, Вы сначала рисуете вашу сцену в буфер кадра, а затем включаете режим выбора и перерисовываете сцену. Когда вы находитесь в режиме выбора, содержимое буфера кадра не изменяется, пока вы не выйдете из этого режима. Когда вы выходите из режима выбора OpenGL, возвращает список примитивов, которые попадают в видимый объем. Каждый пересекающий (попадающий полностью или частично в видимый объем) примитив, вызывает ответ выбора (selection hit). Список примитивов возвращается в виде ответных записей (hit records), представляющих собой массив целочисленных имен и связанных данных, соответствующих текущему содержанию стека имен. Вы создаете стек имен (stack), загружая в него имена, когда вы используете команды рисования примитивов, находясь в режиме выбора. Таким образом, когда возвращен список имен, вы можете его использовать для определения примитивов, которые могли бы быть выбраны пользователем на экране.

В дополнение к этому механизму выбора OpenGL представляет сервисную программу, предназначенную для упрощения выбора в некоторых случаях, ограничивая рисование маленькой областью окна просмотра. Как правило, эта подпрограмма используется, чтобы определить, какие объекты рисуются в близи курсора.

3. ИСПОЛЬЗОВАНИЕ МЕХАНИЗМА ВЫБОРА В ИНТЕРАКТИВНОЙ ГРАФИЧЕСКОЙ ПРОГРАММЕ

Чтобы использовать механизм выбора, вы должны выполнить следующие шаги:

1. Определить с помощью команды glSelectBuffer массив, который будет использоваться для возвращаемых ответных записей.
2. Войти в режим выбора, задав константу GL_SELECT в команде glRenderMode.
3. Инициализировать стек имен, используя команды glInitNames( ) и glPushName().
4. Определить видимый объем, который вы хотите использовать для выбора. Обычно он отличается от видимого объема, в котором первоначально рисуется сцена. Поэтому, вы вероятно захотите сохранить, а затем восстановить текущее состояние преобразования с помощью команд glPushMatrix( ) и glPopMatrix( ).
5. Поочередно вызывать команды рисования примитивов и команды управления стеком имен, чтобы каждому примитиву, представляющему интерес, было назначено соответствующее имя.
6. Выйти из режима выбора и обработать возвращенные данные выбора (ответные записи).

Пример программы, использующей механизм выбора в интерактивной графической программе:

```
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
void drawTriangle (GLfloat x1, GLfloat y1, GLfloat x2,
   GLfloat y2, GLfloat x3, GLfloat y3, GLfloat z)
{
  glBegin (GL_TRIANGLES);
  glVertex3f (x1, y1, z);
  glVertex3f (x2, y2, z);
  glVertex3f (x3, y3, z);
  glEnd ();
}
{
  glColor3f (1.0, 1.0, 1.0);
  glBegin (GL_LINE_LOOP);
  glVertex3f (x1, y1, -z1);
  glVertex3f (x2, y1, -z1);
  glVertex3f (x2, y2, -z1);
  glVertex3f (x1, y2, -z1);
  glEnd ();
  glBegin (GL_LINE_LOOP);
  glVertex3f (x1, y1, -z2);
  glVertex3f (x2, y1, -z2);
  glVertex3f (x2, y2, -z2);
```

```
  glVertex3f (x1, y2, -z2);
  glEnd ();
  glBegin (GL_LINES);      /*  4 lines          */
  glVertex3f (x1, y1, -z1);
  glVertex3f (x1, y1, -z2);
  glVertex3f (x1, y2, -z1);
  glVertex3f (x1, y2, -z2);
  glVertex3f (x2, y1, -z1);
  glVertex3f (x2, y1, -z2);
  glVertex3f (x2, y2, -z1);
  glVertex3f (x2, y2, -z2);
  glEnd ();
}
void drawScene (void)
{
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  gluPerspective (40.0, 4.0/3.0, 1.0, 100.0);

  glMatrixMode (GL_MODELVIEW);
  glLoadIdentity ();
  gluLookAt (7.5, 7.5, 12.5, 2.5, 2.5, -5.0, 0.0, 1.0, 0.0);
  glColor3f (0.0, 1.0, 0.0);   /*  green triangle           */
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -5.0);
  glColor3f (1.0, 0.0, 0.0);   /*  red triangle  */
  drawTriangle (2.0, 7.0, 3.0, 7.0, 2.5, 8.0, -5.0);
  glColor3f (1.0, 1.0, 0.0);   /*  yellow triangles        */
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, 0.0);
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -10.0);
  drawViewVolume (0.0, 5.0, 0.0, 5.0, 0.0, 10.0);
}
void processHits (GLint hits, GLuint buffer[])
{
  unsigned int i, j;
  GLuint names, *ptr;

  printf («hits = %d\n», hits);
  ptr = (GLuint *) buffer;
  for (i = 0; i < hits; i++) {   /*  for each hit  */
    names = *ptr;
    printf (» number of names for hit = %d\n», names); ptr++;
    printf(»  z1 is %g;", (float) *ptr/0x7fffffff); ptr++;
    printf(» z2 is %g\n», (float) *ptr/0x7fffffff); ptr++;
    printf (»   the name is «);
    for (j = 0; j < names; j++) {        /*  for each name */
      printf ("%d «, *ptr); ptr++;
    }
    printf ("\n»);
  }
}
#define BUFSIZE 512
void selectObjects(void)
{
  GLuint selectBuf[BUFSIZE];
  GLint hits;

  glSelectBuffer (BUFSIZE, selectBuf);
  (void) glRenderMode (GL_SELECT);

  glInitNames();
  glPushName(0);

  glPushMatrix ();
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glOrtho (0.0, 5.0, 0.0, 5.0, 0.0, 10.0);
```

```
  glMatrixMode (GL_MODELVIEW);
  glLoadIdentity ();
  glLoadName(1);
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -5.0);
  glLoadName(2);
  drawTriangle (2.0, 7.0, 3.0, 7.0, 2.5, 8.0, -5.0);
  glLoadName(3);
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, 0.0);
  drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -10.0);
  glPopMatrix ();
  glFlush ();

  hits = glRenderMode (GL_RENDER);
  processHits (hits, selectBuf);
}
void init (void)
{
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_FLAT);
}

void display(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  drawScene ();
  selectObjects ();
  glFlush();
}
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
  }
}
/*  Main Loop  */
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (200, 200);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init();
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Отладить и запустить программу, приведенную в тексте лабораторной работы.
2. Составить интерактивную графическую программу, использующую механизм выбора по заданию преподавателя.

Контрольные вопросы

1. Программистская модель интерактивной машинной графики.
2. Классы логических устройств ввода. Селектор.
3. Реализация логического устройства ввода типа «селектор» средствами OpenGL.