МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Методические указания по выполнению
курсового проекта
по дисциплине

Компьютерная графика

для студентов специальности
231000.62
«Программная инженерия»

Томск – 2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав.кафедрой АОИ,
профессор
_____Ю.П.Ехлаков
"__"_____ 2012 г.

Методические указания по выполнению
курсового проекта
по дисциплине

Компьютерная графика

для студентов специальности

231000.62

«Программная инженерия»

Разработчик
Доцент каф. АОИ
_____Т.О.Перемитина

2012

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Курсовой проект является завершающим этапом в изучении дисциплины "Компьютерная графика". Курсовое проектирование должно способствовать закреплению, углублению и обобщению знаний, полученных студентами за время обучения, и применению этих знаний к решению поставленной задачи.

Курсовой проект выполняется с целью:

– систематизации, закрепления и расширения теоретического материала по математическим и алгоритмическим основам компьютерной графики;

– выработки у студента навыков научно-исследовательской работы;

– систематизации, обобщения и анализа фактического материала по проблемам компьютерной графики;

– приобретения практических знаний составления и реализации математических моделей средствами компьютерной графики.

Процесс выполнения и защиты курсового проекта направлен на формирование следующих общекультурных (ОК) и профессиональных (ПК) компетенций:

– владение культурой мышления, способности к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения (ОК-1);
– готовность к кооперации с коллегами, работе в коллективе (ОК-3);
– способности создания программных интерфейсов (ПК-14);
– понимание концепций и атрибутов качества программного обеспечения (надежности, безопасности, удобства использования), в том числе, роли людей, процессов, методов, инструментов и технологий обеспечения качества (ПК-18).

В процессе курсового проектирования студент должен приобрести и закрепить навыки:

– работы со специальной литературой фундаментального и прикладного характера;
– объекно-ориентированного программирования;
– работы с графической библиотекой OpenGL;
– самостоятельной работы над поставленной задачей;
– оформления пояснительной записки к проделанной работе;
– представлению и защите результатов работы.

Курсовой проект выполняется и защищается в сроки, определенные учебным графиком.

Выполнение курсового проекта предполагает консультационную помощь со стороны преподавателя и творческое развитие студентом темы и разделов курсового проекта.

2. ОБЩИЕ ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ

2.1. Курсовой проект выполняется в соответствии с заданием на курсовой проект.

2.2. Объем работы над поставленной задачей должен быть установлен таким образом, чтобы студент мог выполнить его в течение одного семестра.

2.3. Курсовой проект выполняется в виде пояснительной записки в соответствии с общими требованиями и правилами оформления курсовых и дипломных работ (ОС ТУСУР 6.1-97).

2.4. Работа студентом выполняется самостоятельно. Роль руководителя - постановка задачи, контроль за ходом выполнения курсового проекта студентом и консультативная помощь.

2.5. Общие требования к работе:

• работа должна быть выполнена с применением библиотеки OpenGL;

• работа должна включать интерактивные элементы (как минимум навигацию по сцене);

• программа должна представлять законченный продукт, который может использовать сторонний пользователь.

К проверке представляется полностью протестированное откомпилированное приложение, выполненное в среде Delphi, исходные коды программ и пояснительная записка (не более 20 страниц).

3. РАЗРАБОТКА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Пояснительная записка к проекту должна включать в указанной ниже последовательности:

1) титульный лист (Приложение А);
2) задание на проектирование (Приложение Б);
3) содержание;
4) введение;
5) основную часть;
6) заключение;
7) список литературы;
8) приложения.

3.1 Титульный лист оформляется согласно ОС ТУСУР 6.1-97.

3.2. Задание на проектирование выполняется по форме согласно ОС ТУСУР 6.1-97

3.3. Содержание. В содержании перечисляются заголовки разделов, подразделов, список литературы, приложения и указывают страницы, на которых они начинаются.

3.4. В разделе "Введение" приводится:

– формулировка задачи;

– определение цели;
– описание исходных данных (информация о графических объектах).

3.5. Основная часть проекта должна содержать:
– описание используемой среды реализации;
– описание используемых возможностей графической библиотеки;
– описание возможностей и ограничений программного продукта;
– руководство для пользователей программного продукта.

3.6. Заключение должно содержать краткие выводы о проделанной работе, практическое приложение, перспективы использования результатов разработанного проекта.

3.7. В список литературы входят те источники литературы, на которые есть ссылки в пояснительной записке к курсовому проекту.

3.8. В качестве приложений к пояснительной записке помещают листинги программ и результаты их работы.

Пояснительная записка выполняется **в строгом соответствии** правилам, изложенным в п.3.

4. РЕКОМЕНДАЦИИ И ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

В соответствии с вариантом задания необходимо создать средствами OpenGL статическое или анимированное изображение трехмерной сцены. Студенты, заинтересованные в получении более высокой оценки могут проявить фантазию и сделать модель более реалистичной, используя следующие приемы:

– наложение текстур на объекты;
– присутствие в сцене теней;
– реализация устранения ступенчатости;
– использование зеркальных объектов;
– использование сложных моделей освещения;
– введение в сцену прозрачных объектов.

4.1 Основные возможности OpenGL

OpenGL – это графическая библиотека, которая содержит набор функций для работы с двумерной и трехмерной графикой. OpenGL является стандартной библиотекой в большинстве 32-х разрядных операционных системах. Она при-сутствует во всех версиях Windows. Это означает, что программы, использую-щие OpenGL, могут без больших усилий быть перенесены на разные платфор-мы, что является одним из плюсов этой библиотеки.

OpenGL – универсальная, гибкая, популярная графическая библиотека. Она сочетает многочисленные современные достижения в области компьютерной графики с относительной легкостью изучения. OpenGL остается основным «конкурентом» для таких графических библиотек, как DirectX.

Применение OpenGL сводится к описанию структур данных и вызову функций, которые обрабатывают эти данные.

Основные возможности OpenGL

OpenGL – это графическая библиотека, которая содержит набор функций для работы с двумерной и трехмерной графикой. OpenGL является стандартной библиотекой в большинстве 32-х разрядных операционных системах. Она при-сутствует во всех версиях Windows. Это означает, что программы, использую-щие OpenGL, могут без больших усилий быть перенесены на разные платфор-мы, что является одним из плюсов этой библиотеки.

OpenGL – универсальная, гибкая, популярная графическая библиотека. Она сочетает многочисленные современные достижения в области компьютерной графики с относительной легкостью изучения. OpenGL остается основным «конкурентом» для таких графических библиотек, как DirectX.

Применение OpenGL сводится к описанию структур данных и вызову функций, которые обрабатывают эти данные.

4.2 Визуальные эффекты в OpenGl

Для создания реалистических изображений необходимо определить как свойства самого объекта, так и свойства среды, в которой он находится. Первая группа свойств включает в себя параметры материла, из которого сделан объект, способы нанесения текстуры на его поверхность, степень прозрачности объекта. Ко второй группе можно отнести количество и свойства источников света, уровень прозрачности среды. Все эти свойства можно задавать, используя соответствующие команды OpenGL.

4.2.1 Свойства материала

Для задания параметров текущего материала используются команды:

glMaterial[i f](face, pname: GLenum, param: GLtype);

glMaterial[i f]v(face, pname: GLenum, params: ^GLtype).

С их помощью можно определить рассеянный, диффузный и зеркальный цвета материала, а также цвет степень зеркального отражения и интенсивность излучения света, если объект должен светиться. Какой именно параметр будет определяться значением param, зависит от значения pname:

– **GL_AMBIENT** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют рассеянный цвет материала (цвет материала в тени). Значение по умолчанию: (0.2, 0.2, 0.2, 1.0).

– **GL_DIFFUSE** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет диффузного отражения материала. Значение по умолчанию:(0.8, 0.8, 0.8, 1.0).

– **GL_SPECULAR** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет зеркального отражения материала. Значение по умолчанию: (0.0, 0.0, 0.0, 1.0).

– **GL_SHININESS** параметр params должен содержать одно целое или вещественное значение в диапазоне от 0 до 128, которое определяет степень зеркального отражения материала. Значение по умолчанию: 0.

– **GL_EMISSION** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют интенсивность излучаемого света материала. Значение по умолчанию: (0.0, 0.0, 0.0, 1.0).

– **GL_AMBIENT_AND_DIFFUSE** эквивалентно двум вызовам команды **glMaterial**..() со значением pname **GL_AMBIENT** и **GL_DIFFUSE** и одинаковыми значениями params.

В большинстве моделей учитывается диффузный и зеркальный отраженный свет; первый определяет естественный цвет объекта, а второй - размер и форму бликов на его поверхности.

Параметр face определяет тип граней, для которых задается этот материал и может принимать значения **GL_FRONT**, **GL_BACK** или **GL_FRONT_AND_BACK**.

Если в сцене материалы объектов различаются лишь одним параметром, рекомендуется сначала установить нужный режим, вызвав **glEnable**() с параметром **GL_COLOR_MATERIAL**, а затем использовать команду:

glColorMaterial(face, pname: GLenum), где параметр face имеет аналогичный смысл, а параметр pname может принимать все перечисленные значения. После этого, значения выбранного с помощью pname свойства материала для конкретного объекта (или вершины) устанавливается вызовом команды **glColor**..(), что позволяет избежать вызовов более ресурсоемкой команды **glMaterial**..() и повышает эффективность программы.

4.2.2 Источники света

Добавить в сцену источник света можно с помощью команд:

glLight[i f](light, pname: GLenum, param: GLfloat);

glLight[i f]v(light, pname: GLenum, params: ^GLfloat).

Параметр light однозначно определяет источник, и выбирается из набора специальных символических имен вида **GL_LIGHT*i***, где i должно лежать в диапазоне от 0 до **GL_MAX_LIGHT**, которое не превосходит восьми.

Оставшиеся два параметра имеют аналогичный смысл, что и в команде **glMaterial**..(). Рассмотрим их назначение:

- **GL_SPOT_EXPONENT** параметр param должен содержать целое или вещественное число от 0 до 128, задающее распределение интенсивности света. Этот параметр описывает уровень сфокусированности источника света. Значение по умолчанию: 0 (рассеянный свет).
- **GL_SPOT_CUTOFF** параметр param должен содержать целое или вещественное число между 0 и 90 или равное 180, которое определяет максимальный угол разброса света. Значение этого параметра есть половина угла в вершине конусовидного светового потока, создаваемого источником. Значение по умолчанию: 180 (рассеянный свет).

- **GL_AMBIENT** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет фонового освещения. Значение по умолчанию: (0.0, 0.0, 0.0, 1.0).
- **GL_DIFFUSE** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет диффузного освещения. Значение по умолчанию: (1.0, 1.0, 1.0, 1.0) для **LIGHT0** и (0.0, 0.0, 0.0, 1.0) для остальных.
- **GL_SPECULAR** параметр params должен содержать четыре целых или вещественных значения цветов RGBA, которые определяют цвет зеркального отражения. Значение по умолчанию: (1.0, 1.0, 1.0, 1.0) для **LIGHT0** и (0.0, 0.0, 0.0, 1.0) для остальных.
- **GL_POSITION** параметр params должен содержать четыре целых или вещественных, которые определяют положение источника света. Если значение компоненты w равно 0.0, то источник считается бесконечно удаленным и при расчете освещенности учитывается только направление на точку (x,y,z), в противном случае считается, что источник расположен в точке (x,y,z,w). Значение по умолчанию: (0.0, 0.0, 1.0, 0.0).
- **GL_SPOT_DIRECTION** параметр params должен содержать четыре целых или вещественных числа, которые определяют направление света. Значение по умолчанию: (0.0, 0.0, -1.0, 1.0).

Для использования освещения сначала надо установить соответствующий режим вызовом команды **glEnable**(**GL_LIGHTING**), а затем включить нужный источник командой **glEnable**(**GL_LIGHT*n***).

4.2.3 Модель освещения

В OpenGL используется модель освещения Фонга, в соответствии с которой цвет точки определяется несколькими факторами: свойствами материала и текстуры, величиной нормали в этой точке, а также положением источника света и наблюдателя. Для корректного расчета освещенности в точке надо использовать единичные нормали, однако команды типа **glScale**..(), могут изменять длину нормалей. Чтобы это учитывать, используется уже упоминавшийся режим нормализации нормалей, который включается вызовом команды **glEnable**(**GL_NORMALIZE**).

Для задания глобальных параметров освещения используются команды:

glLightModel[i f](pname, param: GLenum);

glLightModel[i f]v(pname: GLenum, const params: ^GLtype).

Аргумент pname определяет, какой параметр модели освещения будет настраиваться и может принимать следующие значения:

- **GL_LIGHT_MODEL_LOCAL_VIEWER** параметр param должен быть булевским и задает положение наблюдателя. Если он равен FALSE, то направление обзора считается параллельным оси -z, вне зависимости от положения в видовых координатах. Если же он равен TRUE, то наблюдатель находится в начале видовой системы координат. Это может улучшить качество освещения, но усложняет его расчет. Значение по умолчанию: FALSE.

- **GL_LIGHT_MODEL_TWO_SIDE** параметр param должен быть булевским и управляет режимом расчета освещенности как для лицевых, так и для обратных граней. Если он равен FALSE, то освещенность рассчитывается только для лицевых граней. Если же он равен TRUE, расчет проводится и для обратных граней. Значение по умолчанию: FALSE .
- **GL_LIGHT_MODEL_AMBIENT** параметр params должен содержать четыре целых или вещественных числа, которые определяют цвет фонового освещения даже в случае отсутствия определенных источников света. Значение по умолчанию:(0.2, 0.2, 0.2,1.0).

4.2.4 Работа с текстурой

Принятый в OpenGL формат хранения изображений отличается от стандартного формата Windows DIB только тем, что компоненты (R,G,B) для каждой точки хранятся в прямом порядке, а не в обратном и выравнивание задается программистом. При создании образа текстуры в памяти следует учитывать следующие требования:

1. Размеры текстуры, как по горизонтали, так и по вертикали должны представлять собой степени двойки. Это требование накладывается для компактного размещения текстуры в памяти и способствует ее эффективному использованию.Использовать только текстуры с такими размерами конечно неудобно, поэтому перед загрузкой их надо преобразовать. Изменение размеров текстуры проводится с помощью команды **gluScaleImage**(format: GLenum, widthin, heightin: GLint, typein: GLenum, const ^datain, widthout, heightout GLint, typeout: GLenum, ^dataout). Параметры widthin, heightin, widhtout, heightout определяют размеры входного и выходного изображений, а с помощью typein и typeout задается тип элементов массивов, расположенных по адресам datain и dataout. Как и обычно, это может быть тип **GL_UNSIGNED_BYTE**, **GL_SHORT**, **GL_INT** и так далее. Результат своей работы функция заносит в область памяти, на которую указывает параметр dataout.
2. Необходимо предусмотреть случай, когда объект по размерам значительно меньше наносимой на него текстуры. Чем меньше объект, тем меньше должна быть наносимая на него текстура и поэтому вводится понятие уровней детализации текстуры. Каждый уровень детализации задает некоторое изображение, которое является как правило уменьшенной в два раза копией оригинала. Такой подход позволяет улучшить качество нанесения текстуры на объект. Например, для изображения размером $2^m x 2^n$ можно построить max(m,n)+1 уменьшенных изображений, соответствующих различным уровням детализации.

Эти два этапа создания образа текстуры в памяти можно провести с помощью команды **gluBuild2DMipmaps**(target: GLenum, components, width, height: GLint, format, type: GLenum, const ^data), где параметр target должен быть равен **GL_TEXTURE_2D**, components определяет количество цветовых компонент текстуры, которые будут использоваться при ее наложении и может

принимать значения от 1 до 4 (1-только красный,2-красный и alpha, 3-красный, синий, зеленый, 4-все компоненты).

Параметры width, height, data определяют размеры и расположение текстуры соответственно, а format и type имеют аналогичный смысл, что и в команде **gluScaleImage**().

В OpenGL допускается использование одномерных текстур, то есть размера 1xN, однако это всегда надо указывать, используя в качестве значения target константу **GL_TEXTURE_1D**. Существует одномерный аналог рассматриваемой команды- **gluBuild1DMipmaps**(), который отличается от двумерного отсутствием параметра height.

При использовании в сцене нескольких текстур, в OpenGL применяется подход, напоминающий создание списков изображений. Вначале, с помощью команды **glGenTextures**(n^ GLsizei, textures: ^GLuint) надо создать n идентификаторов для используемых текстур, которые будут записаны в массив textures. Перед началом определения свойств очередной текстуры следует вызвать команду **glBindTexture**(target: GLenum, texture: GLuint), 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, то есть делает эту текстуру текущей.

При наложении текстуры надо учитывать случай, когда размеры текстуры отличаются от размеров объекта, на который она накладывается. При этом возможно как растяжение, так и сжатие изображения, и то, как будут проводиться эти преобразования, может серьезно повлиять на качество построенного изображения. Для определения положения точки на текстуре используется параметрическая система координат (s,t), причем значения s и t находятся в отрезке [0,1]. Для изменения различных параметров текстуры применяются команды:

glTexParameter[if](target, pname, param: GLEnum)

glTexParameter[if]v(target, pname:Glenum, params: ^GLEnum)

При этом target имеет аналогичный смысл, что и раньше, pname определяет, какое свойство будем менять, а с помощью param или params устанавливается новое значение. Возможные значения pname:

– **GL_TEXTURE_MIN_FILTER** параметр param определяет функцию, которая будет использоваться для сжатия текстуры. При значении **GL_NEAREST** будет использоваться один (ближайший), а при значении **GL_LINEAR** четыре ближайших элемента текстуры. Значение по умолчанию: **GL_LINEAR**.

– **GL_TEXTURE_MAG_FILTER** параметр param определяет функцию, которая будет использоваться для увеличения (растяжения) текстуры. При значении **GL_NEAREST** будет использоваться один (ближайший), а при

значении **GL_LINEAR** четыре ближайших элемента текстуры. Значение по умолчанию: **GL_LINEAR.**

– **GL_TEXTURE_WRAP_S** параметр param устанавливает значение координаты s, если оно не входит в отрезок [0,1]. При значении **GL_REPEAT** целая часть s отбрасывается, и в результате изображение размножается по поверхности. При значении **GL_CLAMP** используются краевые значения: 0 или 1, что удобно использовать, если на объект накладывается один образ. Значение по умолчанию: **GL_REPEAT**.

– **GL_TEXTURE_WRAP_T** аналогично предыдущему значению, только для координаты t.

Использование режима **GL_NEAREST** значительно повышает скорость наложения текстуры, однако при этом снижается качество, так как в отличие от **GL_LINEAR** интерполяция не производится.

Для того, чтобы определить, как текстура будет взаимодействовать с материалом, из которого сделан объект, используются команды

glTexEnv[i f](target, pname, param: GLenum)

glTexEnv[i f]v(target, pname: Glenum, params: ^GLtype)

Параметр target должен быть равен **GL_TEXTURE_ENV**, а в качестве pname рассмотрим только одно значение **GL_TEXTURE_ENV_MODE**, которое применяется наиболее часто. Параметр param может быть равен:

– **GL_MODULATE** конечный цвет находится как произведение цвета точки на поверхности и цвета соответствующей ей точки на текстуре.

– **GL_REPLACE** в качестве конечного цвета используется цвет точки на текстуре.

– **GL_BLEND** конечный цвет находится как сумма цвета точки на поверхности и цвета соответствующей ей точки на текстуре с учетом их яркости.

Перед нанесением текстуры на объект осталось установить соответствие между точками на поверхности объекта и на самой текстуре. Задавать это соответствие можно двумя методами: отдельно для каждой вершины или сразу для всех вершин, задав параметры специальной функции отображения.

Первый метод реализуется с помощью команд

glTexCoord[1 2 3 4][s i f d](coord: type)

glTexCoord[1 2 3 4][s i f d]v(coord: ^type)

Чаще всего используется команды вида **glTexCoord2.**.(s, t: type), задающие текущие координаты текстуры.

4.2.5 Наложение тумана

Реализация тумана в OpenGL связана с изменением яркости объектов сцены. Для включения тумана необходимо вызвать функцию **glEnable**(**GL_FOG**). Для контроля над туманом существуют функции

glFog[i f](pname: GLenum, param: Type);

glFog[i f]v(pname: GLenum, param: ^Type),

где свойство pname может принимать следующие значения:

– **GL_FOG_MODE** , где значение param может принимать значение: **GL_LINEAR**, **GL_EXP**, **GL_EXP2**, определяя функцию вычисления цвета каждой точки изображения при наложении тумана.

– **GL_FOG_DENSITY**, где значение param определяет плотность тумана.

– **GL_FOG_START** – ближайшая граница тумана.

– **GL_FOG_END** – задняя граница тумана.

– **GL_FOG_COLOR** – параметр param является указателем на массив RGBA, определяющий цветовые составляющие тумана.

4.2.6 Работа с буфером трафарета

В OpenGL существует буфер трафарета, с его помощью реализуются разнообразные эффекты, начиная от простого вырезания одной фигуры из другой до реализации теней, отражений и прочих нетривиальных функций. Трафарет это двумерный массив целочисленных переменных. Каждому пикселю в окне соответствует один элемент массива. Использование буфера трафарета происходит в два этапа. Сначала его заполняют, потом, основываясь на его содержимом, отображают объекты.

Рассмотрим функции библиотеки OpenGL для работы с трафаретом. Тест трафарета разрешается при помощи функции **glEnable** с параметром **GL_STENCIL_TEST**. Очищается буфер трафарета при помощи функции **glClear** с параметром **GL_STENCIL_BUFFER_BIT**. Заполнение буфера трафарета происходит при помощи следующих двух функций:

glStencilFunc(func: GLenum, ref: GLint, mask: GLuint);

glStencilOp(fail, zfail, zpass: GLenum).

Первая функция задает правило, по которому будет определяться, пройден тест трафарета или нет. Переменная func может принимать одно из следующих значений:

- **GL_NEVER** (не проходит);
- **GL_LESS** (проходит, если (ref **and** mask) < (stencil **and** mask);
- **GL_LEQUAL** (проходит, если (ref **and** mask) <= (stencil **and** mask);
- **GL_GREATER** (проходит, если (ref **and** mask) > (stencil **and** mask);
- **GL_GEQUAL** (проходит, если (ref **and** mask) >= (stencil **and** mask);
- **GL_EQUAL** (проходит, если (ref **and** mask) = (stencil **and** mask);
- **GL_NOTEQUAL** (проходит, если (ref **and** mask) <> (stencil **and** mask);
- **GL_ALWAYS** (всегда проходит).

Если тест трафарета не пройден, то фрагмент (пиксели) фигуры не прорисовываются в данном месте, т.е. они не попадают в буфер кадра. Если тест пройден, то фигура рисуется. Вторая функция позволяет задать, как будет инициализироваться буфер трафарета. Параметры ***fail*** (тест трафарета не пройден), ***zfail*** (тест трафарета пройден, Z-буфера – нет) и ***zpass*** (пройдены оба теста, либо буфер глубины не используется) могут принимать одно из следующих значений:

- **GL_KEEP** (сохранить текущее значение в буфере трафарета);
- **GL_ZERO** (обнулить);
- **GL_REPLACE** (заменить на ref);
- **GL_INCR** (увеличить на единицу);
- **GL_DECR** (уменьшить на единицу);
- **GL_INVERT** (поразрядно инвертировать).

4.2.7 Эффект прозрачности

За прозрачность отображаемой информации отвечает четвертая цветовая компонента – Alpha. В OpenGl Alpha-компонента может быть обработана двумя способами. Это может быть вывод изображений с отсечением пикселей, не проходящих определенного порогового значения Alpha, либо наложение одного изображения на другое с использованием значения Alpha как уровня прозрачности выводимого изображения относительно уже находящегося в буфере либо наоборот. Рассмотрим оба способа.

За разрешения проверки порогового уровня Alpha отвечает команда **glEnable(GL_ALPHA_TEST)**. После разрешения проверки, для каждого выводимого пикселя на экране будет выполняться проверка Alpha-компоненты по условию заданному с помощью **glAlphaFunc**(func: GLenum, ref: GLclampf), где ref – содержит некоторое пороговое значение, а func может иметь значение:

- **GL_NEVER** (не проходит);
- **GL_LESS** (проходит, если ref < alpha);
- **GL_LEQUAL** (проходит, если ref <= alpha);
- **GL_GREATER** (проходит, если ref > alpha);
- **GL_GEQUAL** (проходит, если ref >= alpha);
- **GL_EQUAL** (проходит, если ref = alpha);
- **GL_NOTEQUAL** (проходит, если ref <> alpha);
- **GL_ALWAYS** (всегда проходит).

В конечном результате, на экране будут отображены лишь пиксели, прошедшие тест.

Для включения режима отработки прозрачности, нам потребуется команда **glEnable(GL_BLEND)**. Аналогично предыдущему случаю, при включении данного режима в действие вступает функция **glBlendFunc**(sfactor, dfactor: GLenum), где параметры sfactor и dfactor определяют соответственно способ формирования исходного (входного изображения) и конечного (отображаемой сцены) цветов. Всего существует 11 методов вычисления цветовых компонент, все они подробно рассмотрены в файле справочной системы, входящей в комплект Delphi. В данном случае нас интересует лишь два значения – **GL_SRC_ALPHA** для sfactor и **GL_ONE_MINUS_SRC_ALPHA** для dfactor. Этот способ работает при отображении сцены, объекты в которой расположены последовательно приближаясь к наблюдателю. В таком случае, при отображении очередного объекта мы, не изменив прозрачности уже созданной сцены, наложим на нее объект, учтя его Alpha-компоненту.

5. ПОДВЕДЕНИЕ ИТОГОВ И ОРГАНИЗАЦИЯ ЗАЩИТЫ КУРСОВОГО ПРОЕКТА

Подведение итогов подготовки курсового проекта включает следующие этапы:

- сдача курсового проекта на проверку руководителю;
- доработка курсового проекта с учетом замечаний руководителя;
- сдача готового курсового проекта на защиту;
- защита курсового проекта.

Выполненный курсовой проект подписывается студентом и представляется на защиту. Курсовой проект, удовлетворяющий предъявленным требованиям, допускается к защите, о чем руководитель делает запись на титульном листе.

Защита курсового проекта, как правило, должна проводиться публично в присутствии группы.

Руководитель проекта определяет требования к содержанию и продолжительности доклада при защите, устанавливает регламент для оппонентов.

Защита курсового проекта, как правило, состоит в коротком докладе (5-7 мин) студента и ответах на вопросы по существу проекта.

Курсовой проект оценивается по пятибалльной системе. Оценка записывается в ведомость, а положительная оценка ставится в зачетную книжку за подписью руководителя.

Оценка проекта производится с учетом:

- обоснованности и качества расчетов и проектных разработок;
- соблюдения требований к оформлению курсового проекта;
- оригинальности решения задач проектирования (один из основных критериев оценки качества курсового проекта);
- содержания доклада и качества ответов на вопросы.

Курсовой проект оценивается комиссией, которая заслушивает доклад студента о проделанной работе.

Студент должен иметь допуск руководителя к защите.

Во время доклада студент демонстрирует работу своей программы, отвечает на вопросы комиссии.

6. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Порев В. Н. Компьютерная графика. – СПб.: БХВ-Санкт-Петербург, 2004. ISBN 5-94157-139-9.

2. Миронов Б.Г., Миронова Р.С., Пяткина Д.А. Инженерная и компьютерная графика: Учебник для ссузов.- 5-е изд., стереотип. - М.: Высшая школа, 2006. – 333 с. ISBN 5-06-004456-4.

3. Люкшин Б. А. Инженерная и компьютерная графика: учебное пособие. - Томск: ТУСУР, 2007. – 99 с. ISBN 978-5-86889-438-8.

4. Перемитина Т.О. Компьютерная графика: методические указания к выполнению лабора-торных работ для студентов специальности 230102. – Томск: Томский межвузовский центр дистанционного образования, 2007. – 35 с.

5. Перемитина Т.О. Компьютерная графика. Учебное пособие. – Томск: ТМЦДО, 2006. - 134 с.

6. Шатохин А. Е. Компьютерная графика: Учебное пособие. - Томск: ТМЦДО, 2002. - 76 с.

Приложение А

Форма титульного листа курсового проекта

Министерство образования и науки РФ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ТЕМА ПРОЕКТА ПРОПИСНЫМИ БУКВАМИ

Пояснительная записка к курсовому проекту
по дисциплине
"Компьютерная графика"
Код по классификатору

Студент гр. (номер)
(подпись) И.О.Фамилия
(дата)

Руководитель
(должность, ученая
степень, звание)
(подпись) И.О.Фамилия
(дата)

Год

Приложение Б

Форма задания для курсового проекта

Министерство образования и науки РФ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
зав.кафедрой АОИ
профессор
_____Ю.П.Ехлаков
"__" _____ 20__ г.

ЗАДАНИЕ

по курсовому проектированию
дисциплина
"Компьютерная графика"

студенту _____
группа _____ факультет _____
1. Тема проекта _____
_____
2. Срок сдачи студентом законченного проекта _____
3. Исходные данные к проекту _____
_____
_____
4. Содержание пояснительной записки (перечень подлежащих разработке вопросов_____
_____
5. Перечень графического материала _____
_____
6. Дата выдачи задания _____
Руководитель _____
(подпись)

(должность, место работы, фамилия, имя, отчество)

Задание принял к исполнению (дата)
_____(подпись студента)