Методичні вказівки до виконання курсової роботи з дисципліни "Комп'ютерна графіка"/ Уклад. О.Н. Романюк -Вінниця : ВДТУ, 2010. – 59ст.

Приведені основні вимоги до виконання курсової роботи з дисципліни "Комп'ютерна графіка", а також рекомендації по розробці програмних засобів . Методичні вказівки розроблені у відповідності з планом кафедри та програми дисципліни " Комп'ютерна графіка".

Укладач Романюк Олександр Никифорович

Редактор В.О. Дружиніна
Коректор З.В. Поліщук

Відповідальний за випуск зав.каф. Пєтух А.М.

Рецензенти: В.П. Кожемяко , доктор технічних наук, професор
В.І. Клочко, доктор педагогічних наук, професор
Г.А.Рудомін , кандидат технічних наук, доцент
С.М. Білан, кандидат технічних наук, доцент

ЗМІСТ

ВСТУП

Бурхливий розвиток засобів обробки інформації, підвищення рівня автоматизації процесів виробництва і керування приводить до зростання ролі людського чинника. Для ефективного розв'язування задач оператору необхідно подати великий об'єм інформації в зручному для нього виді з заданою точністю в реальному масштабі часу.

У цих умовах однією з основних проблем стає організація ефективної інформаційної взаємодії людини з ЕОМ. Найбільш ємне та наглядне подання великих об’ємів даних забезпечує графічна форма, яка дозволяє провести візуальну оцінку результатів обчислення, внести необхідні корективи , відібрати з запропонованого матеріалу дані для наступної машинної обробки. Комп’ютерна графіка дозволяє суттєво збільшити пропускну спроможність інформаційного каналу, через який здійснюється двосторонній зв'язок користувача і комп’ютера. Роль і значення графічного подання результатів обчислень в промисловості та науково-дослідній практиці неперервно зростає.

Комп’ютерна графіка характеризує новий етап застосування комп’ютерів для обробки інформації і забезпечує не тільки підвищення наочності отриманих результатів, але і можливості вирішення принципово нових задач, як, наприклад, геометричне моделювання, дизайн, мультиплікація, автоматизація проектувальних робіт.

Роль машинної графіки (МГ) як однієї з основних підсистем САПР значна, тому що тільки вона дозволяє в умовах сучасного рівня розвитку обчислювальної техніки реалізувати найбільше прийнятну для проектувальника технологію автоматизованого проектування. Це досягається завдяки забезпеченню звичної для нього графічної форми спілкування із системою як при введенні завдань в ЕОМ, так і при оцінюванні результатів автоматизованого проектування. Саме включення засобів МГ у САПР перетворює ЕОМ дійсно в інструмент проектувальника, значно спрощує йому доступ до знань, закладених в ЕОМ у вигляді автоматичних проектних процедур і довідкових даних, дозволяє автоматизувати виконання трудомістких креслярських і розрахунково- графічних робіт.

Широке застосування методів комп’ютерної графіки для рішення інженерних, економічних, дослідних, конструкторських та дизайнерських задач обумовлює необхідність вивчення сучасними фахівцями основ комп’ютерної графіки.

В навчальному посібнику приведені методичні вказівки до виконання курсової роботи з дисципліни "Комп'ютерна графіка" для студентів бакалаврського напрямку 6.0804 - "Комп'ютерні науки "

1 Основні вимоги до курсової роботи

При виконанні курсової роботи студент повинен продемонструвати вміння застосовувати теоретичні знання і практичні навики при розробці закінченого програмного продукту в галузі комп'ютерної графіки.

Курсова робота є комплексною роботою і її виконання вимагає використання знань, отриманих не тільки в одній конкретній дисципліні, але й у ході попереднього вивчення як фундаментальних і загальноосвітніх дисциплін ("Вища математика", "Фізика", "Інженерна графіка", Основи дискретної математики",), так і загально-професійних дисциплін .

Курсова робота орієнтована на розробку закінченого програмного продукту, який дозволяє формувати реалістичні зображення на екрані монітора . Така спрямованість роботи пов'язана з тим, що алгоритми нижнього рівня студенти досить глибоко вивчають у ході освоєння теоретичного матеріалу та виконання лабораторних робіт . Алгоритми верхнього рівня досить трудомісткі , а програми їх реалізації достатньо об'ємні, що не дозволяє здійснити їхню розробку і налагодження в ході виконання лабораторних робіт.

У ході курсового проектування студенти вирішують задачі, зв'язані з обгрунтуванням і розробкою нових або модифікацією і використанням відомих методів і алгоритмів графічного представлення об'єктів, вибору і обгрунтування структури даних, а також технологічні і ергономічні питання.

1.1 Розрахунково-пояснювальна записка

На захист повинні бути представлені комплекс програм і розрахунково- пояснювальна записка.

Комплекс програм являє собою закінчений програмний продукт, який може настроюватися на конкретне програмно-технічне середовище ЕОМ. Для взаємодії користувача з програмною системою студент розробляє інтерфейс користувача, який включає просте загальноприйняте меню, необхідні розділи як з експлуатації програми, так і з інтерпретації одержаних результатів.

Курсова робота включає такі розділи:

1. Завдання на курсову роботу, затверджене керівником.
2. Анотація
3. Перелік скорочень (за необхідністю)
4. Вступ (коротка характеристика сучасного стану напрямку, до якого має безпосереднє відношення об'єкт проектування)
5. Коротке технічне завдання, розроблене студентом у взаємодії з керівником роботи.
6. Обгрунтування оптимального варіанта вирішення основної задачі (приводяться можливі варіанти розв'язування поставленої задачі і доводиться, що запропонований варіант є оптимальним).

7. Розділи та матеріали, які містять усі необхідні розробки та обгрунтування, що супроводжуються відповідними розрахунками та ілюстраціями, посиланнями на літературні джерела, нормативні документи.
8. Розроблені моделі, алгоритми та програми для розрахунків, аналізу, синтезу чи моделювання на ЕОМ
9. Розрахунок економічних показників.
10. Висновки, в яких аналізуються основні підсумки роботи .
11. Список використаних літературних джерел інформації.
12. Додатки.

У вступі дається огляд і аналіз існуючих програмних систем в обраному напрямку, обгрунтовується необхідність розробки нового комплексу програм. Проводиться аналіз і короткий опис відомих алгоритмів розв'язування поставленої задачі . Обсяг вступу 1-3 сторінок.

На основі зробленого у вступі огляду проводиться вибір і обгрунтування запропонованого алгоритму. При використанні відомого алгоритму варто більш докладно продемонструвати його переваги в порівнянні з іншими відомими алгоритмами, привести його алгоритмічну складність, особливості практичної реалізації, шляхи розв'язування задач, що виникають у ході програмної реалізації.

При розробці нового методу або алгоритму необхідно детально викласти отримані самостійно математичні співвідношення, покладені в основу розв'язування задачі, а також описати запропонований алгоритм. При цьому варто чітко виділити основні етапи роботи алгоритму з вказанням необхідних вихідних даних для його роботи і результатів, які отримані на кожному етапі.

Як правило, вибір алгоритму тісно пов'язаний із використаною структурою даних. Тому необхідно привести обгрунтування методів і алгоритмів апроксимації кривих, поверхонь тіл, граней і т.д. Повинні бути зазначені вихідні дані, за допомогою яких задаються відображувані об'єкти, проведено порівняння з іншими можливими способами їхнього задання. При цьому варто проаналізувати надлишковість обраного способу опису об'єктів, показати переваги при оперуванні цими даними.

На основі обраних даних для представлення об'єктів студент здійснює наступний вибір і обгрунтування типів і структур для їхнього машинного представлення. При їхньому виборі варто виходити з можливостей мови програмування для їхнього представлення, витрат пам'яті на зберігання, часу на обробку, розмірності даних.

Вибір вихідних даних і форми їхнього представлення повинний враховувати таку характеристику, як обсяг і зручність користувача при введенні. Зокрема, введення великої кількості даних стомлює користувача і збільшує імовірність введення помилкових даних. У даній частині записки можуть виконуватися розрахунки для визначення об'ємів пам'яті,

необхідної для збереження вихідних даних, проміжних і остаточних результатів, а також розрахунки, що дозволяють оцінити час розв'язування задачі на ЕОМ.

Результати таких розрахунків повинні використовуватися при порівнянні альтернативних варіантів алгоритмів, а також для оцінки можливості практичної реалізації поставленої задачі на наявній технічній базі.

В розрахунково-пояснювальній записці необхідно відобразити обгрунтування технології виготовлення комплексу програм: модульна або об'єктно- орієнтована. Необхідно представити модульну структуру комплексу, обгрунтування обраного принципу розбиття програм на модулі, призначення, взаємозв'язок з іншими складовими частинами кожного модуля. У випадку використання об'єктно-орієнтованого програмування варто обгрунтувати і описати введені класи об'єктів. У цьому ж розділу вирішується питання вибору і обгрунтуванням мови програмування. Велика увага тут повинна бути приділена розробці інтерфейсу користувача, вибору меню, яке у повній мірі відповідало б характеру роботи спроектованого комплексу програм і було зручним і зрозумілим користувачу.

Опис програм повинний закінчуватися викладом керівництва програміста, у якому наводяться вимоги до апаратних засобів і програмного забезпечення ЕОМ, а також викладається порядок роботи з комплексом програм. Ця частина повинна містити такі розділи:

- призначення та умови застосування програми;
- характеристики програми;
- звертання до програми;
- вхідні і вихідні дані;
- повідомлення.

У розділі "Призначення та умови застосування програми" повинні бути описані призначення і основні функції, які виконуються програмою, умови, необхідні для виконання програми (об'єм оперативної пам'яті, вимоги до складу і параметрів периферійних пристроїв, вимоги до програмного забезпечення).

У розділі "Характеристики програми" доводиться опис основних характеристик і особливостей програми (режим роботи, засоби контролю правильності виконання і самовідновлення програми).

У розділі "Звертання до програми" повинен бути приведений опис процедур виклику програми (способи передачі управлінням і параметрів даних).

У розділі " Вихідні дані" повинен бути приведений опис вхідної і вихідної інформації, яка використовується програмою.

У розділі "Повідомлення" повинні бути зазначені тексти повідомлень , які видаються програмою в ході її виконання, описаний їх зміст і дії, які необхідно почати за цими повідомленнями.

У додатках до керівництва програміста можуть приводитися додаткові матеріали (приклади, ілюстрації, таблиці, графіки).

В розрахунково-пояснювальній записці необхідно привести тести для перевірки правильності роботи комплексу програм, результати тестування на тестових прикладах.

Дослідницько-експериментальний розділ є рекомендованою части - ною курсової роботи. Він повинен містити результати теоретичного або експериментального дослідження в ході виконання курсової роботи. У першому випадку це можуть бути результати, отримані при дослідженні математичного методу, покладеного в основу алгоритму. В другому випадку експериментально досліджується розроблений комплекс програм із метою отримання значень тимчасових, об'ємних і інших характеристик комплексу програм (алгоритму) в залежності від кількості об'єктів, що зображуються, їхньої складності, точності представлення (виду апроксимації, кількості граней, які апроксимують криволінійну поверхню і т.д.).

У додатку приводять лістинги програм пакета або його найбільш суттєвих складових частин. Бажано привести тверді копії зображень, одержуваних на екрані монітора і виведені потім на принтер. При наявності аналітичних результатів (у вигляді числових величин) приводять також і їхні роздруківки, графіки, діаграми.

Всі розділи роботи повинні бути пов'язані між собою і являти собою єдине закінчене ціле.

Матеріал записки повинен викладатися грамотною технічною мовою, бути оформлений відповідно до вимог ЄСКД, ДЕРЖСТАНДАРТІВ.

1.2.Захист курсової роботи

Захист курсової роботи підбиває підсумок усієї роботи студента протягом семестру. Перед захистом роботи студенти повинні завчасно проінсталювати на ПЭВМ розроблені програмні продукти. На початку захисту студент робить доповідь із викладом суті пророблено роботи. Для ілюстрації основних положень він використовує графічний матеріал. Після цього, як правило, задають питання члени комісії, на які студенту потрібно відповісти. Друга частина захисту полягає в демонстрації комплексу програм. При цьому необхідно пояснити правила взаємодії користувача з програмою, продемонструвати на заздалегідь підготовлених прикладах характерні особливості реалізованого методу (алгоритму). Потім можуть бути задані питання з практичної частини.

Доповідь повинна бути короткою (3-5 хвилин), чіткою і ясною. У доповіді повинні бути виділені основні задачі, які стояли при виконанні роботи, обгрунтовані шляхи їхнього розв'язування і пояснені отримані результати. Не варто впадати в зайву деталізацію, зупинятися на

другорядних моментах. Члени комісії можуть з'ясувати деякі деталі роботи шляхом постановки відповідних питань. Закінчуватися доповідь повинна висновками з виробленої роботи.

Оцінка курсової роботи складається з ряду показників, серед яких можна виділити:

1) якість, глибину проробки теми, відповідність роботи поставленому технічному завданню;

2) якість, об'єм програмного продукту, зручність його експлуатації;

3) якість доповіді, правильність відповідей на питання.

2 Характеристика пакетів прикладних програм комп'ютерної графіки

В галузі комп'ютерної графіки розроблений ряд високопродуктивних програмних засобів, вибір яких визначається поставленою задачею та областю застосування. В даному розділі приведені основні характеристики найбільш поширених пакетів прикладних програм, що дасть можливість вибрати найбільш доцільний для розв'язування поставленої задачі.

2.1 PhotoImpact

Пакет прикладних програм PhotoImpact 6 від Ulead -програмний продукт для формування Web-сторінок, графічних об'єктів для Web і редагування зображень. Ефективні і різноманітні інструменти редагування зображень дозволяють спростити використання ефектів освітленості, зміни кольорів, фокусування, а також усувати проблеми, із якими стикалися раніше користувачі при опрацюванні зображень, отриманих за допомогою цифрових камер і сканерів. Тепер за допомогою інструментів для ретушування можна дуже просто усувати «негативні ефекти» від подряпин і бруду, що залишилися після сканування зображення.

PhotoImpact 6 дозволяє без програмування створювати Web-сторінки, що містять зображення, тексти й інші складові компоненти. Завдяки тому, що Web-сторінки можуть бути збережені у власному форматі UFO, їх можна змінювати і відображати в будь-який час. Гнучкі інструменти для тексту дозволяють легко створювати текст для графічних об'єктів і постачати його різними мультимедійними можливостями (звуковими й анімаційними ефектами).

Можливість підтримки векторної графіки (4 інструменти) дозволяє формувати 2D і 3D-зображення , а за рахунок збільшеної продуктивності програма дозволяє працювати з великою кількістю графічних файлів із максимальною швидкістю. Панель EasyPalette містить велику кількість доступних текстур і бібліотек, Quick Command Panel дозволяє створювати каталог виконаних задач або процедур.

Зручна функція керування фотографіями в альбомі надає можливість організації файлів, що дозволяє виконувати їхній пошук набагато простіше і легше, ніж це було реалізовано в попередній версії..

У PhotoImpact 6 можна створювати маску безпосередньо на зображенні, використовуючи режим маски. Користувач може створювати маску безпосередньо на зображенні, використовуючи майже будь-який інструмент роботи з кольором. Знову створювані об'єкти можуть бути вільно розміщені на зображенні. За замовчуванням, об'єкти позначені зеленим кольором.

Якщо документ буде збережений у форматі UFO, то при його відкритті всі об'єкти залишаються над фоном або базовим зображенням і кожний із них може бути відредагований і переміщений.

У PhotoImpact 6 Web-об'єкти відіграють спеціальну роль на Web-сторінці. Вони можуть функціонувати як навігаційні пристрої або як зображення-гіперпосилання. Є можливість зазначити, які об'єкти є об'єктами Web, за допомогою кольору.

Складові компоненти можуть використовуватися як навігаційні або інші типові елементи Web-сторінки. Component Designer (у меню Web) містить великий набір заздалегідь заданих елементів, що можна настроювати і використовувати при створенні Web-сторінки.

Анімації можна легко формувати , застосовуючи деякі спеціальні ефекти (Animation Studio, Lighting і Type Effect).

Наявний оптимізатор контролює стискання зображень так, щоб розміри файлів були прийнятними для їхнього використання в Web-сторінках. Файли можуть бути оптимізовані в одному із трьох стандартних форматів: JPG, GIF і PNG.

Для відновлення програми і завантаження додаткових модулів можна використовувати модуль Smart Download.

2.2 Adobe Illustrator

Adobe Illustrator вважається одним із головних інструментів для формування векторних ілюстрацій

Він призначений для формування ілюстрацій і орієнтований на виведення готових зображень із високим розподіленням . Пакет дозволяє створювати фігури і символи довільної форми, а потім масштабувати, повертати і деформувати їх. Крім того, Illustrator містить широкий спектр інструментів для роботи з текстом і багатосторінковими документами.

В Adobe Illustrator реалізовані потужні засоби керування шарами, що дозволяє маніпулювати ієрархіями шарів, допоміжними шарами і групами об'єктів. Будь-який об'єкт можна наділити прозорістю або визначити як маску шару; при цьому до розташованих нижче об'єктів застосовуються різні рівні прозорості. Однак слід відзначити, що атрибути об'єктів, у тому числі заливання і спеціальні ефекти, фіксуються в окремій палітрі Appearance .

До векторних об'єктів можна застосовувати такі "растрові" за своєю природою функції, як прозорість, режими побудови колірних переходів і сумісні з Adobe Photoshop фільтри. При цьому всі аспекти зображення залишаються цілком редагованими. На жаль, застосування будь-якого складного ефекту в реальному часі істотно знижує швидкість перерисовування екрана.

Реалізовані в Illustrator засоби підготування Web-графіки достатньо обмежені . Можна настроїти палітру користувача відповідно до кольорів палітри Web-safe. У режимі оптимізації Illustrator дозволяє попередньо переглядати одночасно виведені на екран здвоєні або зчетверені зображення, щоб порівнювати результати застосування різних установок параметрів. Передбачено також можливість формування "активних точок" для переходів на URL-адреси, але засоби фрагментування зображень і побудови керованих кнопок, які трансформуються, у програмі відсутні.

У пакетах Illustrator і Photoshop застосовується той самий механізм керування кольором (він потрібний для того, щоб домагатися точної відповідності кольорів видрукуваного і відображуваного на екрані зображення.

2.3 Canvas

Пакет Canvas 7 фірми Deneba Software дозволяє працювати з будь-яким типом графічного зображення в рамках єдиного інтерфейсу. Відмінна риса підходу розробників Canvas полягає в тому, що їхній виріб забезпечує повну можливість редагування всіх графічних елементів - ефектів, ілюстрацій, тексту і растровых зображень.

Завдяки унікальному інтерфейсу програми Canvas екран не захаращується непотрібними деталями, а користувач одержує швидкий доступ до необхідних інструментів. Canvas дає користувачам потужні засоби роботи, але, на жаль, для цього пакета характерний цілий ряд прикрих недоліків і обмежень. Наприклад, у деяких елементах керування єдиною припустимою одиницею вимірювання є піксел. У програмі Canvas використовується метафора макета документа: в ілюстраціях, публікаціях, анімаціях можуть застосовуватися як растрові, так і векторні об'єкти. Палітра Document Layout допомагає організовувати сторінки, шари та об'єкти.

Редагування елементів забезпечується за допомогою засобу Sprite Effects, який дозволяє застосовувати до векторних і растрових зображень, а також до тексту різні фільтри (у тому числі більшість Photoshop-сумісних зовнішніх модулів). На жаль, число включених у пакет фільтрів обмежено.

Робота з растровими зображеннями достатньо проста. Коли за допомогою векторного способу обробки прозорих об'єктів користувач формує градієнтну маску, її характеристики автоматично зберігаються в альфа-каналі..

Засоби для роботи з векторною графікою дозволяють підготовляти як художні, так і технічні ілюстрації. Так, зручний інструмент Push дає можливість легко редагувати криві Безьє. Програма Canvas забезпечує формування CAD-файлів. Але в ряді функцій органи керування розраховані на введення даних тільки в пікселах, навіть якщо в настроюванні документа задані інші одиниці вимірювання. Засоби керування текстом дозволяють використовувати прийоми макетування на "типографському" рівні.

У пакеті Canvas реалізовані лише базові способи підготування Web-графіки, у числі яких - функції одночасного відображення на екрані декількох варіантів того самого зображення у форматах GIF і JPEG для попередньої оцінки результатів оптимізації.

2.4 Adobe Photoshop

Програма Adobe Photoshop - професійний графічний редактор, орієнтований на широке коло задач. Можлива робота як із растровими зображеннями, так і з векторними об'єктами. В даний час ця програма є практично стандартним інструментом в арсеналі дизайнера, художника, оформлювача і других користувачів, які працюють в областях, пов'язаних з обробкою зображень.

Основне призначення програми Adobe Photoshop - формування та обробка зображень. Сюди включаються рисування, ретушування, зміна яскравості і контрастності, комбінування зображень і т.д. Як і будь-яка інша програма для роботи з зображеннями, дозволяє як завантажувати готові зображення, так і створювати нові. Редагування раніше створених зображень виконується дуже легко. Можна ввести ілюстрацію в комп'ютер за допомогою сканера або цифрової фотокамери. Можливо імпортувати графічний файл, отриманий за допомогою іншої програми. Інструменти Adobe Photoshop дозволяють легко виконати ретушування і відновлення фотографій. Змінюючи яскравість, контрастність і насиченість, коректуючи кольори, забираючи порошини і подряпини, ви можете істотно поліпшити погані фотознімки.

Обравши один із множини інструментів: олівець, пензель, ручку, аэрограф або якийсь інший, мишею або графічною ручкою ви наносите мазки на створюваний рисунок, як у традиційній техніці рисування. Множина спеціальних ефектів дозволяє додати зображенню або його частині оригінальний вид.

При складному редагуванні неможливо обійтися без засобів виділення. Різноманіття інструментів роботи з виділенням дозволяє виділяти будь-які, найскладніші фрагменти, змінювати виділення, зберігати його і завантажувати в наступній роботі.

Відмінною рисою програми Adobe Photoshop є робота із шарами. Шар - це окреме зображення, яке підлягає редагуванню. Зображень у процесі

роботи може бути декілька .Основна ідея полягає в тому, що результувальне зображення - це накладення всіх наявних шарів один на одного в тому порядку, який вказує розробник. Можна змінювати прозорість кожного із шарів і в такий спосіб домагатися потрібного результату. Програма Photoshop допускає використання 8 тис. шарів, а при роботі над складними проектами ви можете виділяти шари кольором і групувати їх. Кінцевим етапом роботи є об'єднання всіх шарів у єдине зображення.

Більшість функцій, які забезпечують підготування Web-графіки, реалізовано в модулі Adobe ImageReady. У числі цих функцій - потужний засіб фрагментування зображень, прекрасні інструменти експорту й оптимізації, а також об'єктів, за допомогою котрих можна швидко будувати професійно оформлені кнопки. Як і в пакеті Fireworks, у ImageReady передбачена можливість виділення найбільше важливих ділянок будь-якого зображення, щоб їхні ключові фрагменти (наприклад, текст) не піддавалися спотворенням при скороченні розміру файла. Водночас серйозний недолік полягає в тому, що Photoshop і ImageReady мають різні інтерфейси.

Photoshop цілком успішно справляється з підготуванням графічних матеріалів до типографської преси і до роздруківки на принтері. Завдяки реалізованим у цій програмі добротним засобам керування кольором видрукуване на принтері зображення буде точно відповідати тому, що ви бачите на екрані.

Виведення зображень із векторними текстом і формами можливе у популярному форматі Adobe Portable Document Format (PDF) або безпосередньо з Photoshop 6.0 на PostScript-принтери.

Програма дозволяє формувати ілюстрації, які включають доступні для редагування векторні форми і тексти на зображеннях і забезпечує просте рисування прямокутників, прямокутників з округленими кутами, еліпсів і багатокутників.

Для тексту можливо застосування додаткових опцій, включаючи перехід на новий рядок, вирівнювання в абзаці, повне вирівнювання текстового блока, розставлення переносів (тільки для англійської мови) і автоматичне визначення кінця рядка.

Для швидкого створення Web-графіки пропонуються удосконалені функції нової версії ImageReady. Швидкий перехід між ImageReady 3.0 і Photoshop 6.0 і автоматичне зберігання файла спрощують процес створення Web-графіки. Після формування зображень безпосередньо в Photoshop до них можна застосовувати операції стиснення. Для створення кнопок і інших Web-елементів спрощено використання інструментів векторної графіки для форм і стилів шарів. У палітрі Styles програми ImageReady з'явилася можливість зберігання rollover-графіки. Також можна скористатися динамічними секторами на основі шарів, які змінюються автоматично при модифікації вмісту шару.

Photoshop 6.0 пропонує інтуїтивний інтерфейс користувача для швидкого освоєння можливостей програми. Тепер можна збирати часто використовувані палітри в Palette Well і швидко вибирати інструменти з контекстно-залежної панелі. Крім того, доступний швидкий пошук користувальних пензлів, градієнтів, форм і інших інструментів за допомогою нового диспетчера попередніх настроювань (Preset Manager).

Завдяки можливості швидкого маскування (інструмент Quick Mask) можна створювати і переглядати маски безпосередньо на зображенні. У інтерактивному режимі можна швидко спотворювати або деформувати зображення при переміщенні пікселей за допомогою нової команди Liquify. У зображеннях можна застосовувати привабливі ефекти, що використовують більше ніж 95 спеціальних фільтрів, включаючи збільшення різкості зображення, зм'якшення, стилізацію, перекручування, видалення пилу, освітлення й інші.

Упорядкування зображень із множини елементів на сотнях шарів із розширеними опціями керування дозволяє організовувати шари в набори. При цьому можна виконувати злиття шарів, їхнє дзеркальне відображення, копіювання, вирівнювання, розподіл і блокування, а також копіювання ефектів з одного шару в інший у межах списку палітри. У розпорядженні користувачів є доступні для редагування градієнти, візерунки, кольори, а також колірна корекція вмісту шарів.

Настроювання різних параметрів шарів, включаючи опції спеціальних ефектів, насиченості і прозорості, виконується з одного діалогового вікна, для зберігання параметрів у вигляді стилів шарів і їхнього безпосереднього застосування здійснюються з нової палітри Styles.

2.5 CorelDRAW

Основним поняттям у CorelDRAW як і в будь-якому іншому редакторі векторної графіки, є поняття об'єкта. Робота над будь-якою ілюстрацією полягає в формуванні об'єктів, їхньому редагуванні і розташуванні в потрібних місцях. При цьому спочатку створюється приблизна форма об'єктів, після чого форма уточнюється шляхом додавання, видалення і переміщення вузлів контуру. Після формування необхідної форми об'єкта задається колір контуру і вибирається заливання об'єкта.

Створити в редакторі можна як стандартні фігури: прямокутники, еліпси, багатокутники, автофігури, спіралі і решітки; так і довільної форми, які складаються із прямих і кривих ліній. Серед стандартних фігур є досить складні рисунки. Засобами роботи з текстом CorelDRAW можна по можливостях наблизитися до текстового редактора. Багаті можливості форматування дозволяють прямо в редакторі створювати невеликі текстові документи, оформлені рисунками. Застосування оригінальних ефектів дає можливість сформувати реалістичний рисунок із простих об'єктів.

Кожний рисунок, створений у редакторі, складається з одного або декількох об'єктів, які можуть накладатися і цілком або частково закривати один одного. У якості об'єктів можуть використовуватися растрові рисунки, підготовлені раніше за допомогою будь-якого редактора растрової графіки, імпортовані в CorelDRAW. Тому що в постачання CorelDRAW включається набір готових картинок, який називається clipart . Часто робота над ілюстрацією зводиться до вставки декількох готових рисунків, їхнього невеликого редагування і додавання тексту.

Основні підходи до роботи з CorelDRAW.

- Формування простих геометричних фігур або довільних кривих і ламаних, замкнутих і розімкнутих. Вставка і форматування тексту.
- Редагування будь-якого об'єкта, зміна кольору контуру і зафарбовування , зміна форми об'єкта.
- Вставка готових зображень або раніше створених ілюстрацій у документ.
- Застосування різноманітних художніх ефектів.
- Розміщення всіх об'єктів у потрібних місцях, визначення порядку взаємного перекриття об'єктів.

Пакетом CorelDRAW 10 Graphics Suite користувачам пропонуються широкі можливості формування векторних і растрових зображень, а також анімацій. Основу цього графічного пакета складають три програми - CorelDRAW 10 (макетування сторінок і створення ілюстрацій), PHOTO-PAINT 10 (редагування зображень і рисування) і R.A.V.E. (формування анімаційних ефектів).

Попередній перегляд і оптимізація зображень для впровадження в HTML-код за допомогою фільтрів для файлів форматів JPG, GIF і PNG полегшують публікацію графіки в Web, а новий оптимізатор зменшує розмір графічного файла і автоматично встановлює параметри для більш швидкого завантаження графічних об'єктів.

Інтегрований механізм попереджень забезпечує корисний зворотний зв'язок дизайнера і створюваного ним документа. У процесі підготування документа з'являються попередження про можливі проблеми, що дозволяє створювати досконалі документи, а також зберігати попередні стилі для загального потоку документів. Необхідні настроювання механізму попереджень виконуються програмою Preflight, яка доступна в декількох діалогових вікнах - Print, Flash Export, Publish to PDF і Publish to the Web.

При формуванні Web-графіки можливий попередній перегляд кожного графічного символу і всієї Web-сторінки за допомогою Web-браузера. Крім того, можна створювати rollover-графіку і застосовувати різні стани (Normal, Down і Over) у відповідь на переміщення миші і тактильні дії.

Зображення можна експортувати в різні графічні формати , а rollover-графіку - у формат SWF. Можна імпортувати файли з Macromedia

FreeHand 8 і Microsoft Visio 5, а також за допомогою фільтрів працювати з документами AutoCAD .

За допомогою альбомного режиму можна переглядати всі сторінки документа і швидко переміщати їх. Крім того, можливо розміщати символи кількома мовами в одному текстовому полі, а засоби перевірки правопису і граматики дозволяють автоматично виправляти текст, використовуючи зазначену мову.

Робочий простір CorelDRAW 10 можна пересилати по електронній пошті.

Користувачі можуть швидко переглядати і застосовувати параметри інтерактивних інструментів. Додатково можна створювати установки для інструментів , які найчастіше викликаються, що дозволить заощадити час при багатократному застосуванні того самого ефекту.

2.6 3D Studio Max Release 2

Версія 3D Studio VIZ - орієнтована в основному на проектні задачі. Частина функцій, що існують у 3D Studio MAX 1.x були виключені, але включені нові засоби, що істотно збільшують продуктивність, об'єктні прив'язки в стилі "AutoCAD", CAD-подібні засоби роботи зі сплайнами, зручна робота з текстурами, суміщення об'єктів сцени з растровою фоновою підкладкою , реалізація умов сонячного освітлення, засоби контролю над масштабом текстур і матеріалів, що привласнюються об'єкту , підтримка файлів у форматі DWG. 3D Studio Max R2 містить усі засоби, що і 3D Studio VIZ плюс ряд додаткових можливостей (усього додано біля 1000 нових функцій).

Виборче променеве трасування (Selective Raytracing) - це опція скануючого рендеринга . Вона може бути привласнена окремим об'єктам сцени для відтворення ефектів відбиття, переломлення і геометрично точних тіней, які враховують ступінь прозорості об'єкта. Новий варіант включає ефект розфокусування, пов'язаний із далекістю об'єкта.

Фізичні взаємодії (Integrated Dynamics). Цей інструмент використовує закони фізики жорстких взаємодій для відтворення динамічних взаємодій об'єктів сцени без урахування деформації.

NURBS - моделювання. NURBS - поверхні є особливим типом сплайнових поверхонь, які дозволяють дуже точно створити і анімувати плавні вигини, характерні для тканин або шкірних покривів тварин. NURBS- поверхні дають плавні безшовні з'єднання між собою або з полігональними об'єктами. Надзвичайно точно відтворюються при рендеринзі. Дуже легко управляються та анімуються.

Складні системи частинок у новій версії одержали подальший розвиток і тепер системи частинок можуть використовуватися для створення ефектів полум'я, води, що тече, бджолиного рою і т.д.

Оптичні ефекти лінз - за допомогою цього засобу можна добавляти в

сцену різні світіння, спалахи та інші візуальні ефекти, пов'язані з горінням або вибухом.

Camera Matching (суміщення камери) - це спосіб дозволяє розташувати і вирівняти камеру в сцені відповідно до умов натурної зйомки фото зображення або послідовності кінокадрів.

Camera Mapping - карта фронтальної проекції. Дозволяє вирівняти растрове зображення по видовій проекції об'єкта для точного вирівнювання текстур або фотозображень об'єктів по об'єктах сцени. Макропрограмування. . MAXScript - вбудована мова, яку має програма. Досить легкий синтаксис. Існує, щоб оптимизувати часто повторювані дії. Є набір готових макросів. Макроси дозволяють управляти всіма параметрами об'єктів і, крім того, забезпечують зв'язок із зовнішніми джерелами даних, наприклад, із таблицями Excel.

Weighted morphing - поліпшений морфінг. Дозволяє використовувати множину цілей і в такий спосіб створювати складні ефекти, у тому числі: міміку персонажа. При цьому використовуються ключі впливу, що об'єднують усі цілі морфінга.

Промислові стандарти лінз - відтворюють оптичні властивості стандартних промислових виробів. Потрібний формат вибирається зі списку.Крім того, можливе ручне настроювання.

Drag-and-Drop дозволяє істотно підвищити продуктивність за рахунок зменшення кількості маніпуляцій.

Незалежні фонові зображення - програма дозволяє призначати фонові зображення видовим екранам незалежно від інших. Таким чином, дозволяє призначати різні растрові підкдладки різним проекціям сцени.

Контроль за рухом можна здійснювати, використовуючи мишку, джойстик і медіаінструменти.

У редакторі матеріалів залишився колишній інтерфейс, але додалося багато нових функціональних можливостей. Є переглядач матеріалів і карт. Тепер можна одночасно переглядати до 24 матеріалів, одержувати збільшене представлення будь-якого зразка в окремому вікні, обертати будь-який матеріал, використовувати різні підкладки, умови освітлення і нестандартні зразки геометрії.

Поліпшена анімація. Переміщення або деформація об'єкта в процесі анімації може здійснюватися уздовж привласненого йому сплайнового шляху. Вражаючих ефектів можна домогтися за допомогою Space warps (перекручувачі простору), основані на модифікаторах, а також використовуючи різні спецеффекти: Particle Bomb - розриває об'єкти на шматочки, що розлітаються в різні сторони.

Auto-Bone - дозволяє на основі будь-якої ієрархії об'єктів, пов'язаних відношеннями інверсної кінематики автоматично формувати кістяк. Це дозволяє дуже точно відтворювати рухи живих істот.

Освітлення - усі види джерел освітлення мають однакові можливості і можуть створювати тіні, створювати ефект ослаблення освітлення зі

збільшенням відстані від джерела, проектувати зображення. Можлива анімація контролю рівня освітленості, управляти такими характеристиками: зсув краю, ближня і далека межа загасання світлового потоку й ін.

Є дві установки для якості: остаточна і попередня і три методи для створення ефекту розмитості зображення при русі: швидкий, згладжений і залежний від швидкості. Інтерактивна візуалізація в реальному часі забезпечується підтримкою технології апаратного прискорення Open GL і Direct 3D. Рендеринг може здійснюватися і контролюватися на робочій станції з використанням TCP/IP - протоколу обміну даними.

У Video Post реальні сцени можуть інтегруватися з тривимірними комп'ютерними об'єктами і маскованими зображеннями. Підтримуються різні типи масок, включаючи Alpha канал, RGB фільтри і світіння. Вбудовані фільтри можуть генерувати ефекти типу зоряних систем або анімувати полум'я. Можливе застосування відеоефектів із Adobe Premier. Є можливість імітації старіння і потертостей. Є фільтр глибини різкості.

Plug-in – модулі, що підключаються. Можуть використовувати модулі, розроблені іншими фірмами (більше 200 розроблювачів).

3 Розробка програмних засобів з використанням графічного інтерфейсу OpenGL

OpenGL є на даний момент одним із найпопулярніших програмних інтерфейсів (API) для розробки додатків в області двовимірної і тривимірної графіки. Стандарт OpenGL був розроблений і затверджений головними фірмами в області розробки програмного забезпечення, а його основою стала бібліотека IRIS GL, розроблена Silicon Graphics.

У стандарті немає регламентацій з опису структури графічних об'єктів. Головна увага приділяється опису процесу їхньої візуалізації. Завдяки цьому зростає ефективність роботи наявної апаратури: від простих пристроїв, що використовують тільки буфер кадру до сучасних графічних систем, здатних на апаратному рівні візуалізувати тривимірні об'єкти. У OpenGL передбачені тільки можливості з виведенням зображень, організація введення цілком передається конкретній віконній системі, яка дозволяє досягати додаткової апаратної незалежності додатків.

Через мінімальне використання складних структур для формування тривимірних об'єктів можливо застосування OpenGL у якості бази для побудови бібліотек керування структурованими об'єктами. Прикладами таких бібліотек можуть бути об'єктно-орієнтовані інструментальні пакети, які використовуються для візуалізації і моделювання складних графічних структур .

З метою розвитку промислового стандарту OpenGL була створена незалежна організація Architecture Review Board (ARB), постійними членами якої є DEC, IBM, Intel, Evans й Sutherland, Intergraph, Microsof і SGI. У задачу ARB входить розробка пропозицій для подальшого розвитку OpenGL, розбір зауважень, які надходять від користувачів, розробка специфікацій і тестування нових версій.

На даний момент реалізація OpenGL містить у собі декілька бібліотек (опис базових функцій OpenGL, GLU,GLUT,GLAUX і інші), призначення яких буде описано нижче.

Характерними рисами OpenGL, що забезпечили поширення і розвиток цього графічного стандарту, є:

Стабільність

Доповнення і зміни в стандарті реалізуються таким чином, щоб зберегти сумісність із розробленим раніше програмним забезпеченням.

Надійність

Додатки, що використовують OpenGL, гарантують однаковий візуальний результат незалежно від типу операційної системи та організації відображення інформації. Крім того, ці додатки можуть виконуватися як на персональних комп'ютерах, так і на робочих станціях і суперкомп'ютерах.

Легкість застосування

Стандарт OpenGL має продуману структуру та інтуїтивно зрозумілий інтерфейс, що дозволяє з меншими витратами розробляти ефективні додатки. Необхідні функції для забезпечення сумісності з різним обладнанням реалізовані на рівні бібліотеки і значно спрощують розробку додатків.

3.1 Основні можливості OpenGL:

- Набір базових примітивів: точки, лінії, багатокутники і т.п.
- Видові і координатні перетворення.
- Видалення невидимих ліній і поверхонь (z-буфер).
- Використання сплайнів для побудови ліній і поверхонь.
- Накладення текстури і застосування освітлення.
- Використання спеціальних ефектів: туманність , зміна прозорості, поєднання кольорів (blending), усунення ступінчастості (anti-aliasing).

Як уже було сказано, існує реалізація OpenGL для різних платформ, для чого було зручно розділити базові функції графічної системи і функції для відображення графічної інформації та взаємодії з користувачем. Були розроблені бібліотеки для відображення інформації за допомогою віконної підсистеми для операційних систем Windows і Unix (WGL і GLX

відповідно), а також бібліотеки GLAUX і GLUT, які використовуються для формування так званих консольних додатків.

Бібліотека GLAUX поступається за популярністю написаній дещо пізніше бібліотеці GLUT, хоча вони дають приблизно однакові можливості. Надалі в даному посібнику як основна буде розглядатися бібліотека GLUT, яка дає широкий набір засобів взаємодії з користувачем.

До складу бібліотеки GLU увійшла реалізація більш складних функцій, таких як набір популярних геометричних примітивів (куб, куля, циліндр, диск), функції побудови сплайнів, реалізація додаткових операцій над матрицями і т.п. Всі вони реалізовані через базові функції OpenGL.

Архітектура й особливості синтаксису

Графічна система OpenGL є конвеєром, що складається з декількох етапів обробки даних:

- апроксимація кривих і поверхонь;
- обробка вершин і формування примітивів;
- растеризація та обробка фрагментів;
- операції над пікселами;
- підготування текстури;
- передача даних у буфер кадру.

Для забезпечення інтуїтивно зрозумілих назв у OpenGL повне ім'я команди має вигляд:

type **glCommand_name[1 2 3 4][b s i f d ub us ui][v]** (type1 arg1,…,typeNargN).

Таким чином, ім'я складається з декількох частин:

gl - це ім'я бібліотеки, у якій описана ця функція: для базових функцій бібліотек GLU, GLUT, GLAUX це gl, glu, glut, aux відповідно; Command_name - ім'я команди; [1 2 3 4] -число аргументів команди; [b s i f d ub us ui] - тип аргументу: символ b означає тип GLbyte (аналог char у C\C++),символ f -тип GLfloat (аналог float), символ i - тип GLint (аналог int) і так далі; [v] - наявність цього символу показує, що як параметр функції використовується покажчик на масив значень

Символи в квадратних дужках у деяких назвах не використовуються. Наприклад, команда glVertex2i( ) описана як базова в бібліотеці OpenGL і використовує як параметри два цілих числа, а команда glColor3fv( ) використовує як параметр покажчик на масив із трьох чисел.

Структура консольного додатка

Будемо розглядати побудову консольного додатка за допомогою бібліотеки GLUT або GL Utility Toolkit, що одержала останнім часом широке поширення. Ця бібліотека забезпечує єдиний інтерфейс для роботи

з вікнами незалежно від платформи. Структура додатка залишається незмінною для операційних систем Windows, Linux і багатьох інших.

- Функції GLUT можуть бути класифіковані на декілька груп за їхнім призначенням:
- ініціалізація;
- початок обробки подій;
- керування вікнами;
- керування меню;
- реєстрація функцій;
- керування палітрою кольорів;
- відображення шрифтів;
- відображення додаткових геометричних фігур (тор, конус і ін.).

Ініціалізація проводиться за допомогою функції

glutInit (int *argcp, char **argv).

Змінна argcp є покажчиком на стандартну змінну argc, яку описує у функції main( ), а argv - покажчик на параметри, які передаються програмі при запуску. Ця функція виконує необхідні початкові дії для побудови вікна додатка і тільки декілька функцій GLUT можуть бути викликані до неї. До них відносяться:

glutInitWindowPosition (int x, int y) ,

glutInitWindowSize (int width, int height),

glutInitDisplayMode (unsigned int mode) .

Перші дві функції задають відповідно положення і розмір вікна, а остання функція визначає різні режими відображення інформації, що можуть спільно задаватися з використанням операції побітового “або” (“ | “) :

- **GLUT_RGBA** - режим RGBA. Використовується за замовчуванням, якщо не зазначені явно режими GLUT_RGBA або GLUT_INDEX ;
- **GLUT_RGB** - та ж, що і GLUT_RGBA;
- **GLUT_INDEX -р**ежим індексованих кольорів (використання палітри). Скасовує GLUT_RGBA;
- **GLUT_SINGLE- в**ікно з одиночним буфером. Використовується за замовчуванням.
- **GLUT_DOUBLE - в**ікно з подвійним буфером. Скасовує GLUT_SINGLE;
- **GLUT_DEPTH** - вікно з буфером глибини.

Це неповний список параметрів для даної функції, однак для більшості випадків цього буває досить.

Подвійний буфер звичайно використовують для анімації. Буфер глибини або z-буфер використовується для видалення невидимих ліній і поверхонь.

Функції бібліотеки GLUT реалізують так званий подійно-керований механізм. До подій відносяться: рух миші, закриття вікна, зміна

властивостей вікна, переміщення курсору, натискання клавіші і "порожня" (idle) подія, коли нічого не відбувається. Для проведення періодичної перевірки здійснення тієї або іншої події треба зареєструвати функцію, що буде її обробляти. Для цього використовуються функції виду:

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)).

Параметром для них є ім'я відповідної функції заданого типу. За допомогою glutDisplayFunc( ) задається функція рисування для вікна додатка, яке викликається при необхідності формування або відновлення зображення. Для явного вказання, що вікно треба обновити, іноді зручно використовувати функцію void **glutPostRedisplay** (void).

Через glutReshapeFunc( ) встановлюється функція зміни розмірів вікна користувачем, якій передаються нові розміри.

glutMouseFunc() визначає обробку команд від миші, а glutIdleFunc( ) задає функцію, яка буде викликатися щораз, коли немає подій від користувача.

Контроль усіх подій відбувається всередині нескінченного циклу у функції void **glutMainLoop** (void), котра звичайно викликається наприкінці будь-якої програми, яка використовує GLUT.

Структура додатка, який використовує анімацію, буде такою:

```
#include <GL/glut. h>
void MyIdle(void){
//--Код, що змінює змінні, які визначають наступний кадр --//
....
};
void MyDisplay(void){
//-- Код OpenGL, що відображає кадр --//
 ....
//-- Після рисування переставляємо буфери --//
glutSwapBuffers();
};
void main(int argcp, char **argv){
……
//-- Ініціалізація GLUT --//
glutInit(&argcp, argv);
glutInitWindowSize(640, 480);
glutInitWindowPosition(0, 0);
 //--Відкрити вікно--//
glutCreateWindow("My OpenGL Application");
 //-- Вибір режиму: Подвійний буфер і RGBA кольори --//
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
```

```
  //-- Реєстрація функцій , які викликаються --//
  glutDisplayFunc(MyDisplay);
  glutIdleFunc(MyIdle);
  //-- Запуск механізму обробки подій --//
  glutMainLoop();
};
```

У випадку, якщо додаток будує статичне зображення, можна замінити GLUT_DOUBLE на GLUT_SINGLE, тому що одного буфера в цьому випадку буде досить.

3.2 Вершини і примітиви

Під вершиною розуміється точка в тривимірному просторі, координати якого можна задавати таким способом:

void **glVertex[2 3 4][s i f d]** (type coords) **,**

void **glVertex[2 3 4][s i f d]v** (type *coords) **.**

Координати точки задаються максимум чотирма значеннями: x, y, z, w, при цьому можна вказувати два (x,y) або три (x,y,z) значення, а для інших змінних у цих випадках використовуються значення за замовчуванням: z=0, w=1.

Координатні осі розташовані так, що точка (0,0) знаходиться в лівому нижньому кутку екрана, вісь x спрямована вліво, вісь y - вгору, а вісь z - з екрана. Це розташування осей світової системи координат, у якій задаються координати вершин об'єкта.

Однак , щоб задати фігуру одних координат вершин недостатньо, і ці вершини треба об'єднати в одне ціле, визначивши необхідні властивості. Для цього в OpenGL використовується поняття примітивів, до яких відносяться точки, лінії, зв'язані або замкнуті лінії, трикутники і так далі. Завдання примітива відбувається всередині командних дужок:

void **glBegin** (GLenum mode)

void **glEnd** (void)

Параметр mode визначає тип примітива, що задається, і може приймати такі значення:

GL_POINTS - кожна вершина задає координати деякої точки;

GL_LINES - кожна окрема пара вершин визначає відрізок; якщо задане непарне число вершин, то остання вершина ігнорується**;**

GL_LINE_STRIP - кожна наступна вершина задає відрізок разом із попередньою;

GL_LINE_LOOP - відмінність від попереднього примітива тільки в тому, що останній відрізок визначається останньою і першою вершиною, створюючи замкнуту ламану;

GL_TRIANGLES - кожна окрема трійка вершин визначає трикутник; якщо задане число вершин не кратне трьом, то останні вершини ігноруються;

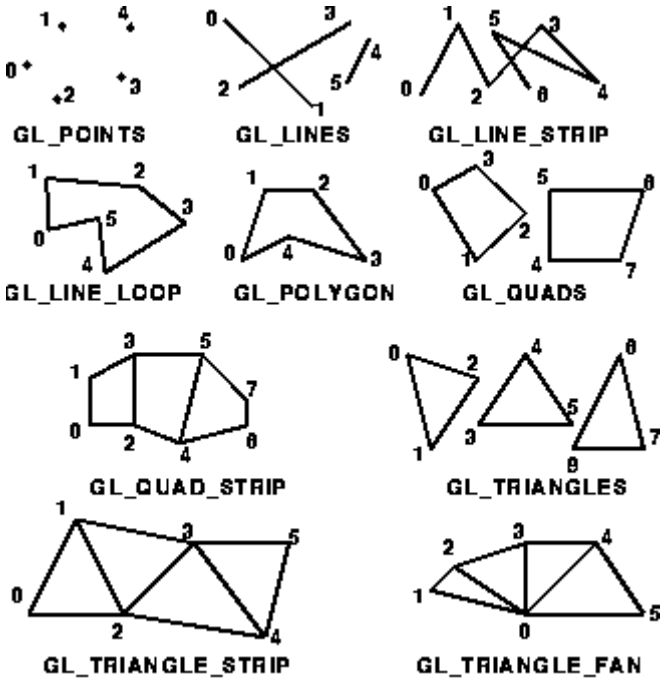
GL_TRIANGLE_STRIP - кожна наступна вершина задає трикутник разом із двома попередніми;

GL_TRIANGLE_FAN - **т**рикутники задаються першою і кожною наступною парою вершин (пари не перетинаються);

GL_QUADS - кожна окрема четвірка вершин визначає чотирикутник; якщо задане не кратне чотирьом число вершин, то останні вершини ігноруються;

GL_QUAD_STRIP - чотирикутник із номером n визначається вершинами з номерами 2n-1, 2n, 2n+2, 2n+1;

GL_POLYGON - послідовно задаються вершини опуклого багатокутника.



Для завдання поточного кольору вершини використовуються команди

void **glColor[3 4][b s i f]** (GLtype components)

void **glColor[3 4][b s i f]v** (GLtype components)

Перші три параметри задають R, G, B компоненти кольору, а останній параметр визначає alpha- компоненту , яка задає рівень прозорості об'єкта. Якщо в назві команди зазначений тип ‘f' (float), то значення всіх параметрів повинні належати відрізку [0,1], при цьому за замовчуванням значення alpha-компоненти встановлюється рівним 1.0, що відповідає повній непрозорості. Якщо зазначений тип ‘ub' (unsigned byte), то значення повинні лежати у відрізку [0,255].

Різним вершинам можна призначати різні кольори і тоді буде проводитися лінійна інтерполяція кольорів по поверхні примітива.

Для керування режимом інтерполяції кольорів використовується команда void **glShadeModel** (GLenum mode), виклик якої з параметром GL_SMOOTH включає інтерполяцію, а з GL_FLAT - відключає.

Наприклад, щоб нарисувати трикутник із різними кольорами у вершинах, досить написати:

```
GLfloat BlueCol[3] = {0,0,1};
glBegin(GL_TRIANGLE);
    glColor3f(1.0, 0.0, 0.0); // червоний
    glVertex3f(0.0, 0.0, 0.0);
    glColor3ub(0,255,0); // зелений
    glVertex3f(1.0, 0.0, 0.0);
    glColor3fv( BlueCol ); // синій
    glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Для задання кольору фону використовується команда

void **glClearColor** (GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha).

Значення повинні знаходитися у відрізку [0,1] і за замовчуванням дорівнюють нулю. Після цього виклик команди

void **glClear** (GLbitfield mask)

з параметром GL_COLOR_BUFFER_BIT встановлює колір фону в усі буфери, доступні для запису кольору (іноді зручно використовувати декілька буферів кольору).

Крім кольору аналогічним способом можна визначити нормаль у вершині, використовуючи команди

void **glNormal3[b s i f d]** (type coords)

void **glNormal3[b s i f d]v** (type coords).

Вектор може не мати одиничної довжини, але він буде нормалізований автоматично в режимі нормалізації, який включається викликом команди glEnable(GL_NORMALIZE).

Команди void **glEnable** (GLenum mode)

void **glDisable** (GLenum mode)

здійснюють включення або відключення того чи іншого режиму роботи конвеєра OpenGL. Ці команди застосовуються досить часто, і їхній вплив буде розглядатися в конкретних випадках.

Взагалі, всередині командних дужок glBegin( ) і glEnd( ) можна робити виклик лише декількох команд, в які входять glVertex. .( ), glColor. .( ) glNormal. .( ), glRect. .( ), glMaterial. .( ) і glTexCoord. .( ).

За допомогою команд

void **glRect[s i f d]** (GLtype x1, GLtype y1, GLtype x2, GLtype y2)

void **glRect[s i f d]v** (GLtype *v1, GLtype *v2)

можна нарисувати прямокутник у площині z=0 із координатами протилежних кутів (x1,y1) і (x2,y2), або набір прямокутників із координатами кутів у масивах v1 і v2.

Крім задання самих примітивів можна визначити метод їх відображення на екрані, де під примітивами в даному випадку розуміються багатокутники.

Однак спочатку треба визначити поняття лицьових і зворотних граней.

Під гранню розуміється одна зі сторін багатокутника. За замовчуванням лицьовою вважається та сторона, вершини якої обходяться проти годинникової стрілки. Напрямок обходу вершин лицьових сторін можна змінити викликом команди void **glFrontFace** (GLenum mode) зі значенням параметра mode , який дорівнює GL_CW, а скасувати - із GL_CCW.

Щоб змінити метод відображення багатокутника використовується команда

void **glPolygonMode** (GLenum face, Glenum mode).

Параметр mode визначає, як будуть відображатися багатокутники, а параметр face встановлює тип багатокутників, до яких буде застосовуватися ця команда і може приймати такі значення:

GL_FRONT - для лицьових граней,

GL_BACK - для зворотних граней ,

GL_FRONT_AND_BACK - для всіх граней.

Параметр mode може дорівнювати:

GL_POINT - при такому режимі будуть відображатися тільки вершини багатокутників;

GL_LINE - при такому режимі багатокутник буде представлятися набором відрізків;

GL_FILL -при такому режимі багатокутники будуть зафарбовуватися поточним кольором .

Крім того, можна вказувати, який тип граней відображати на екрані. Для цього спочатку треба установити відповідний режим викликом команди glEnable (GL_CULL_FACE), а потім вибрати тип граней за допомогою команди void **glCullFace** (GLenum mode).

Виклик команди з параметром GL_FRONT приводить до видалення зі зображення всіх лицьових граней, а з параметром GL_BACK - зворотних (установка за замовчуванням).

Крім розглянутих стандартних примітивів у бібліотеках GLU і GLUT описані більш складні фігури, такі як сфера, циліндр, диск (у GLU) і сфера, куб, конус, тор, тетраедр, додекаедр, ікосаедр, октаедр і чайник (у GLUT). Автоматичне накладення текстури передбачено тільки для фігур із бібліотеки GLU (формування текстур у OpenGL буде розглядатися нижче).

Наприклад, щоб нарисувати сферу або циліндр, треба спочатку створити об'єкт спеціального типу GLUquadricObj за допомогою команди GLUquadricObj* **gluNewQuadric**(void), а потім викликати відповідну команду:

void **gluSphere** (GLUquadricObj * qobj, GLdouble radius, GLint slices, GLint stacks)

void **gluCylinder** (GLUquadricObj * qobj**,**GLdouble baseRadius**,**GLdouble topRadius**,** GLdouble height**,** GLint slices**,** GLint stacks), де параметр slices задає число розбивок уздовж осі z, a stacks - уздовж осі z.

Важливо відзначити, що для коректної побудови перерахованих примітивів необхідно видаляти невидимі лінії і поверхні, для цього треба включити відповідний режим викликом команди

glEnable(GL_DEPTH_TEST).

Масиви вершин

Якщо вершин багато, то зручно об'єднувати вершини в масиви, використовуючи команду

void **glVertexPointer** (GLint size, GLenum type, GLsizei stride, void *ptr).

При цьому size визначає число координат вершини (може дорівнювати 2, 3, 4), type визначає тип даних (може бути дорівнює GL_SHORT, GL_INT, GL_FLOAT, GL_DOUBLE). Іноді зручно зберігати в одному масиві інші атрибути вершини, і тоді параметр stride задає зсув від координат однієї вершини до координат наступної; якщо stride дорівнює нулю, це значить, що координати розташовані послідовно. У параметрі ptr вказується адреса, де знаходяться дані.

Аналогічно можна визначити масив нормалей, кольорів і деяких інших атрибутів вершини, використовуючи команди

void **NormalPointer** (GLenum type, GLsizei stride, void *pointer),

void **ColorPointer** (GLint size, GLenum type, GLsizei stride, void *pointer).

Для того, щоб ці масиви можна було використовувати надалі, треба викликати команду

void **glEnableClientState** (GLenum array)

з параметрами GL_VERTEX_ARRAY, GL_NORMAL_ARRAY, GL_COLOR_ARRAY відповідно. Після закінчення роботи з масивом бажано викликати команду

void **glDisableClientState** (GLenum array) з відповідним значенням параметра array.

Для відображення вмісту масивів використовується команда

void **glArrayElement** (GLint index),

яка передає OpenGL атрибути вершини, використовуючи елементи масиву з номером index. Це аналогічно послідовному застосуванню команд виду glColor. .(…),glNormal....(…), glNormal. .(…) з відповідними параметрами. Однак замість неї звичайно викликається команда

void **glDrawArrays** (GLenum mode, GLint first, GLsizei count),

яка рисує count примітивів, обумовлених параметром mode, використовуючи елементи з масивів з індексами від first до first+count-1. Це еквівалентно виклику команди glArrayElement() із відповідними індексами.

У випадку , коли одна вершина входить у декілька примітивів, то замість дублювання її координат у масиві зручно використовувати її індекс.

Для цього треба викликати команду void **glDrawArrays** (GLenum mode, GLsizei count, GLenum type, void *indices), де indices - це масив номерів вершин, які треба використовувати для побудови примітивів; type- визначає тип елементів цього масиву: GL_UNSIGNED_BYTE, GL_UNSIGNED_SHORT, GL_UNSIGNED_INT, а count- задає кількість.

Списки зображень

Якщо потрібно декілька разів звертатися до однієї і тієї ж групи команд, ці команди можна об'єднати в так званий список зображень (display list) і викликати його при необхідності. Для того, щоб створити новий список зображень треба помістити всі команди, що повинні в нього ввійти, між командними дужками:

void **glNewList** (GLuint list, GLenum mode)

void **glEndList**().

Для розрізнення списків використовуються цілі додатні числа, що задаються при формуванні списку значенням параметра list, а параметр mode визначає режим обробки команд, що входять у список:

GL_COMPILE - команди записуються в список без виконання;

GL_COMPILE_AND_EXECUTE - команди спочатку виконуються, а потім записуються в список

Після того, як список створений, його можна викликати командою

void **glCallList** (GLuint list), вказавши в параметрі list ідентифікатор потрібного списку. Щоб викликати відразу декілька списків, можна використати команду

void **glCallLists** (GLsizei n, GLenum type, const GLvoid *lists) ,

яка викликає n списків з ідентифікаторами з масиву lists, тип елементів якого вказується в параметрі type. Це можуть бути типи GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_INT, GL_UNSIGNED_INT і деякі інші. Для видалення списків використовується команда

void **glDeleteLists** (GLint list, GLsizei range),

яка видаляє списки з ідентифікаторами ID із діапазону list <= ID <= list+range-1.

3.3 Перетворення координат і проекції

У OpenGL використовуються як основні три системи координат: лівостороння, правостороння і віконна. Перші дві системи є тривимірними і відрізняються одна від одної напрямком осі z: у правосторонньої вона спрямована на спостерігача, а в лівосторонньої - у глиб екрана. Розміщення осей x і y аналогічне описаному вище. Лівостороння система використовується для задання значень параметрам команди gluPerspective(

), glOrtho( ), а правостороння або світова система координат -у всіх інших випадках.

Для задання різних перетворень об'єктам сцени в OpenGL використовуються операції над матрицями, при цьому розрізняють три типи матриць: видова, проекцій і текстури. Всі вони мають розмір 4x4. Видова матриця визначає перетворення об'єкта у світових координатах, таких як паралельне перенесення, зміна масштабу і поворот. Матриця проекцій задає, як будуть проектуватися тривимірні об'єкти на площину екрана (у віконні координати), а матриця текстури визначає накладення текстури на об'єкт.

Для того щоб вибрати, яку матрицю треба змінити, використовується команда void **glMatrixMode** (GLenum mode) , виклик якої зі значенням параметра mode, рівний GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE, включає відповідно режим роботи з видовою матрицею, матрицею проекцій і матрицею текстури . Для виклику команд, які задають матриці того або іншого типу необхідно спочатку установити відповідний режим.

Для визначення елементів матриці поточного типу викликається команда void **glLoadMatrix[f d]** (GLtype *m), де m вказує на масив із 16 елементів типу float або double відповідно до назви команди. При цьому спочатку в ньому повинний бути записаний перший стовпець матриці, потім другий, третій і четвертий.

Команда void **glLoadIdentity** (void) заміняє поточну матрицю на одиничну. Часто потрібно зберегти вміст поточної матриці для подальшого використання, для чого використовують команди

void **glPushMatrix** (void),

void **glPopMatrix** (void) .

Вони записують і відновлюють поточну матрицю зі стека, причому для кожного типу матриць стек свій. Для видових матриць його глибина дорівнює як мінімум 32, а для двох інших типів - як мінімум 2.

Для множення поточної матриці зліва на іншу матрицю використовується команда void glMultMatrix[f d] (GLtype *m), де m повинний задавати матрицю розміром 4x4 у вигляді масиву з описаним розташуванням даних. Однак звичайно для зміни матриці того або іншого типу зручно використовувати спеціальні команди, які за значеннями своїх параметрів створюють потрібну матрицю і перемножують її з поточною.

До видових перетворень будемо відносити перенесення, поворот і зміну масштабу вздовж координатних осей. Для проведення цих операцій достатньо помножити на відповідну матрицю кожну вершину об'єкта і одержати змінені координати цієї вершини:

$(x', y', z', 1)^T = M * (x, y, z, 1)^T$,

де M - матриця видового перетворення. Перспективне перетворення і проектування здійснюється аналогічно. Сама матриця може бути сформована за допомогою таких 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. .( ) здійснює перенесення об'єкта, збільшуючи до координат
його вершин значення своїх параметрів.

glRotate. .( ) здійснює поворот об'єкта проти годинникової стрілки на кут angle (вимірюється в градусах) навколо вектора (x,y,z).

glScale. .( ) здійснює масштабування об'єкта , домножуючи відповідні координати його вершин на значення своїх параметрів.

У випадку , якщо треба, наприклад, повернути один об'єкт сцени, а іншій залишити нерухомим, зручно спочатку зберегти поточну видову матрицю в стеку командою glPushMatrix( ), потім викликати glRotate. .( ) із потрібними параметрами, описати примітиви, із яких складається цей об'єкт, а потім відновити поточну матрицю командою glPopMatrix( ).

Крім зміни положення самого об'єкта іноді буває потрібно змінити положення точки спостереження, що також приводить до зміни видової матриці. Це можна зробити за допомогою команди

void **gluLookAt** (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz), де точка (eyex,eyey,eyez) визначає точку спостереження; (centerx, centery, centerz) задає центр сцени, яка буде проектуватися в центр області виведення, а вектор (upx, upy, upz) задає додатній напрямок осі , визначаючи поворот камери. Якщо, наприклад, камеру не треба повертати, то задається значення (0,1,0), а зі значенням (0,-1,0) сцена буде повернена.

Фактично, ця команда здійснює перенесення і поворот об'єктів сцени, але в такому вигляді задавати параметри буває зручніше.

Проекції

У OpenGL існують орфографічна (паралельна) і перспективна проекція. Перший тип проекції може бути заданий командами

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

void **gluOrtho2D** (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top).

Параметри команди задають точки (left, bottom, -near) і (right, top, -near), які відповідають лівому нижньому і правому верхньому куткам вікна виведення. Параметри near і far задають відстань до ближньої і дальньої площин відсікання по дальності від точки (0,0,0) .

В другій команді, на відміну від першої , значення near і far установлюються рівними -1 і 1 відповідно.

Перспективна проекція визначається командою

void **gluPerspective** (GLdouble angley, GLdouble aspect, GLdouble znear, GLdouble zfar) , яка задає усічений конус видимості в лівосторонній системі координат. Параметр angley визначає кут видимості в градусах по осі y і повинний знаходитися в діапазоні від 0 до 180. Кут видимості вздовж осі x задається параметром aspect, що звичайно задається як відношення сторін області виведення. Параметри zfar і znear задають відстань від спостерігача до площин відсікання по глибині і повинні бути додатними . Чим більше відношення zfar/znear, тим гірше в буфері глибини будуть розрізнятися розміщені поруч поверхні, тому що за замовчуванням в нього буде записуватися 'стиснута' глибина в діапазоні від 0 до 1 .

Область виводу

Після застосування матриці проекцій на вхід наступного перетворення подаються так звані усічені (clip) координати, для яких значення усіх компонент $(xc, yc, zc, wc)^T$ знаходяться на відрізку [-1,1]. Після цього знаходяться нормалізовані координати вершин за формулою:

$(xn, yn, zn)^T = (xc/wc , yc/wc, zc/wc)^T$.

Область виведення - прямокутник у віконній системі координат, розміри якого задаються командою:

void **glViewPort** (GLint x, GLint y, GLint width, GLint height).

Значення всіх параметрів задаються в пікселах і визначають ширину і висоту області виведення з координатами лівого нижнього кутка (x,y) у віконній системі координат. Розміри віконної системи координат визначаються поточними розмірами вікна додатку, точка (0,0) знаходиться в лівому нижньому кутку вікна.

Використовуючи параметри команди glViewPort(), обчислюються віконні координати центру області виведення (ox,oy) за формулами ox=x+width/2, oy=y+height/2.

Нехай px=width, py=height, тоді можна знайти віконні координати кожної вершини:

$(xw, yw, zw)^T = ((px/2) xn+ ox , (py/2) yn+ oy , [(f-n)/2] zn+(n+f)/2)^T$.

При цьому цілі додатні розміри n і f задають мінімальну і максимальну глибину точки у вікні і за замовчуванням рівні 0 і 1 відповідно. Глибина кожної точки записується в спеціальний буфер глибини (z-буфер), який використовується для видалення невидимих ліній і поверхонь. Установити значення n і f можна викликом функції

void **glDepthRange** (GLclampd n, GLclampd f).

Команда glViewPort() звичайно використовується у функції, зареєстрованій за допомогою команди glutReshapeFunc().

3.4 Матеріали та освітлення

Для створення реалістичних зображень необхідно визначити як властивості самого об'єкта, так і властивості середовища, у якому він знаходиться. Перша група властивостей містить у собі параметри матеріалу, із якого зроблений об'єкт, способи нанесення текстури на його поверхню, рівень прозорості об'єкта. До другої групи можна віднести кількість і властивості джерел світла, рівень прозорості середовища. Всі ці властивості можна задавати, використовуючи відповідні команди OpenGL.

Властивості матеріалу

Для задання параметрів поточного матеріалу використовуються команди

void **glMaterial[i f]** (GLenum face, GLenum pname, GLtype param),

void **glMaterial[i f]v** (GLenum face, GLenum pname, GLtype *params).

За їхньою допомогою можна визначити розсіяний, дифузійний і дзеркальний кольори матеріалу, а також колір, ступінь дзеркального відбитка та інтенсивність випромінювання світла, при умові, що об'єкт повинний світитися. Який саме параметр буде визначатися значенням param, залежить від значення pname:

GL_AMBIENT, параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, які визначають розсіяний колір матеріалу (колір матеріалу в тіні). Значення за замовчуванням: (0.2, 0.2, 0.2, 1.0).

GL_DIFFUSE , параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, які визначать колір дифузійного відбитку матеріалу. Значення по умовчанню: (0.8, 0.8, 0.8, 1.0).

GL_SPECULAR, параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, які визначають колір дзеркаль-Ного відбитка матеріалу. Значення за замовчуванням: (0.0, 0.0, 0.0, 1.0).

GL_SHININESS , параметр params повинний містити одне ціле або дійсне значення в діапазоні від 0 до 128, яке визначає ступінь дзеркального відбитка матеріалу. Значення за замовчуванням: 0.

GL_EMISSION, параметр params повинний містити чотирьох цілих або дійсних значення кольорів RGBA, які визначають інтенсивність світла, яку випромінює матеріал. Значення за замовчуванням: (0.0, 0.0, 0.0, 1.0).

GL_AMBIENT_AND_DIFFUSE еквівалентно двом викликам команди glMaterial.( ) зі значенням pname GL_AMBIENT і GL_DIFFUSE і однаковими значеннями params.

З цього випливає, що виклик команди glMaterial[i f]( ) можливий тільки для установки ступеня дзеркального відбитка матеріалу. У більшості моделей враховується дифузійне і дзеркальне відбите світло; перший визначає природний колір об'єкта, а другий - розмір і форму відблисків на його поверхні.

Параметр face визначає тип граней, для яких задається цей матеріал і може приймати значення GL_FRONT, GL_BACK або

GL_FRONT_AND_BACK.

Якщо в сцені матеріали об'єктів розрізняються лише одним параметром, рекомендується спочатку установити потрібний режим, викликавши glEnable( ) с параметром GL_COLOR_MATERIAL, а потім використати команду

void **glColorMaterial** (GLenum face, GLenum pname),

де параметр face має аналогічний зміст, а параметр pname може приймати всі перераховані значення.

Додати в сцену джерело світла можна за допомогою команд

void **glLight[i f]** (GLenum light, GLenum pname, GLfloat param),

void **glLight[i f]** (GLenum light, GLenum pname, GLfloat *params).

Параметр light однозначно визначає джерело, і вибирається з набору спеціальних символічних імен вигляду GL_LIGHTi , де і повинно знаходитися в діапазоні від 0 до GL_MAX_LIGHT, що не перевершує восьми.

Два параметри , що залишилися , мають аналогічний зміст, що й у команді glMaterial. .( ). Розглянемо їхнє призначення (спочатку описуються параметри для першої команди, потім для другої):

GL_SPOT_EXPONENT - параметр param повинний містити ціле або дійсне число від 0 до 128, яке задає розподіл інтенсивності кольору. Цей параметр описує рівень сфокусованності джерела світла. Значення за замовчуванням: 0 (розсіяне світло).

GL_SPOT_CUTOFF - параметр param повинний містити ціле або дійсне число між 0 і 90 або рівне 180, яке визначає максимальний кут відхилення світла . Значення за замовчуванням: 180 (розсіяне світло).

GL_AMBIENT - параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, яке визначають колір фонового освітлення. Значення за замовчуванням: (0.0, 0.0, 0.0, 1.0).

GL_DIFFUSE - параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, які визначають колір дифузійного освітлення.Значення за замовчуванням: (1.0, 1.0, 1.0, 1.0) для LIGHT0 і (0.0, 0.0, 0.0, 1.0) для інших.

GL_SPECULAR - параметр params повинний містити чотири цілих або дійсних значення кольорів RGBA, які визначають колір дзеркального відбитка. Значення по умовчанню: (1.0, 1.0, 1.0, 1.0) для LIGHT0 і (0.0, 0.0, 0.0, 1.0) для інших.

GL_POSITION - параметр params повинний містити чотири цілих або дійсних значень, які визначають положення джерела світла. Якщо значення компоненти w дорівнює 0.0, то джерело вважається нескінченно віддаленим і при розрахунку освітленості враховується тільки напрямок на точку (x,y,z), у протилежному випадку вважаються, що джерело

розташоване в точці (x,y,z,w). Значення за замовчуванням: (0.0, 0.0, 1.0, 0.0).

GL_SPOT_DIRECTION - параметр params повинний містити чотири цілих або дійсних числа, які визначають напрямок світла. Значення за замовчуванням: (0.0, 0.0, -1. 0, 1.0).

Модель освітлення

У OpenGL використовується модель освітлення Фонга, відповідно до якої колір точки визначається декількома чинниками: властивостями матеріалу і текстури, розміром нормалі в цій точці, а також положенням джерела світла і спостерігача. Для коректного розрахунку освітленості в точці треба використовувати одиничні нормалі, однак команди типу glScale. .( ), можуть змінювати довжину нормалей. Щоб це враховувати, використовується режим нормалізації, який включається викликом команди glEnable(GL_NORMALIZE).

Для задання глобальних параметрів освітлення використовуються команди

void **glLightModel[i f]** (GLenum pname, GLenum param),

void **glLightModel[i f]v** (GLenum pname, const GLtype *params).

Аргумент pname визначає, який параметр моделі освітлення буде настроюватися і може приймати такі значення:

GL_LIGHT_MODEL_LOCAL_VIEWER параметр param повинний бути булевим і задавати положення спостерігача. Якщо він дорівнює FALSE, то напрямок огляду вважається рівнобіжним осі -z. Якщо ж він дорівнює TRUE, то спостерігач знаходиться на початку видової системи координат. Це може поліпшити якість освітлення, але ускладнює його розрахунок. Значення за замовчуванням: FALSE.

GL_LIGHT_MODEL_TWO_SIDE - параметр param повинний бути булевим і керувати режимом розрахунку освітленості як для лицьових, так і для зворотних граней. Якщо він дорівнює FALSE, то освітленість розраховується тільки для лицьових граней. Якщо ж він дорівнює TRUE, розрахунок проводиться і для зворотних граней. Значення за замовчуванням: FALSE.

GL_LIGHT_MODEL_AMBIENT - параметр params повинний містити чотири цілих або дійсних числа, які визначають колір фонового освітлення навіть у випадку відсутності джерел світла. Значення замовчуванням: (0.2, 0.2, 0.2,1.0).

3.5 Текстури

Накладення текстури на поверхню об'єктів сцени підвищує її реалістичність, однак при цьому треба враховувати, що цей процес вимагає значних обчислювальних витрат. Під текстурою будемо розуміти деяке

зображення, яке треба певним чином нанести на об'єкт. Для цього варто виконати такі етапи:

- вибрати зображення і перетворити його до потрібного формату, завантажити зображення в пам'ять;
- визначити, як текстура буде наноситися на об'єкт, і як вона буде з ним взаємодіяти.

Розглянемо кожний із цих етапів.

Підготування текстури

Прийнятий у OpenGL формат збереження зображень відрізняється від стандартного формату Windows DIB тільки тим, що компоненти (R,G,B) для кожної точки зберігаються в прямому порядку, а не в зворотному і вирівнювання задається програмістом. Зчитування графічних даних із файла і їх перетворення можна проводити і вручну, однак зручно скористатися функцією, що входить до складу бібліотеки GLAUX (для її використання треба додатково підключити glaux. lib), що сама проводить необхідні операції. Це функція

AUX_RGBImageRec* **auxDIBImageLoad** (string file),

де file - назва файла з розширенням *.bmp або *.dib. У якості результату функція повертає покажчик на область пам'яті, де зберігаються перетворені дані.

При створенні способу текстури в пам'яті варто враховувати такі вимоги.

По-перше, розміри текстури як по горизонталі, так і по вертикалі повинні бути ступенями двійки. Ця вимога накладається для компактного розміщення текстури в пам'яті і сприяє її ефективному використанню. Використовувати тільки текстури з такими розмірами звичайно незручно, тому перед завантаженням їх треба перетворити. Зміна розмірів текстури проводиться за допомогою команди

void **gluScaleImage** (GLenum format, GLint widthin, GL heightin, GLenum typein,

const void *datain, GLint widthout, GLint heightout, GLenum typeout, void *dataout).

У якості значення параметра format звичайно використовується значення GL_RGB або GL_RGBA, які визначають формат збереження інформації. Параметри widthin, heightin, widhtout, heightout визначають розміри вхідного і вихідного зображень, а за допомогою typein і typeout задається тип елементів масивів, розташованих за адресами datain і dataout.

По-друге, треба передбачити випадок, коли об'єкт по розмірах значно менший текстури, яка на нього наноситься. Чим менший об'єкт, тим менша повинна бути текстура і тому вводиться поняття рівнів деталізації текстури. Кожний рівень деталізації задає деяке зображення, що є, як правило , зменшеною в два рази копією оригіналу. Такий підхід дозволяє поліпшити якість нанесення текстури на об'єкт. Наприклад, для

зображення розміром 2mx2n можна побудувати max(m,n)+1 зменшених зображень, що відповідають різним рівням деталізації.

Ці два етапи створення способу текстури в пам'яті можна провести за допомогою команди

void **gluBuild2DMipmaps** (GLenum target, GLint components, GLint width, GLint height, GLenum format, GLenum type, const void *data),

де параметр target , який повинний дорівнювати GL_TEXTURE_2D, components , визначає кількість колірних компонент текстури, що будуть використовуватися при її накладенні і може приймати значення від 1 до 4 (1-тільки червоний, 2-червоний і alpha, 3-червоний, синій, зелений, 4- всі компоненти).

Параметри width, height, data визначають розміри і розташування текстури відповідно, а format і type мають аналогічний зміст, що й у команді gluScaleImage().

При використанні в сцені декількох текстур, у OpenGL застосовується підхід, що нагадує створення списків зображень. Спочатку, за допомогою команди

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

треба створити n ідентифікаторів для використовуваних текстур, які будуть записані в масив textures. Перед початком визначення властивостей чергової текстури варто викликати команду

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

де target може приймати значення GL_TEXTURE_1D або GL_TEXTURE_2D, а параметр texture повинен дорівнювати ідентифікатору тієї текстури, до якої будуть відноситися наступні команди.

Методи накладення текстури

При накладенні текстури треба враховувати випадок, коли розміри текстури відрізняються від розмірів об'єкта, на який вона накладається. При цьому можливо як розтяг, так і стиск зображення, і те, як будуть проводитися ці перетворення може серйозно вплинути на якість побудованого зображення. Для визначення положення точки на текстурі використовується параметрична система координат (s,t), причому значення s і t знаходяться у відрізку [0,1]. Для зміни різних параметрів текстури застосовуються команди:

void **glTexParameter[i f]** (GLenum target, GLenum pname, GLenum param),

void **glTexParameter[i f]v** (GLenum target, GLenum pname, GLenum *params).

Можливі значення pname:

- **GL_TEXTURE_MIN_FILTER-** параметр param визначає функцію, яка буде використовуватися для стиску текстури. При значенні GL_NEAREST буде використовуватися один (найближчий), а при

значенні GL_LINEAR чотири найближчих елементи текстури. Значення за замовчуванням: GL_LINEAR.

- **GL_TEXTURE_MAG_FILTER** -параметр param визначає функцію, що буде використовуватися для збільшення (розтягу) текстури. При значенні GL_NEAREST буде використовуватися один (найближчий), а при значенні GL_LINEAR чотири найближчих елементи текстури. Значення за замовчуванням: GL_LINEAR;
- **GL_TEXTURE_WRAP_S -** параметр param встановлює значення координати s, якщо вона не входить у відрізок [0,1]. При значенні GL_ REPEAT ціла частина s відкидається, і в результаті зображення розмножується по поверхні. При значенні GL_CLAMP використовуються крайні значення: 0 або 1. Значення за замовчуванням: **GL_REPEAT;**
- **GL_TEXTURE_WRAP_T** аналогічно попередньому значенню, тільки для координати t.

Використання режиму **GL_NEAREST** значно підвищує швидкість накладення текстури, однак при цьому знижується якість, тому що на відміну від GL_LINEAR інтерполяція не здійснюється.

Для того, щоб визначити, як текстура буде взаємодіяти з матеріалом, із якого зроблений об'єкт, використовуються команди

void **glTexEnv[i f]** (GLenum target, GLenum pname, GLtype param),

void **glTexEnv[i f]v** (GLenum target, GLenum pname, GLtype *params).

Параметр target повинен дорівнювати GL_TEXTURE_ENV, а в якості pname розглянемо тільки одне значення GL_TEXTURE_ENV_MODE, яке найчастіше застосовується. Параметр param може дорівнювати:

- **GL_MODULATE -** кінцевий колір знаходиться як добуток кольору точки на поверхні і кольори відповідної їй точки на текстурі;
- **GL_REPLACE -** у якості кінцевого кольору використовується колір точки на текстурі;
- **GL_BLEND -** кінцевий колір знаходиться як сума кольору точки на поверхні і кольору відповідної їй точки на текстурі з урахуванням їхньої яскравості.

Координати текстури

Перед нанесенням текстури на об'єкт залишилося встановити відповідність між точками на поверхні об'єкта і на самій текстурі. Задавати цю відповідність можна двома методами: окремо для кожної вершини або відразу для усіх вершин, задавши параметри спеціальної функції відображення.

Перший метод реалізується за допомогою команд

void **glTexCoord[1 2 3 4][s i f d]** (type coord),

void **glTexCoord[1 2 3 4][s i f d]v** (type *coord).

Частіше усього використовуються команди вигляді glTexCoord2. .(type s, type t), які задають поточні координати текстури. Знаходження відповідних координат текстури є досить трудомістким заняттям, тому в бібліотеці GLU крім команд, що проводять побудову таких примітивів, як сфера, циліндр і диск, передбачене також накладення на них текстур. Для цього досить викликати команду

void **gluQuadricTexture** (GLUquadricObj *quadObject, GLboolean textureCoords)

з параметром textureCoords , який дорівнює GL_TRUE, і тоді поточна текстура буде автоматично накладатися на примітив.

Другий метод реалізується за допомогою команд

void **glTexGen[i f d]** (GLenum coord, GLenum pname, GLtype param),

void **glTexGen[i f d]v** (GLenum coord, GLenum pname, const GLtype *params).

Параметр coord визначає , для якої координати задається формула, і може приймати значення GL_S,GL_T; pname визначає тип формули і може дорівнювати GL_TEXTURE_GEN_MODE, GL_OBJECT_PLANE, GL_EYE_PLANE. За допомогою params задаються необхідні параметри, а param може бути рівним GL_OBJECT_LINEAR, GL_EYE_LINEAR, GL_SPHERE_MAP. Розгляд усіх можливих комбінацій значень аргументів цієї команди зайняло б занадто багато місця, тому як приклад розглянемо, як можна задати дзеркальну текстуру. При такому накладенні текстури зображення будуть наче відбиватися від поверхні об'єкта, викликаючи цікавий оптичний ефект. Для цього спочатку треба створити два цілочислені масиви коефіцієнтів s_coeffs і t_coeffs із значеннями (1,0,0,1) і (0,1,0,1) відповідно, а потім викликати команди:

```
glEnable (GL_TEXTURE_GEN_S);
glTexGeni (GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGendv (GL_S, GL_EYE_PLANE, s_coeffs);
```

і такі ж команди для координати t із відповідними змінами.

3.6 Формування додатків

Додаток містить інформацію в основному прикладного характеру, що може знадобитися при вивченні описаного вище матеріалу.

Стандартні геометричні примітиви

Розглянемо стандартні команди побудови примітивів, що реалізовані в бібліотеках GLU і GLUT.

Як уже було сказано, щоб побудувати примітив із бібліотеки GLU, треба спочатку створити покажчик на quadric-об'єкт за допомогою команди gluNewQuadric( ), а потім викликати одну з команд gluSphere( ), gluCylinder( ), gluDisk(), gluPartialDisk( ). Розглянемо ці команди окремо:

void **gluSphere** (GLUquadricObj *qobj, GLdouble radius, GLint slices, GLint stacks) будує сферу з центром на початку координат і радіусом

radius. При цьому число розбивок сфери навколо осі z задається параметром slices, а уздовж осі z - параметром stacks;

void **gluCylinder** (GLUquadricObj *qobj**,**GLdouble baseRadius**,**GLdouble topRadius**,** GLdouble height**,** GLint slices**,** GLint stacks) будує циліндр без основ, поздовжня вісь паралельна осі z, задня основа має радіус baseRadius, і розташована в площині z=0, передня основа має радіус topRadius і розташована в площині z= height. Якщо задати один із радіусів рівним нулю, то буде побудований конус.

Параметри slices і stacks мають аналогічний зміст, що й у попередній команді.

void **gluDisk** (GLUquadricObj *qobj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops) будує плоский диск (тобто, коло) із центром на початку координат і радіусом outerRadius. При цьому, якщо значення innerRadius ненульове, то в центрі диска буде знаходитися отвір радіусом innerRadius. Параметр slices задає число розбивок диска навколо осі z, а параметр loops - число концентричних основ, перпендикулярних осі z.

void **gluPartialDisk** (GLUquadricObj *qobj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops, GLdouble startAngle, GLdouble sweepAngle)- відмінність цієї команди від попередньої полягає в тому, що вона будує сектор кола, початковий і кінцевий кути якого відраховуються проти годинникової стрілки від додатного напрямку осі y і задаються параметрами startAngle і sweepAngle. Кути вимірюються в градусах.

Команди, що проводять побудову примітивів із бібліотеки GLUT, реалізовані через стандартні примітиви OpenGL і GLU. Для побудови потрібного примітива досить викликати відповідні команди.

void **glutSolidSphere** (GLdouble radius, GLint slices, GLint stacks)

void **glutWireSphere** (GLdouble radius, GLint slices, GLint stacks)

Команда glutSolidSphere() будує сферу, а glutWireSphere()-каркас сфери радіусом radius. Інші параметри мають аналогічний зміст, що й у попередніх командах.

void **glutSolidCube** (GLdouble size)

void **glutWireCube** (GLdouble size)

Ці команди будують куб або каркас куба з центром на початку координат і довжиною ребра size.

void **glutSolidCone** (GLdouble base, GLdouble height, GLint slices, GLint stacks)

void **glutWireCone** (GLdouble base, GLdouble height, GLint slices, GLint stacks)

Ці команди будують конус або його каркас, висотою height і радіусом основи base, розташованої вздовж осі z. Основа знаходиться в площині z=0. Інші параметри мають аналогічний зміст, що й у попередніх командах.

void **glutSolidTorus** (GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings)

void **glutWireTorus** (GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings)

Ці команди будують тор або його каркас у площині z=0. Внутрішній і зовнішній радіуси задаються параметрами innerRadius, outerRadius. Параметр nsides задає число сторін у кільцях, які складають ортогональний переріз тора, а rings-число радіальних розбивок тора.

void **glutSolidTetrahedron** (void)

void **glutWireTetrahedron** (void)

Ці команди будують тетраедр (правильну трикутну піраміду) або його каркас, при цьому радіус описаної сфери навколо нього дорівнює 1.

void **glutSolidOctahedron** (void)

void **glutWireOctahedron** (void)

Ці команди будують октаедр або його каркас. Радіус описаної навколо нього сфери дорівнює 1.

void **glutSolidDodecahedron** (void)

void **glutWireDodecahedron** (void)

Ці команди будують додекаедр або його каркас, радіус описаної навколо нього сфери дорівнює квадратному кореню з трьох.

void **glutSolidIcosahedron** (void)

void **glutWireIcosahedron** (void)

Ці команди будують ікосаедр або його каркас, радіус описаної навколо нього сфери дорівнює 1.

Формування додатка в середовищі MS Visual C++ 5.0

Перед початком роботи необхідно скопіювати файли glut.h, glut32.lib glut32.dll у каталоги ..\MSVC\Include\Gl, ..\MSVC\Lib, ..\Windows\System відповідно. Також у цих каталогах треба перевірити наявність файлів gl.h, glu.h, opengl32.lib, glu32.lib, opengl32.dll, glu32.dll, які входять до складу Visual C++ і Windows. При використанні команд із бібліотеки GLAUX до перерахованих файлів треба додати glaux. h, glaux. lib.

Для створення додатка треба виконати такі дії:

- Створення проекту: для цього треба вибрати *File->New->Projects->Win32 Console Application*, набрати ім'я проекту, OK.
- У вікні , що з'явилося , вибрати *'An empty* project', Finish,OK.

- Текст програми можна або розмістити в створеному текстовому файлі (обравши *File->New->Files->Text File*), або додавши файл із розширенням *. с або *. cpp у проект (обравши *Project->Add To Project->Files*).
- Підключити до проекту бібліотеки OpenGL. Для цього треба вибрати *Project->Settings->Link* і в поле *Object/library modules* набрати назви потрібних бібліотек: opengl32. lib, glu32. lib, glut32. lib і, якщо треба, glaux. lib.
- Для компіляції вибрати *Build->Build program. exe*, для виконання - *Build->Execute program. exe*.
- Щоб при запуску не з'являлося текстове вікно, треба вибрати *Project->Settings->Link* і в поле *Project Options* замість ‘subsystem:console' набрати ‘subsystem:windows', і набрати там же рядок ‘/entry:mainCRTStartup'
- Коли програма готова, рекомендується перекомпілювати її в режимі ‘Release' для оптимізації по швидкодії та об'єму. Для цього спочатку треба вибрати *Build->Set Active Configuration…* і відзначити ‘…-Win32Release ’, а потім наново підключити необхідні бібліотеки.

Формування додатка в середовищі Borland C++

Як і для Visual C++, спочатку треба забезпечити наявність файлів glut.h, glut32.lib, glut32.dll у каталогах ..\Borland\Include\Gl, ..\Borland\Lib, ..\Windows\System відповідно. Також у цих каталогах треба перевірити наявність файлів gl.h, glu.h, opengl32.lib, glu32.lib, opengl32.dll, glu32.dll, що звичайно входять до складу Borland++ і Windows. При цьому треба враховувати, що версії Microsoft файлів opengl32.lib, glu32.lib, glut32.lib для Borland C++ не підходять і варто використовувати тільки сумісні версії. Щоб створити такі версії, треба використовувати стандартну програму ‘implib', що знаходиться в каталозі ..\Borland\Bin. Для цього треба виконати команди вигляду

implib ..\Borland\Lib\filename.lib ..\filename.dll

для перерахованих файлів, які формують потрібний *. lib файл із відповідного *. dll файла. Крім того, треба відзначити, що компілятор Borland не може з невідомих причинах використовувати файл glaux. lib, що входить до складу Borland++5.02, при компіляції додатків, що використовують бібліотеку GLAUX, тому можливо від цієї бібліотеки прийдеться відмовитися. Для створення додатка треба виконати такі дії:

- Створення проекту: для цього треба вибрати *Project->New Project* і заповнити поля у вікні *Target Expert* у такий спосіб: у поле *Platform* вибрати Win32,у поле *Taget Model* вибрати *Console*, натиснути *Advanced* і скасувати вибір пунктів ‘ *. rc ‘ і ‘ *. def ‘.

- Підключити до проекту бібліотеки OpenGL. Для цього треба вибрати у вікні проекту назву файла проекту (*. exe) і натиснувши праву кнопку миші вибрати в контекстному меню пункт *Add node*. Потім треба визначити положення файлів opengl32. lib, glu32. lib, glut32. lib.

4 Рекомендації до сканування фотореалістичних зображень

Скануванню для наступної електронної публікації підлягає графічний матеріал, який існує у вигляді твердої копії. Це фотографії, рисунки, схеми, креслення, логотипи, ескізи.

Щоб уникнути втрати інформації через часткову некоректну роботу програми сканування , розміри прямокутного вікна області сканування необхідно задавати на 5-10% більшими від розмірів даної області для подальшого кадрування .

Необхідно відразу вибрати той колірний режим (RGB, Grayscale, Bitmap), який у найменшій мірі необхідний для кінцевого зображення.

Старі або ушкоджені оригінали варто сканувати зі збільшеною роздільною здатністю, у кольорі, з наступною ретушшю і зменшенням у графічному редакторі.

4.1 Вибір роздільної здатності для сканування

Головне правило: для одержання максимальної чіткості скануйте з розподільною здатністю, кратною оптичній роздільній здатності сканера. Наприклад, якщо у вас сканер з оптичною здатністю 600ppi (600x1200ppi), вибирайте роздільну здатність 75, 150, 300, 600, 1200 ppi - сканеру простіше розділити 600 на 2, 4, 8 ніж на 1.2 або 1. 33333... , тому якість обробки буде вищою.

Рекомендуються такі значення роздільної здатності

Таблиця. 4.1 - Рекомендована роздільна здатність

Розпізнавання тексту 75 – 300	**75 - 300**
Бланки, візитки, фотографії, журнали для виведення на екран і на принтер	**300 - 600**
Те ж, якщо картинка маленька і її потрібно збільшити	**600 - 1200**
Веб-дизайн	**75 - 300**
Професійні задачі: сканування якісних оригіналів для поліграфії	**600 - 2000**
Професійні задачі: сканування слайдів і негативних фотоплівок	**1200**

Якщо зображення призначене для типографського відтворення, то роздільна здатність зображення повинна мінімум у півтора, максимум у два рази, перевищувати щільність растра (лініатури) пластини при використанні стандартної АМ технології растрування.

При використанні ЧМ-технології (стохастичного) растрування лініатура збільшується на 1.2. Це відноситься і до струминних принтерів.

4.2 Проробка тіней і детальність зображення

Якість проробки тіней і яскравих ділянок прямо залежить від класу сканера і діапазону оптичної щільності останнього, однак, навіть на сканері бізнес - класу можна одержувати зображення дуже високої якості, якщо правильно оцінити вихідний оригінал і підібрати установки сканування.

Щоб не втратити деталі в тіньових областях, необхідно встановити значення точки чорного(мінімальна яскравість - shadow) трохи вищим нуля, а для якісного відтворення відблисків і дуже яскравих деталей, необхідно встановити значення точки білого(максимальна яскравість - highlight) біля 255.

Після установки граничних значень яскравості варто відкоригувати середні тони (параметр Gamma)

Для одержання максимальної виразності рекомендуються такі значення:

- *Контрастні оригінали: 1.2 - 1.3*
- *Збалансовані оригінали 1.4 - 1.6*
- *Темні і малоконтрастні оригінали 1.6 - 1.9*

У більшості випадків, функція автопідбору яскравості в програмі сканування справляється з задачею краще людського ока, встановлюючи оптимальні значення. Стежте за тим, щоб виділена для сканування область не містила сторонніх деталей: край фотоплівки, фотографії, притискну кришку і т.д., тому що автопідбір здійснюється на основі виділеної області.

4.3 Видалення друкарського растру

Якщо ви скануєте оригінали, надруковані в друкарні з застосуванням стандартної технології растрування (АМ - технологія), друкарський растр, що міститься в зображенні, може створити проблеми при подальшому опрацюванні, а особливо, при повторному раструванні: конфлікт нової і залишкової растрової структур (інтерференція) практично завжди викликає появу муару в зображенні.

Більшість інтерфейсів сканерів мають вбудовані фільтри усунення растра (descreen), однак користуватися ними треба з великою обережністю: при неправильному підборі роздільної здатності і установок дерастрування , можна знизити чіткість зображення.

Насамперед, потрібно оцінити частоту растра (лініатуру) зображення, щоб правильно зазначити її в установках фільтра. Інтерфейс сканера, як

правило, пропонує варіанти найбільше поширених установок, які в більшості випадків дозволяють ефективно видалити друкарський растр із мінімальними втратами якості.

Практичні поради:

- *При скануванні встановіть розподілення трохи більшим, ніж лініатура оригіналу.*
- *Додайте монохроматичний шум в область періодичної структури зображення і трохи розмийте її фільтром Gaussian Blur (Размиття по Гауссу) або радіально.*
- *Відновіть різкість зображення після розмиття.*
- *У ділянках однорідного кольору (літери, логотипи і ін.) растр ефективно віддалиться за допомогою "заливання" області основним кольором.*

При скануванні растрових поліграфічних зображень із книг і журналів можливо поява муару (геометричних регулярних візерунків, сітки). Фахівцями вироблений ряд досить ефективних прийомів з його нейтралізації, наприклад, легкий поворот оригіналу на сканері, або збільшення роздільної здатності сканування з наступною обробкою цифрового зображення в графічному редакторі, яка полягає в наступному посиленні різкості і зменшенні роздільної здатності.

4.4 Підвищення чіткості зображення

Ви не зможете повернути зображенню втрачену різкість, якщо оригінал споконвічно був нечітким, однак навіть у безнадійних випадках, використовуючи описані методи, можна поліпшити сприйняття зображення і створити відчуття різкості. Процес сканування, як правило, приводить до невеликої втрати різкості, а процедура видалення растра це тільки збільшує.

Для підвищення чіткості існує декілька способів:

Накладення нерізкої маски (Unsharp Masking)

Застосовуйте такі параметри:

- *Amount(Розмір): 100 (двократне збільшення різкості) для якісних зображень і біля 200% для нерізких.*
- *Radius(Радіус): Розділіть вихідне розподілення на 200 для одержання оптимального значення. Звичайно не варто перевищувати значення 1.0, для маленьких зображень задавайте радіус не більший 0.5*
- *Threshold(Поріг): від 2 до 6 (від 4 до 6 для портретів і великих планів)*

Конвертуйте зображення в режим LAB і підсильте контраст і детальність у каналі Luminance(Яскравість), застосувавши для цього нерізке маскування, а потім конвертуйте зображення назад у свою колірну модель. Таким способом можна одержати добрі результати.

Сприйняття чіткості може також погіршуватися через порушення структури кольору в зображенні. У цьому випадку можливо прийдеться серйозно попрацювати над чистотою кольорів, підсилити контраст і насиченість, поліпшивши в такий спосіб сприйняття чіткості зображення.

Додавання монохромного шуму в окремих частинах зображення може також поліпшити сприйняття чіткості, але необхідно стежити за тим, щоб зображення не стало зернистим.

Процес поліграфічного відтворення злегка зм'якшує зображення, роблячи його менш чітким, тому при підготуванні до друку варто злегка збільшити різкість більше оптимальної, щоб одержати результат необхідної якості.

5 Аналіз форматів графічних файлів та рекомендації до їх вибору

У зв'язку з технічним прогресом виникнула потреба в перенесенні на платформу ПК і адаптації різних форматів кодування і збереження графічної інформації з інших платформ (наприклад Macintosh, де подібні розробки розвиваються протягом уже другого десятка років), або ж в розробці власних, орієнтованих на ПК графічних форматів, що цілком враховують всі особливості архітектури відеоадапторів.

Таблиця. 5.1 - Поширені формати файлів растрової графіки

Формат	Макс. глиб. Кольору	Макс. кількість кольорів	Макс. розмір зображення, піксел	Методи стиснен-ня	Кодування декількох зображень
BMP	24	16'777'216	65535 x 65535	RLE*	-
GIF	8	256	65'535 x 65535	LZW	+
JPEG	24	16'777'216	65535 x 65535	JPEG	-
PCX	24	16'777'216	65535 x 65535	RLE	-
PNG	48	281'474'976'710'656	2'147'483'647 x 2 147 483 647	Deflation (варіант LZ77)	-
TIFF	24	16'777'216	усього 4'294'967'295	LZW, RLE і інші*	+

* Стиск виконується факультативно.

5.1 Формат BMP

Для імені файла, поданого в BMP-форматі, частіше всього використовується розширення bmp, хоча деякі файли мають розширення rle, що означає run length encoding (кодування довжини серій). Розширення RLE вказує на те, що здійснено стиснення растрової інформації файла

одним із двох способів стиснення rle, які припустимі для файлів BMP-формату.

У файлах BMP інформація про колір кожного піксела кодується 1, 4, 8, 16 або 24 бітами (біт/піксел). Зображення при глибині 1 біт/піксел може мати всього два кольори, а при глибині 24 біт/піксел - більше 16 млн. різних кольорів.

Таблиця. 5.1. - Структура файла BMP

Структура файла BMP
Заголовок файла растрової графіки (14 байт) Сигнатура файла BMP (2 байт) Розмір файла (4 байт) Не використовується (2 байт) Не використовується (2 байт) Місцезнаходження даних растрового масиву (4 байт)
Інформаційний заголовок растрового масиву (40 байт) Довжина заголовка (4 байт) Ширина зображення (4 байт) Висота зображення (4 байт) Число колірних площин (2 байт) Біт/піксел (2 байт) Метод стиску (4 байт) Довжина растрового масиву (4 байт) Горизонтальне розподілення (4 байт) Вертикальне розподілення (4 байт) Число кольорів зображення (4 байт) Число основних кольорів (4 байт)
Таблиця кольорів (довжина змінюється от 8 до 1024 байт)
Власне дані растрового масиву

На приведеній схемі показана структура типового BMP-файла, що містить 256-кольорове зображення (із глибиною 8 біт/піксел). Файл розбитий на чотири частини : заголовок файла растрової графіки, інформаційний заголовок растрового масиву, таблиця кольорів і власне дані растрового масиву. Заголовок файла растрової графіки містить інформацію про файл, у тому числі про адресу, з якої починається область даних растрового масиву. В інформаційному заголовку растрового масиву містяться зведення про зображення, що зберігається у файлі, наприклад, його висоту і ширину в пікселах. У таблиці кольорів подані значення основних кольорів RGB (червоний, зелений, синій), які використовуються у зображенні. Програми, що зчитують і відображають BMP-файли, у випадку використання відеоадаптерів, які не дозволяють відображати

більше 256 кольорів, для точної передачі кольору можуть програмно встановлювати такі значення RGB у колірних палітрах адаптерів.

Формат даних растрового масиву у файлі BMP залежить від числа біт, які використовують для кодування даних про колір кожного піксела. При 256-кольоровому зображенні кожний піксел у тій частини файла, де містяться власне дані растрового масиву, описується одним байтом (8 біт). Цей опис піксела не представляє значень кольорів RGB, а служить покажчиком для входу в таблицю кольорів файла. Таким чином, якщо в якості першого значення кольору RGB у таблиці кольорів файла BMP зберігається R/G/B=255/0/0, то значенню піксела 0 у растровому масиві буде поставлений у відповідність яскраво-червоний колір. Значення пікселів зберігаються в порядку їхнього розташування зліва направо, починаючи (як правило) із нижнього рядка зображення. Таким чином, у 256-кольоровому BMP-файлі перший байт даних растрового масиву являє собою індекс для кольору піксела, що знаходиться в нижньому лівому кутку зображення; другий байт представляє індекс для кольору сусіднього справа піксела і т.д. Якщо число байтів у кожному рядку непарне, то до кожного рядка добавлять додатковий байт, щоб вирівняти дані растрового масиву .

Не всі файли BMP мають структуру, подібну показаній на схемі. Наприклад, файли BMP із глибиною 16 і 24 біт/піксел не мають таблиць кольорів; у цих файлах значення пікселів растрового масиву безпосередньо характеризують значення кольорів RGB. Також можуть розрізнятися внутрішні формати збереження окремих розділів файла. Наприклад, інформація растрового масиву в деяких 16 і 256-кольорових BMP-файлах може стискуватися за допомогою алгоритму RLE, який заміняє послідовності ідентичних пікселів зображення на лексеми.

5.2 Формат PCX

PCX став першим стандартним форматом графічних файлів для збереження файлів растрової графіки в комп'ютерах IBM PC. На цей формат, що застосовувався в програмі Paintbrush фірми ZSoft, на початку 80-х рр. фірмою Microsoft була придбана ліцензія, і потім він поширювався разом із виробами Microsoft. Надалі формат був перетворений у Windows Paintbrush і почав поширюватися з Windows. Хоча область застосування цього популярного формату скорочується, файли формату PCX широко поширені сьогодні.

Файли PCX розділені на такі три частини: заголовок PCX, дані растрового масиву і факультативної таблиці кольорів. 128-байтний заголовок PCX містить декілька полів, у тому числі поля розміру зображення і числа біт для кодування інформації про колір кожного піксела. Інформація растрового масиву стискується з використанням простого методу стиску RLE; факультативна таблиця кольорів наприкінці файла містить 256 значень кольорів RGB, що визначають кольори

зображення. Кодування кольору кожного піксела в сучасних зображеннях PCX може виконуватисяз глибиною 1, 4, 8 або 24 біт.

5.3 Формат TIFF

Розроблений корпораціями Aldus і Microsoft. Цей формат підтримується цілим рядом систем підготування документації і є найбільш реальним претендентом на стандарт для збереження і транспортування растрових зображень.

Якщо PCX - один із найпростіших для декодування форматів растрової графіки, то TIFF (Tagged Image File Format) - один із найскладніших.
Файли TIFF мають розширення tiff. Кожний файл починається 8-байтовим заголовком файла зображення (IFH), найважливіший елемент якого - каталог файла зображення (Image File Directory, IFD) - служить покажчиком до структури даних. IFD являє собою таблицю для ідентифікації однієї або декількох порцій даних змінної довжини, які називаються тегами. Теги зберігають інформацію про зображення. У специфікації формату файл TIFF визначено більше як 70 різних типів тегів. Наприклад, тег одного типу зберігає інформацію про ширину зображення в пікселах, другого - інформацію про його висоту. У тегі третього типу зберігається таблиця кольорів (при необхідності), а тег четвертого типу містить самі дані растрового масиву. Зображення, закодоване у файлі TIFF, цілком визначається його тегами, і цей формат файла легко розширюється, оскільки для надання файлу додаткових властивостей досить лише визначити додаткові типи тегів.

Основною концепцією формату TIFF є колірна модель зображення. Під цим розуміється набір характеристик зображення, що визначають спосіб представлення кольору. Стандартизовано такі колірні моделі:

· дворівневе зображення (bi-level image);
· монохромне зображення (gray-scale image);
· індексоване кольорове зображення (paletted color image);
· повне кольорове зображення (full RGB image).

TIFF є відкритим форматом і дозволяє створити будь-яку модель зображення. Природно, що вибір необхідної моделі визначається задачею, яка розв'язується .Наприклад, дворівнева модель найбільш зручна в системах підготування документації. Індексоване кольорове зображення сумісне з форматом збереження графічної інформації в найбільше поширених у даний час растрових графічних дисплеях.

Крім інформації про модель зображення формат TIFF містить метричні характеристики, такі як : розміри зображення, щільність (кількість пікселів на одиницю довжини) . Ці характеристики особливо корисні в системах підготування документації. TIFF не накладає практично ніяких обмежень на параметри зображення. Так, наприклад розміри зображення можуть досягати 4 мільярдів. Кількість бітів на піксел обмежено цим же числом.

Формат TIFF дозволяє зберегти в одному файлі будь-яку кількість зображень. Крім того, є можливість зберігати декілька копій одного зображення з різними характеристиками. Так, наприклад, можна мати декілька варіантів зображення, які відрізняються різною щільністю, що корисно знову ж у видавничих системах для роботи з декількома принтерами.

У форматі TIFF є можливість упаковувати зображення різними методами. У тому числі зображення може зберігатися і у неупакованому вигляді, що представляється зручним, тому що, наприклад, при формуванні зображення важливий довільний доступ до будь-якого елемента зображення за досить малий час. Одним із методів кодування є LZW (Lempel, Ziv & Welch), що дає високий коефіцієнт стиску.

Упорядкування програм, що розрізняють усі типи тегів, досить складне. У більшості програм для читання файлів TIFF реалізується тільки підмножина тегів. Саме тому сформований однією програмою файл TIFF іноді не може бути прочитаний іншою. Крім того, програми, що формують файли TIFF, можуть визначати власні типи тегів, що мають зміст тільки для них. Програми читання файлів TIFF можуть пропускати незрозумілі для них теги . Завжди існує небезпека, що це вплине на зовнішній вигляд зображення.

Ще одна складність полягає в тому, що файл TIFF може містити декілька зображень, кожне з яких супроводжують власний IFD і набір тегів. Дані растрового масиву у файлі TIFF можуть стискуватися з використанням декількох методів, тому в надійній програмі для читання файлів TIFF повинні бути засоби розпакування RLE, LZW.

Незважаючи на свою складність, файловий формат TIFF залишається одним із кращих для передачі растрових масивів з однієї платформи на іншу завдяки своїй універсальності, що дозволяє кодувати у двійковому вигляді практично будь-яке зображення без втрати його візуальних або будь-яких інших атрибутів.

5.4 Формат GIF

Структура файла GIF залежить від версії GIF-специфікації, якій відповідає файл. Незалежно від номера версії, файл GIF починається з 13-байтів заголовка, що містить сигнатуру, яка ідентифікує цей файл у якості GIF-файла, номер версії GIF і іншу інформацію. Якщо файл зберігає всього одне зображення, за заголовком звичайно розташовується загальна таблиця кольорів, яка визначає кольори зображення. Якщо у файлі зберігається декілька зображень (формат GIF, аналогічно TIFF, дозволяє в одному файлі кодувати два і більше зображень), то замість загальної таблиці кольорів кожне зображення супроводжується локальною таблицею кольорів.

У файлі GIF87a за заголовком і загальною таблицею кольорів розміщається зображення, що може бути першим із декількох, які розташовуються підряд . Кожне зображення включає 10-байтів опису зображення, розташованої слідом за ним локальної таблиці кольорів і біт растрового масиву. Для підвищення ефективності використання пам'яті дані растрового масиву стискуються за допомогою алгоритму LZW.

Файли GIF89a мають аналогічну структуру, але вони можуть містити факультативні блоки розширення з додатковою інформацією про кожне зображення. У специфікації GIF89a визначені чотири типи блоків розширення. Це блоки розширення для керування графікою, які описують, як зображення повинно виводитися на екран (наприклад, чи накладається воно на попереднє зображення подібно діапозитиву або просто заміняє його); блоки розширення зі звичайним текстом, що містять текст, який відображається разом із графікою; блоки розширення для коментарю, які містять коментар у коді ASCII; і блоки розширення прикладних програм, у яких зберігається інформація, що належить тільки програмі , яка створила цей файл. Блоки розширення можуть знаходитися практично в будь-якому місці файла після загальної таблиці кольорів.

У GIF'ах також існують *блоки спеціальної інформації* - фреймы.

Відомості , які в них закладені, визначають час демонстрації кожного кадру, наявність і розмір локальних колірних палітр, методи видалення зображення з екрана та інші параметри. Крім того, у керуючому фреймі «Розширення коментарів» можна розмістити довільний текст, який складається із символів першої половини таблиці ASCII (керуючі, псевдографіка і символи англійського алфавіту). Частіше всього коментар використовується для вказання імені автора і дати створення файла - досить примітивний, але забезпечує хоч якійсь спосіб захистити свої роботи від несанкціонованого копіювання.

Однієї з особливостей формату GIF є і те, що розміри і місця різних блоків інформації, будь-то дані зображення або керуючі розширення, не піддаються ніякому попередньому обчисленню. Це значною мірою ускладнює реалізацію алгоритмів перегляду GIF-зображень.

Основні переваги GIF полягають у широкому поширенні цього формату і його компактності. Але йому властиві два досить серйозних недоліки. Один із них полягає в тому, що в зображеннях, які зберігаються у вигляді GIF-файла, не може бути використано більш 256 кольорів. Другий, можливо, ще більш серйозний, полягає в тому, що розроблювачі програм, які використовують у них формати GIF, повинні мати ліцензійну угоду з CompuServe і вносити плату за кожний екземпляр програми.

5. 5 Формат PNG

Формат PNG (Portable Network) був розроблений для заміни GIF, щоб обминути юридичні перешкоди, що стоять на шляху використання GIF-

файлів. PNG успадкував багато можливостей GIF і, крім того, він дозволяє зберігати зображення з більшою кількістю кольорів.

PNG- значно більш новий формат для стиску як індексованих , так і повнокольорових зображень. При роботі з індексованими зображеннями, у деяких випадках, здатний дати півторакратний виграш у порівнянні з GIF. Формат може застосовувати різну кількість біт для кодування інформації. Однак, багато графічних редакторів, хоча і здатні відкрити PNG файл, можуть зберігати дані в тільки в кодуванні 24 або 8 біт . Навіть якщо у зображенні використовується усього 4 кольори , вони все одно будуть кодуватися 8 бітами. PNG не підтримується браузерами ранніх версій. Більш того, різні браузеры по різному відображають складову яскравості PNG картинки. Ефективна сфера застосування формату-середні і великі зображення зі значними рівномірно залитими ділянками. Наприклад, скриншот вікна драйвера Windows (440x430 pix) стискується PNG до 4,7 Кб при 11 кольорах і дизерингі 10%. Формат GIF дозволяє одержати 8,5 Кб при тих же установках.

5.6 Формат JPEG

Формат файла JPEG (Joint Photographic Experts Group) був розроблений компанією C-Cube Microsystems як ефективний метод збереження зображень із великою глибиною кольору, наприклад, тих, які отримують при скануванні фотографій із численними відтінками кольору. Найбільша відмінність формату JPEG від інших форматів полягає в тому, що в JPEG використовується алгоритм стиску з втратами (а не алгоритм без втрат) інформації. Алгоритм стиску без втрат так зберігає інформацію про зображення, що розпаковане зображення в точності відповідає оригіналу. При стиску з втратами приноситься в жертву частина інформації про зображення, щоб досягти більшого коефіцієнта стиску. Розпаковане зображення JPEG рідко відповідає оригіналу абсолютно точно, але дуже часто ці розходження настільки незначні, що їх ледь можна (якщо взагалі можна) виявити.

Процес стиску зображення JPEG досить складний і часто для досягнення прийнятної продуктивності вимагає спеціальної апаратури. Спочатку зображення розбивається на квадратні блоки зі стороною розміром 8 піксел. Потім робиться стиск кожного блока окремо за три кроки. На першому кроці за допомогою формули дискретного косинусоїдального перетворення Хури (DCT) здійснюється перетворення блока 8x8 з інформацією про пікселі у матрицю 8x8 амплітудних значень, які відображають різні частоти (швидкості зміни кольору) у зображенні. На другому кроці значення матриці амплітуд діляться на значення матриці квантування, яка зміщена так, щоб відфільтрувати амплітуди, які незначно впливають на загальний вигляд зображення. На третьому, останньому кроці, квантована матриця амплітуд стискується з використанням алгоритму стиску без утрат.

Оскільки в квантованій матриці відсутня значна доля високочастотної інформації, наявної у вихідної матриці, то перша часто стискується до половини свого початкового розміру або навіть ще більше. Реальні фотографічні зображення часто зовсім неможливо стиснути за допомогою методів стиску без втрат, тому 50%-ний стиск варто визнати досить гарним. З іншого боку, застосовуючи методи стиску без втрат, можна стискувати деякі зображення на 90%. Такі зображення погано підходять для стиску методом JPEG.

При стиску методом JPEG втрати інформації відбуваються на другому кроці процесу. Чим більше значення в матриці квантування, тим більше відкидається інформації з зображення і тим більш щільно стискується зображення. Компроміс полягає в тому, що більш високі значення квантування приводять до гіршої якості зображення. При формуванні зображення JPEG користувач установлює показник якості, розмір котрого "управляє" значеннями матриці квантування. Оптимальні показники якості, які забезпечують кращий баланс між коефіцієнтом стиску і якістю зображення, різні для різних зображень і звичайно можуть бути знайдені тільки методом проб і помилок.

5.7 Рекомендації для вибору графічних форматів

1. Формат JPEG краще підходить для фотографічних зображень, а формат GIF - для зображень, які містять рівні кольори, такі, як ілюстрації і штрихова графіка

2. Розміри зображень підбирайте за допомогою відповідних графічних редакторів. Хоча можливо змінити в редакторі видимий розмір зображень, фактичний розмір файла при цьому не змінюється; наприклад, якщо ви вставили в веб-сторінку зображення 2x2 см із розміром файла 10 Кб, а потім зменшили його в редакторі в два рази, то його файл при цьому все одно залишиться розміром у 10 Кб.

3. Якщо ви створюєте зображення, що містить тільки чорний і білий колір, то збережіть його як чорно-біле зображення . Це допоможе скоротити розмір його файла. У Microsoft Image Composer ви можете зробити це, установивши при зберіганні колірний формат Black and White.

4. Якщо ви формуєте зображення, яке містить плавні переходи кольору або прямі лінії, і буде збережено у форматі GIF, то постарайтеся зробити їх горизонтальними. Це пояснюється тим, що для формату GIF виконується стиск по горизонтальних рядках, тому одноколірні лінії будуть стискуватися краще, що дозволить зменшити розмір файла

5. Перевірте, як виглядають ваші зображення при різній роздільній здатності і різній кількості кольорів екрана, а також при використанні різних браузерів.

6.Якщо ви працюєте з Adobe Photoshop і хочете зберегти зображення у форматі GIF, то змініть колірний режим RGB на Indexed Color (Індексовані кольори) і виберіть мережну палітру (опція Palette: Safety)без дифузії (опція Dither: none)

7. Якщо ви використовуєте Photoshop і зберігаєте зображення як GIF, і при цьому знаєте, що в зображенні менше 256 кольорів, то спробуйте використовувати точну палітру (Palette: Exact). Точні палітри містять рівно стільки кольорів, скільки є в зображенні, тим самим зменшуючи розмір файла.

8. При розробці зображень для веб-сторінок намагайтеся, щоб підготовлені вами зображення не перевищували 15-20 Кб (для орієнтування - типовий розмір більшості оптимізованих картинок 300x200 px складає 5 - 10 Кб). Якщо необхідно вставити зображення великого розміру, то краще окремо підготувати його зменшену копію, а вже з неї зробити посилання на велике зображення. Якщо буде необхідно, читач самостійно перегляне велике зображення в окремому вікні браузера.

9. GIF на відміну від JPEG підтримує анімаційні зображення.

10. Власний формат PSD пакета PhotoShop вимагає великих об'ємів пам'яті, тому якщо шари не мають значення, краще зберігати зображення у PNG.

11 . Не зберігайте проміжні результати в JPG! Це призведе до втрати якості зображення. Зберігайте в JPG тільки кінцевий результат .

12. Один із прийомів поєднати невеликі розміри та якість зображення полягає в розрізані зображення на ділянки і зберіганні кожної ділянки в найбільш доцільному для неї форматі. Наприклад, якщо поверх фотографії наноситься текст, то для нього неприпустимо використання формату JPG, оскільки в цьому випадку буде мати місце ореол навколо букв. В цьому випадку фотографія розрізається на декілька фрагментів таким чином, щоб напис виявився в окремому фрагменті невеликого розміру. Цей фрагмент доцільно запам'ятати в форматі GIF.

Отримані фрагменти потім розміщають в таблицю з невказаними розмірами, що гарантує їхнє щільне стикування між собою.

13. Якщо Вам потрібний прозорий фон, прийдеться використовувати GIF або PNG. При цьому необхідно пам'ятати, що зберігання в режимі anti-alisieng контурів приводить до утворення помітної кольорової кайми, якщо колір, обраний «прозорим» сильно відрізняється від кольору фону розміщення картинки. «Сгладжування» робить картинку приємнішою для очей шляхом додавання на межах колірних переходів пікселів середнього кольору. Крім того, небезпечні режими «взаємопроникнення» (Diffusion) або «змішування» (Dithering) різних кольорів, які використовуються для зменшення палітри. Можливо одержати «дірчастий» рисунок, якщо деякі його точки виявляться прозорими.

14. Підсилити стиснення і зменшити втрати якості в JPG можна попереднім застосуванням фільтра «Blur» («розмиття») , наявного в більшості графічних редакторів.

15. Найбільш помітний виграш від зберігання (GIF або progressiv JPG) спостерігається при виведенні великої за площою картинки. На жаль, це негативно позначається на деяких броузерах (наприклад, Netscape Navigator може просто підвиснути). Тому доцільно організувати попередній перегляд: спочатку завантажується картинка такого ж розміру, але з меншою розподільною здатністю або чорно-біла, а потім, поверх неї оригінал, збережений без чергування рядків (GIF) або з покроковим поліпшенням зображення (progressiv JPG). Цей спосіб здійснюється вказанням в тегі HTML <*img src=«... »*> параметра *lowsrc=«... »* із вказанням імені файла гіршої якості.

16. В таблиці приведені різни типи графічних файлів для різних застосувань.

Таблиця. 5.2 - Застосування графічних файлів

Тип зображення	GIF	JPEG	PNG
1. Фотографії (великого розміру)	-	так	-
2. Фотографії (малого розміру)	так	так	-
3. Векторні зображення	так	-	так
4. Діаграми і графіки	так	-	так
5. Текст у вигляді зображення	так	-	так
6. Скриншоти системних вікон Windows	так	-	так

6 Вибір колірної гами

6.1 Суміщення кольорів

Виділяють три групи кольорів за суміщенням - родинні, родинно-контрастні і контрастні.

Родинні кольори розташовуються в одній чверті колірного кола і мають хоча б один загальний колір, наприклад, жовтий, жовтогарячий і жовто-червоний. Існують чотири групи родинних кольорів - жовто-червоні, червоно-сині, синьо-зелені, зелено-жовті. При цьому в суміщенні не повинно бути одночасно двох контрастних кольорів.

Родинно-контрастні кольори розташовуються в двох сусідніх чвертях колірного кола і мають один загальний (головний) колір, два інші складові кольори - взаємододаткові. Існують чотири групи родинно-контрастних кольорів - жовто-червоні і червоно-сині, червоно-сині і синьо-зелені, синьо-зелені і зелено-жовті, зелено-жовті і жовто-червоні. Ці суміщення досить активні.

Контрастні кольори розташовуються в протилежних чвертях колірного кола. Око відразу відчуває такі суміщення, тому вони застосовуються там, де необхідно привернути увагу.

Колірна гармонія. Розглянемо основний чинник колірної гармонії - рівновага колірних тонів. Сумістити можна будь-які кольори за принципом контрасту, родинного контрасту . Два родинних кольори можуть бути урівноважені рівною кількістю присутніх в них одного з головних кольорів, тобто однаковою мірою можуть бути жовтуваті, зеленуваті і т.д. Гармонізація родинних кольорів можлива за умови ослаблення насиченості головних або інших кольорів або зменшення їхньої активності шляхом затемнення.

Наприклад, щоб родинні кольори (чисто жовтий і жовто-червоний) стали гармонічними, необхідно послабити насиченість жовтого сумішшю білого в такій кількості, скільки в жовто-червоному червоного кольору. Гармонічні суміщення родинних кольорів неактивні, спокійні, м'які, особливо якщо кольори слабко насичені і зближені за світлістю.

Гармонічні контрастні кольори розташовані на кінцях діаметрів колірного кола. Однак не всі контрастні кольори гармонічні.

Випадки триколірної гармонії. До двох гармонічних родинно-контрастних кольорів може бути доданий третій - головний колір, що їхріднить, але ослабленої насиченості. Кольори стануть попарно родинними-контрастними і попарно додатковими. Такі суміщення високогармонічні і колористично багаті. До двох гармонічних родинних кольорів може бути доданий один контрастний. Так, гармонія утвориться, якщо родинні зеленувато-жовтий і листяно-зелений кольори доповнити червоно-синім кольором, тобто додатковим проміжному двох перших.

На додаток до відзначених колірних суміщень існує множина монохромних суміщень. Монохромні - це ті кольори, що мають одну назву, але різну світлість і насиченість.

Ефектно сприймається сірий колір у суміщенні з чорним і білим. При побудові композиції на суміщенні хроматичних кольорів із чорним, білим і рідше сірим є свої особливості. Чорний колір має сильний вплив на хроматичні, особливо на середні за світлістю і високонасичені червоні і зелені кольори : ці кольори світлішають, насиченість їх зростає, вони як би світяться. Зі збільшенням площі чорного кольору в композиції цей ефект стає більш відчутним : білий колір, особливо білі обвідки хроматичних кольорів, надають композиції своєрідний колорит; кольори стають повітряними, пастельними.

6.2 Рекомендації до вибору кольорів

Колірна гама сторінки дає можливість виділити такі найбільше важливі частини сторінки як навігаційні елементи, змістовні блоки, акценти, взаємозв'язки і т.д Уміння правильно використовувати колір є атрибутом професіоналізму дизайнерів. Дизайнер завжди спочатку продумує колірну схему, потім тільки починає створювати графіку.

Суворих законів колірної гармонії не існує. Ті самі суміщення кольорів дозволяють домогтися різних ефектів у залежності від співвідношення в композиції кольорів, порядку їхнього чергування, а також від розмірів і форми колірних елементів.

Мистецтво оформлення приходить із досвідом і завжди залежить від кількості експериментів розроблювача.

При виборі кольорової гами необхідно притримуватись таких рекомендацій.

1. Використовуйте на сторінках максимум три-чотири кольори. Багато сайтів, які використовують сотні колірних відтінків болісно сприймаються і слабко запам'ятовуються. Якщо використовуються на сайті кольорові фотографії, то можна використовувати їхній домінуючий колір для заголовків.

2. Дотримуйтесь узгодженості відтінків. Використовуйте для різних типів даних різні відтінки. Наприклад, заголовки робіть синім кольором, попередження краще робити червоним, колір основного тексту повинний бути теж визначений однозначно. При цьому дуже зручно користуватися каскадними таблицями стилів, де можна заздалегідь визначити стиль оформлення сторінки.

3. Кольори фону і основного тексту повинні бути контрастними. При більшій контрастності зручніше буде читати текст. На темному фоні найкраще використовувати білий або жовтий кольори, при цьому блакитний читається складніше. При виборі кольору фону і тексту враховуйте аудиторію, з якою будете працювати. Для молоді з домашнім Інтернетом більш звичний нічний вихід в Інтернет. Для них краще використовувати темний фон і світлий текст. Якщо Ваша аудиторія складається з людей , які користуються Інтернетом на своїй роботі (в офісах, установах), варто використовувати світлий фон із темним текстом. За статистикою світлий фон краще темного.

4. Кольори Web-сайти, що розробляються для глобальної аудиторії, повинні бути вибрані , базуючись на розумінні їх культурного й емоційного значення. Web-дизайнер повинен серйозно поставитися до аналізу символіки кольору. Наприклад, червоний - колір вогню, пожежі не асоціюється зі спокоєм . Зелений капелюх у Китаї сигналізує зраду , але в багатьох інших регіонах (наприклад, в Індії), зелений колір асоціюється з процвітанням і благополуччям.

5. Кольори повинні підбиратися в залежності від аудиторії або цільової групи сайта. При розробці дизайну сайта для молодої аудиторії використовують більш яскраві кольори, тоді як для літніх людей повинні використовуватися більш помірні, спокійні і стримані кольори.

Дослідження показали, що чоловіки віддають перевагу синьому коліру над червоним, у той час як жінки навпаки - червоному над синім. Інший приклад - чоловіки віддають перевагу жовтогарячому кольору над жовтим, тоді як жінки віддають перевагу жовтому над жовтогарячому.

6. Необхідно враховувати колірні асоціації.

Мозок людини в життєвому процесі створює асоціації. При цьому, зовнішні подразники викликають часом дуже сильні реакції.

Визначено, що з кольорами пов'язані такі асоціації.

Червоний:

Збудження , енергія, пристрасть, бажання, швидкість, потужність, тепло, любов, агресія, небезпека, вогонь, кров, війна, насильство, все інтенсивне і жагуче.

Жовтий:

Радість, щастя, оптимізм, ідеалізм, уява, надія, сонячне світло, літо, золото, філософія, шахрайство, малодушність, зрадництво, ревнощі, жадібність, обман, хвороба, ризик.

Синій:

Спокій, гармонія, об'єднання, довіра, істина, консерватизм, безпека, чистота, порядок, лояльність, небо, вода, холод, технологія, депресія.

Жовтогарячий:

Енергія, баланс, тепло, ентузіазм, оживленість , експансивність, пишність, все, що вимагає уваги.

Зелений:

Природа, середовище, удача, відновлення, молодість, бадьорість, весна, щедрість, родючість, ревнощі, недосвідченість, заздрість, невдача.

Пурпурний:

Духовність, шляхетність, церемонія, незбагненність, перетворення, мудрість, просвітництво, жорстокість, зарозумілість.

Сірий:

Безпека, надійність, інтелект, скромність, гідність, завершеність, твердість, консерватизм, практичність, старість, сум.

Коричневий:

Земля, вогнище, будинок, на відкритому повітрі, надійність, зручність, витривалість, простота.

Білий:

Чистота, простота, сумирність, точність, невинність, молодість, народження, зима, сніг, стерильність, шлюб (західні культури), смерть (східні культури), холодний, клінічний.

Чорний:

Потужність, складність, формальність, елегантність, багатство,

таємниця, страх, зло, анонімність, невдоволення , глибина, стиль, зло, сум, гнів, метрополітен, гарний технічний колір, жалоба, смерть (західні культури).

Всі ці асоціації варто використовувати при оформленні сайта. Якщо відвідувача потрібно зарядити енергією використовуйте сильні відтінки. Якщо, навпаки, потрібно створити мирну атмосферу, краще використовувати пастельні тони, послабити агресивні кольори.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дуда Р., Харт П., Распознавание образов и анализ сцен . -М: Мир, 1976.
2. Пратт У. Цифровая обработка изображений. -М: Мир,1982 .
3 . Ньюмен У., Спрулл Р. Основы интерактивной машинной графики . -М: Мир, 1976 .
4 . Гилой В., Интерактивная машинная графика . -М: Мир, 1981.
5 . Павлидис Т., Алгоритмы машинной графики и обработки изображений . -М: Радио и связь, 1986.
6 . Фоли Дж., Вэн Дэм А., Основы интерактивной машинной графики . -М: Мир, 1987.
7. Роджерс Р. Алгоритмические основы машинной графики . -М: Мир, 1989.
8. О.Н.Романюк. Комп'ютерна графіка. Навчальний посібник/ Вінниця: ВДТУ, 2000, - 129 с.
9. Тихомиров Ю. Программирование трехмерной графики . Санкт- Петербург: BHV, 1998.
10. Гардон И., Люка М. Машинная графика и автоматизация конструирования. –М: Мир, 1987.
11. Эргон Ж. Синтез изображений. Базовые алгоритмы. –М: Радио и связь, 1993.
12. Яблонский Ф.М., Троицкий Ю.Ф. Средства отображения информации. –М: Высшая школа, 1995.
13. Хирн Д., Бейкер М. Микрокомпьютерная графика. –М: Мир,1987.
14. Лапшин У. Графика для IBM PC. –М.: СОЛОН, 1995.
15. Григорьев В.Л.. Видеосистемы ПК фирмы IBM. М: Радиосвязь, 1993.
16. Справочник по машинной графике в программировании/ Под. ред. В.Е.Михайленко, А.А.Лященко. –К.:Будівельник, 1994.
17. Уокер Б.С., Гурд Дж. Р.., Дроник Е.А. . Интерактивная машинная графика . –М.: Машиностроение, 1980.
18. Тормышев Ю.И. Технические средства машинной графики . -Минск: Наука и техника, 1987.
19. Росінський Ю.М. Комп'ютерна графіка. Навчальний посібник.

Житомир, 1997.
20. Боресков А. В., Шикин Е. В. Компьютерная графика. Динамика, реалистические изображения. - М : ДИАЛОГ- МИФИ ,1995.
21. Методичні вказівки до лабораторних робіт з курсу "Машинна графіка і діалогові системи " / Укладач О.Н.Романюк. -Вінниця : ВПІ, 1992
22.Яблонский Ф.М., Троицкий Ю.Ф. Средства отображения информации.
–М: Высшая школа, 1995.
23. Лапшин У. Графика для IBM PC. –М.: СОЛОН, 1995.
24. Григорьев В.Л.. Видеосистемы ПК фирмы IBM. М: Радиосвязь, 1993.
25. Майкл Ласло. Вычислительная геометрия и компьютерная графика на С++. М.: "Издательство БИНОМ ", 1997.
26. Аммерал Л. Принципы программирования в машинной графике. - М.:
Сол. Систем, 1992.
27. Шеймос М. Вычислительная геометрия. М.:Мир,1989.

Міністерство освіти та науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
До виконання курсової роботи з дисципліни
“Комп’терна графіка”
для студентів бакалаврського напрямку 6.08.04 – “Комп’ютерні науки”

Затверджено
на засіданні кафедри ПЗ
Протокол 8 від 3.12.2002
Зав.кафедри ПЗ _____Пєтух А.М.

Вінниця ВНТУ 2003

Міністерство освіти та науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
До виконання курсової роботи з дисципліни
“Комп’терна графіка”
для студентів бакалаврського напрямку 6.08.04 – “Комп’ютерні науки”

Вінниця ВНТУ 2003