Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ГРАФИЧЕСКОЕ ТРЕХМЕРНОЕ ПРОГРАММИРОВАНИЕ

Методические указания к курсовой работе и организации
самостоятельной работы для студентов направления
«Программная инженерия»(уровень бакалавриата)

2018

Перемитина Татьяна Олеговна
Графическое трехмерное программирование: Методические указания к курсовой работе и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) / Т.О. Перемитина. – Томск, 2018. – 28 с.

Оглавление

1 Введение

Курсовая работа по дисциплине «Графическое трехмерное программирование» является логическим продолжением изученной ранее дисциплины «Компьютерная графика». Выполнение курсовой работы должно способствовать закреплению, углублению и обобщению знаний, полученных студентами за время обучения, и применению этих знаний к решению поставленной задачи.

Целью дисциплины является изучение математических и алгоритмических основ компьютерной графики, а также освоение средств разработки программного обеспечения для визуализации реалистичных изображений сложных трехмерных сцен.

Задачи дисциплины:

– сформировать взгляд на компьютерную графику как на систематическую научно-практическую деятельность, носящую как теоретический, так и прикладной характер;

– сформировать базовые теоретические понятия, лежащие в основе компьютерной графики, освоить особенности восприятия растровых и векторных изображений;

– дать представление о методах геометрического моделирования;

– научить практическому использованию алгоритмов и методов компьютерной графики при проектировании пользовательских интерфейсов программных систем.

В результате изучения дисциплины студент должен ***знать*** основные принципы компьютерной графики; базовые алгоритмы создания и преобразования двумерных и трехмерных объектов; наиболее распространенные форматы графических файлов.

Уметь разрабатывать графические приложения; пользоваться специальными процедурами и функциями графических библиотек и современными пакетами графических прикладных программ.

Владеть методами создания реалистических трехмерных изображений.

Данные методические указания предназначены для выполнения курсовой и самостоятельной работы по дисциплине «Графическое трехмерное программирование» подготовки бакалавров направления «Программная инженерия».

2 Методические указания к выполнению курсовой работы

2.1 Изучение общих требований к курсовой работе

Цель занятия: ознакомление с требованиями выполнения и защиты курсовой работы.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– ознакомится с данными учебно-методическими указаниями по теме «Изучение общих требований к курсовой работе».

Порядок проведения занятия:

– повторить теоретический материал по теме практической работы;

– ответить на вопросы по теме «Изучение общих требований к курсовой работе занятия» (собеседование).

Учебно-методические указания:

Этап выполнения курсовой работы – от выбора темы до успешной защиты – всегда останется актуальной проблемой для студентов. При написании курсовой работы студент может встретиться с рядом проблем, которые не всегда удается решить вовремя, что в итоге может привести к ухудшению качества проделанной работы или и вовсе невозможности её закончить.

Для разрешения имеющихся вопросов возникает необходимость в постоянных консультациях с руководителем, проходящих в рамках аудиторных занятий в соответствии с расписанием.

Курсовые работы могут быть нескольких направлений, каждое из которых имеет отличительные черты не только в оформлении отчета, но и в ходе выполнения самой работы. Так в Положении по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе выделяют четыре направления курсовых работ - реферативный, расчетно-практический, опытно-экспериментальный и программно-исследовательский. В рамках курсовой работы «Графическое трехмерное программирование» возможны только два направления курсовых работ: программного и программно-исследовательского характера.

Курсовые работы, которые имеют ***программный*** характер, основаны на решении некоторой задачи или проблемы. Примером таких работ могут быть любые реализации трехмерных сцен с использованием методов математического моделирования. Главное отличие этого вида работы в том, что на защите должны быть представлены отчет, содержащий полную информацию о программной реализации и листинг программы.

Согласно положению ТУСУРа целью курсовых работ ***программно-исследовательского*** характера является исследование недокументированных (плохо документированных) функций, параметров и характеристик разработанной другими лицами программы или программного пакета. В курсовой работе по данной дисциплине могут исследоваться возможности определенного пакета для создания трёхмерной компьютерной графики. Для эффективного и разностороннего изучения имеющихся функций пакета студент с его помощью может создать выбранную им трёхмерную модель с дополнительными настройками материала, текстур, освещения и анимации.

Таким образом, основная часть курсовой работы программно-исследовательского характера будет состоять из следующих разделов:

– раздела, содержащего общую известную информацию об исследуемом пакете с описанием его характеристик и возможностей, полученных из литературных источников, включая Интернет, и обоснование выбора для изучения именно данного пакета;

– практической части, содержащей план и основные этапы исследования, обработку, анализ и формулировку полученных результатов в виде исследованных функций и оценки доступных источников информации для ознакомления с возможностями пакета.

Основные этапы выполнения курсовых работ программного и программно-исследовательского характера одинаковы:

1) Согласование темы курсовой работы и разработка технического задания;
2) Анализ предметной области и разработка алгоритма решения поставленной задачи;
3) Программная реализация и тестирование;
4) Оформление пояснительной записки к курсовой работе;

5) Защита курсовой работы.

Работа студентом выполняется ***самостоятельно.*** Роль руководителя - постановка задачи, контроль за ходом выполнения курсовой работы студентом и консультативная помощь.

Общие требования к работе:

– работа должна быть выполнена с применением свободно распространяемых графических библиотек;

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

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

2.2 Согласование темы и разработка технического задания к курсовой работе

Цель занятия: выбор и согласование темы курсовой работы.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– повторить теоретические основы учебного пособия Компьютерная графика: Учебное пособие / Перемитина Т. О. — 2012. 144 с. (https://edu.tusur.ru/publications/5613);

– ознакомится с данными учебно-методическими указаниями теме «Согласование темы и разработка технического задания к курсовой работе».

Порядок проведения занятия:

– повторить теоретический материал по теме практической работы;

– ответить на вопросы по теме «Согласование темы и разработка технического задания к курсовой работе» (собеседование).

Учебно-методические указания:

Этап выбора темы наиболее важен во всем процессе выполнения курсовой работы. Здесь необходимо правильно оценить свои возможности, знание предметной области, выделить наиболее

предпочтительные сферы или же выявить актуальную и корректную проблему.

Тема курсовой работы может быть предложена самим студентом при условии обоснования им ее целесообразности, соответствия содержания работы дисциплине. Выбор темы влияет на весь ход выполнения курсовой и изменение темы в ходе выполнения работы грозит потерей большого количества времени и сил.

Примеры тем курсовой работы:

1. Реализовать задачу трехмерного отсечения для различных объектов: многогранников, круглых тел или их сочетаний. Решить задачу определения взаимного расположения объектов и отсекающего объема. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.
2. Работа с освещением. Разработать программу, осуществляющую имитацию движения луча по поверхности. Программа должна обладать дружественным интерфейсом и предоставлять пользователю возможность влиять на свойства поверхности и луча. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.
3. Построение редактора векторных шрифтов. Разработать редактор, позволяющий формировать символы в векторном представлении, записывать их в файл и использовать для создания текстового файла. Редактор должен иметь соответствующий сервис и режимы работы.
4. Построение трехмерной динамической сцены. Реализация движения тела по заданной траектории. В работе предусмотреть возможность задания траектории облета трехмерного тела и выдачу изображения этого тела с точек траектории, взятых с определенным шагом. Возможны вариации за счет смены траекторий и способа представления тел - со сплошной заливкой или каркасное отображение.
5. Построение изображения тел в различных проекциях. Предусмотреть построение каркасных изображений различных трехмерных геометрических проекций с возможностью изменения точек наблюдения.
6. Построение реалистических изображения с учетом теней. Требуется построить тени для выбранных объектов при расположении источника света на конечном расстоянии от объекта (вне поля зрения).
7. Работа с буфером трафарета. Реализовать трехмерную сцену, содержащую изображение куба с вырезанными (с помощью буфера

трафарета) в гранях отверстиями. Отверстия каждой грани должны быть различными. В каждом отверстии поместить полупрозрачную фигуру, с помощью которой данное отверстие было получено. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.

8. Программная визуализация трехмерной модели лабиринта вместе с путем его прохождения. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.

9. Программная визуализация трехмерной модели картинной галереи. Использовать команды переопределения свойств материала.

10. Реализовать трехмерную сцену, содержащую изображение вращающейся модели Солнечной системы. Реализовать возможность отображения подписи названий планет.

11. Реализация программы – имитатора сложного станкового механизма. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.

12. Изобразить трехмерную сцену, изображающую работающие механические часы. Полученную сцену вращать по таймеру и использовать команды переопределения свойств материала.

13. Реализовать трехмерную сцену имитирующую движение мяча, падающего на пол и отскакивающего от пола. Сначала следует смоделировать отскок в ту же точку, откуда началось падение, затем высота отскока постепенно уменьшается. Число отскоков, скорость движения должны задаваться в программе.

14. Хранители экрана. Создать программу – хранитель экрана (Screen Saver) поддерживающую опции настройки, различающую состояния активного режима и режима конфигурации, осуществляющую выход, если пользователь нажал клавишу или переместил мышь.

15. Программа графического дизайна. Разработать программу для дизайна ландшафта с возможностью моделирования расположения различных видов растительности и других предметов (беседки, фонтаны, скульптуры, осветительные приборы и т.д.). Использовать всевозможные спецэффекты: туман, текстуры и др.

16. Программа графического дизайна. Разработать программу для проектирования кухонной мебели с возможностью изменения текстуры материала и фурнитуры. Программа должна предусматривать просмотр 2D (ортографические проекции) и 3D макетов.

17. Программа графического дизайна. Разработать программу для проектирования офисной мебели с возможностью изменения текстуры материала и фурнитуры. Программа должна предусматривать просмотр 2D (ортографические проекции) и 3D макетов.

18. Программа графического дизайна. Разработать программу для проектирования мебели для гостиной с возможностью изменения текстуры материала и фурнитуры. Программа должна предусматривать просмотр 2D (ортографические проекции) и 3D макетов.

19. Программа графического дизайна. Разработать программу для проектирования мягкой мебели с возможностью изменения текстуры материала и фурнитуры. Программа должна предусматривать просмотр 2D (ортографические проекции) и 3D макетов.

20. Программа графического дизайна. Разработать программу для дизайна интерьера помещений с возможностью размещения различных объектов мебели, бытовой техники и других предметов интерьера. Программа должна предусматривать просмотр 2D (ортографические проекции) и 3D макетов.

После окончательного выбора темы на основании результата анализа задачи составляется техническое задание. Форма задания определяется кафедрой, обеспечивающей руководство курсовой работы. После составления задание согласуется и утверждается (подписывается) руководителем и исполнителем курсовой работы.

2.3 Анализ предметной области и разработка алгоритма решения поставленной задачи

Цель занятия: ознакомление с этапами выполнения курсовой работы.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– повторить теоретические основы учебного пособия Компьютерная графика: Учебное пособие / Перемитина Т. О. — 2012. 144 с. (https://edu.tusur.ru/publications/5613);

– ознакомится с данными учебно-методическими указаниями теме «Анализ предметной области и разработка алгоритма решения поставленной задачи» (собеседование).

Порядок проведения занятия:

– повторить теоретический материал по теме практической работы;

– ответить на вопросы по теме «Анализ предметной области и разработка алгоритма решения поставленной задачи» (собеседование).

Учебно-методические указания:

При выполнении курсовой работы программного характера на данном этапе необходимо ознакомиться с уже существующими алгоритмами решения поставленной задачи и, проанализировав их, предложить новый алгоритм решения поставленной задачи. Следует обосновать выбора алгоритма, расписать его преимущества и недостатки перед остальными и предоставить его детальное описание.

На этом же этапе стоит выбрать среду разработки, в которой будет разрабатываться программа, и обосновать ее выбор.

При выполнении курсовой работы программно-исследовательского характера необходимо провести сравнительный анализ программных пакетов, используемых для создания трехмерной графики. Сравнение может включать в себя описание функциональных возможностей, системные требования, стоимость пакета, наличие документации и обучающих пособий по работе в программе, а также оценку сложности овладения навыками работы в программе (согласно сторонним источникам). В итоге из приведенного перечня доступных программных пакетов выбирается наиболее приемлемый вариант и приводится обоснование принятого решения.

2.4 Программная реализация и тестирование

Цель занятия: ознакомление с требованиями к программной реализации и тестированию приложения.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– повторить теоретические основы учебного пособия Компьютерная графика: Учебное пособие / Перемитина Т. О. — 2012. 144 с. (https://edu.tusur.ru/publications/5613);

– ознакомится с данными учебно-методическими указаниями теме «Программная реализация и тестирование» (собеседование).

Порядок проведения занятия:

– повторить теоретический материал по теме практической работы;

– ответить на вопросы по теме «Программная реализация и тестирование» (собеседование).

Учебно-методические указания:

При выполнении курсовой работы программного характера данный этап представляет реализацию программы, следуя техническому заданию. **Важно:** При написании кода не стоит забывать комментировать важные моменты. При написании отчета по проделанной работе все эти моменты помогут освежить память, а также дополнить отчет.

При выполнении курсовой работы программно-исследовательского характера в ходе данного этапа выполняется исследование функций программного пакета посредством трехмерного моделирования. В процессе выполняются шаги алгоритма, и параллельно с этим изучаются и документируются используемые функции программы. **Важно:** рекомендуется регулярно делать скриншоты различных стадий моделирования и сохранять резервные копии во избежание потери данных.

В соответствии с вариантом задания необходимо создать 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 сложных моделей освещения;
- введение в сцену прозрачных объектов.

Ниже приведено описание реализации всех перечисленных выше приемов повышения реалистичности трехмерной сцены с применением библиотеки OpenGL, изученной ранее в рамках дисциплины «Компьютерная графика».

2.4.1 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 компьютер-ной 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 компьютер-ной 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.

2.4.2 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.

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.

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***).

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).

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.

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.

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).

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.

2.5 Оформление пояснительной записки к курсовой работе

Цель занятия: ознакомление с требованиями к оформлению пояснительной записки к курсовой работе.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– изучить Образовательный стандарт вуза (ОС ТУСУР).

– повторить теоретические основы учебного пособия Компьютерная графика: Учебное пособие / Перемитина Т. О. — 2012. 144 с. (https://edu.tusur.ru/publications/5613);

– ознакомится с данными учебно-методическими указаниями теме «Оформление пояснительной записки к курсовой работе» (собеседование).

Порядок проведения занятия:

– повторить теоретический материал по теме практической работы;

– ответить на вопросы по теме «Оформление пояснительной записки к курсовой работе» (собеседование).

Учебно-методические указания:

Пояснительная записка к курсовой работе должна включать:

- титульный лист;
- задание на курсовую работу;
- содержание;
- введение;
- основную часть;
- заключение;
- список литературы;
- приложения.

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

В разделе «Введение» приводится:

- определение цели;
- формулировка задач;

– описание исходных данных (информация о графических объектах).

Основная часть работы должна содержать:

– описание используемой среды реализации;
– описание используемых возможностей графической библиотеки;
– описание возможностей и ограничений программного продукта;
– руководство для пользователей программного продукта.

Заключение должно содержать краткие выводы о проделанной работе, практическое приложение, перспективы использования результатов курсовой работы.

В список литературы входят те источники литературы, на которые есть ссылки в пояснительной записке к курсовой работе.

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

2.6 Подведение итогов и организация защиты курсовой работы

Цель занятия: ознакомление с требованиями и подготовка к защите курсовой работы.

Рекомендации по подготовке к занятию

– изучить Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе;

– повторить теоретические основы учебного пособия Компьютерная графика: Учебное пособие / Перемитина Т. О. — 2012. 144 с. (https://edu.tusur.ru/publications/5613);

– ознакомится с данными учебно-методическими указаниями теме «Подведение итогов и организация защиты курсовой работы» (собеседование).

Порядок проведения занятия:

– сдача курсовой работы на проверку руководителю;
– доработка курсовой работы с учетом замечаний руководителя;
– сдача готовой курсовой работы на защиту;
– подготовка презентации и доклада;
– защита курсовой работы.

Учебно-методические указания:

Выполненная курсовая работа подписывается студентом и представляется на защиту. Курсовая работа, удовлетворяющая предъявленным требованиям, допускается к защите, о чем руководитель делает запись на титульном листе.

Защита курсовой работы, как правило, должна проводиться публично в присутствии группы.

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

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

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

Оценка работы производится с учетом:

- оригинальности решения поставленных задач (один из основных критериев оценки качества курсовой работы);
- своевременности выполнения всех этапов курсовой работы;
- соблюдения требований к оформлению пояснительной записки к курсовой работе;
- содержания доклада и качества ответов на вопросы.

Студент должен иметь допуск руководителя к защите.

Во время доклада студент демонстрирует работу своей программы.

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Самостоятельная работа студентов рассматривается как вид деятельности, позволяющий целенаправленно формировать и развивать самостоятельность студента как личностное качество при выполнении различных видов заданий и проработке дополнительного учебного материала.

Критериями оценки внеаудиторной самостоятельной работы студентов могут быть:

– уровень развития логического мышления студента (гибкость, рациональность, оригинальность мышления);

– сформированность умений самообразования студента (способность находить, систематизировать и применять информацию из различных источников для решения поставленных задач);

– степень развития коммуникативных умений (умение работать в малых группах, выступать с докладом);

– грамотность в изложении материала;

– сформированность самоконтроля и самооценки.

Самостоятельная работа является важной составляющей в изучении дисциплины и заключается в самостоятельном изучении теоретического материала, подготовки к выполнению контрольных работ, подготовки к практическим занятиям и лабораторным работам.

3.2 Подготовка к собеседованию

Подготовка к собеседованию включает в себя изучение рекомендуемых и дополнительных литературных источников, а также материала, излагаемого преподавателем на лекциях по дисциплине «Компьютерная графика».

Во время собеседования выясняется подготовка студента по обсуждаемой теме, объем специальной литературы, с которой он ознакомился, обсуждение вопросов, не до конца понятых студентами.

3.3 Подготовка презентации и доклада

Доклад – это самостоятельная работа, анализирующая и обобщающая публикации по заданной тематике, предполагающая выработку и обоснование собственной позиции докладчика в отношении рассматриваемых вопросов.

Презентация и доклад должны включать в себя несколько частей:

– Тема работы – ее нужно назвать в начале защиты.

– Актуальность темы – обязательно нужно знать, чем же выбранная тема поможет науке и обществу.

– Цель курсовой работы – цель курсовой работы всегда одна – цель это то, что должно быть достигнуто при успешном выполнении курсовой работы.

– Задачи курсовой работы – задач может быть несколько, они позволяют добиться цели курсовой работы.

– Теоретическая часть работы – немного теоретических положений (необходимо привести самое основное).

– Аналитическая часть работы – что и как исследовалось, какие результаты получены, какие выявлены недостатки или проблемы.

– Практическая часть работы – общая характеристика переложенных алгоритмов, программ и оценка их эффективности.

– Общий вывод по проделанной работе – повторить цель работы и сказать, что цель работы полностью достигнута.

Объем презентации должен составлять не менее 10 слайдов.

При подготовке к докладу необходимо подготовить план выступления, выстроить доклад в краткой лаконичной форме, последовательно, с соблюдением логических связок между фрагментами выступления. Речь должна быть грамотной и внятной. Желательно по время выступления не читать весь текст. Следует продумать взаимосвязь выступления с показом демонстрационных материалов.

Работа студента над докладом-презентацией включает отработку умения самостоятельно обобщать материал и сформулировать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Докладчики должны знать и уметь: сообщать новую информацию; использовать технические средства; хорошо ориентироваться в теме курсовой работы; дискутировать и быстро отвечать на заданные вопросы;

четко выполнять установленный регламент (не более 10 минут); иметь представление о композиционной структуре доклада.

3.4 Рекомендуемая литература

1. Компьютерная графика: Учебное пособие / Перемитина Т. О. - 2012. 144 с. [Электронный ресурс] - Режим доступа: https://edu.tusur.ru/publications/5613.
2. Компьютерная геометрия и графика: Учебное пособие / Буймов Б. А. - 2012. 108 с. [Электронный ресурс] - Режим доступа: https://edu.tusur.ru/publications/2437.
3. Компьютерная графика: Учебное пособие / Люкшин Б. А. - 2012. 127 с. [Электронный ресурс] - Режим доступа: https://edu.tusur.ru/publications/1864.