МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Миколаївський національний університет
імені В. О. Сухомлинського

О. С. Булгакова,

В. В. Зосімов,

Г. В. Ходякова

Комп'ютерна графіка (2D/3D): теорія

Навчальний посібник

для дистанційної форми навчання

Миколаїв 2021

УДК 004.8

Б 90

РЕЦЕНЗЕНТИ:

ЛИТВИНЕНКО В. І., доктор технічних наук, професор, завідувач кафедри інформатики і комп'ютерних наук Херсонського національного технічного університету;

АТАМАНЮК І. П., доктор технічних наук, професор, завідувач кафедри вищої та прикладної математики Миколаївського національного аграрного університету

Рекомендовано до друку вченою радою
Миколаївського національного університету імені В. О. Сухомлинського
(протокол № 27 від 29.06.2021 р.)

Булгакова О. С.

Б 90 Комп'ютерна графіка (2D/3D): теорія : навчальний посібник для дистанційної форми навчання / О. С. Булгакова, В. В. Зосімов, Г. В. Ходякова. – Миколаїв: СПД Румянцева, 2021. – 150 с.

ISBN 867-345-684-658-1

Навчальний посібник для дистанційного навчання є складовою частиною курсу «Комп'ютерна графіка» для студентів спеціальностей галузі знань 12 Інформаційні технології та спеціальності 113 Прикладна математика, який орієнтований на студентів підготовки ступеня бакалавр. В навчальному посібнику розглядаються теоретичні та практичні питання комп'ютерної графіки, математичні та алгоритмічні основи комп'ютерної графіки, значна увага приділяється створенню ілюстрацій і редагування зображень, тобто алгоритмам векторної і растрової графіки, а також особливостям створення тривимірних моделей. Всі головні положення підручника розглядаються з використанням навчальних прикладів і ілюструються графічними матеріалами.

В навчальному посібнику викладено основний курс лекційного матеріалу та завдання до самостійної роботи, приклади їх рішення, контрольні запитання, тестові завдання.

УДК 004.8

ISBN 867-345-684-658-1

З М І С Т

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Computer Vision (CV)
Image Processing (IP)
Computer Graphics (*CG*)
База знань (БЗ)
Формальна система (ФС)

РОЗДІЛ І. ВСТУП ДО КОМП'ЮТЕРНОЇ ГРАФІКИ

1.1. Напрямки обробки графічної інформації

При обробці інформації, пов'язаної із зображенням на моніторі, прийнято виділяти три основні напрямки: розпізнавання образів (або система комп'ютерного зору), обробку зображень та машинно-комп'ютерну графіку, рис.1.

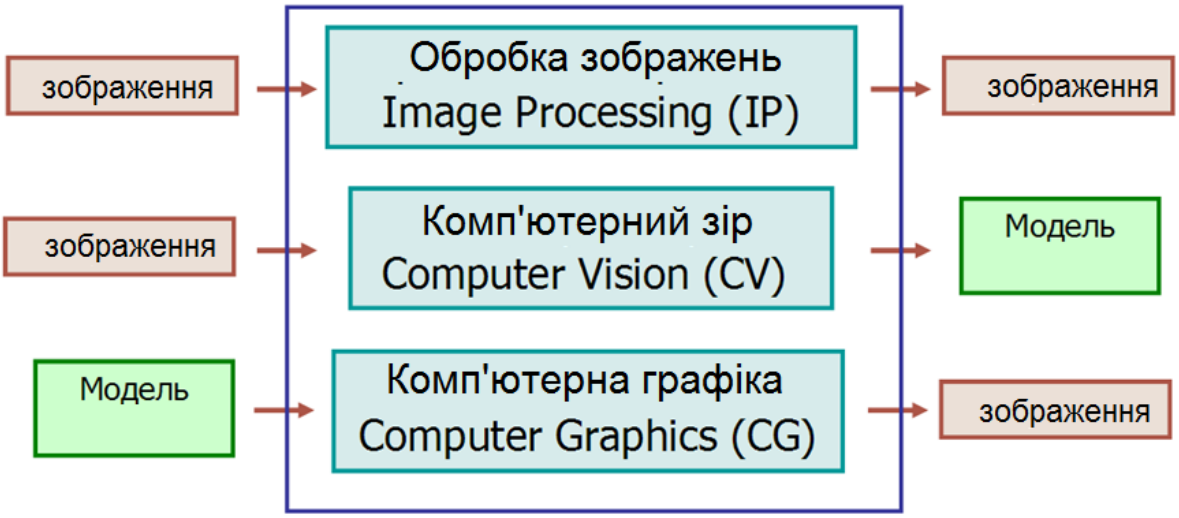


Рис.1. Основні напрями роботи з зображеннями

Розглянемо більш детально ці напрями:

- *Обробка зображень* (Image Processing) розглядає завдання в яких і вхідні і вихідні дані є зображеннями. Наприклад, передача зображення з усуненням шумів і стисненням даних, перехід від одного виду зображення до іншого (від кольорового до чорно-білого) і т.п. Таким чином, під обробкою зображень розуміють діяльність над зображеннями (перетворення зображень), рис.2. Завданням обробки зображень може бути як поліпшення в залежності від певного критерію (реставрація, відновлення), так і спеціальне перетворення, що кардинально змінює зображення.

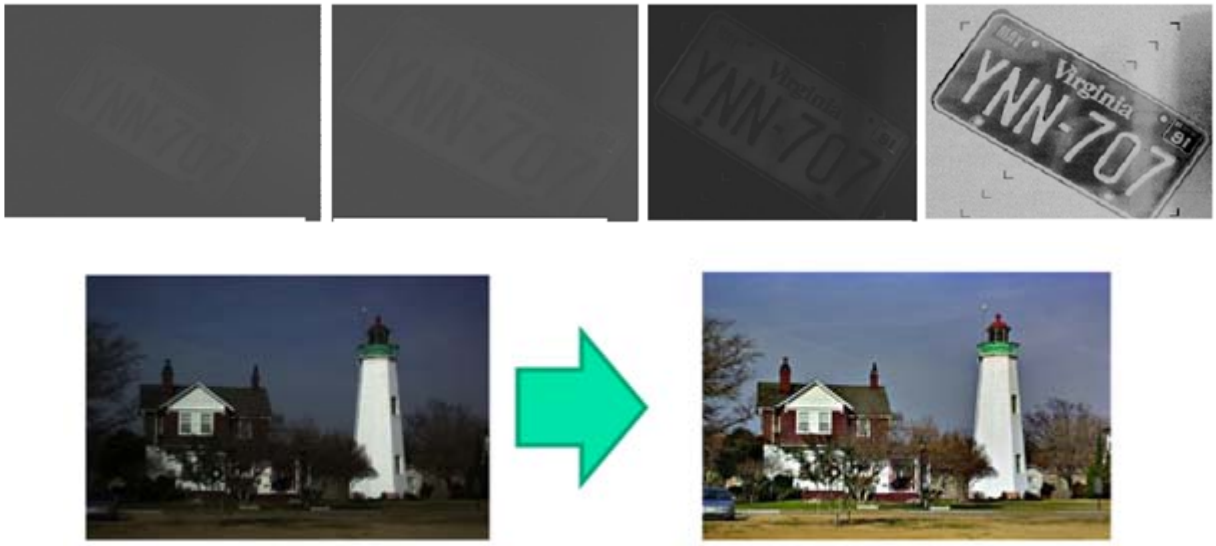


Рис.2. Приклади перетворення зображень

- *комп'ютерний зір (Computer Vision)*

Основне завдання розпізнавання образів (або комп'ютерного зору) полягає в перетворенні вже наявного зображення на формально зрозумілу мову символів. Розпізнавання образів або система комп'ютерного зору – це сукупність методів, що дозволяють отримати опис зображення, поданого на вхід, або віднести задане зображення до певного класу, рис.3.

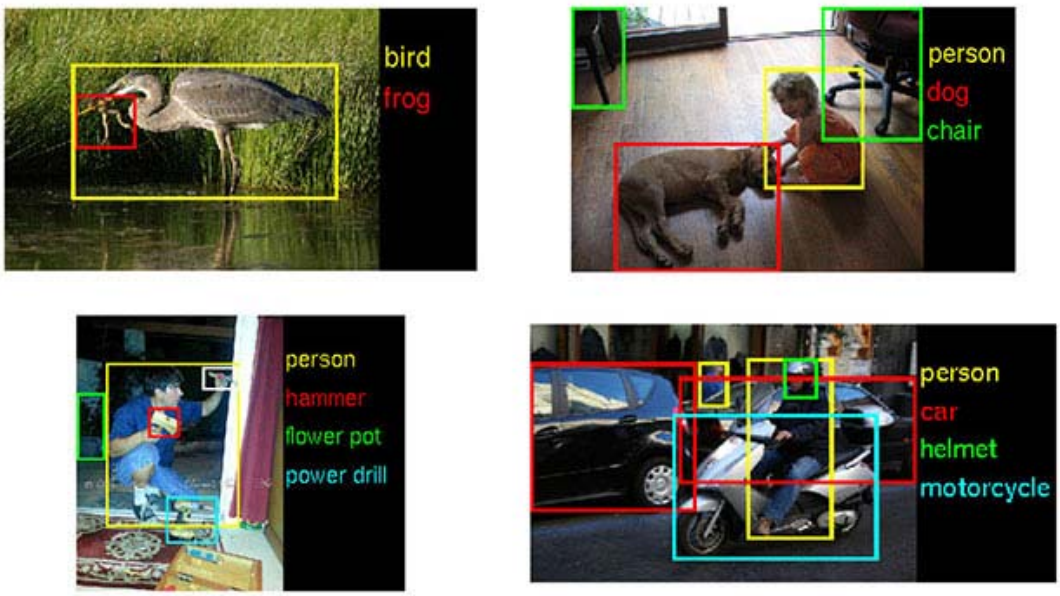


Рис.3. Приклади системи комп'ютерного зору

Одним із завдань CV є так звана скелетизація об’єктів, при якій відновлюється якась основа об’єкта, його «скелет».

1.2. Зображення: визначення, історія, завдання

Зображення – об’єкт, образ, явище, в тій чи іншій мірі подібне (але не ідентичне) зображуваного або сам процес їх створення. Подібність досягається внаслідок фізичних законів отримання зображення.

Говорячи про зображення, необхідно згадати історію виникнення зображення. Отже все почалося з відкриття камери-обскура (в перекладі з латині – темна кімната) – це прототип фотографічного апарату, темне приміщення з одним малим отвором (діаметром порядку 0,3-1,0 мм), через який на протилежну стіну проектується перевернуте зменшене зображення предметів ззовні, рис.4.

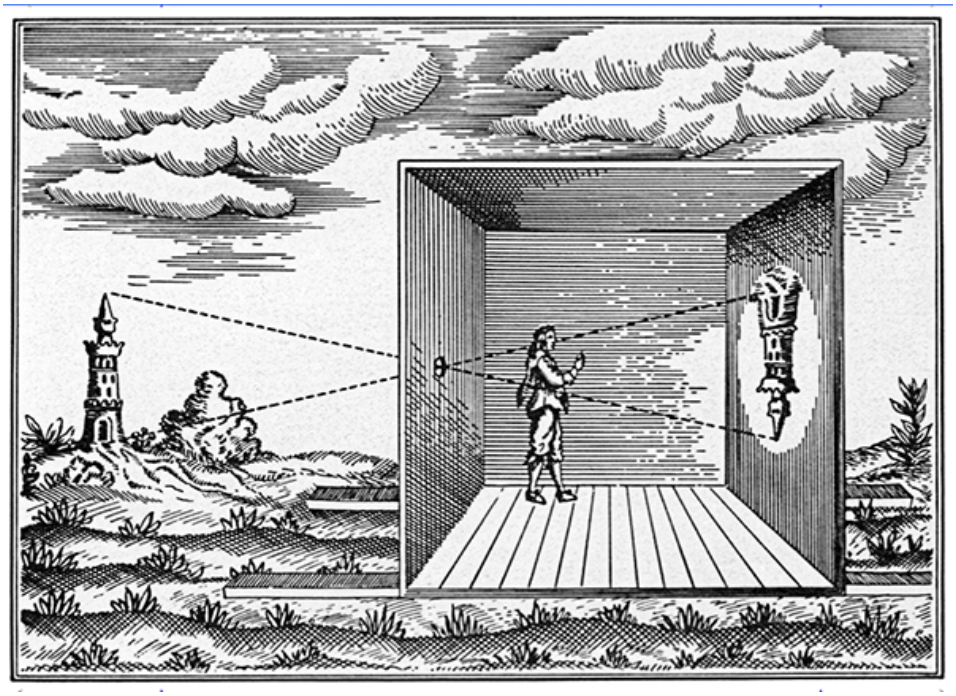


Рис.4. Зображення роботи камери-обскура

7 січня 1839 р. у Парижі Луї-Жак Дагер продемонстрував свій винахід щодо отримання зображення з використанням камери-обскури, хоча вже в V столітті до н. е. китайський філософ Мі Ті описав виникнення зображення на

стіні затемненої кімнати. Згадував про таку камеру і Аристотель у своїй праці «Problemata» (близько 350 року до н.е.). У X столітті арабський учений Ібн ал-Хайсам (Альхазен) з Басри користувався спеціальними наметами для спостережень за затемненням Сонця.

У наш час камери-обскури, встановлені в багатьох містах світу, використовуються для науки і освіти, а також заради забави. Фотографії, виконані за допомогою камери-обскури без лінзи, відзначаються м'якістю малюнка, напіврозмитістю, делікатним контрастом і повною відсутністю дисторсії (distortion), властивої складнішим оптичним пристроям.

Подальший розвиток зображення тісно пов’язаний з історією комп’ютерної графіки.

1.3. Історія розвитку комп’ютерної графіки

Розвиток комп'ютерної графіки, особливо на її початкових етапах, в першу чергу пов'язано з розвитком технічних засобів і особливо дисплеїв:

- довільне сканування променя;
- растрове сканування променя;
- запам'ятовують трубки;
- плазмова панель;
- рідкокристалічні індикатори;
- Електролюмінісцентне індикатори;
- дисплеї з емісією полем.

З повним описом історичного розвитку комп’ютерної графіки можна ознайомитися на https://www.youtube.com/watch?v=gJaxyk6kykU

1.4. Контроль знань

1. Комп'ютерна графіка це …

А. наука, предметом вивчення якої є розробка ПЗ за допомогою ЕОМ, тобто це розділ інформатики, який займається проблемами розробки ПЗ;

Б. відтворення зображень, при якому користувач має можливість оперативно вносити зміни в зображення безпосередньо в процесі його відтворення, тобто передбачається можливість роботи з графікою в режимі діалогу в реальному масштабі часу;

В. наука, предметом вивчення якої є створення, зберігання і обробка моделей і їх зображень за допомогою ЕОМ, тобто це розділ інформатики, який займається проблемами отримання різних зображень (малюнків, креслень, мультиплікації) на комп'ютері.;

Г. вірної відповіді немає;

2. Інтерактивна графіка надає можливість…..

А. роботи з графікою в режимі діалогу статично;

Б. роботи з графікою в режимі діалогу в реальному масштабі часу;

В. роботи з графікою в режимі дистанційного навчання;

Г. вірної відповіді немає;

3. В якому році з’явилися векторні дисплеї.

А. 1950;

Б. 1945;

В. 1982;

Г. 1990;

Д. 2000;

4. В якому році з’явилися цифрові дисплеї.

А. 1960;

Б. 1945;

В. 1982;

Г. 1990;

Д. 2000;

5. Де і коли була розроблена комп’ютерна математична модель руху кішки.

А) 1968; групою з Московського державного університету під керівництвом Н.Н. Константинова

Б) 1982; групою з Московського державного університету під керівництвом Н.Н. Лебедєва

В) 1990; групою з Массачусетського технологічного університету під керівництвом Т. Муаer

Г) 2000; групою з інституту Кібернетики під керівництвом Н.Н. Глушкова

6. В якому році з’явилася технологія мультимедіа.

А. 1986;

Б. 1945;

В. 1982;

Г. 1990;

Д. 2000;

7. За допомогою розв'язку якого математичного апарату у 1968 році була розроблена комп’ютерна математична модель руху кішки.

А. Розв’язок системи нелінійних рівнянь;

Б. Розв’язок інтегральних рівнянь;

В. Розв’язок диференціальних рівнянь;

Г. вірної відповіді немає;

8. Комп’ютерна анімація це …

А. створення статичних графічних об’єктів;

Б. створення графічних об’єктів в динаміці;

В. довільне малювання і креслення на екрані ЕОМ;

Г. вірної відповіді немає;

9. В якому році з’явилась трубка Брауна

А. 1945;

Б. 1897;

В. 1917;

Г. 1996;

10. Який спосіб отримання графічних об’єктів використовується в Adobe Photoshop?

А. растровий

Б. векторний

В. вірної відповіді немає;

РОЗДІЛ ІІ. ПРЕДСТАВЛЕННЯ КОЛЬОРУ

2.1. Поняття кольору та його характеристики

2.2. Кольорові моделі та їх види

2.2.1. Колірна модель RGB

Це одна з найбільш поширених і часто використовуваних моделей. Вона застосовується в приладах, що випромінюють світло, таких, наприклад, як монітори, прожектори, фільтри та інші подібні пристрої.

Дана колірна модель базується на трьох основних кольорах: Red - червоному, Green - зеленому і Blue - синьому. Кожна з перерахованих вище складових може варіюватися в межах від 0 до 255, утворюючи різні кольори, таким чином, доступ до всіх 16 мільйонам (загальна кількість квітів, запропонованих відповідною моделлю одно 256 * 256 * 256 = 16 777 216.).

Дана модель формує відтінки кольору за допомогою трьох основних кольорів: червоного (Red), зеленого (Green) та синього (Blue). Насиченість тієї чи Іншої складової змінюється від 0 до 255. Колір Відображається шістнадцятірічнім числом від $ 00000000 до $ 02FFFFFF. Перші дві цифри праворуч вказують на насиченість червоного кольору, наступні дві цифри - насиченість зеленого кольору, а п'ята и шоста Цифри з правої сторін - на насиченість синього кольору. Наприклад, $ 0000FF - червоний колір, $ 00FF00 - зелений колір. Якщо додамо ці два кольори отримаємо жовтий колір ($ 0000FF + $ 00FF00 = $ 00FFFF).

Рис.2.1 Модель RGB

2.2.2. Колірна модель HSB

Система HSB зручна для користувача. У ній можна синтезувати нові кольори і отримувати різні варіанти заданого кольору, спираючись на інтуїцію. Наприклад, ми знаємо, що чистий синій колір лежить на колірному колі під кутом 240 градусів. Якщо потрібно змістити тон в сторону пурпурного відтінку, то для цього достатньо збільшити кут повороту. Колір здається дуже насиченим? Рішення відомо. Треба змістити точку в радіальному напрямку ближче до центру. Велика яскравість? Зменшуємо відповідну координату. Подібну стратегію синтезу кольору неможливо реалізувати в системі RGB, оскільки важко передбачити наслідки навіть невеликих змін колірних координат. Ще одним безсумнівним достоїнством системи HSB є її незалежність від апаратури. Приблизно таку оцінку могли б дати системі HSB користувачі і розробники комп'ютерних програм.

2.2.3. Колірна модель CMY (Cyan Magenta Yellow)

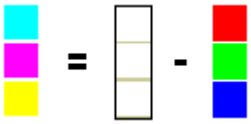


Рис.2.2. Отримання CMY з RGB

Кольори, які використовують білий світ, віднімаючи з нього певні ділянки спектру називаються субтрактивні. Основні кольори цієї моделі: блакитний (білий мінус червоний), фуксин (в деяких книгах його називають пурпуровим) (білий мінус зелений) і жовтий (білий мінус синій). Ці кольори є поліграфічної тріадою і можуть бути легко відтворені поліграфічними машинами. При змішання двох субтрактивних кольорів результат затемнюється (в моделі RGB було навпаки). При нульовому значенні всіх компонент утворюється білий колір (білий папір). Ця модель являє відбитий колір, і її називають моделлю субтрактивних основних кольорів. Дана модель є основною для поліграфії і також є апаратно-залежною.

Рис.2.3. Модель CMY

2.2.4. Колірна модель CMYK

Модель CMYK (Cyan Magenta Yellow Key, причому Key означає чорний колір) - є подальшим поліпшенням моделі CMY і вже чотирьохканальна. Оскільки реальні друкарські фарби мають домішки, їх колір не збігається в точності з теоретично розрахованим блакитним, жовтим і пурпурним. Особливо важко отримати з цих фарб чорний колір. Тому в моделі CMYK до тріади додають чорний колір. Чорний колір зашифрований як K (від слова Key - ключ).

2.2.5. Колірна модель Lab

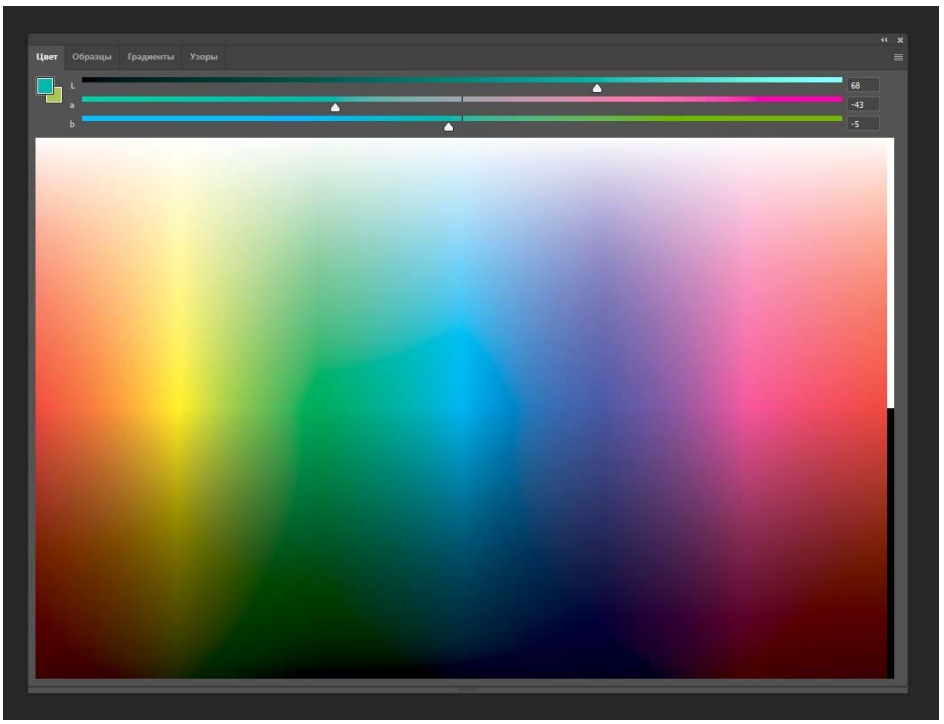
Одна з ранніх моделей, яка лежить в основі системи управління кольором в Photoshop. Модель Lab - це система координат з трьох осей:

L - Lightness, яскравість об'єкта;

a - вісь, по якій відкладені градації від червоного до зеленого;

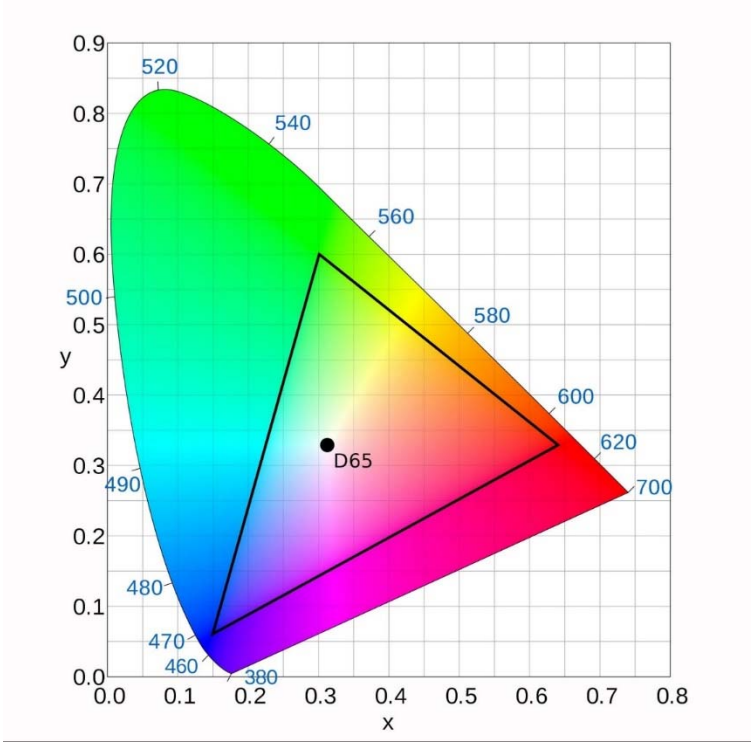
b - вісь з градаціями від жовтого до синього.

Давайте, наприклад, візьмемо бірюзовий колір і поглянемо на нього на діаграмі. На шкалі L показано, наскільки він світлий. На шкалі a - то, що він ближче до зеленого, ніж до червоного. На шкалі b - що в ньому більше синього, ніж жовтого:



Принцип роботи Lab аналогічний тому, як нейрони сітківки людського ока кодують кольору. Кожен колір ми сприймаємо виходячи з трьох координат. Світлий він або темний? Ближче до зеленого або до червоного? У ньому більше жовтизни або синяви? Це називається опонентном сигнали.

За одиницю в моделі приймається мінімальне колірна відмінність, сприймається людським оком. Тому Lab має максимальний колірний обхват.



Саме з Lab зручно працювати при корекції, ретуші та підготовці до друку. Її головна перевага - можливість змінювати яскравість без зміни колірних значень: для цього змінюють значення по осі L.

2.3. Кодування кольору

Для моделі RGB кожна з компонент може представлятися числами, обмеженими деяким діапазоном - наприклад, дробовими числами від 0 до 1 або цілими числами від 0 до деякого максимального значення. В даний час досить поширеним є формат True Color, в якому кожна компонента представлена у вигляді байта, що дає 256 градацій для кожної компоненти: R = 0 ... 255, G = 0 ... 255, B = 0 .. .255. Кількість квітів становить 256x256x256 = 16.7 млн (224).

Такий спосіб кодування кольорів можна назвати компонентним. У комп'ютері коди зображень True Color представляються у вигляді трійок байтів, або упаковуються в довге ціле (четирехбайтное) - 32 біта.

1.5. Контроль знань

1. CMYK - це?

А. антивірус;

Б. формат картинки;

В. колірна система;

Г. програма для перегляду зображень.

2. Які основні поняття використовує колірна модель HSB?

А. відтінок, насиченість, яскравість;

Б. глибина кольору, відтінок, яскравість;

В. роздільна здатність зображення, відтінок, яскравість;

Г. рівні кольору, глибина кольору, яскравість.

3. В кольоровій моделі RGB встановлені наступні параметри: 0.255.0. Який колір буде відповідати цим параметрам?

А. жовтий;

Б. зелений;

В. червоний;

Г. чорний.

4. Lab - це?

А. антивірус;

Б. формат картинки;

В. колірна система (модель);

Г. програма для перегляду зображень.

5. Оберіть правильні відповіді. Запишіть код червоного кольору в двійковому, шістнадцятирічному вигляді.

А. 11111111.00000000.00000000;

Б. FF0000;

В. 00000000.11111111.00000000;

Г. 00FF00;

Д. 11111111.00000000.11111111;

Е. FF00FF;

6. Оберіть правильні відповіді. Запишіть код зеленого кольору в двійковому, шістнадцятирічному вигляді.

А. 11111111.00000000.00000000;

Б. FF0000;

В. 00000000.11111111.00000000;

Г. 00FF00;

Д. 11111111.00000000.11111111;

Е. FF00FF;

7. Оберіть правильні відповіді. Запишіть код зеленого кольору в шістнадцятирічному і десятковому вигляді.

А. FF0000;

Б. 255.0.0

В. 00FF00;

Г. 0.255.0

Д. FF00FF;

Е. 255.0.256

8. Оберіть правильні відповіді. Запишіть код синього кольору в шістнадцятирічному і десятковому вигляді.

А. FF0000;

Б. 255.0.0

В. 00FF00;

Г. 0.255.0

Д. 0000FF;

Е. 0.0.255

9. Оберіть правильні відповіді. Запишіть код червоного кольору в двійковому, шістнадцятирічному вигляді.

А. 11111111.00000000.00000000;

Б. FF0000;

В. 00000000.11111111.00000000;

Г. 00FF00;

Д. 11111111.00000000.11111111;

Е. FF00FF;

10. Оберіть правильні відповіді. Запишіть код зеленого кольору в двійковому, шістнадцятирічному вигляді.

А. 11111111.00000000.00000000;

Б. FF0000;

В. 00000000.11111111.00000000;

Г. 00FF00;

Д. 11111111.00000000.11111111;

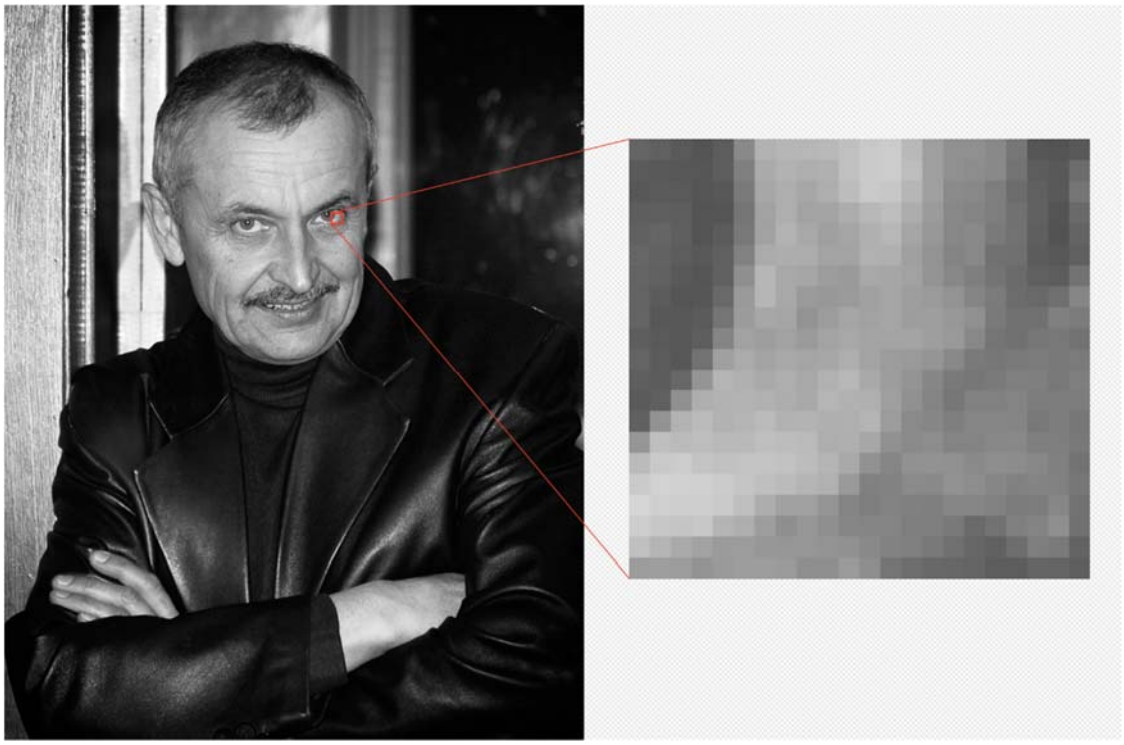
FF00FF

РОЗДІЛ ІІІ. РАСТРОВА ГРАФІКА

3.1. Растрова графіка

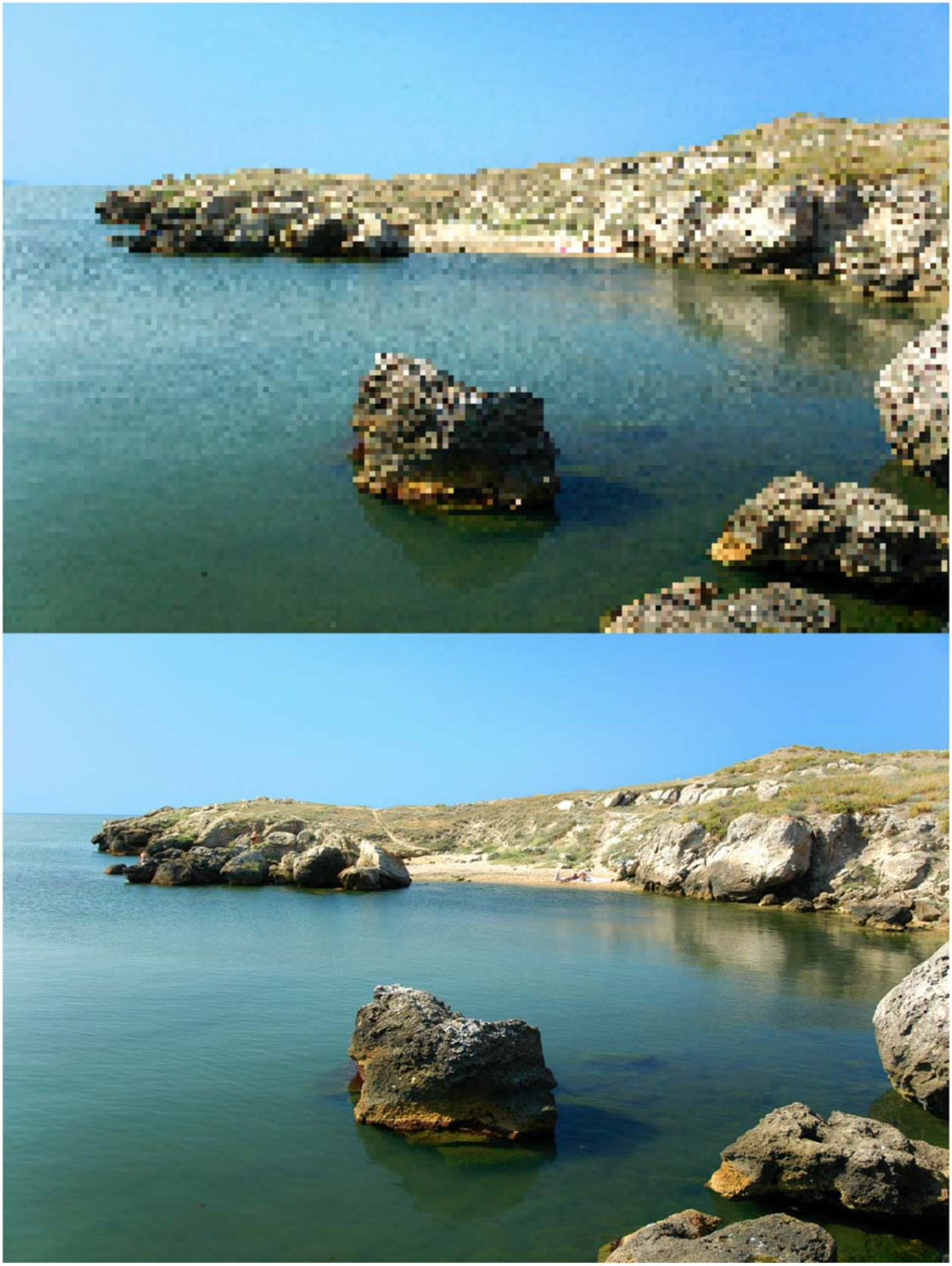
Растр - це множина дрібних точок, з яких може складатися зображення. У випадку з комп'ютером растр - це пікселі, з яких складається фотографія.

Наприклад, коли ви фотографуєте на смартфон або цифровий фотоапарат, ви отримуєте растрове зображення, яке складається з безлічі окремих точок. Якщо дивитися на екрані телефону або комп'ютера, вони не видно, але якщо сильно збільшити, то ці точки стануть помітні.



У растрової графіки є два головних параметра: розмір зображення і глибина кольору.

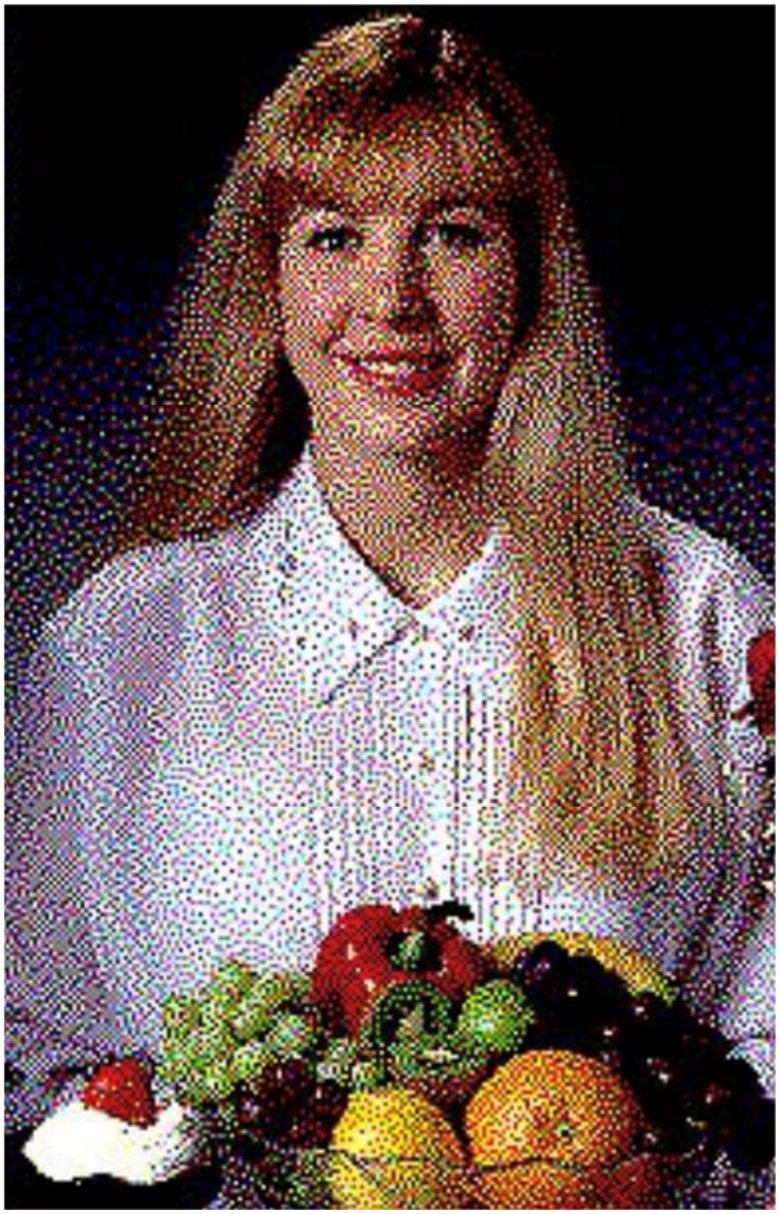
Розмір зображення - це кількість пікселів по горизонталі і вертикалі. Чим більше розмір, тим сильніше можна збільшувати картинку без втрати якості. Наприклад, візьмемо одну і ту ж фотографію, але в однієї буде розмір 100 на 200 пікселів, а в іншої - 1000 на 2000 пікселів:



В одному і тому ж масштабі друга картинка виглядає набагато краще, тому що в ній більше пікселів, які передають більше деталей

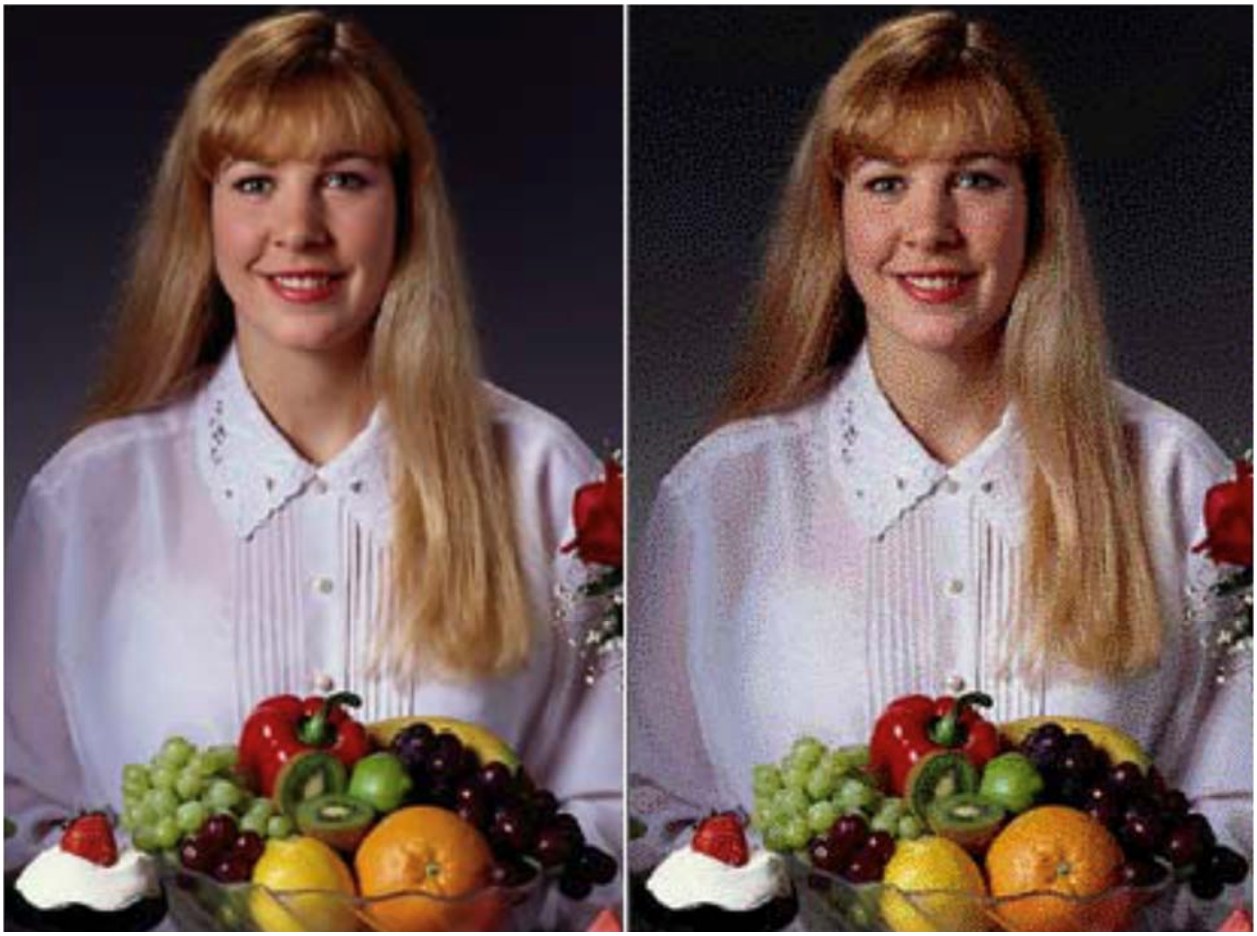
Загальне правило таке: чим більше пікселів на фотографії, тим більше дрібних деталей можна на ній розгледіти. Саме тому виробники камер і смартфонів постійно збільшують кількість пікселів у себе в пристроях.

Глибина кольору. Уявіть, що ваша камера в телефоні може розрізняти тільки 16 кольорів. В цьому випадку фотографії виходили б такими:



В цілому зрозуміло, що тут зображено, але виглядає дивно

Це і є глибина кольору - скільки різних відтінків присутній на зображенні. У нашому прикладі 16 кольорів - це 4 біта, тому що 2 в 4 ступеня = 16. Порівняйте, як виглядає та ж фотографія з глибиною кольору 16 і 8 біт:



Чим більше глибина кольору, тим плавніше колірні переходи на фото

Головне застосування растрової графіки - фотографії і зображення з великою глибиною кольору і безліччю деталей. Фотографії - це растр. Малюнки від руки - найчастіше растр. Якщо на зображенні природа, люди, водичка або що завгодно з безліччю деталей, швидше за все, таке зображення буде растрових.

Комп'ютери класно справляються з растровими зображеннями, тому що растр досить простий в обробці. Комп'ютер ставить поспіль потрібну кількість пікселів і фарбує їх в потрібні кольори. Операція проста, математика мінімальна, просто потрібно повторити її багато разів. Комп'ютери в цьому сильні.

3.2. Кодування растрових зображень

У процесі кодування зображення проводиться його просторове розбиття (дискретизація) на окремі маленькі фрагменти - пікселі (pixel), причому кожному фрагменту присвоюється значення його кольору, тобто код кольору (червоний, зелений, синій і так далі).

Якість кодування зображення залежить від двох параметрів.

По-перше, якість кодування зображення тим вище, чим менше розмір точки і відповідно більшу кількість точок становить зображення.

По-друге, чим більша кількість квітів, тобто більшу кількість можливих станів точки зображення, використовується, тим якісніше кодується зображення (кожна точка несе більшу кількість інформації). Сукупність використовуваних в наборі квітів утворює палітру кольорів, яка пов'язана з кількістю пам'яті, призначеної для зберігання одного пікселя - глибиною кольору.

$$\text{КІЛЬКІСТЬ КОЛЬОРІВ (К)} = 2^{(N)\text{ГЛУБІНА КОЛЬОРУ}}$$

Приклад 1. Відомо, що відеопам'ять комп'ютера має обсяг 512 Кбайт. Роздільна здатність екрану 640 на 200. Скільки сторінок екрану одночасно розміститься в відеопам'яті при палітрі 256 квітів?

Якщо палітра складається з 256 кольорів, то 8 біт глибина кольору.

Кількість пікселів екрану 640□200 = 128000. Для збереження картинки екрану необхідно 128000□8 = 1024000біт = 128000байт = 125 Кбайт.

512/125 = 4,096. Отже, в відеопам'яті розміститися 4,096 сторінок екрану.

Приклад 2. Скільки бітів відеопам'яті займає інформація про одному пікселі на чорно-білому екрані (без напівтонів)?

Для чорно-білого зображення без напівтонів К = 2. Отже 2N = 2. Звідси N = 1 біт на піксель.

Приклад 3. Сучасний монітор дозволяє отримувати на екрані 16 777 216 різних кольорів. Скільки бітів пам'яті займає 1 піксель?

Оскільки К = 16 777 216 = 224, то N = 24 біта на піксель.

Приклад 4. На екрані з роздільною здатністю 640 x 200 висвічуються тільки двоколірні зображення. Який мінімальний обсяг відеопам'яті необхідний для зберігання зображення?

Так як бітова глибина двоколірного ізображе¬нія дорівнює 1, а відеопам'ять, як мінімум, повинна вміщувати одну сторінку зображення, то обсяг відеопам'яті дорівнює 640 • 200 • 1 = 128 000 бітів = 16 000 байт.

3.3. Стиснення растрових зображень

RLE (Run Length Encoding) - метод стиснення, що полягає в пошуку послідовностей однакових пікселів в точки растрового зображення («червоний, червоний, ..., червоний» записується як «N червоних»).

Алгоритм Хаффмана заснований на теорії ймовірності. Спочатку елементи зображення (пікселі) упорядковано відповідно до частоті. Потім з них будується кодове дерево Хаффмана. Кожному елементу зіставляється кодове слово. При прагненні розміру зображення до нескінченності досягається максимальність стиснення. Цей алгоритм також використовується в архіваторах.

Стиснення застосовується і для векторної графіки, але тут вже немає таких простих закономірностей, так як формати векторних файлів досить сильно відрізняються за змістом.

3.4. Стиснення по алгоритму Хаффмана

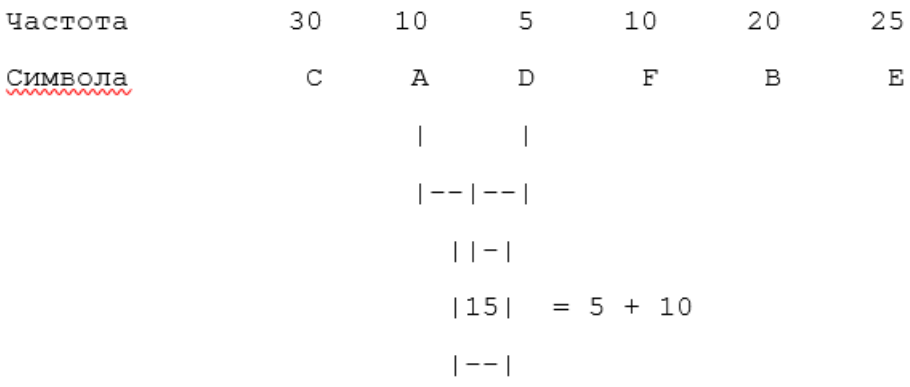
Ми маємо файл довжиною в 100 байт і має 6 різних символів в собі. Ми підрахували входження кожного із символів в файл і отримали наступне:

символ	A	B	C	D	E	F
число вхождений	10	20	30	5	25	10

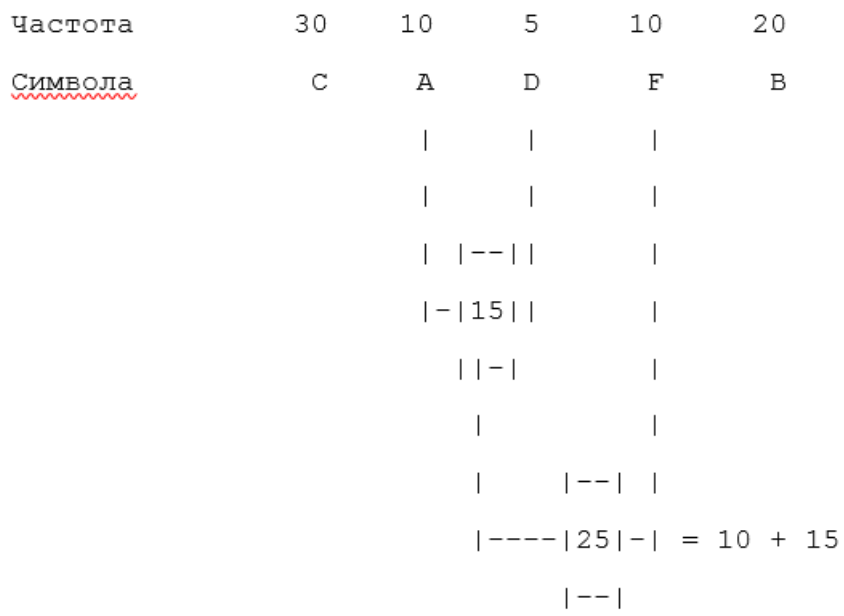
Тепер ми беремо ці числа і будемо називати їх частотою входження для кожного символу. Розмістимо таблицю як нижче.

символ	C	E	B	F	A	D
число вхождений	30	25	20	10	10	5

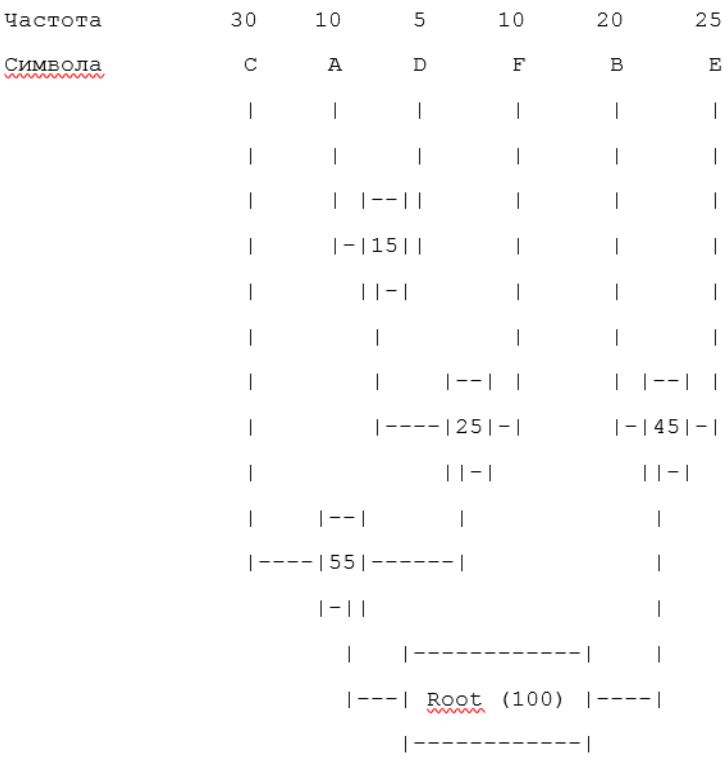
Сформуємо з "вузлів" D і A новий "вузол", частота входження для якого буде дорівнює сумі частот D і A:



Номер в рамці - сума частот символів D і A. Тепер ми знову шукаємо два символи з найнижчими частотами входження. Виключаючи з перегляду D і A і розглядаючи замість них новий "вузол" з сумарною частотою входження. Найнижча частота тепер у F і нового "вузла". Знову зробимо операцію злиття вузлів:



Розглядаємо таблицю знову для наступних двох символів (B і E). Ми продовжуємо в цей режим поки все "дерево" не сформоване, тобто поки все не зведеться до одного вузла.



Тепер коли наше дерево створено, ми можемо кодувати файл. Ми повинні завди починати з кореня (Root). Кодуючи перший символ (лист дерева C) Ми простежуємо вгору по дереву все повороти гілок і якщо ми робимо лівий поворот, то запам'ятовуємо 0-й біт, і аналогічно 1-й біт для правого повороту. Так для C, ми будемо йти вліво до 55 (і запам'ятаємо 0), потім знову вліво (0) до самого символу. Код Хаффмана для нашого символу C - 00. Для наступного символу (A) у нас виходить - ліво, право, ліво, ліво, що виливається в послідовність 0100. Виконавши вище сказане для всіх символів отримаємо

```
C = 00   ( 2 бита )
A = 0100 ( 4 бита )
D = 0101 ( 4 бита )
F = 011  ( 3 бита )
B = 10   ( 2 бита )
E = 11   ( 2 бита )
```

Кожен символ спочатку представлявся 8-ю бітами (один байт), і так як ми зменшили число бітів необхідних для подання кожного символу, ми отже зменшили розмір вихідного файлу.

```
Первинний размір файлу : 100 байт - 800 біт;
            Розмір стисненого файлу :  30 байт - 240 біт;
       240 - 30% з 800
```

3.5. Завдання для самостійної роботи

З теми «Кодування кольору»

1. Растровий графічний файл містить чорно-біле зображення (без градацій сірого) розміром 100x100 точок. Який інформаційний обсяг цього файлу?

   A. 10 000 біт

Б. 1 024 байт

В. 10 Кбайт

Г. 1 000 біт

2. Для зберігання 256-кольорового зображення на кодування одного пікселя виділяється:

А. 2 байта

Б. 4 біт

В. 4 байта

Г. 1 байт

3. Растровий графічний файл містить чорно-біле зображення з 16-ю градаціями сірого кольору розміром 10х10 пікселів. Який інформаційний обсяг цього файлу?

А. 100 біт

Б. 400 біт

В. 800 біт

Г. 400 байт

4. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 640х480, якщо глибина кольору на одну точку становить 4 біт.

А. 234 Кбайт

Б. 384 Кбайт

В. 150 Кбайт

Г. 640 Кбайт

5. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 800х600, якщо глибина кольору на одну точку становить 4 біт.

А. 234 Кбайт

Б. 384 Кбайт

В. 250 Кбайт

Г. 640 Кбайт

5. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1280х1024, якщо глибина кольору на одну точку становить 4 біт.

А. 234 Кбайт

Б. 150 Кбайт

В. 384 Кбайт

Г. 640 Кбайт

6. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1024x768, якщо глибина кольору на одну точку становить 4 біт.

А. 234 Кбайт

Б. 150 Кбайт

В. 384 Кбайт

Г. 640 Кбайт

7. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 640x480, якщо глибина кольору на одну точку становить 8 біт.

А. 1,25 Мбайт

Б. 469 Кбайт

В. 768 Кбайт

Г. 300 Кбайт

8. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 800x600, якщо глибина кольору на одну точку становить 8 біт.

А. 1,25 Мбайт

Б. 469 Кбайт

В. 768 Кбайт

Г. 300 Кбайт

9. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1280x1024, якщо глибина кольору на одну точку становить 8 біт.

А. 1,25 Мбайт

Б. 469 Кбайт

В. 768 Кбайт

Г. 300 Кбайт

10. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1024x768, якщо глибина кольору на одну точку становить 8 біт.

А. 1,25 Мбайт

Б. 469 Кбайт

В. 768 Кбайт

Г. 300 Кбайт

11. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 640x480, якщо глибина кольору на одну точку становить 16 біт.

А. 2,5 Мбайт

Б. 1,5 Мбайт

В. 600 Кбайт

Г. 938 Кбайт

12. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 800x600, якщо глибина кольору на одну точку становить 16 біт.

А. 2,5 Мбайт

Б. 1,5 Мбайт

В. 600 Кбайт

Г. 938 Кбайт

13. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1280x1024, якщо глибина кольору на одну точку становить 16 біт.

А. 2,5 Мбайт

Б. 1,5 Мбайт

В. 600 Кбайт

Г. 938 Кбайт

14. Визначити необхідний обсяг відеопам'яті для графічного режиму екрана монітора 1024x768, якщо глибина кольору на одну точку становить 8 біт.

А. 2,5 Мбайт

Б. 1,5 Мбайт

В. 600 Кбайт

Г. 938 Кбайт

15. Для зберігання растрового зображення розміром 128x128 пікселів відвели 4 Кбайт пам'яті. Яке максимально можливе число кольорів у палітрі зображення.

А. 2

Б. 4

В. 16

Г. 8

З теми «Стиснення графічних даних»

1. На скільки зміниться розмір файлу після стиснення за алгоритмом Хаффмана. Файл містить рядок символів

символ	A	I	C	B
Число входжень	15	14	13	27

На кожний символ відводиться 1 байт. Одиниця виміру відповіді - біт.

Правильна відповідь: 190 біт

2. На скільки зміниться розмір файлу після стиснення за алгоритмом Хаффмана. Файл містить рядок символів

символ	A	B	C	D
Число входжень	15	10	11	22

На кожний символ відводиться 1 байт. Одиниця виміру відповіді - біт.

Правильна відповідь: 125 біт

3. На скільки зміниться розмір файлу після стиснення за алгоритмом Хаффмана. Файл містить рядок символів

символ	A	B	C	D
Число входжень	10	9	9	25

На кожний символ відводиться 1 байт. Одиниця виміру відповіді - біт.

Правильна відповідь: 101 біт

4. На скільки зміниться розмір файлу після стиснення за алгоритмом Хаффмана. Файл містить рядок символів

символ	A	B	C	D
Число входжень	15	10	9	22

На кожний символ відводиться 1 байт. Одиниця виміру відповіді - біт.

Правильна відповідь: 115 біт

5. На скільки зміниться розмір файлу після стиснення за алгоритмом Хаффмана. Файл містить рядок символів

символ	A	B	C	D
Число входжень	20	10	9	22

На кожний символ відводиться 1 байт. Одиниця виміру відповіді - біт.

Правильна відповідь: 145 біт

3.6. Контроль знань

1. Сітка з горизонтальних та вертикальних стовпців, яку на екрані утворюють пікселі, називається:

А. видеопамять;

Б. видеоадаптер;

В. растр;

Г. дисплейный процессор;

2. Піксель на екрані дисплея являє собою:

А. мінімальну ділянку зображення, якій незалежним чином можна задати колір;

Б. двійковий код графічної інформації;

В. електронний промінь;

Г. сукупність 16 зерен люмінофора.

3. Відеоконтроллер - це:

А. дисплейний процесор;

Б. програма, що розподіляє ресурси відеопам'яті;

В. електронне енергозалежний пристрій для зберігання інформації про графічне зображення;

Г. пристрій, що управляє роботою графічного дисплея.

4. Колір точки на екрані дисплея з 16-кольоровий палітрою формується з сигналів:

А. червоного, зеленого і синього;

Б. червоного, зеленого, синього і яскравості;

В. жовтого, зеленого, синього і червоного;

Г. жовтого, синього, червоного і яскравості.

5. Найменшим елементом поверхні екрану, для якого можуть бути задані адреса, колір та інтенсивність, є:

А. символ;

Б. зерно люмінофора;

В. піксель;

Г. растр.

6. Графіка з представленням зображення у вигляді сукупностей точок називається:

А. прямолінійною;

Б. фрактальною;

В. векторною;

Г. растровою.

7. Рідний формат ОС Windows:

А. PNG;

Б. GIF;

В. BMP;

Г. JPEG.

8. Кількість пікселів, що припадає на одиницю площі - це?

А. глибина кольору;

Б. якість зображення;

В. якісна здатність зображення;

Г. роздільна здатність зображення.

9. Характеристика зображення, що визначає кількість бітів, які використовують для подання кольору під час кодування одного піксела растрового зображення?

А. якісна здатність зображення;

Б. глибина кольору;

В. роздільна здатність зображення;

Г. якісна здатність зображення.

10. RGB - це?

А. антивірус;

Б. формат картинки;

В. колірна система (модель);

Г. програма для перегляду зображень.

11. Який з графічних редакторів є растровим?

А. Adobe Illustrator

Б. Paint

В. Corel Draw

РОЗДІЛ IV. ВЕКТОРНА ГРАФІКА

4.1. Векторна графіка: основні поняття

На відміну від растрової графіки, векторна складається не з пікселів, а з математичних формул. У такій графіку кожне зображення намальовано за допомогою окремих елементів:

точок,
еліпсів,
прямокутників,
багатокутників,
кривих будь-якої складності.

Щоб це намалювати, у кожного елемента є свої параметри, наприклад:

координати,
колір,
розмір,
товщина лінії,
товщина контуру,
колір контуру,
прозорість,
радіус кривизни і так далі.

Якщо комп'ютеру потрібно намалювати зоряне небо, ми можемо дати йому такі команди:

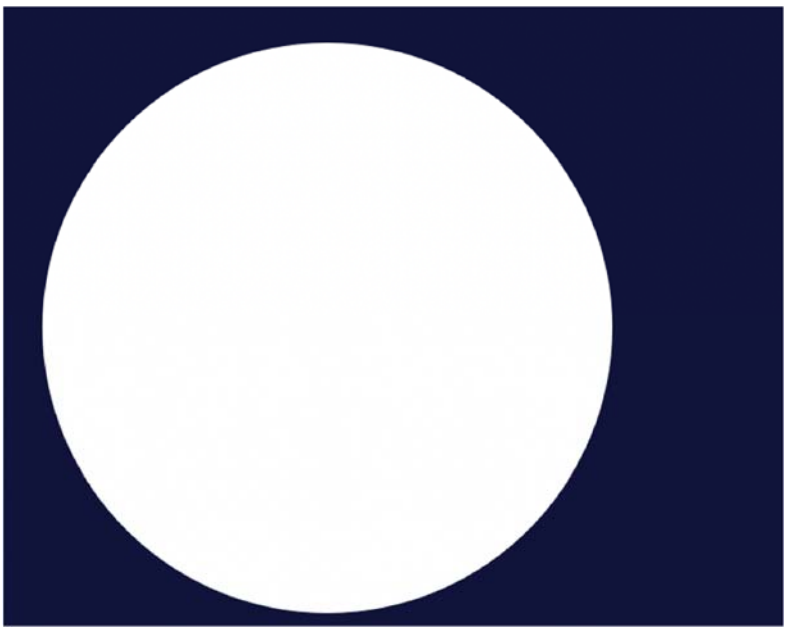
1. Створи порожній малюнок.
2. Залий його градієнтом зверху вниз від темно-синього до синього.
3. Постав крапку {біла, розмір 0,5, непрозорість 100%} за координатами 10,8.

4. Постав крапку {біла, розмір 0,4, непрозорість 100%} за координатами 14,9.
5. Постав крапку {біла, розмір 1,1, непрозорість 80%} за координатами 19,31.
6. ... додаємо ще 113 зірок.

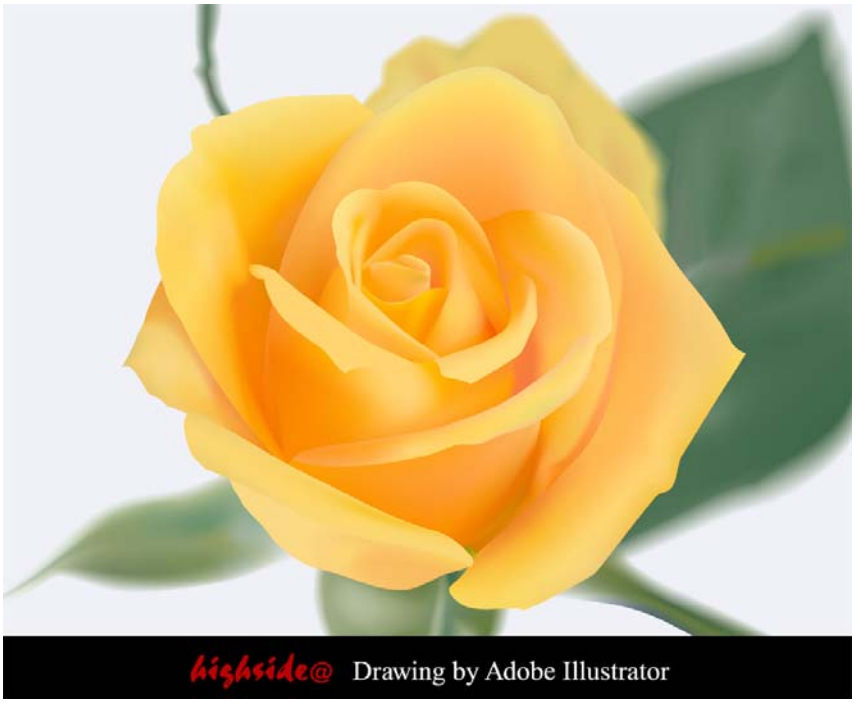
В результаті отримаємо такий малюнок:

Так як ми не прив'язані до розміру зображення, то за цими формулами комп'ютер може нам отрисовать зоряне небо будь-якого розміру - від шпалер на телефон до рекламного білборда 4 на 6 метрів. При цьому при збільшенні втрати якості не відбувається - комп'ютер просто отримує від нас фінальний розмір зображення і малює все в потрібних пропорціях.

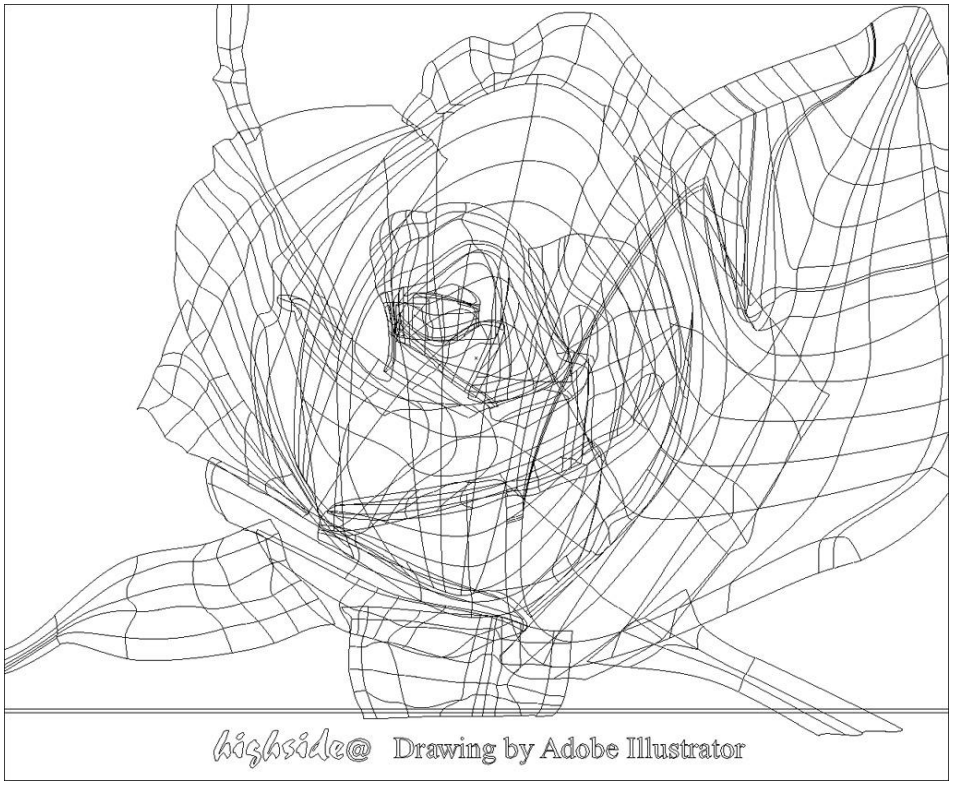
Сила векторної графіки - в можливості нескінченно збільшувати і зменшувати розмір зображення без втрати якості. При зміні розміру комп'ютер відразу перераховує всі формули і отрісовиває картинку заново. Тому при збільшенні векторної графіки не з'являються пікселі і розмиття, навіть якщо нам потрібно збільшити одну зірку в 100 разів:



Мінус векторної графіки в тому, що в ній дуже складно створити фотореалістичне зображення. Справа в тому, що кожна деталь, кожен новий колір і кожен колірний перехід - це нова формула. Щоб побудувати фотореалістичну картинку, потрібно дуже багато формул, які будуть складно обраховуватися, і все одно по деталях можна зрозуміти, що перед нами не фотографія:



Кожен елемент на цій картинці задається своєю формулою. Тут багато деталей, але все одно видно, що це не фотографія, а векторна ілюстрація



Векторна графіка найчастіше застосовується там, де не потрібна фотореалістичність - іконки, піктограми, рекламні матеріали. Головне завдання такого зображення - щоб його можна було збільшити або зменшити як завгодно без втрати якості.

4.2. Математичні основи векторної графіки

4.3. Криві Безьє

На початку 70-х років професор П'єр Безьє, проектуючи на комп'ютері корпусу автомобілів «Рено», вперше застосував для цієї мети особливий вид кривих, описуваних рівнянням третього порядку, які згодом стали відомими під назвою криві Безьє (функція Bezier).

Оскільки ці лінії мають особливе значення як для векторної, так і растрової графіки, має сенс розглянути їх більш детально.

В даний час криві Безьє присутні в будь-якому сучасному графічному пакеті. Досить сказати, що всі комп'ютерні шрифти складаються з кривих Безьє. Криві Безьє знаходять також широке застосування і в растровій графіці. Так, в програмі Photoshop використовується термін контур (path), який базується на кривих Безьє. Саме за допомогою цього інструменту ви можете виділити на сканованою фотографії потрібний об'єкт (наприклад, для його вирізання), який буде використаний при створенні фотомонтажу.

Відрізками такої кривої можна апроксимувати як завгодно складний контур. У цьому випадку він буде складатися з набору кривих Безьє. У заклепках сформована з відрізків кривої Безьє лінія може мати злами. Однак за допомогою функції згладжування (smooth) керуючі точки сусідніх відрізків легко вибудовуються в одну лінію, після чого злам зникає. Гнучкість в побудові і редагуванні кривих Безьє багато в чому визначається характеристиками вузлових і керуючих точок, властивості яких будуть розглянуті в наступному розділі.

Поява кривих Безьє викликало справжній переворот в відео і тривимірній графіці. Це пов'язано з тим, що до появи формул Безьє контури комп'ютерних персонажів були ламаними, поверхні - гранованими, а рух - переривчастим, стрибкоподібним, неприродним. Використання кривих Безьє дозволило реалізувати найбільш загальний і інтуїтивно зрозумілий спосіб управління рухом. Відповідно до нього параметрами кривої можна поставити у відповідність параметри руху комп'ютерного персонажа. В результаті рух буде відбуватися за тими ж розглянутим нами правилам. Таким чином, знаменита крива використовується не тільки в двомірної комп'ютерної графіки, але і в тривимірній графіці, відео та анімацій.

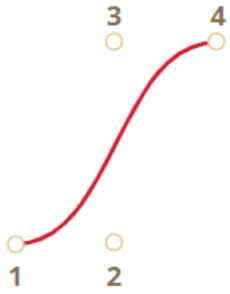
Крива Безьє задається опорними точками.

Їх може бути дві, три, чотири або більше. наприклад:

По двох точках:

По трьох точкам:

По чотирьох точкам:

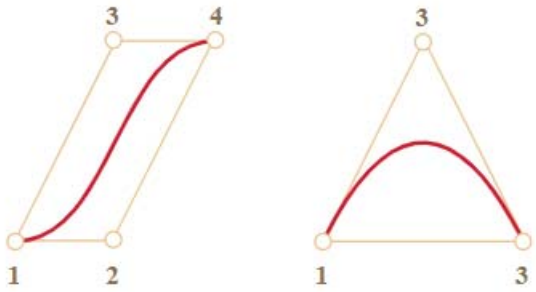


Якщо ви подивіться уважно на ці криві, то помітите:

1. Точки не завжди на кривій. Це абсолютно нормально, як саме будується крива ми розглянемо трохи пізніше.

2. Ступінь кривої дорівнює числу точок мінус один. Для двох точок - це лінійна крива (тобто пряма), для трьох точок - квадратична крива (парабола), для чотирьох - кубічна.

3. Крива завжди знаходиться всередині опуклої оболонки, утвореної опорними точками:

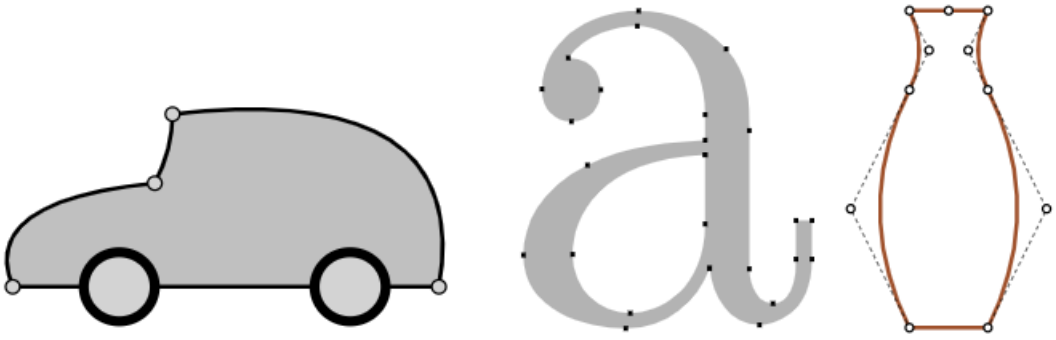


Завдяки останній властивості в комп'ютерній графіці можна оптимізувати перевірку перетину двох кривих. Якщо їх опуклі оболонки не перетинаються, то і криві теж не перетнуться. Таким чином, перевірка перетину опуклих оболонок в першу чергу може дати швидку відповідь на питання про наявність перетину. Перевірити перетинання або опуклі оболонки набагато простіше, тому що це прямокутники, трикутники і т.д. (Див. Рисунок вище), набагато більш прості фігури, ніж крива.

Основна цінність кривих Безьє для малювання в тому, що, рухаючи точки, криву можна міняти, причому крива при цьому змінюється інтуїтивно зрозумілим чином.

Після невеликої практики стає зрозуміло, як розташувати точки, щоб отримати потрібну форму. А, поєднуючи кілька кривих, можна отримати практично що завгодно.

Ось деякі приклади:



4.4. Алгоритм де Кастельжо

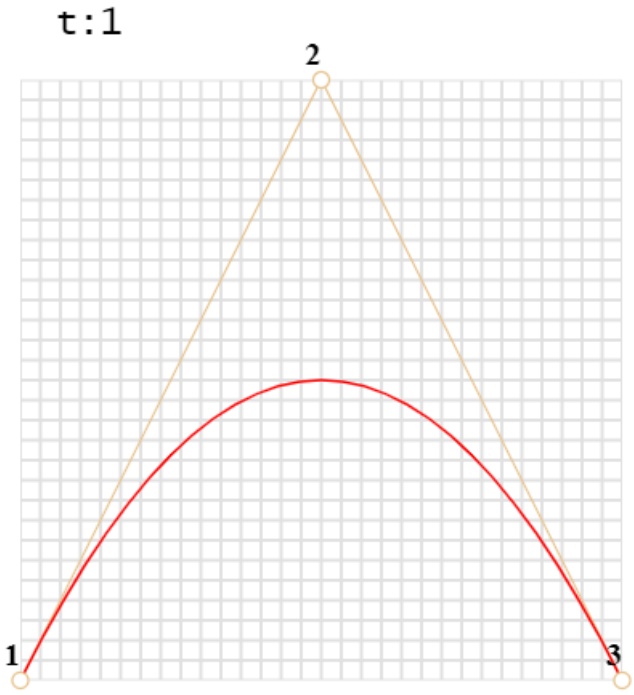
Геометрична інтерпретація алгоритму де Кастельжо:

1. Задана крива Безьє з опорними точками P{0}, ..., P{n}. Поєднавши послідовно опорні точки з першої по останню, отримуємо ламану лінію.

2. Поділяємо кожен отриманий відрізок цієї ламаної в співвідношенні t: (1-t) і з'єднуємо отримані точки. В результаті отримуємо ламану лінію з кількістю відрізків, меншим на один, ніж вихідна ламана лінія.

3. Повторюємо процес до тих пір, поки не отримаємо єдину точку. Ця точка і буде точкою на заданій кривій Безьє з параметром t.

Розглянемо його на прикладі трьох точок.



1. Малюються опорні точки. У прикладі це: 1, 2, 3.

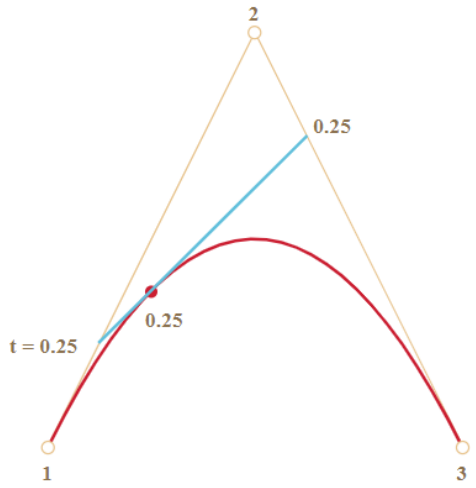
2. Будуються відрізки між опорними точками в такому порядку 1 → 2 → 3. На малюнку вони коричневі.

3. Параметр t «пробігає» значення від 0 до 1. У прикладі використаний крок 0.05, тобто в циклі 0, 0.05, 0.1, 0.15, ... 0.95, 1.

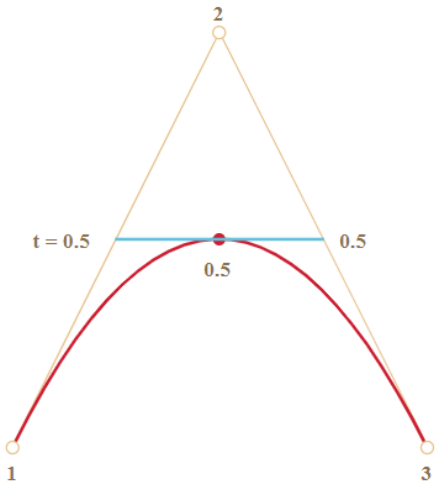
- Для кожного з цих значень t:

- На кожному з коричневих відрізків береться точка, що знаходиться на відстані, пропорційному t, від його початку. Так як відрізків два, то і точок дві.
- Наприклад, при t = 0 - точки будуть на початку, при t = 0.25 - на відстані в 25% від початку відрізка, при t = 0.5 - 50% (на середині), при t = 1 - в кінці відрізків.
- Ці точки з'єднуються. На малюнку нижче з'єднує їх відрізок зображений синім.

При `t=0.25`



При `t=0.5`



4. На отриманому синьому відрізку береться точка на відстані, відповідному t. Тобто, для t = 0.25 (лівий малюнок) отримуємо точку в кінці першої чверті відрізка, для t = 0.5 (правий малюнок) - в середині відрізка. На малюнках вище ця точка відзначена червоним.

5. У міру того, як t «пробігає» послідовність від 0 до 1, кожне значення t додає до кривої точку. Сукупність таких точок для всіх значень утворює криву Безьє. Вона червона і має параболічну форму на картинках вище.

4.5. Контроль знань

1 Чому файл векторного зображення має невеликий обсяг?

- Тому що пікселі дуже малі
- Тому що зображення складається з геометричних примітивів, які можна описати математичними рівняннями
- Тому що немає можливості змінити розміри та пропорції зображення
- Векторне зображення завжди має великий обсяг

2 Що розташовано на самому нижньому рівні ієрархії структури векторної ілюстрації

- обьекти, які представляють собою різноманітні векторні форми.
- картинка, яка об'єднує в своєму складі об'єкти + вузли + лінії + заливки.
- вузли та відрізки ліній, що з'єднують між собою сусідні вузли
- сегменти, які використовуються для побудови контурів

3 Базовий елемент векторної графіки –

- Піксель
- Точка
- Лінія
- n-кутник

4 Парабола в векторній графіці – це

- Крива третього порядку
- Крива другого порядку
- Крива першого порядку
- Крива нульового порядку

5 Крива третього порядку описується …

- Чотирма параметрами
- Двома параметрами
- дев'ятьма параметрами

- одним параметром

6 Який тип вузлових точок не використовується в векторній графіці?

- гладкий вузол (smooth node)
- симетричний вузол (symmetrical node)
- парний вузол (paired node)
- гострий вузол (cusp node)

7 Розмір об'єктів та опис колірних характеристик майже не збільшує розміри файлу

- Так
- Ні

8 Найбільш популярний формат векторної графіки для WEB

- CDR
- SWF
- SVG
- WMF

9 Яка можливість відсутня в векторних редакторах?

- Визначення порядку взаємного розміщення об'єктів
- Вставка і форматування простого і фігурного тексту
- Ретуш фотографії
- Створення об'єктів з примітивів

10 Зовнішній вигляд якого файлу не змінюється при перенесенні на іншу систему та не залежить від версій ПО.

- SWF
- AI
- PDF
- WMF

РОЗДІЛ V. ФРАКТАЛЬНА ГРАФІКА

5.1. Фрактальна графіка: основні поняття

Історія розвитку ідей фрактальної геометрії тісно пов'язана з іменами таких відомих математиків, як Вейєрштрасс, Кантор, Пеано, Хаусдорф, Безикович, Кох, Серпинський і ін. Так Вейєрштрасс уперше ввів в обіг безперервну, але ніде не диференційовану функцію. Хаусдорф в 1919 р. ввів поняття про дробову розмірність множин і привів перші приклади таких множин. Серед них були канторова множина, крива Коха й інші екзотичні об'єкти, мало в той час відомі за межами чистої математики. Оригінальні ідеї Хаусдорфа згодом були істотно розвинені Безиковичем.

Великий внесок у майбутню фрактальну геометрію внесли також знамениті роботи французьких математиків Г. Жуліа й П. Фату, які на початку 20 століття займалися теорією раціональних відображень у комплексній площині. Практично повністю забута, їхня діяльність одержала несподіваний розвиток на початку вісімдесятих років, коли за допомогою комп'ютерів математикам вдалося одержати прекрасні картини, що показують приклади таких відображень. Це вже була ера фрактальної геометрії, оскільки незадовго до цього, у середині 70-х років, у науці з'явився зовсім новий термін "фрактал", що характеризує нерегулярний, але самоподібний об'єкт, що зручно було характеризувати нецілочисленою розмірністю.

Для багатьох стало очевидно, що старі, добрі форми евклідової геометрії дуже програють більшості природних об'єктів через відсутність у них деякої нерегулярності, безладдя й непередбачуваності. Може бути, у майбутньому нові ідеї фрактальної геометрії допоможуть нам вивчити багато загадкових явищ навколишньої природи.

Фрактальні об'єкти, відповідно до свого початкового визначення, мають розмірність, строго перевищуючи топологічну розмірність елементів, з яких вони побудовані.

5.2. Класифікація фракталів

Для того щоб представити все різноманіття фракталів зручно вдатися до їхньої загальноприйнятої класифікації:

- геометричні;

- алгебраїчні;

-стохастичні.

5.3. Методи побудови фракталів

5.4.1. Метод послідовних наближень

Щоб побудувати самоподібний фрактал, наприклад серветку Серпинського, потрібно взяти правильний трикутник, потім вирізати його середину , у результаті чого виходить три маленьких трикутничка. З кожним з них проробляємо ту ж саму операцію й т.д. Це трохи наївне, але наочне пояснення.

Візьмемо довільний набір перетворень подоби площини

$$f_1(x), f_2(x), \ldots, f_m(x) \qquad (5.2)$$

с коефіцієнтами подоби $k_1, k_2, \ldots, k_m$, де $k_i < 1$ для будь-якого $i=1,2,\ldots, m$. Тоді справедлива наступна теорема:

Теорема1. Існує єдина непуста обмежена замкнута (інакше кажучи, компактна) множина F, для якої:

$$F = \bigcup_{i=1}^{m} f_i(F)$$

Множина F називається інваріантною множиною, або аттрактором, системи перетворень (5.1).

Теорема1 дозволяє будувати нові фрактальні об'єкти. Для будь-якої множини А визначимо множину $\Phi(A)$ як:

$$\Phi(A) = \bigcup_{i=1}^{m} f_i(A)$$

Для аттрактора ми маємо рівність *F= Ф(F).*

Як нульове наближення виберемо компактну множину F_0 таку, що:

$$\Phi(F_0) = F_0$$

і покладемо:

$$F_1 = \Phi(F_0), \qquad F_2 = \Phi(F_1), \qquad \ldots$$

Тоді послідовність $F_0, F_1, \ldots, F_n, \ldots$ буде спадною: $F_0 \supset F_1 \supset F_2 \ldots$ і перетин $F = \cap F_n$ є шуканий аттрактор.

У конкретних прикладах нульове наближення легко знайти. Так, для канторівської множини $F_0 = [0,1]$, для серветки Серпинського - вихідний правильний трикутник.

5.4.2. Побудова по точкам або імовірнісний метод

Це найбільш легкий для реалізації на комп'ютері метод. Для простоти розглянемо випадок плоскої самоафінної множини. Отже, нехай $\{S_1,..,S_m\}$ - деяка система афінних стиснень. Відображення S_i представимо у вигляді: $S_i(x)=A_i(x-o_i)+o_i$, *де* A_i - фіксована матриця розміру *2x2* і o_i - двовимірний вектор стовпець.

- Візьмемо нерухливу точку першого відображення S_1 як початкова точка:

x:=1;

Тут ми користуємося тим, що всі нерухливі точки стиснення $S_1,..,S_m$ належать фракталу. Як початкову точку можна вибрати довільну точку й породжена нею послідовність точок стягнеться до фракталу, але тоді на екрані з'являться кілька зайвих точок.

- Відзначимо поточну точку $x=(x_1,x_2)$ на екрані:

putpixel(x_1,x_2,15);

- Виберемо випадковим чином число *j* від *1* до *m* і перерахуємо координати точки *x*:

  $j:=Random(m)+1;$

  $x:=S_j(x);$

- Переходимо на крок 2, або, якщо зробили досить велику кількість ітерацій, то зупиняємося.

Примітка. Якщо коефіцієнти стиснення відображень S_i різні, то фрактал буде заповнюватися точками нерівномірно. У випадку, якщо відображення S_i є подобами, цього можна уникнути невеликим ускладненням алгоритму. Для цього на 3-му кроці алгоритму число *j* від *1* до *m* треба вибирати з імовірностями $p_1=r_1^s,..,p_m=r_m^s$, де r_i позначають коефіцієнти стиснення відображень S_i, а число *s* (розмірність подоби) відшукується з рівняння $r_1^s+...+r_m^s=1$. Рішення цього рівняння можна знайти, наприклад, методом Ньютона [5].

5.4.3. Система ітерованих функцій (IFS)

5.4.4. Метод L-систем

Найбільш простим способом побудови геометричних фракталів є метод *L- систем*, розроблений Аристидом Ліденмайером. Біолог за освітою, Ліденмайер запропонував метод описання складних природних об'єктів і процесів за допомогою простих складових і деяких правил їхнього перетворення. При цьому він використовував певну формальну граматику, що опирається на правила генерації й перетворення символьних рядків.

Нехай є деяка рядок, що складається з довільних символів, називана аксіомою, і набір рядків, називаних правилами. Кожне правило має вигляд **символ → рядок**.

Наприклад:

Аксіома: **acb**

Правила: **a → ab**

b (a)

Спочатку (на 0 кроці) покладемо результуючий рядок рівній аксіомі.

Далі почнемо переглядати рядок ліворуч праворуч. Якщо черговий символ не задає ніякого правила, то він просто переноситься в новий результуючий рядок. Якщо ж черговий символ є першим символом одного із правил, то він заміняється на рядок з відповідного правила. Для розглянутого прикладу:

0-й крок: **a c b** Результуючий рядок: **acb**

1-й крок: **ab c a** Результуючий рядок: **abca**

2-й крок: **ab a c ab** Результуючий рядок: **abacab**

3-й крок: **ab a ab c ab a** Результуючий рядок: **abaabcaba**

і так далі.

Лінденмайер розглядав L-Системи, як формальний спосіб опису розвитку біологічних об'єктів, але пізніше L-Системи знайшли застосування в комп'ютерній графіці. Виявилося, що з їхньою допомогою дуже зручно малювати фрактали й різні природні об'єкти із самоподібною структурою. Метод побудови графічних об'єктів за допомогою L-Систем ще називають "черепашачою графікою" (turtle geometry).

Нехай є деякий виконавець ("черепашка"), що може виконати набір команд. Черепашка переміщається по площині. Поточний стан черепашки задається координатами x , y і кутом a, що визначає напрямок, у якому повзе черепашка. Припустимо, що в черепашки є пам'ять, організована у вигляді стека (тобто черепашка може запам'ятати кілька значень, але згадувати їх вона буде у зворотному порядку: те, що запам'ятала останнім, згадає першим, те, що запам'ятала передостаннім, згадає другим і т.д.). Нехай початкове положення черепашки задається координатами x_0 , y_0 і напрямком руху a_0. Крім того, нехай задане значення кроку h, на який переміщається черепашка

по команді "уперед" і кут b, на який повертає черепашка по команді "повернути праворуч" або по команді "повернути ліворуч".

Нехай черепашка вміє виконувати наступні команди (кожна команда кодується одним символом):

"**F**" – переміститися вперед на крок h у напрямку a, залишивши слід (намалювавши відрізок);

"**f**" – переміститися вперед на крок h у напрямку a, не залишаючи сліду;

"+" – повернути праворуч (за годинниковою стрілкою) на кут b (змінити напрямок руху);

"**–**" – повернути ліворуч (проти годинникової стрілки) на кут b (змінити напрямок руху);

"**[**" – запам'ятати (відкласти в стек) поточний стан (x, y, a);

"**]**" – згадати (взяти зі стека й установити) останнє збережене в пам'яті стан (x, y, a);

Програмою для черепашки є рядок, у якій, крім зазначених символів, можуть зустрічатися й будь-які інші. Черепашка переглядає рядок-програму символ за символом. Команди вона виконує, а символи, що не є командами, пропускає.

Для одержання кривої досить задати значення кроку h, початкове положення черепашки й напрямок руху, а потім виконати отриманий програму-рядок.

Приклад побудови фрактала - трикутник Серпінського. Для цього фрактала:

аксіома: FXF--FF--FF

правила: F → FF

X (-iFXF++FXF++FXF-i

кут b=360/6=60 (

На 0 кроці: FXF--FF--FF

На 1 кроці: FF --FXF++FXF++FXF-- FF -- FF FF -- FF FF …

У цьому прикладі керуючий рядок містить символ не є командою для черепашки (символ X). При побудові чергового предфрактала цей символ буде ігноруватися.

За допомогою L-Систем можна зберегти складне зображення, запам'ятавши тільки аксіому й утворюючі правила. Але з ростом числа кроків швидко росте й довжина керуючого рядка й, отже, час виконання програми.

5.5. Серветка й килим Серпинського

Регулярний фрактал, називаний серветкою Серпинського (Sierpinski gasket), виходить послідовним вирізанням центральних рівносторонніх трикутників так, як показано на рис. 5.6.

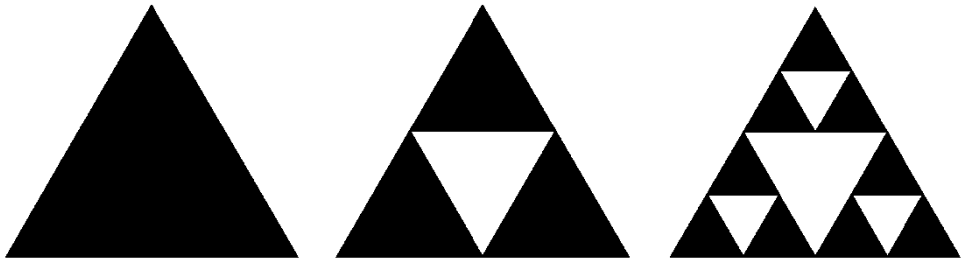


Рис.5.6. Побудова серветки Серпинського

У результаті виходить "дірява" фігура (рис. 5.6.), що складається з нескінченного числа ізольованих точок. Фрактальна розмірність серветки Серпинського підраховується по формулі (1.4)

$$D = \frac{\ln 3}{\ln 2} = 1.5849\ . \qquad (5.2)$$

Підрахуємо тепер периметр виключених областей. Якщо сторона вихідного трикутника дорівнює 1, то на першому кроці побудови периметр

центрального трикутника дорівнює 3/2. На другому кроці до нього додаються три нових трикутники із загальним периметром, рівним 9/4 і т.д. Очевидно, що на п-*му* кроці периметр *P* визначається сумою геометричної прогресії.

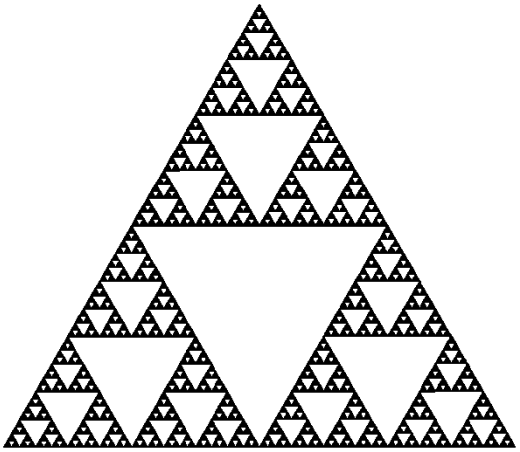


Рис. 5.7. Серветка Серпинського

$$\mathcal{P} = \sum_{k=1}^{n} \left(\frac{3}{2}\right)^k = 3\left[\left(\frac{3}{2}\right)^n - 1\right] \approx 3\left(\frac{3}{2}\right)^n \tag{5.8}$$

З іншого боку, масштаб довжини на *n-му* кроці дорівнює $l = 1/2^n$. Тому формула для периметра, вираженого через цей масштаб, здобуває вид, схожий з формулою (5.8) для довжини берегової лінії

$$\mathcal{P} \approx l\left(\frac{2}{l}\right)^D \tag{5.9}$$

де *D* визначається формулою (5.9).

Аналогічно серветці Серпинського можна побудувати квадратний **килим Серпинського** (triadic Sierpinski carpet), що є двовимірним аналогом канторівської множини виключених середніх третин.

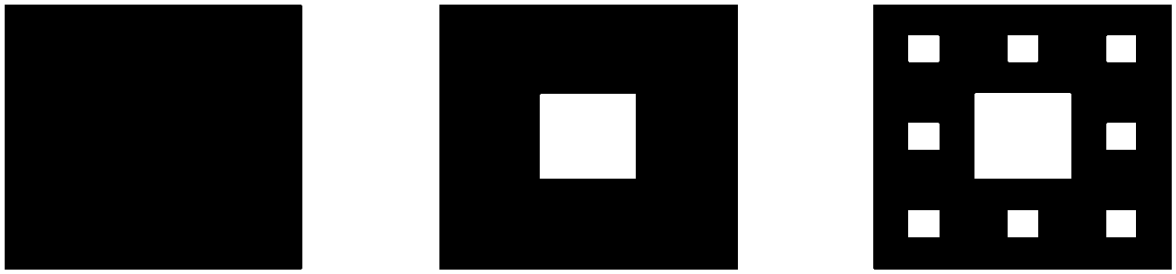


Рис.5.10. Побудова квадратного килима Серпинського

Рецепт його створення полягає в наступному. Спочатку береться квадрат з довжиною сторони, рівній одиниці. Потім кожна зі сторін квадрата ділиться на три рівні частини, а весь квадрат, відповідно, на дев'ять однакових квадратиків зі стороною, рівної *1/3*. З отриманої фігури вирізьблюється центральний квадрат. Потім такій же процедурі піддається кожний з 8 квадратиків, що залишилися, і т. д (рис. 5.11).

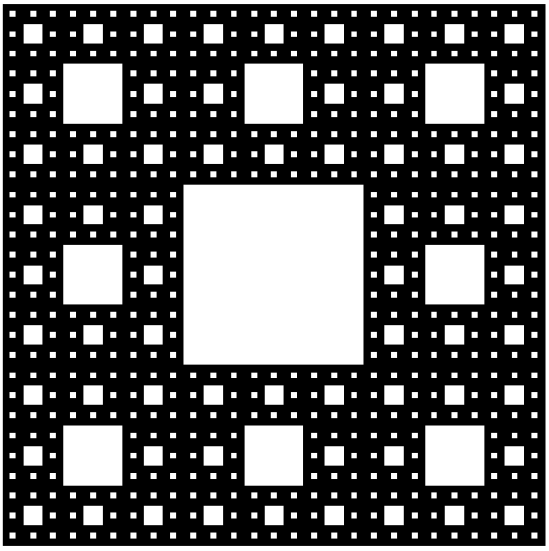


Рис. 5.11. Квадратний килим Серпинського

У результаті виходить дірявий квадратний килим Серпинського зі значенням фрактальної розмірності

$$D = \frac{\ln 8}{\ln 3} = 1.8928 \tag{5.4}$$

Він також являє собою приклад ідеального самоподібного фрактала. Його фрактальна розмірність, однак, більше, ніж у серветки Серпинського, тобто він є менш дірявим.

5.6. Триадна крива Кох

На рис. 5.11 показано, як побудувати триадну криву Кох. Триадна крива Кох-Один зі стандартних прикладів, що приводяться на підтвердження того, що крива може мати фрактальну розмірність *D> 1*.

Побудова кривої Кох починається із прямолінійного відрізка одиничної довжини *L(1) = 1*. Цей вихідний відрізок називається затравкою і може бути замінений яким-небудь багатокутником, наприклад рівностороннім трикутником, квадратом. Затравка- це 0-ві покоління кривої Кох. Побудова кривої Кох: кожну ланку затравки заміняємо утворюючим елементом, позначеним на рис. 2.5 через *n* = 1. У результаті такої заміни ми одержуємо 1-е покоління - криву із чотирьох прямолінійних ланок, кожне довжиною по 1/3.

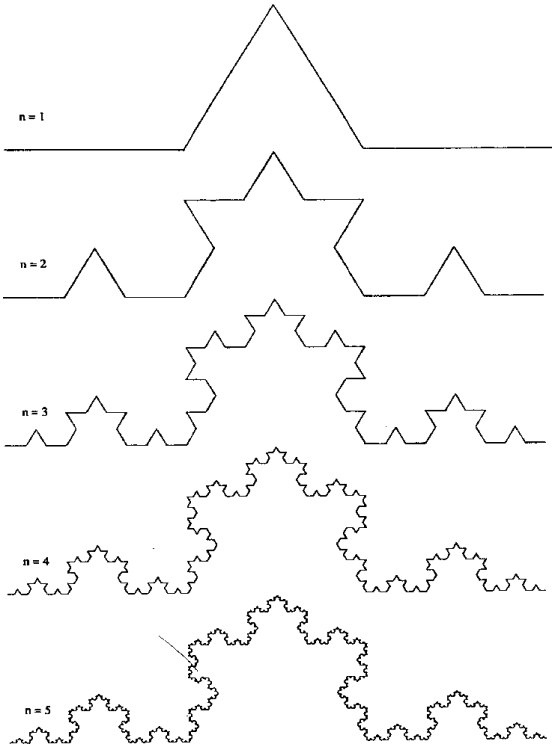


Рис.5.11. Побудова триадної кривої Кох

Довжина всієї кривої 1-го покоління становить *L(1/3) = 4/3*. Наступне покоління виходить при заміні кожної прямолінійної ланки зменшеним утворюючим елементом. У результаті ми одержуємо криву 2-го покоління, що складається з $N = 4^2 = 16$ ланок, кожна довжиною $б=3^{-2}=1/9$. Довжина кривої

2-го покоління дорівнює $L(1/9) = (4/3)^2 = 16/9$. Заміняючи всі ланки попереднього покоління кривої зменшеним утворюючим елементом, одержуємо нове покоління кривої. Крива *n-го* покоління при будь-якому кінцевому n називається предфракталом.

У вигляді виключення простежимо у всіх подробицях за тим, як виходить вираження для *D*. Довжина предфрактала *n-го* покоління визначається формулою

$$L(б) = (4/3)''.$$

Довжина кожної ланки становить

$$б = 3^{-n}.$$

Зауважуючи, що число поколінь n представимо у вигляді

$$n = — ln\ б/ln\ 3,$$

запишемо довжину предфрактала у вигляді

$$L(\delta) = (4/3)^n = \exp\left(- \frac{\ln\ \delta[\ln\ 4 - \ln\ 3]}{\ln\ 3} \right) = \delta^{1-D} \tag{5.5}$$

Формула (2.5) має вигляд наближеної формули (2.1), у якій

$$D = ln\ 4/ln\ 3 \sim 1{,}2628.$$

Число сегментів дорівнює $N(б) = 4'' = 4^{-\ln б/\ln 3}$ і може бути записане у вигляді

$$N(б) = б^{-p}. \tag{5.6}$$

Як буде показано далі, *D*-фрактальна розмірність триадної кривої Кох. Насамперед помітимо, що побудова Коха дозволяє в будь-якому поколінні одержувати нормальну криву кінцевої довжини. Мандельброт називає такі криві предфракталами. Але при збільшенні числа поколінь величина б прагне до нуля й довжина кривої розходиться. Ясно, що множина точок, що одержує межу нескінченно великої кількості ітерацій процедури Кох, не є кривою, для якої довжина є зручною мірою [36]. Але якщо ми виберемо пробну функцію $h(б) = б^d$, то одержимо *d*-міру

$$M_d = \sum h(\delta) = N(\delta)\, h(\delta) = \delta^{-D} \delta^d.$$

Видно, що міра M_d залишається кінцевою і дорівнює одиниці тільки в тому випадку, якщо розмірність *d*, що входить у пробну функцію *h(б),* дорівнює *D*. Критична розмірність і, отже, розмірність Хаусдорфа-Безиковича для триадної кривої Кох дорівнює $D = ln\ 4/ln3$. На кожній стадії побудови предфрактали Кох можуть бути розтягнуті в пряму лінію, тому топологічна розмірність триадної кривої Кох дорівнює $D_г = 1$. Тому що розмірність Хаусдорфа-Безиковича *D* для кривої Кох більше її топологічної розмірності D_m - що крива Кох є фрактальна множина із фрактальною розмірністю $D = ln\ 4/ln3$.

5.7. Множина Мандельброта

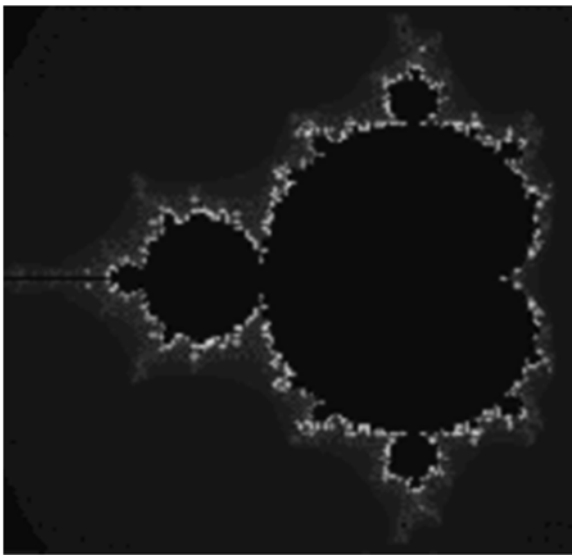


Рис 5.12. Множина Мандельброта

Множини Мандельброта й Жулиа, імовірно, дві найпоширеніші серед складних фракталів. Їх можна знайти в багатьох наукових журналах, обкладинках книг, листівках, і в комп'ютерних хоронителях екрана.

Множина Мандельброта, що була побудовано Бенуа Мандельбротом, напевно перша асоціація, що виникає в людей, коли вони чують слово фрактал.

Цей фрактал генерується простою формулою *Zn+1=Zna+C*, де *Z і C* - комплексні числа й а - додатнє число.

Множина Мандельброта, що найчастіше можна побачити - це множина Мандельброта 2-го ступеня, тобто *a=2*. Той факт, що множина Мандельброта не тільки *Zn+1=ZnI+C*, а фрактал, показник у формулі якого може бути будь-яким позитивним числом увів в оману багатьох. На рис.3.7 приклад множини Мандельброта для різних значень показника *a*.

Рис 5.13. Поява пухирців при а=3.5

Також популярний процес *Z=Z*tg(Z+C)*. Завдяки включенню функції тангенса, виходить множина Мандельброта, оточена областю, що нагадує яблуко. При використанні функції косинуса, виходять ефекти повітряних пухирців. Коротше кажучи, існує нескінченна кількість способів настроювання множини Мандельброта для одержання різних картинок.

Крок 0:

вибрати0 *pmin, pmax, qmin, qmax*, .

вибрати число *M*, що вважається нескінченно великим.

покласти *dp=(pmax-pmin)/(A-1), dq=(qmax-qmin)/(B-1).*

для всіх пар *(Np, Nq),* де *Np=0,1... A-1* і *Nq=0,1...B-1* виконати наступну процедуру.

Крок 1:

покласти *p(0)=pmin+Np*dp, q(0)=qmin+Nq*dq, k=0, x(0)=y(0)=0. x* і *y* - обнуляються при кожному звертанні до 1 кроку оскільки в процесі ітерації значення x и y змінюються.

Крок 2 (ітерація):

обчислити $x(k+1)$ і $y(k+1)$, використовуючи формули

$x(k+1)=x(k)^2*x(k)-y(k)^2 + p,$

збільшити лічильник k на 1 $(k:=k+1)$.

Крок 3 (оцінка):

обчислити $r=x(k)^2+y(k)^2$.

якщо $r>M$, то вибрати колір k і йти на крок 4.

якщо $k=M$, то вибрати колір 0 (чорний) і йти на крок 4.

$r<=M$, $k<M$. Повернутися на крок 2.

Крок 4:

Приписати колір k точці екрана $(p(n), q(n))$ і перейти до наступної точки, починаючи із кроку 1.

5.8. Множина Жюліа

Дивно, але множина Жюліа утворюється по тій же самій формулі, що й множина Мандельброта. Множину Жюліа було винайдено французьким математиком Гастоном Жюліа, по імені якого й була названа множина. Перше питання, що виникає після візуального знайомства з множинами Мандельброта й Жюліа це "якщо оба фрактала згенеровані по одній формулі, чому вони такі різні?"

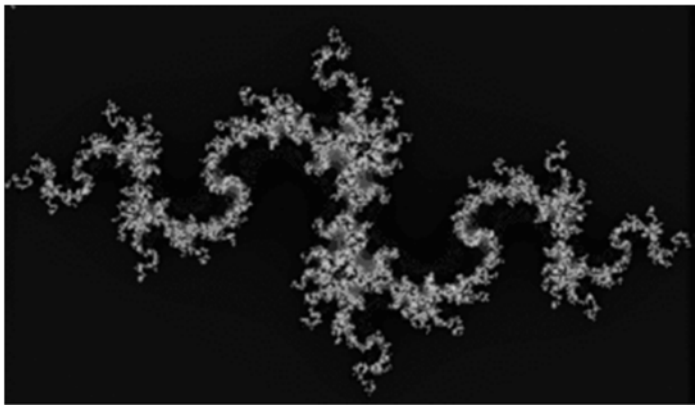


Рис 5.13. Множина Жюліа

Фрактал Мандельброта - це, насправді, множина фракталів Жюліа, з'єднаних разом (рис 5.14). Кожна точка (або координата) множини Мандельброта відповідає фракталу Жюліа.

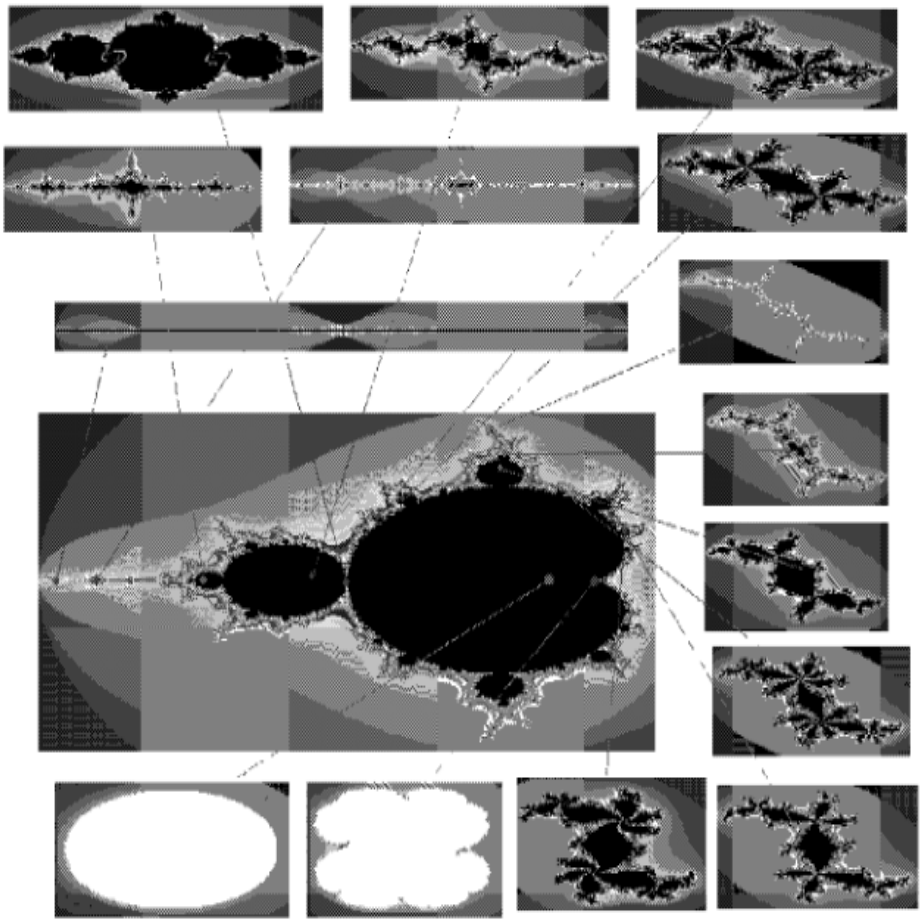


Рис 5.14. Фрактал Мандельброта - це, насправді, множина фракталів Жюліа, з'єднаних разом

Множинау Жюліа можна згенерувати використовуючи ці точки як початкові значення в рівнянні *Z=ZI+C*. Але це не виходить, що якщо вибрати точку на фракталі Мандельброта й збільшити її, то можна одержати фрактал Жюліа. Ці дві точки ідентичні, але тільки в математичному сенсі. Якщо взяти цю точку й прорахувати її по даній формулі, можна одержати фрактал Жюліа, що відповідає певній точці фрактала Мандельброта.

Крок 0:

Вибрати параметр *c=p+iq*, вибрати *Xmin = -1.5, Ymin = -1.5, Xmax=1.5,Ymax=1.5.*

Це означає, що *p*- реальна змінна*, iq* - уявлювана, але теж цілком реальна змінна. Наприклад, *p = 0.32 c_img = 0.043*. Ці параметри є константою при побудові одного відображення і їхнього значення визначають малюнок. Масштабувати відображення можна міняючи значення *Xmax, Xmin, Ymax, Ymin.*

Вибрати число *M,* що вважається нескінченно великим.

Цей параметр визначає максимальне число ітерацій

Обчислити *dx=(Xmax-Xmin)/(A-1), dy=(Ymax-Ymin)/(B-1).*

Для всіх пар (*Nx, Ny), де Nx=0,1... A-1* і *Ny=0,1...B-1* виконати наступну процедуру.

Крок 1(перший і другий цикли):

Покласти *x(n)=xmin+Nx*dx, y(n)=ymin+Ny*dy, k=0*. Тут послідовно змінюються тільки *Nx* і *Ny*, на одну змінну *Nx* приходиться *B* змінних *Ny*. Піксель який буде надалі офарблюватися має координати *(x(n),y(n)).*

Крок 2 (третій цикл, ітерація):

Обчислити *x(k+1) і y(k+1),* по формулах

$x(k+1)=x(k)^2*x(k)-y(k)^2 + p,$

$y(k+1)=2*x(k)*y(k) + q.$

Збільшити лічильник *k* на 1 (*k:=k+1*).

Крок 3 (оцінка усередині 3 цикли):

Обчислити $r=x(k)^2+y(k)^2$.

Якщо r>M, то йти на крок 4.

Якщо *k=K*, то вибрати колір 0 (чорний) і йти на крок 4. У цьому рядку щось не так, помилка напевно, справа в тому, що кількість кольрів К може рівнятися 2, 16, 255 і т.д. і переривати цикл із *M>K* нема рації. У цілому ця умова стосується тільки алгоритму розфарбовування. Рядок одержує законний і концептуальний вид переривання ітерації при досягненні припустимого максимуму якщо *K=M*, або повністю:

Якщо *k=M* то вибрати колір 0 (чорний) і йти на крок 4.

Якщо $r<=M$ і $k<M$. Повернутися на крок 2.Тут теж була помилка типу (Якщо $r<=M$, $k<K$).

Крок 4:

Приписати колір k точці екрана $(x(n), y(n))$ і перейти до наступної точки, починаючи із кроку 1.

5.9. Застосування фракталів

Мова фрактальної геометрії необхідна, також, при вивченні поглинання або розсіювання випромінювання в пористих середовищах, для характеристики сильної турбулентності, при моделюванні властивостей поверхні твердих тіл, для опису діелектричного пробою й блискавки, при аналізі процесів руйнування матеріалів, при дослідженні різних стадій росту речовини за рахунок дифузії й наступної агрегації, у квантовій механіці при описі геометричної структури хвильових функцій у точці переходу Андерсона метал-діелектрик. Дивно те, що подібні геометричні форми зустрічаються у зовсім різних галузях науки: в астрофізику при описі процесів кластеризації галактик у Всесвіті, у картографії при вивченні форм берегових ліній і розгалуженої мережі річкових русел і, наприклад, у біології, при аналізі будови кровоносної системи або розгляді складних поверхонь клітинних мембран.

При фрактальному підході хаос перестає бути синонімом безладдя й знаходить тонку структуру. Фрактальна наука ще дуже молода, і в неї має бути велике майбутнє. Краса фракталів далеко не вичерпана й ще подарує нам чимало шедеврів - тих, які тішать око, і тих, які доставляють насолоду розуму.

Застосування теорії фракталів у фізиці

Розвиток фрактальних концепцій у фізиці наповнило багато напрямків фізики новим змістом. У результаті досліджень у цьому напрямку стало зрозуміло, що конденсований стан речовини може являти собою не тільки суцільне середовище, але й мати пористу фрактальну структуру, особливо, якщо процес утворення конденсованої системи протікає при нерівновагих умовах. Тому фрактальні концепції вважаються фізичними концепціями XX в.

Одне із властивостей фрактальних систем пов'язане з їхньою взаємодією з електромагнітними хвилями. Фрактальні системи, як правило, мають більш високі питомі випромінювальні параметри, ніж суцільні системи.

Фрактальний клубок і фрактальні нитки

Поряд з випромінюванням систем, що містять найпростіші фрактальні структури - фрактальні агрегати, коротко проаналізована і більш складна фрактальна структура - фрактальний клубок, що володіє рядом специфічних властивостей, у тому числі й випромінювальними. Фрактальний клубок як фізичний об'єкт був уведений як модель каркасу кульової блискавки.

Фрактальна концепція кульової блискавки опирається на наявність у неї твердого каркаса, і його моделі. Перша фрактальна модель кульової блискавки використовувала концепцію фрактальних агрегатів і пов'язану з ними інформацію. Основою для наступної моделі - моделі фрактального клубка - були експерименти по утворенню фрактальних ниток у результаті лазерного опромінення металевої поверхні. Фрактальні нитки(рис. 5.16) утворюються в результаті швидкої конденсації слабко іонізованної металевої пари в зовнішньому електричному полі, а фрактальний клубок являє собою систему переплетених фрактальних ниток.

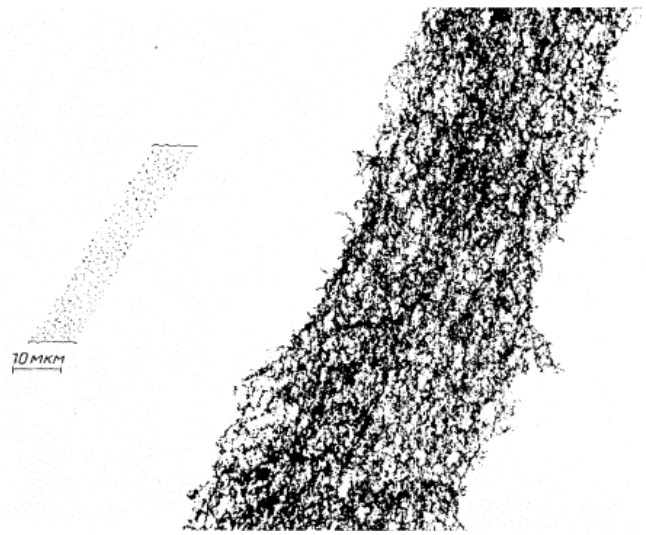


Рис. 5.16. Елемент фрактальної нитки. Ліворуч - фотографія на просвіт на звичайному мікроскопі. Праворуч - фотографія на електронному мікроскопі.

Фрактальні кластери

Фрактальний кластер (фрактальний агрегат) — хаотичний фрактал, що утворюється при асоціації твердих аерозолів у газі у випадку дифузійного характеру їхнього руху, а також у результаті конденсації в складних нерівновагих умовах, наприклад, при злипанні твердих часток, що рухаються за певним законом, кластера (агрегату). Як приклад на рис.5.16 наведений тетраэдричний фрактальний кластер.

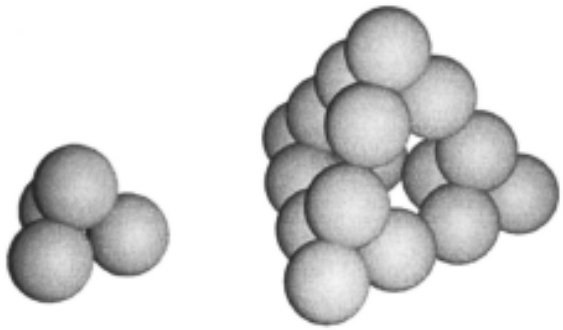


Рис. 5.16. Фрактальний кластер

Активним дослідником і автором численних публікацій, присвячених властивостям фрактальних кластерів, є Б. М. Смирнов.

Фрактальні кластери мають зв'язок й аналогії з рядом фізичних систем і процесів. Сюди ставляться процеси й структури при утворенні кластерів і затвердінні колоїдних розчинів, коагуляція, процеси перколяції, утворення полімерів, діелектричний пробій, деякі біофізичні процеси й т. д . Хоча кожній із цих проблем властиві свої специфічні особливості, загальні подання про фрактальні кластери створюють позиції, з яких можна проводити більш глибокий аналіз досліджуваних систем і явищ.

Серед експериментальних досліджень фрактальних структур перше місце займає дослідження плівок, що утворюються на поверхні. Якщо плівка створюється із твердих аерозолів, напиляемых на поверхню, то при відповідних сортах аерозолю й підложки плівка розвивається у вигляді фрактального кластера. Це має місце, якщо аерозолі погано зв'язуються з підложкою, але добре зв'язуються один з одним.

Інший приклад спостережуваного поверхневого фрактального кластера. На підложці, що одночасно служить електродом і поміщено в електроліт, під дією прикладеної до електродів напруги виділяється метал – цинк. Поки різниця потенціалів між електродами не перевищує деякої критичної величини, осадження цинку на підложку відбувається у вигляді фрактального кластера із фрактальною розмірністю 1,66 ± 0,03 (рис. 5.17). Збільшення різниці потенціалів над критичною змінює характер осадження металевих часток на підложці й приводить до збільшення фрактальної розмірності утвореного кластера.

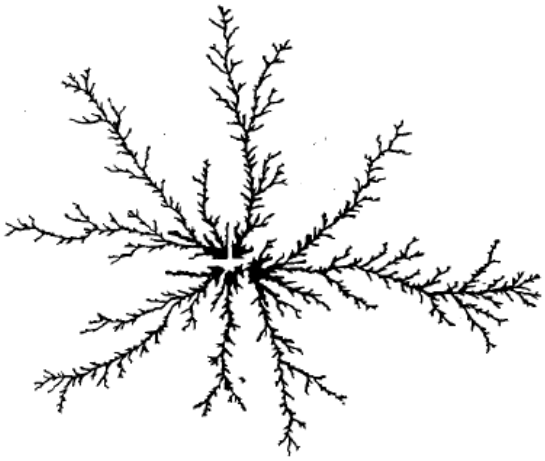


Рис. 5.17. Фрактальний кластер цинку, утворений при виділенні цинку на поверхні в процесі електролізу77. Фрактальна розмірність кластера 1,08 ± 0,03

Поряд із процесами утворення фрактальних кластерів із твердих аерозолів в обсязі й на поверхні існують інші процеси і явища, де корисні знання про фрактальну структуру кластерів. На рис. 3.8 наведені фігури Лихтенберга, які дають картину діелектричного пробою в газовому, рідкому або твердому ізоляторі. В умовах роботи один з електродів розташовувався в центрі, а інший являв собою окружність на діелектричній поверхні.

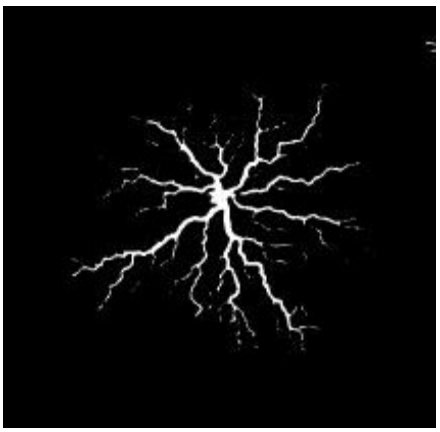


Рис. 5.18. Фігура Лихтенберга (проінтегрована за часом інтенсивність світіння від діелектричного пробою)

Світні області діелектрика, по яких протікає пробійний струм, нагадують фрактальний кластер. Його фрактальна розмірність, отримана зі співвідношення між розмірами й числом елементів, становить приблизно 1,7. Через обмежений дозвіл не вдається визначити цю величину по кореляції між плотностями елементів. Найпростіша модель такого розряду, що включає рух світної точки по ламаних лініях із заданим розміром прямих відрізків, дає для фрактальної розмірності фігури Лихтенберга значення $n = 1{,}75 \pm 0{,}02$ [3].

Наведені приклади показують, що представлення про фрактальні кластери й фрактальні структури розширюють наші знання про ряд систем і явищ і сприяють більш детальному аналізу реальних процесів і явищ. Ці представлення є тим фундаментом, на якому можуть успішно розвиватися експериментальні дослідження фрактальних структур і динаміки їхнього утворення.

РОЗДІЛ VI. ТРИВИМІРНА ГРАФІКА

6.1. Основні типи представлення об’єктів у 3D-просторі

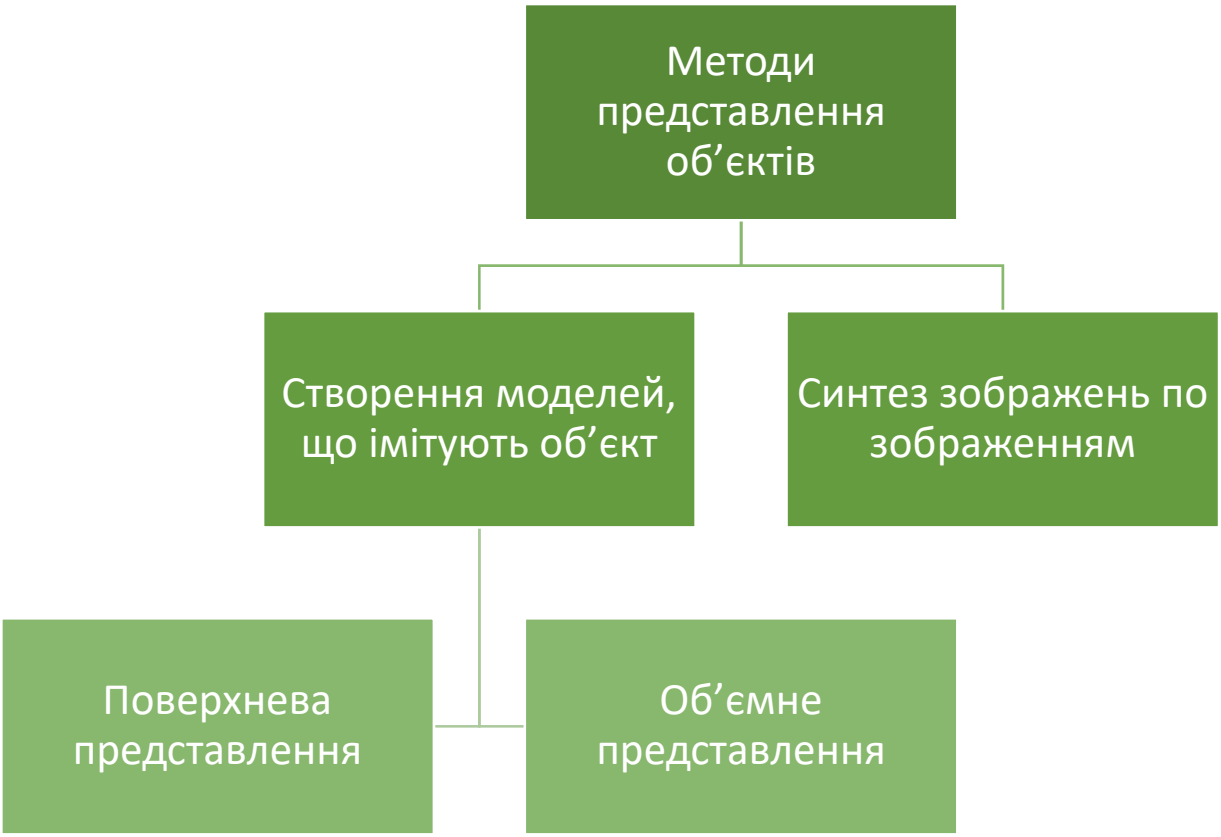
Отже, ми хочемо навчитися працювати з тривимірними об'єктами, наша мета - обробка (зокрема, візуалізація) 3D об'єктів і сцен за допомогою ЕОМ. Для цього необхідно мати моделі відповідних об'єктів у формі, придатній для такої обробки.

Теоретично нам може знадобитися:

1. Властивості тіла, пов'язані з простором (його положення щодо інших тіл, геометрична форма, розмір тіла і т.д.)
2. Інформація про матеріал тіла (колір, текстура, відбивна здатність, прозорість і т.д.)
3. Інформація, що характеризує взаємодію об'єктів (щільність, можливість деформуватися, маса і т.д.)

Цей набір даних може змінюватися в залежності від поставлених перед нами завдань і конкретних об'єктів, які ми хочемо представити.

Існує дві великі групи підходів для вирішення такого завдання. Відповідно до першими, ми створюємо окрему модель, яка імітує наш тривимірний об'єкт, і потім відображаємо її на екран, а відповідно з другими ми синтезуємо необхідне нам зображення тривимірних об'єктів по вже існуючим зображенням. Методи першої групи, в свою чергу, діляться на дві підгрупи: поверхневе уявлення (зберігається тільки межа об'єкта) і об'ємне уявлення (зберігається інформації про всі точки об'єкта). Однак, варто зазначити, що метод синтезу зображень із зображень було трохи відособлене, так як не дозволяє здійснювати ніяких дій, крім візуалізації об'єктів.



Поверхневе уявлення (Boundary representation, B-rep)

У поверхневому поданні об'єкт створюється за допомогою набору тонких поверхонь, що складають його кордон. Як правило, поверхневе уявлення використовується в тих областях, де немає необхідності обробляти будь-яким чином внутрішність тіла: дизайнерські проекти, моделювання обводів якого-небудь виробу, створення об'єктів з нестандартними елементами (наприклад, округлення із змінним радіусом, гвинтоподібна равлик) і т. д.

Існує кілька основних способів завдання границі тіла.

Неявна функція (implicit function)

Завдання 3 D об'єкта за допомогою неявної функції полягає в тому, що ми вказуємо якусь функцію $F(x,y,z)$, нулями якої будуть точки (x_i,y_i,z_i), що утворюють нашу поверхню, або дають її наближення. Так, наприклад, неявна функція для сфери радіуса r (див. Рис. 1), з центром в точці (x_0,y_0,z_0) має вигляд:

$$F(x,y,z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2$$

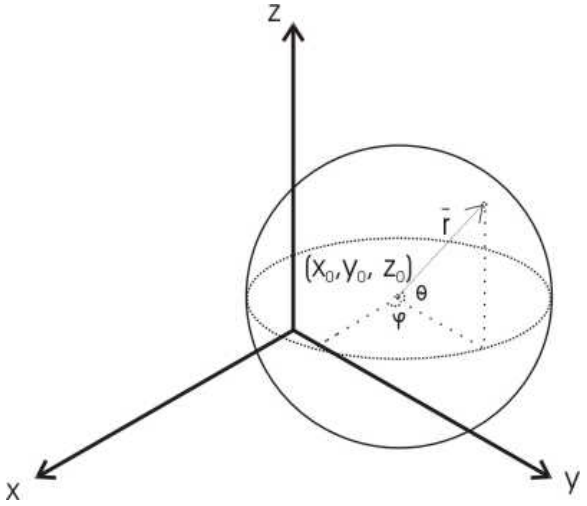


Рис.6.1. Сфера.

Параметричне завдання, сплайни, NURBS

При параметричному завданні координати точок поверхні розглядаються як певні функції від двох параметрів, що пробігають деякий набір значень.

$$\begin{cases} x = x(u,v) \\ y = y(u,v) \\ z = z(u.v) \end{cases}$$

Так, наприклад, параметризація тієї ж сфери (див. Рис. 1) щодо двох кутів φ (довгота) і θ (широта) матиме вигляд:

$$\begin{cases} x = r\cos\theta\cos\varphi \\ y = r\cos\theta\sin\varphi, \quad 0 \le \varphi < 2\pi, \ -\dfrac{\pi}{2} \le \theta \le \dfrac{\pi}{2} \\ z = r\sin\theta \end{cases}$$

Крім цього використовуються різного роду особливі параметричні поверхні, наприклад, поверхні Безьє, B -сплайни, зокрема, NURBS ("Non-Uniform Rational B-Spline", неоднорідний раціональний B-сплайн) .

Якщо говорити коротко, то поверхня Безьє будується за деякою топологічно прямокутної сітці відповідно до такої формули:

$$Q(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j} J_{n,i}(u) K_{m,j}(w),$$

де $J_{n,i}(u)$ і $K_{m,j}(w)$ - базисні функції Берштейна в параметричних напрямках *u* та *w* :

$$J_{n,i}(u) = \binom{n}{i} u^i (1-u)^{n-i}, \quad K_{m,j}(w) = \binom{m}{j} w^j (1-w)^{m-j}$$

Елементи $B_{i,j}$ - вершини задає полігональної сітки. Кожна з граничних кривих поверхні Безьє є кривої Безьє.

B - сплайни будуються у відповідності з наступними формулами:

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,l}(w),$$

де $N_{i,k}(u)$ і $M_{j,l}(w)$ - базисні функції B -сплайна в біпараметріческіх напрямках:

$$N_{i,1}(u) = \begin{cases} 1, \text{ если } x_i \le u \le x_{i+1} \\ 0, \text{в противном случае} \end{cases}$$

$$N_{i,k}(u) = \frac{(u - x_i) N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u) N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}}$$

$$M_{i,1}(u) = \begin{cases} 1, \text{ если } y_j \le w \le y_{j+1} \\ 0, \text{в противном случае} \end{cases}$$

$$M_{j,l}(u) = \frac{(w - yj) M_{j,l-1}(w)}{y_{j+l-1} - y_j} + \frac{(y_{j+l} - w) M_{j+1,l-1}(w)}{y_{j+l} - y_{j+1}}$$

Елементи $B_{i,j}$ - вершини задає полігональної сітки,

Ці параметричні поверхні дуже широко використовуються в різних CAD - системах (в Росії використовується термін САПР - системи автоматичного проектування). Так, за допомогою них моделюються і розраховуються обводи автомобілів, форми деталей і т.п.

Полігональні поверхні

При використанні даного методу поверхню (surface) представляється у вигляді набору деяких багатокутників (face) в просторі (див. рис. 6.2).

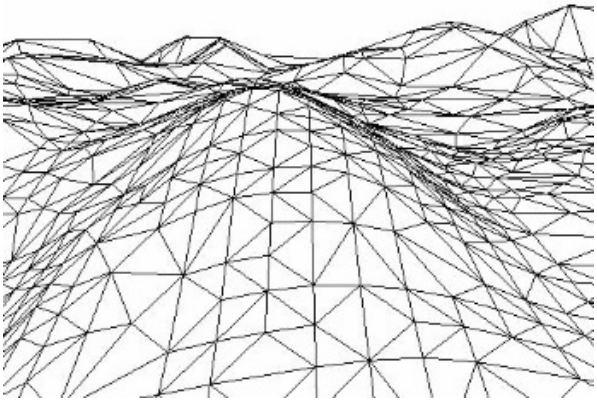


Рис. 6.2**.** Триангуляція поверхні.

Часто в якості багатокутників використовуються трикутники, в цьому випадку саме розбиття поверхні (наближене уявлення трикутниками) називається тріангуляції поверхні. Топологія отриманої при цьому сітки описується наступним чином:

1. Об'єктами сітки є вершини, ребра і трикутники (задаються трьома вершинами або трьома ребрами), а в більш загальному випадку - face .

2. У будь-якої вершини є властивість валентності (тобто число багатокутників, що містять її).

3. Для будь-якого трикутника існує не більше одного іншого трикутника, інцидентних для першого по фіксованому ребру. Тобто, зокрема, не може бути ось такий ситуації:

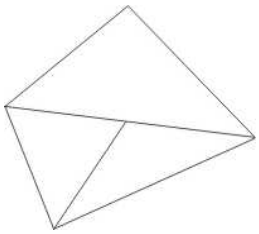


Рис 6.3 . Порушення умови про инцидентности.

При такому способі завдання ми можемо, наприклад, організувати обхід в деякому напрямку по трикутниках, що містить фіксовану вершину, просто переходячи кожен раз до трикутника, інцидентних по ребру для попереднього.

Зрозуміло, при тріангуляції якост ство одержуваної поверхні тим краще, чим більше трикутників ми будемо для цього використовувати (див. Рис. 6.4).

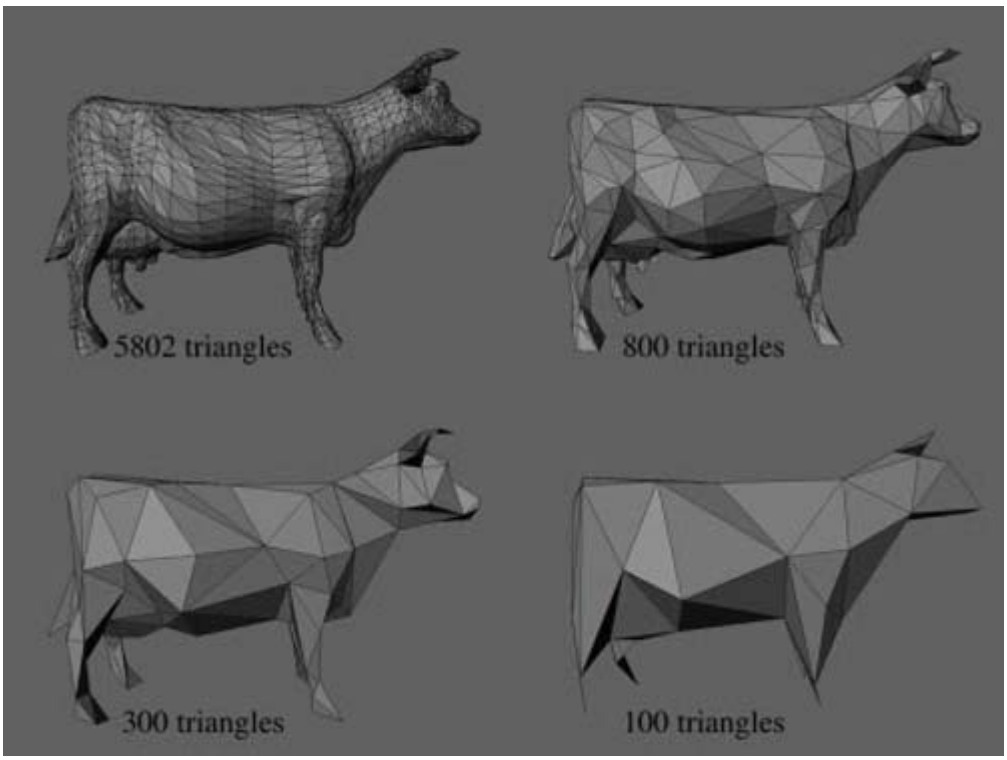


Рис. 6.4 **.** Залежність якості зображення від числа трикутників.

Конструктивна геометрія тел (Constructive Solid Geometry , CSG)

Метод конструктивної геометрії тел полягає в тому, що ми отримуємо потрібну нам поверхню як результат застосування послідовності різних множинних операцій (об'єднання, перетинання, різниці і т.д.) до деяких примітивам (тобто мінімальним об'єктами для побудови). Як примітивів можу виступати паралелепіпеди, кулі, конуси, піраміди і т.д. Сама поверхня задається за допомогою дерева побудови (див. Рис. 6.5).

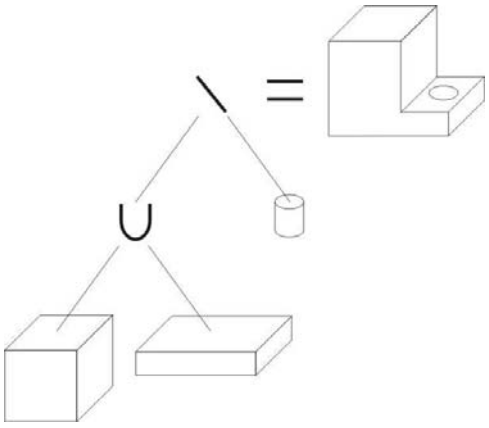


Рис. 6.5 Дерево побудови.

Об'ємне подання (Volume representation)

Методи об'ємного уявлення тел зберігають інформацію про будь-яких, не обов'язково видимих, частинах тіла. Всі вони, як правило, вимагають набагато більше місця для зберігання об'єкта, але як уже було сказано, дають додаткову інформацію про об'єкти, яка може нам знадобитися. Також ці методи необхідні, коли тіла не мають як такої межі (наприклад, туман, суспензія в рідини і т.д.)

3 D растр є набором кубиків (див. рис. 6.6) (voxel ' ов, від « volume element ») в просторі. Кожен воксел може мати деякий числове значення, що є атрибутом відповідної точки в просторі.

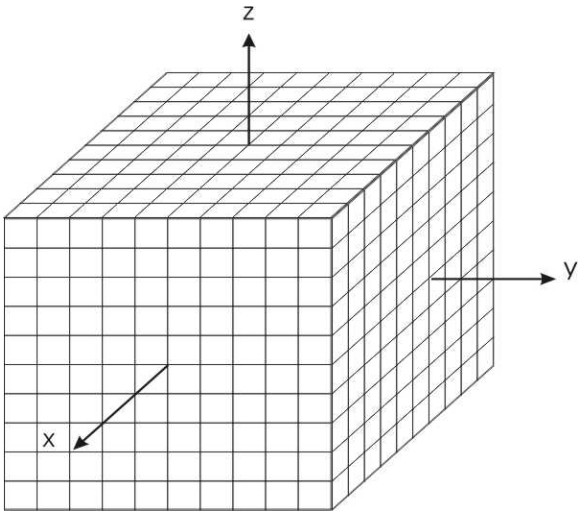


Рис. 6.6. 3 D растр.

Даний метод, незважаючи на свою простоту, має серйозний недолік: для зберігання навіть невеликого об'єкта в прийнятній якості потрібно дуже багато місця. Наприклад, якщо ми зберігаємо кубик з ребром 1024 і виділяємо по одному байту (8 біт) на атрибут воксель, то нам буде потрібно $1024 \times 1024 \times 1024 \times 1 байт = 1 Гб.$ Враховуючи той факт, що зберігати об'єкт нам, швидше за все, треба буде в оперативній пам'яті комп'ютера, застосування цього методу в чистому вигляді стає вкрай скрутним.

Вісімкове дерево

У більшості випадків в 3 D растре є деякі області вокселов, що мають однакові атрибути. Цим можна скористатися для того, щоб скоротити обсяг займаного об'єктом місця. Будемо зберігати растр у вигляді дерева (див. Рис.

7), вершини якого мають ступінь 0 або 8. Спочатку перевіримо, чи не лежать у всіх воксель в растрі однакові значення, якщо так, то нам досить буде зберегти це значення (*p)* і позначити, що далі дробити кубик не треба, наприклад, між іншим в вершину це значення *p* і 0 (позначення термінального вершини). Якщо у нас є воксели з різними значеннями, то продовжимо процес. А саме, розділимо растр на октанти, відповідні координатним площинам. Для кожного октанта знову перевіримо, чи не містить він однакові воксели. І т.д. Отримаємо дерево, в вершинах якого лежать 1 (що означає, що відповідний октант не однорідний) або 0 і деякі числа p_i, що відповідають значенням всіх вокселов в даному Октант.

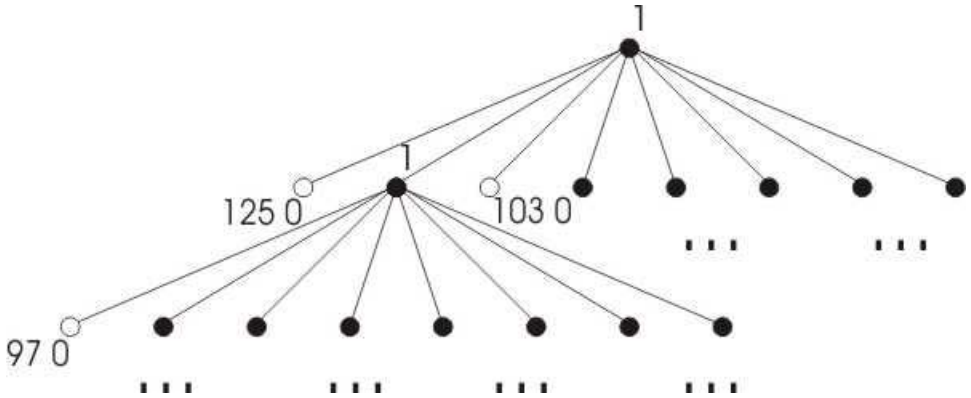


Рис. 6.7 Вісімкове дерево 3 D растра.

Таким чином, якщо растр складається з вокселов, що мають один і той же атрибут, то ми зекономимо $1\,Гб - (1байт + 1бит)$; якщо ж все вокселі різні, то зайве місце, яке ми витратимо (на 1 і 0), складе $(1+8+8^2+...+8^{10})\,битов = 1227133513\,битов \approx 146\,мб$. Але остання ситуація зустрічається вкрай рідко, тому використання вісімкових дерев в більшості випадків дозволяє значно скоротити обсяг пам'яті для зберігання об'єкта.

Двійкове дерево

Ідея побудови двійкового дерева повністю аналогічна побудові вісімкового дерева. Але в даному випадку растр послідовно поділяється не на октанти, а на половинки: спочатку паралельно площині OXY , потім OYZ і т.д. Відповідно, аналогічно будується дерево, вершини якого мають ступінь 0 або 2.

Синтез зображень по зображеннях (Image - based rendering , immersive imaging)

Ідея використання синтезу зображень із зображень полягає в тому, що ми генеруємо зображення тривимірного об'єкту на моніторі по набору існуючих зображень (наприклад, фотографій). При цьому нам не доводиться вирішувати, іноді вельми трудомістку, завдання створення моделей об'єктів.

Загальна ідея

Отже, ми хочемо генерувати зображення тривимірної сцени по набору зображень. На це завдання можна подивитися як на знаходження значень деякої функції f (тобто атрибутів пікселів) від 5 параметрів - x,y,z,φ і θ, де x,y,z - координати спостерігача, а φ та θ - кути, що задають напрямок променя зору (див. рис . 6.8).

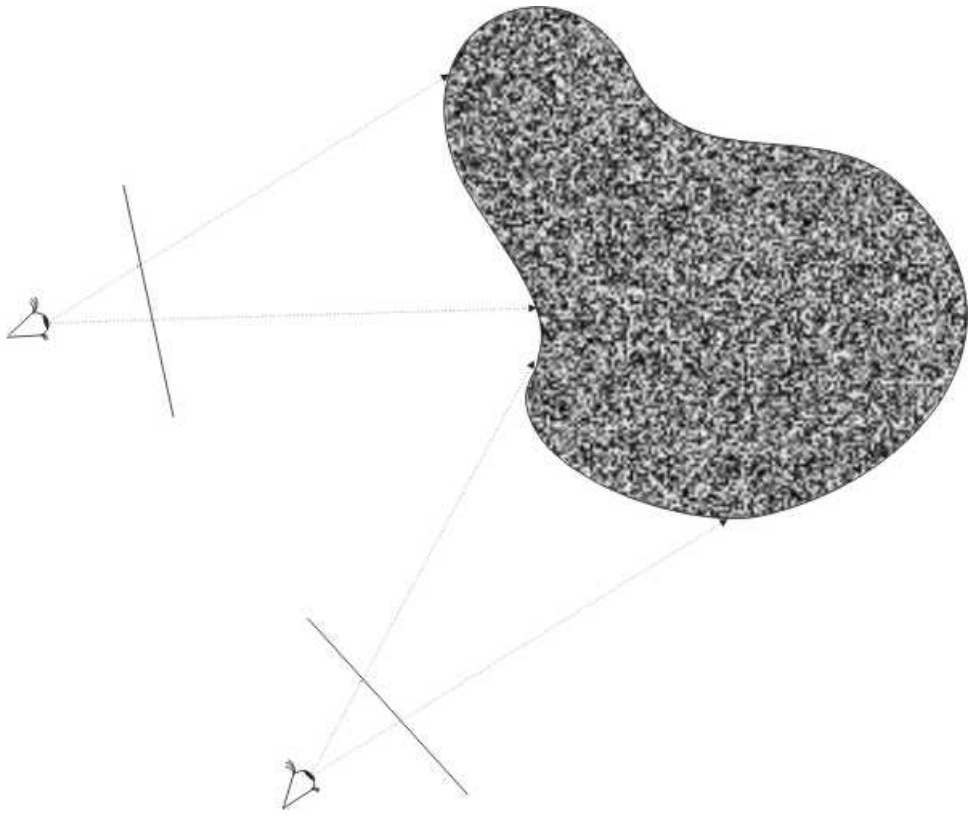


Рис. 6.8 . Визначення атрибутів пікселів в залежності від положення спостерігача.

Розмірність масиву даних, які нам будуть потрібні при такому підході, складе, таким чином, 5 D .

Lumigraph

У тому випадку, якщо ми знаємо, що спостерігач не може перебувати за деякою лінією, можна скоротити необхідний нам обсяг даних. А саме: сфотографувати об'єкт уздовж цієї лінії (на всіх рівнях по вертикалі), а потім в якості значень пікселя, відповідного деякому променю зору, будемо брати значення, відповідне найближчого променю камери (див. Рис. 6.9).

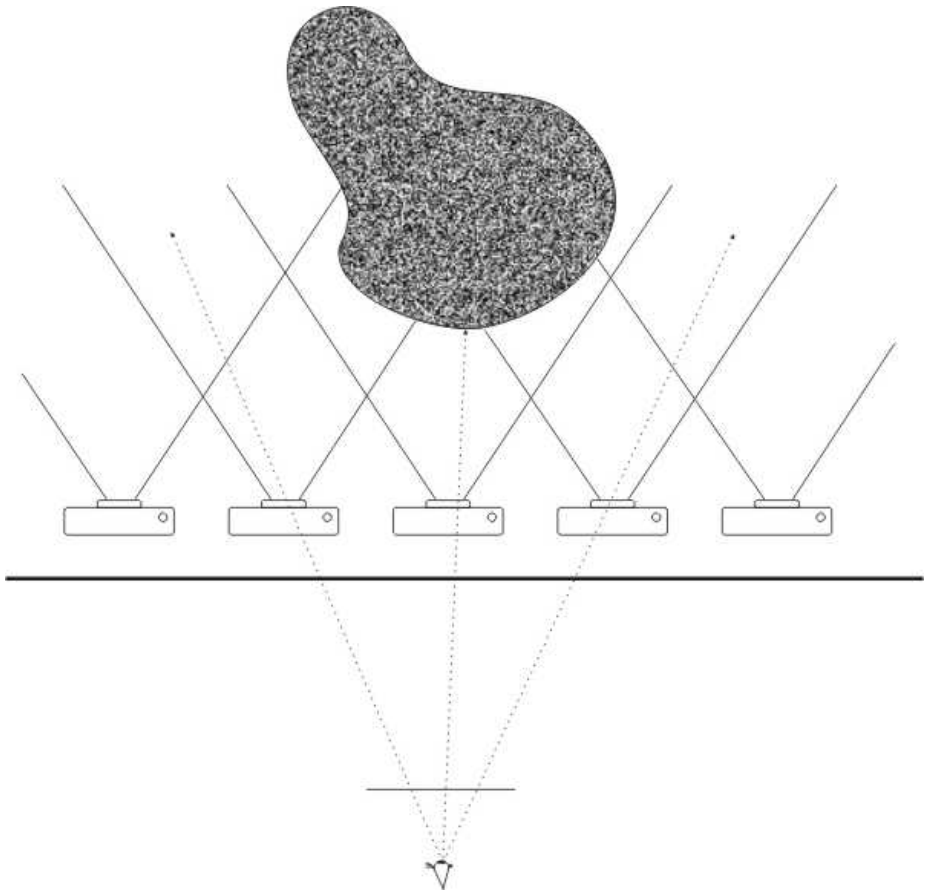


Рис.6.9**.** Lumigraph .

Розмірність масиву даних в цьому випадку становитиме 4 D (2 D на положення камери і 2 D на зображення від кожної камери).

Concentric mosaic

Даний метод полягає в наступному: нехай нам необхідно отримати посвідку деякої кімнати зсередини. Зробимо в центрі кімнати набір вертикальних лінійних (тобто шириною в 1 піксель) знімків у всіх напрямках. Потім зробимо аналогічні знімки з точок концентричних кіл по дотичним напрямками (див. Рис. 6.10) (як за годинниковою стрілкою, так і проти; на малюнку показано тільки один напрямок).

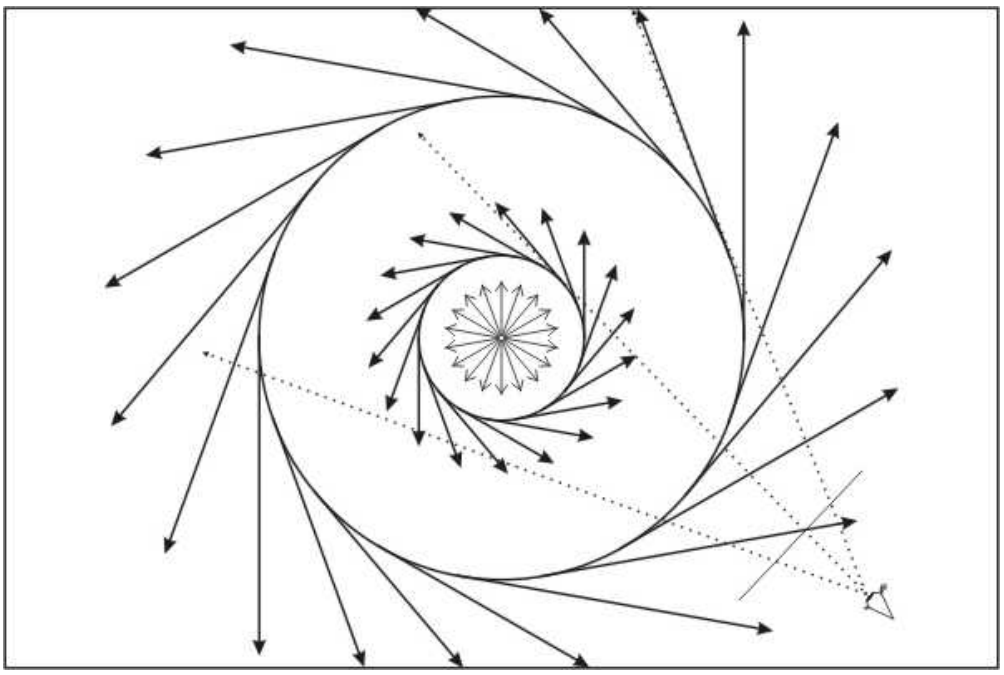
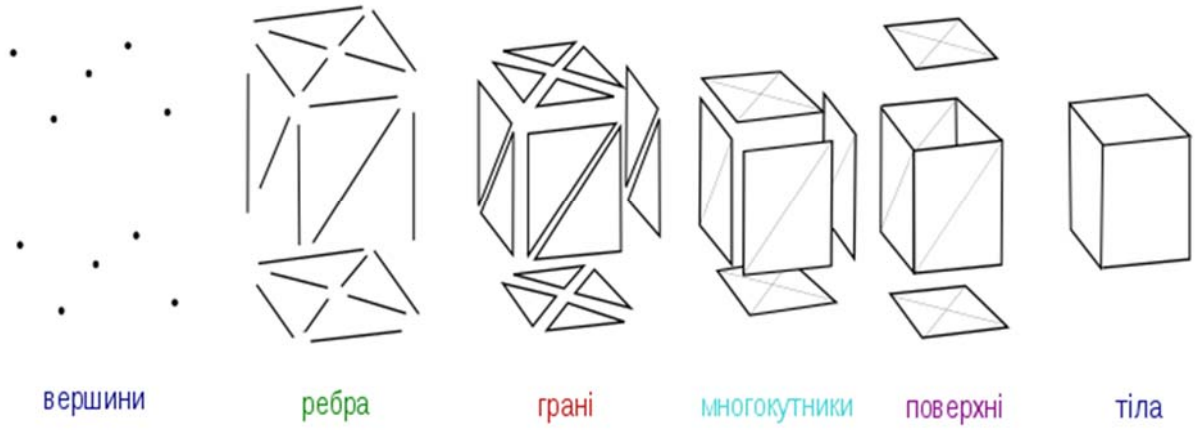


Рис.6.10. Concentric mosaic .

Як значення пікселя, відповідного деякому променю зору, будемо брати значення, відповідне найближчого дотичному променю. В цьому випадку розмірність масиву даних складе 3D (по 1 D на радіус кола, кут і саме зображення).

6.2. Основні полігонального моделювання

Будь-який полігональний об'єкт задається набором полігонів (інакше іменованих полігональними гранями) і тому об'єднує множину таких однотипних елементів, або підб'єктів, як вершини (Vertex, F9), ребра (Edge, F10) і грані (Face, F11)



Моделювання за допомогою вершин

Для вершин існує цікавий спосіб їх стесуванню (своєрідний аналог фаски) командою **Edit Polygons => Chamfer Vertex** (Правка полігонів => стесати вершини), що призводить до створення з однієї грані відразу декількох нових граней за рахунок того, що вершини як би зрізаються (рис . 6.11). Після цього до нових вершин можна застосувати будь-які перетворення, наприклад витягнути з них нові грані, застосувавши команду **Edit Polygons => Extrude Vertex** (Видавити вершину) - рис. 6.12.

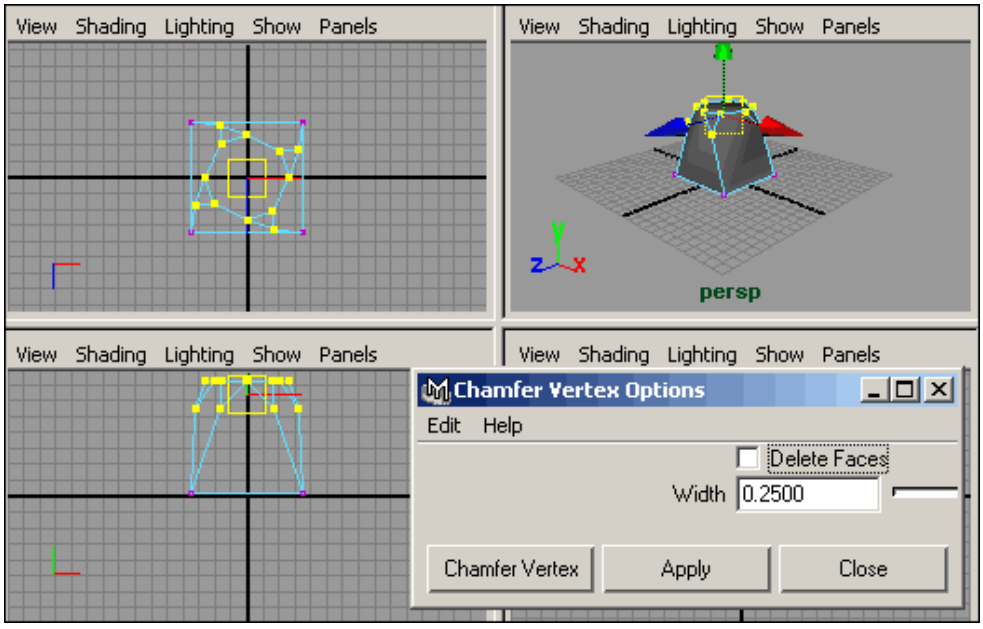


Рис.6.11. Моделювання за допомогою вершин

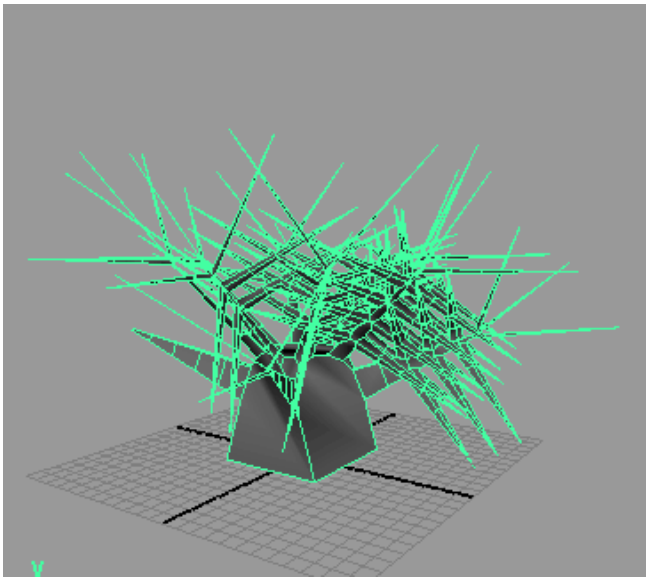


Рис.6.12. Моделювання за допомогою вершин

Моделювання за допомогою ребер

Можна додати до обраних ребер фаску, що може стати в нагоді, наприклад, при створенні огранованих моделей. Для прикладу візьміть звичайну кулю, натисніть клавішу **F10**, виділіть всі ребра і застосуєте команду **Edit Polygons => Bevel** (Правка полігонів => Фаска) - гладка куля стане граненою

Моделювання за допомогою полігонів

Можна додати до обраних граней пряму фаску, що здійснюється шляхом вставки площин замість загальних ребер виділених граней і необхідно при ручному згладжування форми моделі. Для додавання фаски виділіть весь об'єкт або його окремі грані, клацніть на квадратику праворуч від команди **Edit Polygons => Bevel** (Правка полігонів => Фаска) і в вікні, встановіть розмір фаски.

6.3. Робота з матеріалами

Під матеріалом розуміють набір характеристик, які в тій чи іншій мірі впливають на відображення поверхні об'єкту в процесі візуалізації сцени; і від того, наскільки вдало буде підібраний матеріал, залежить природність фінального виду модельованої сцени.

Для призначення матеріалу об'єкту необхідно виділити об'єкт, натиснути правою кнопкою миші на зразку матеріалу і вибрати в контекстному меню команду **Assign Material to Selection** (Назначити матеріал виділеному об’єкту). При бажанні також можна конвертувати матеріал в текстуру, застосувавши команду **Convert to File Texture** (Конвертувати в файл текстури) з меню **Edit**.

Принципи створення матеріалів

- шляхом зміни параметрів тонованого розфарбовування одного з існуючих базових зразків;

- шляхом автоматичного перетворення однієї з вхідних в поставку текстурних карт в матеріал;
- автоматично згенерувати матеріал на базі текстурних карт, серед яких є зразки для імітації різних поверхонь.

Параметри тонування розфарбовування:

- **Color** (Колір) - задає основний колірний фон матеріалу об'єкта;
- **Transparency** (Прозорість) - регулює ступінь прозорості матеріалу;
- **Ambient Color** (колір підсвітки) - визначає колір ділянок поверхні об'єкта, які не освітлені прямими променями світла, тобто колір тіні на поверхні об'єкта;
- **Incandescence** (самосвітіння) - задає особливості самосвітіння об’єкта;
- **Bump Mapping** (Карта рельєфу) - дозволяє створювати рельєфні поверхні;
- **Diffuse** (Дифузний колір) - визначає інтенсивність розсіюваних матеріалом світлових променів при його освітленні прямими променями світла;
- **Translucence** (просвічування) - дозволяє імітувати відтінок світла, що просвічує крізь матеріал, використовується при створенні напівпрозорих тканин (наприклад, матового скла);
- **Translucence Focus** (фокусування просвічування) - визначає спосіб віддзеркалення світла від поверхні: при низьких значеннях даного параметра імітується ефект м'якого просвічування, а при високих - інтенсивнішого;
- **Eccentricity** (Ексцентриситет) - вказує розмір бликової плями: як правило, для матових поверхонь встановлюється більший розмір відблиску, а у блискучих - менший;
- **Specular Roll Off** (Сила блиску) - використовується тільки при наявності на поверхні відблиску і визначає його інтенсивність;
- **Specular Color** (дзеркальний колір) - встановлює кольоровий тон світлових відблисків, що з'являються на поверхні об'єкта;

• **Reflectivity** (відбивна здатність) - визначає яскравість відображення навколишніх об'єктів дзеркальною поверхнею;

• **Reflected Color** (Колір відбитого світла) - встановлює колірний тон відбитого світла, визначається найчастіше через карту текстур

6.4. Основи NURBS моделювання

Будь-яка NURBS-модель являє собою деякий набір NURBS-поверхонь, утворених NURBS-кривими. Останні, в свою чергу, являються неоднорідними раціональними сплайнами Безьє (Non-Uniform Rational Bezier Splines, NURBS). Дані криві описуються математичними формулами - в результаті відпадає необхідність запам’ятовувати кожну точку кривої, досить знати координати її початку і кінця і математичну формулу, яка описує криву. Це дозволяє створювати складні криволінійні поверхні з невеликою кількістю керуючих вершин і легко позбавлятися від грубої огранки об’єктів, надаючи їм плавну викривлену форму шляхом простого збільшення деталізації.

NURBS-примітиви зазвичай використовуються в якості основи для формування більш складних моделей і створюються командою **Create=>NURBS Primitives** або вибором потрібного примітиву на вкладці **Surfaces** (Поверхні) панелі **Shelf**.

При редагуванні примітивів на рівні підоб’єктів деформація NURBS-поверхонь заснована на інтерполяції кривих, а деформація полігональних моделей в першу чергу пов’язана зі зміною орієнтації граней.

Побудова кривих за допомогою визначення їх вершин або редагованих точок:

- **CV Curve Tool** (Крива по керуючим вершинам) - встановлює керуючі вершини кривої точно в тих точках, де клацнуто мишкою, точки редагування створюються автоматично на основі керуючих вершин;

- **EP Curve Tool** (Крива по точкам редагування) - формує точки редагування кривої точно там, де клацнуто мишкою; керуючі вершини створюються автоматично на базі точок редагування;
- **Pencil Curve Tool** (Олівцева крива) - дозволяє малювати криву, перетягуючи мишку, даний тип кривої відрізняється створенням дуже великої кількості точок редагування і керуючих вершин, що може призвести до складнощів при редагуванні.

Команда **Edit Curves=>Rebuild Curve** (Редагувати криві=>Перебудувати криву) допомагає редагувати (н-р, видаляти точки) криву. Дана команда може бути застосована до будь-яких типів NURBS-кривих.

- **Метод обертання** - моделювання NURBS-поверхонь шляхом обертання NURBS-кривих навколо опорної точки, що призводить до отримання тіл обертання, тобто моделей, що мають центральну осьову симетрію (**Surfaces=>Revolve** (Поверхні=>Обертання)
- **Метод отримання плоских поверхонь** полягає в вирізанні області площини, заданої деякою NURBS-кривою. NURBS-криві повинні лежати на площині (тому створювати їх краще в проекціях Top або Persrective) і бути замкнутими (**Surfaces=>Planar** (Поверхні=>Планарний).
- **Метод видавлення з скосом** дозволяє формувати NURBS-поверхні з різноманітними фасками і базується на застосуванні одного з двох інструментів: **Bevel** (Скіс) або **Bevel Plus** (Покращений скіс). Перший дозволяє створювати поверхні на основі як відкритих, так і замкнутих кривих, а другий — тільки замкнутих (тому частіше потребується попереднє закриття контуру кривої за допомогою команди Open/Close Curves). Розмір скосу і глибина видавлювання, а також вид фаски (тільки при використанні інструмента Bevel Plus) регулюється. Сформовані операцією Bevel поверхні виходять відкритими з торців, але ці отвори нескладно закрити спеціально вирізаними

плоскими поверхнями, створеними за допомогою інструмента Planar (Планарний). Поверхні, отримані за допомогою команди Bevel Plus, з торців закриті, тому для них подібних перетворень не потрібно.

Дозволяє будувати поверхні на базі двох кривих: кривої-профіля і кривої-шляху. Крива-профіль являє собою перетин необхідної поверхні, який потім видавлюється вздовж кривої-шляху. Тому даним способом зручно створювати такі об'єкти, як звивисті тунелі, шланги для поливу, пружини, будівельні профілі і т.п.

В якості кривої-профіля може виступати як одна крива, так і декілька об'єднаних в групу кривих, в якості кривої-шляху - тільки одна крива. Крива-профіль може бути відкритою або закритою, незалежною або ізопараметричною. Якість отриманої в результаті поверхні напряму залежить від кількості керуючих точок, що знаходяться на кривих, - при їх нестачі можливе небажане скручення поверхні. Для побудови поверхонь даним методом необхідно виділити криву-профіль, після, утримуючи натиснутою клавішу Shift - криву-шлях і застосувати команду **Surfaces**=>**Extrude** (Поверхні=>Видавити).

6.5. Завдання до самостійної роботи по темі «Полігональне моделювання»

Створіть наступні об'єкти засобами полігонально моделювання

1. Яйце з куба

Створіть витягнутий у висоту полігональний куб (рис. 1.1). Виділіть його та тричі проведіть операцію згладжування, скориставшись командою **Polygons => Smooth** (Полігони => Згладити). Це призведе до перетворення кубічної поверхні в еліптичну (рис. 1.2), проте для того, щоб вона стала нагадувати за формою яйце, потрібно додатково стиснути верхні перерізи (сечения, рос.) і,

навпаки, розширити нижні, що досягається послідовним масштабуванням. Перейдіть в режим редагування вершин, натиснувши клавішу **F9**, виділіть в проекції **Side** всі вершини перерізів, розташованих у верхній половині об'єкта, і інструментом **Scale Tool** злегка зменшите їх радіус (рис. 1.3). Потім скоротіть кількість виділених перерізів на одне знизу і знову проведіть ту ж саму операцію масштабування і т.д. Потім виділіть всі вершини перерізів з нижньої третини об'єкта і трохи збільште їх радіус (рис. 1.4). Ту ж саму операцію багаторазово повторіть, зменшуючи кожен раз розмір виділеної області на один розташований вище переріз і намагаючись надати об'єкту форму яйця (рис. 1.5). Після закінчення натисніть клавішу F8 і знову застосуйте до об'єкту згладжування. Ви отримаєте приблизно таке яйце, як показано на рис. 1.6.

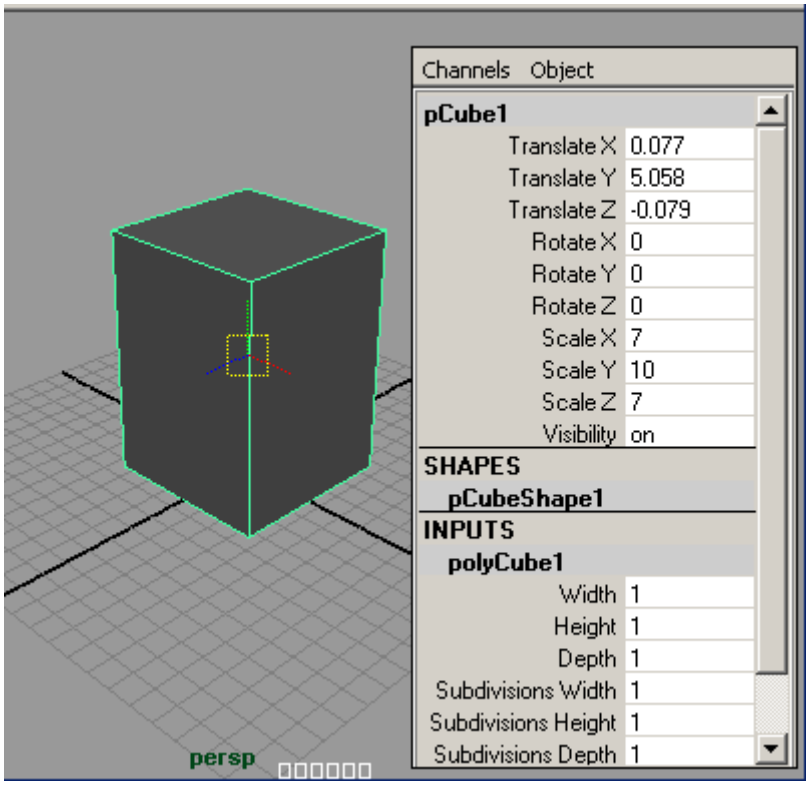


Рис. 1.1. Початковий куб

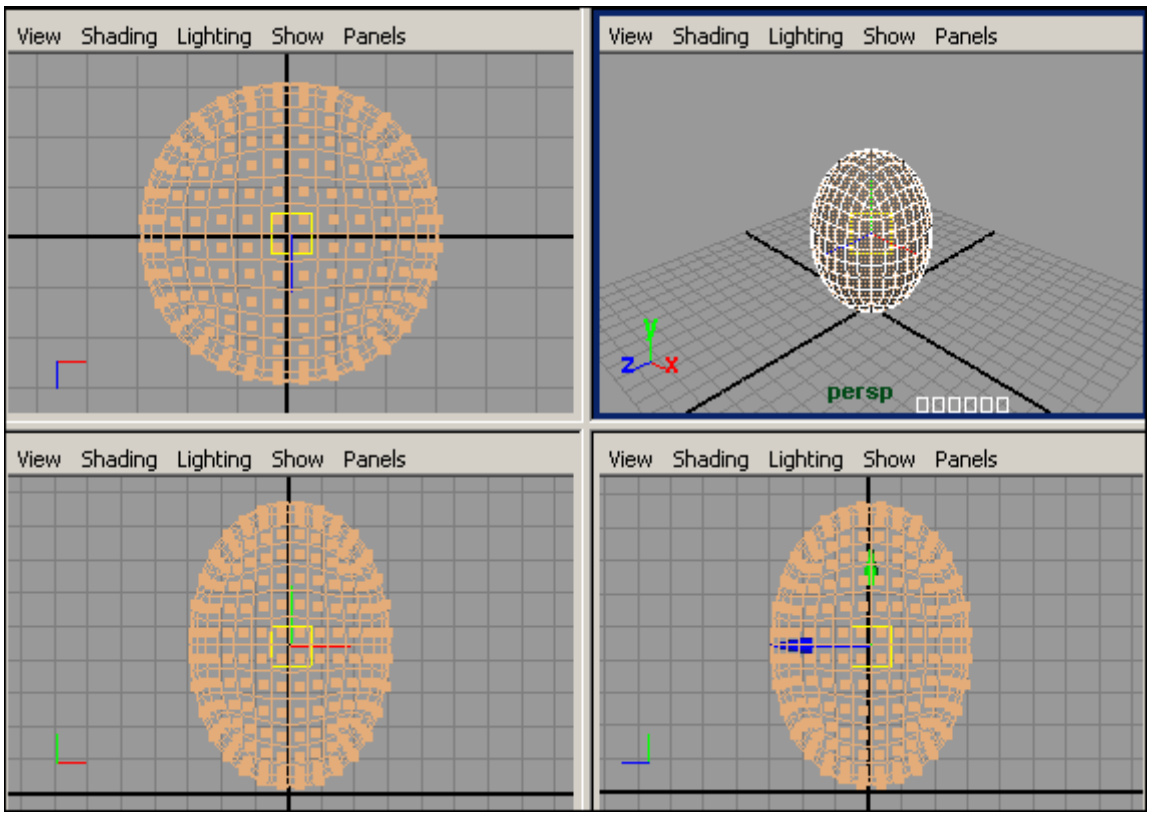


Рис. 1.2. Результат триразового згладжування

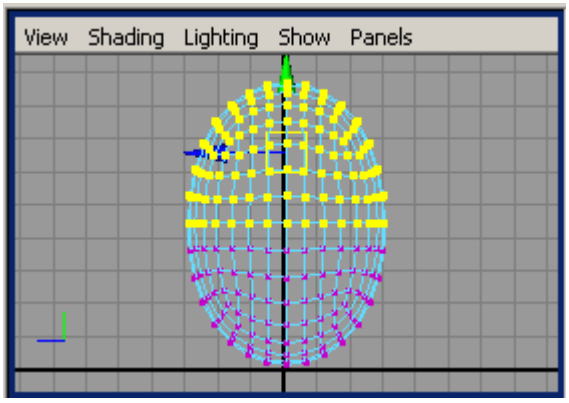


Рис. 1.3. Стиснення верхніх перерізів

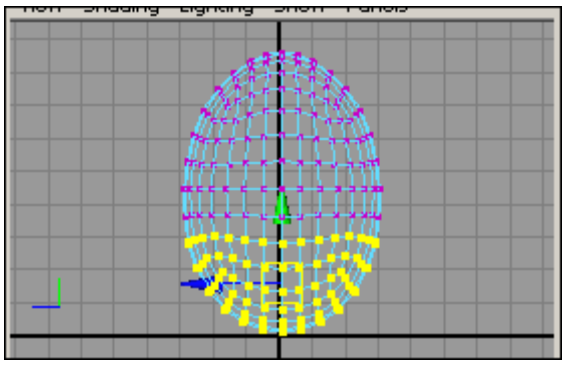


Рис. 1.4. Розширення нижніх перерізів

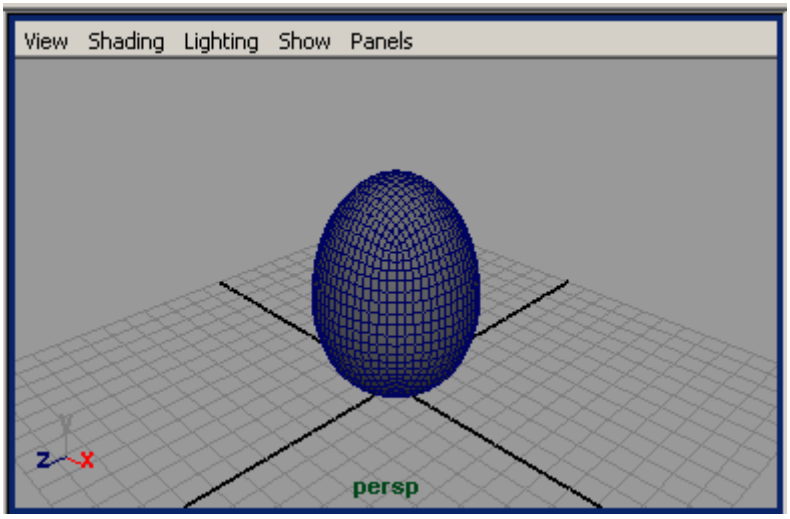


Рис. 1.5. Результат останнього масштабування

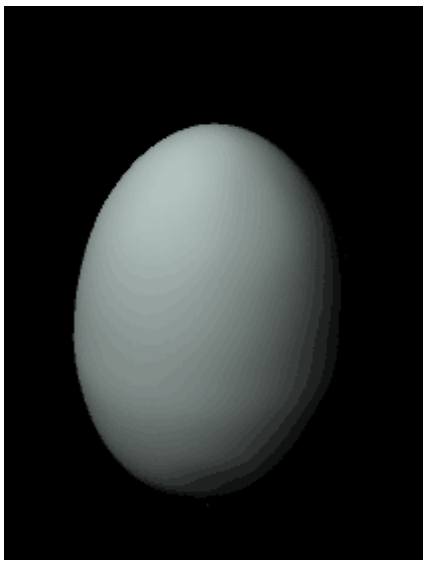


Рис. 1.6. Яйце

2. Морський їжак зі сфери

Створіть полігональну сферу (рис. 2.1), натисніть клавішу F9 для переходу в режим редагування вершин і виділіть всі вершини (рис. 2.2). Застосуйте до вершин операцію **Extrude** (Вдавлювання), клацнувши на квадратику праворуч від команди **Edit Poligons => Extrude Vertex** (Редагування полігонів => Видавлювання вершин), і налаштуйте параметри видавлювання приблизно так само, як показано на рис. 2.3. Не знімаючи виділення, накладіть на підоб'єкти фаску за допомогою команди **Edit Poligons => Bevel** (Редагування полігонів => Фаска) з параметрами за замовчуванням. Отримана в результаті модель нагадує морського їжака і представлена на рис. 2.4.

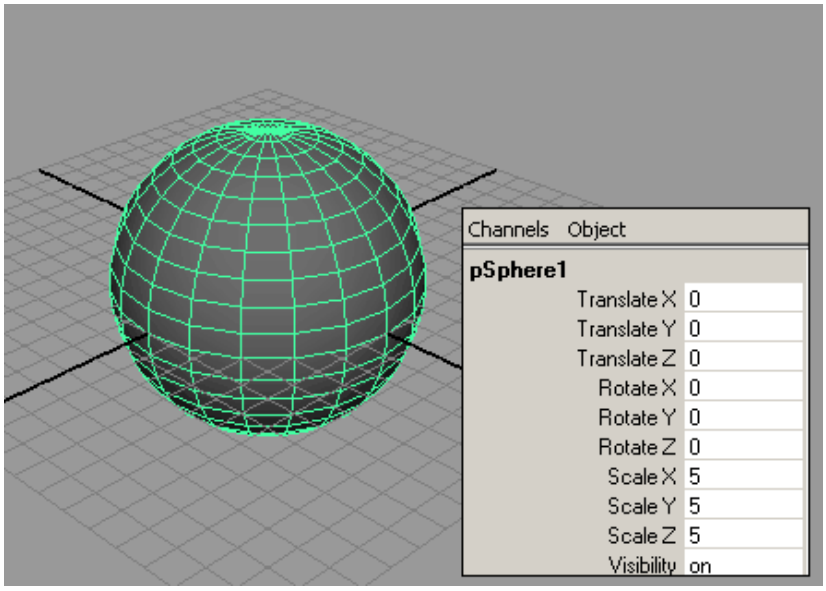


Рис. 2.1. Початкова сфера

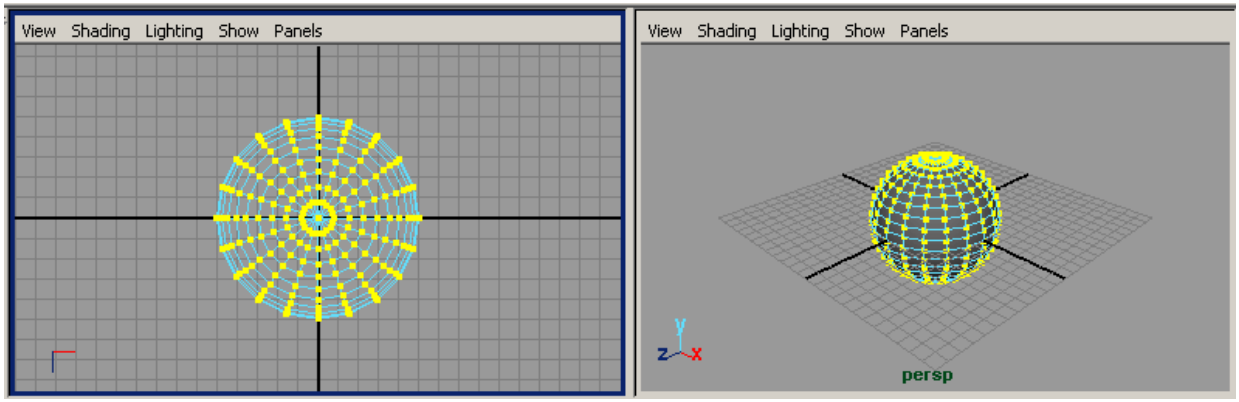


Рис. 2.2. Виділення вершин сфери

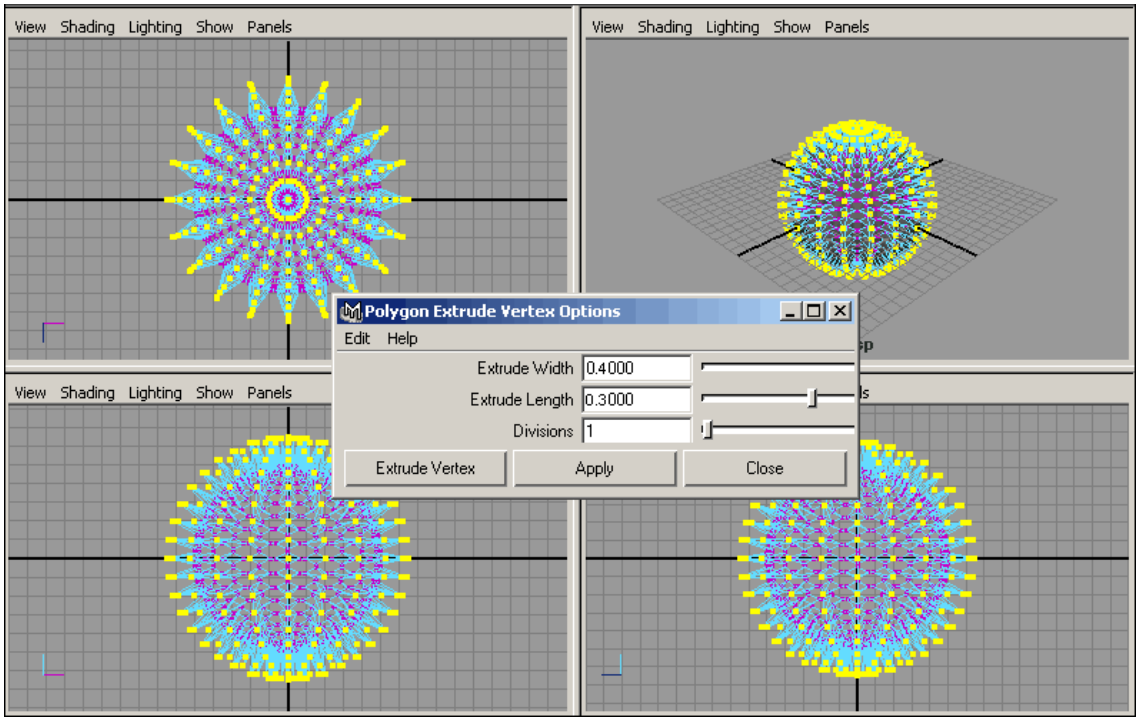


Рис. 2.3. Застосування операції ***Extrude***

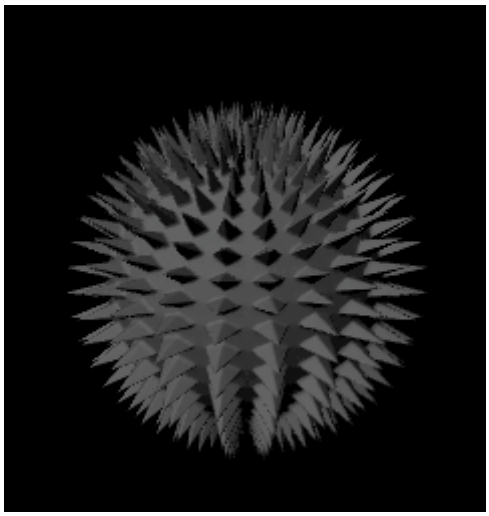


Рис. 2.4. Морський їжак

3. Кубик Рубіка з куба

Спробуємо змоделювати кубик Рубіка не з набору окремих, а на основі одного куба (рис. 3.1). Зверніть увагу на число сегментів по глибині, висоті і ширині, яке в точності відповідає запланованому числу кубиків на кожній зі

сторін. Перейдіть в режим редагування граней, виділіть всі грані на одній зі сторін куба і застосуєте до них операцію **Extrude Face** (Видавлювання граней) при таких параметрах, як показано на рис. 3.2. Зверніть увагу, що для необхідного в даному випадку варіанту видавлювання прапорець **Keep Faces Together** з меню **Polygons => Tool Options** (Полігони => Налаштування опцій) повинен бути відключений. Потім послідовно повторіть ту ж саму операцію для всіх інших граней, розташованих на зовнішніх сторонах куба. В результаті куб виявиться розбитим на окремі кубічні фрагменти, правда основні ребра куба будуть виражені нечітко, що трохи псує його зовнішній вигляд (рис. 3.3). Тому постараємося виділити дані ребра шляхом зняття глибокої, але дуже вузької фаски. Утримуючи **Shift**, виділіть ребра (для цього доведеться кілька разів змінити огляд перегляду) і застосуйте до них операцію **Bevel** (Фаска) з меню Edit Polygons. Параметри настройки фаски представлені на рис. 3.4, а отриманий кубик Рубіка - на рис. 3.5.

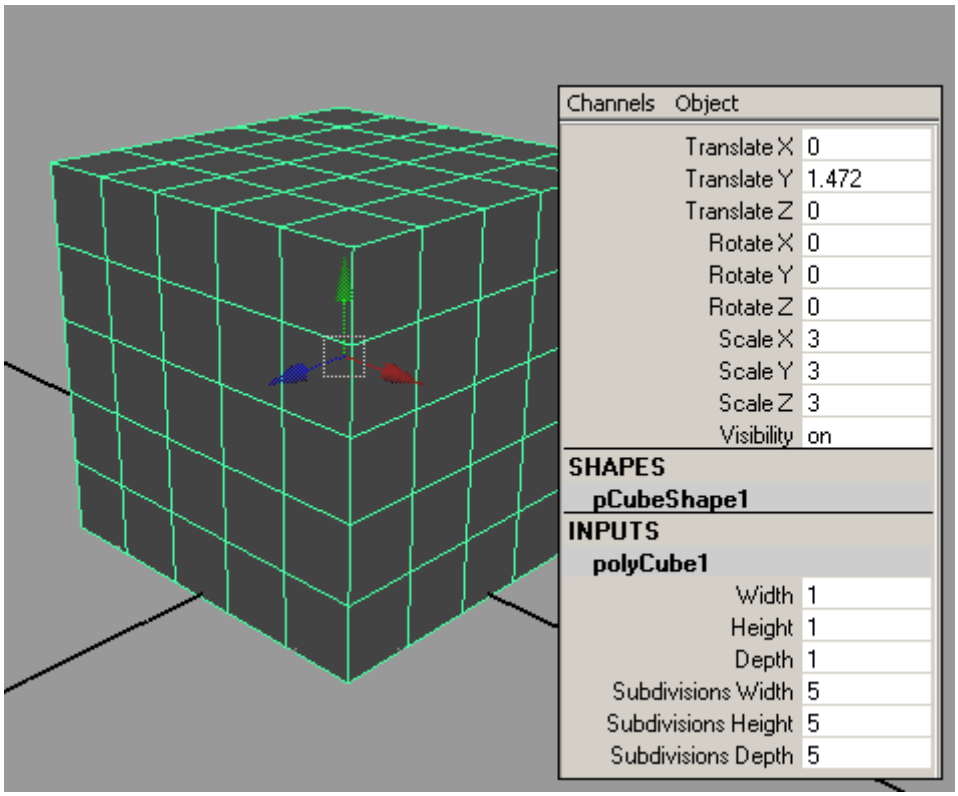


Рис. 3.1. Початковий куб

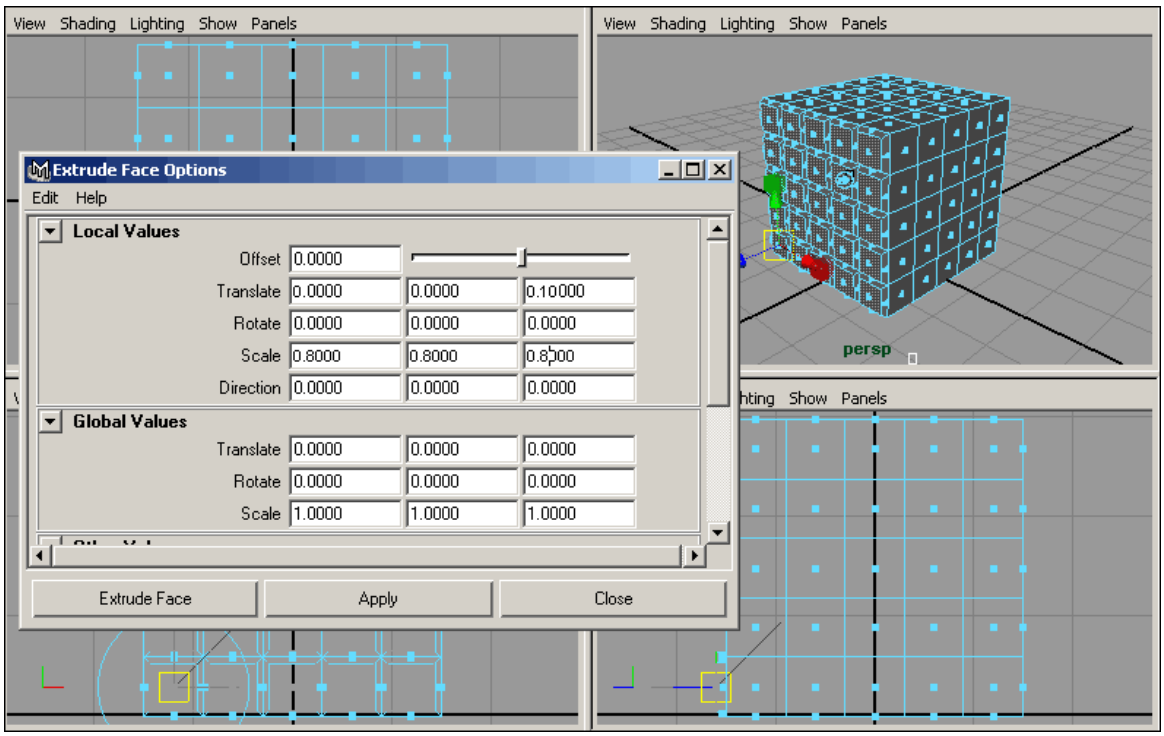


Рис. 3.2. Настройка витягування граней для одной з сторін куба

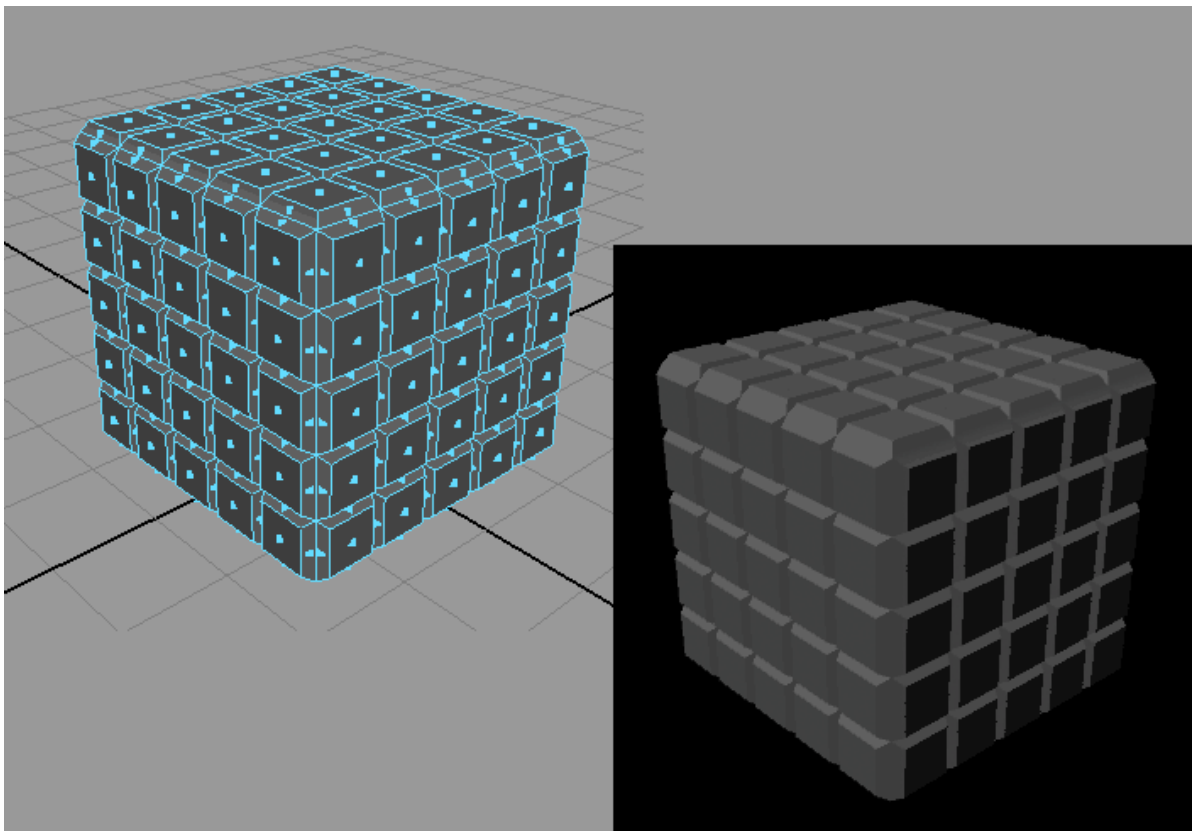


Рис. 3.3. Результат витягування граней

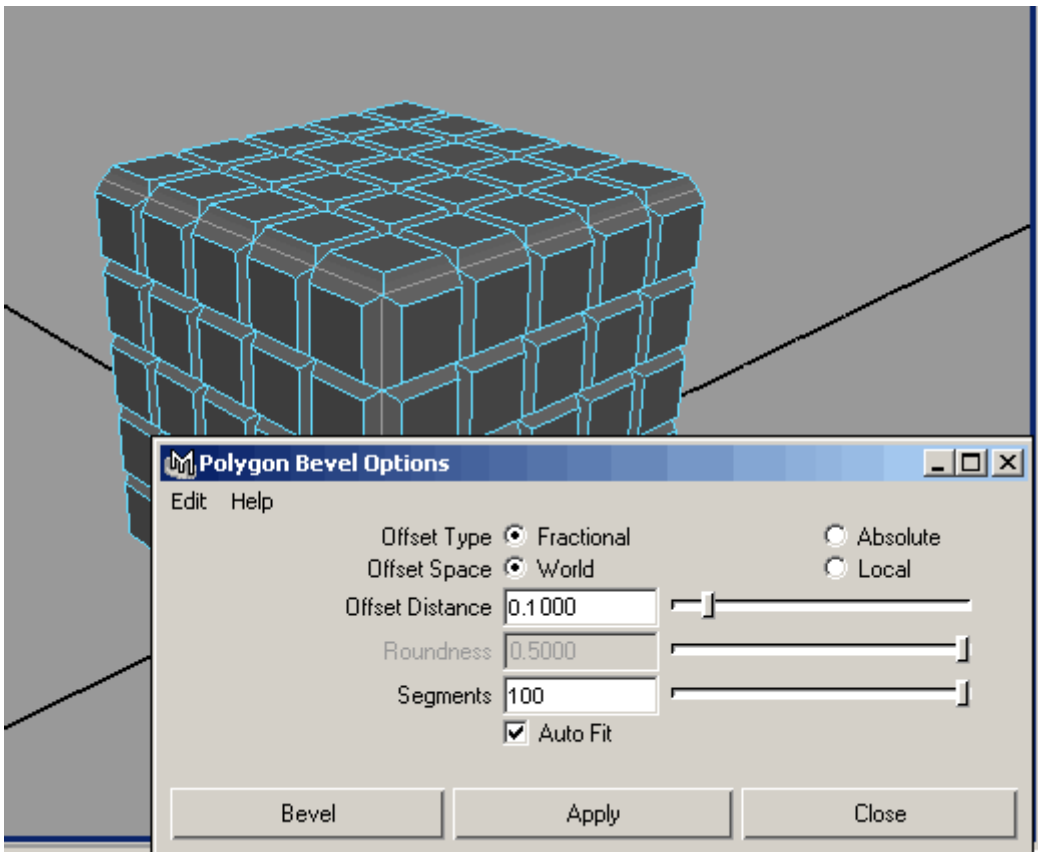


Рис. 3.4. Застосування операції ***Bevel***

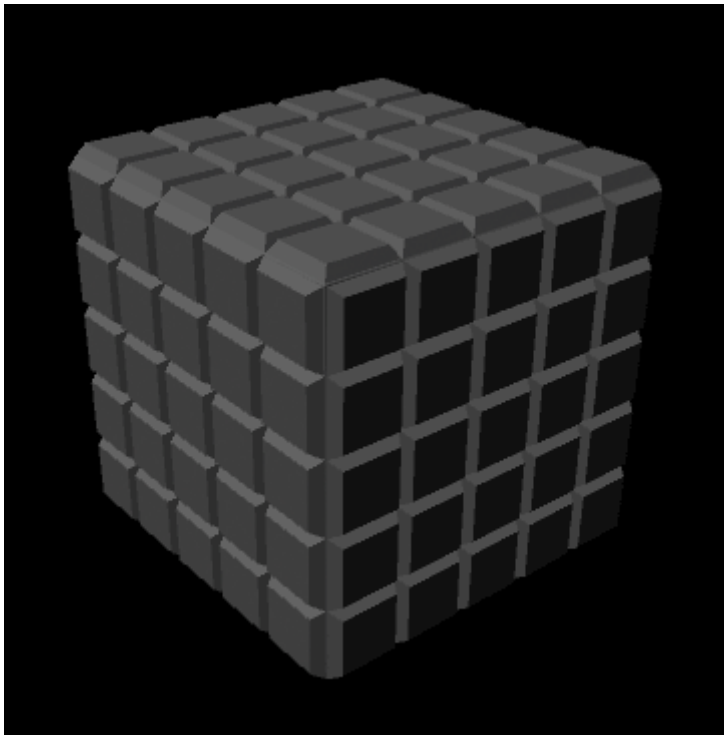


Рис. 3.5. Кубик Рубіка

4. Плитка шоколаду з куба

Спробуємо створити плитку шоколаду, взявши за основу звичайний полігональний куб з певним чином скоригованим у вікні каналів (**Channel Box**) числом сегментів по глибині і ширині (рис. 4.1), що визначає число плиток на кожній зі сторін. Виділіть об'єкт і перейдіть в режим редагування граней, натиснувши клавішу F11. Перейдіть до роботи з чотирма вікнами проекцій і в одній з проекцій - **Side** (Збоку) або **Front** (Спереду) - виділіть верхні грані (рис. 4.2). Застосуйте до них операцію **Extrude** (команда **Edit Poligons => Extrude Face** - (Редагування полігонів => Видавлювання граней) при параметрах за замовчуванням, а потім у вікні каналів збільште параметр **Local Translate Z** до 0,1 (рис. 4.3). Дана дія приведе до невеликого витягуванню виділених граней по осі Z. Після цього, не знімаючи виділення, застосуєте до граней операцію **Bevel** (команда **Edit Poligons => Bevel** - Редагування полігонів => Фаска) з параметрами як на рис. 4.4. В результаті об'єкт виявиться розбитим на окремі фрагменти і дійсно буде нагадувати плитку шоколаду (рис. 4.5).

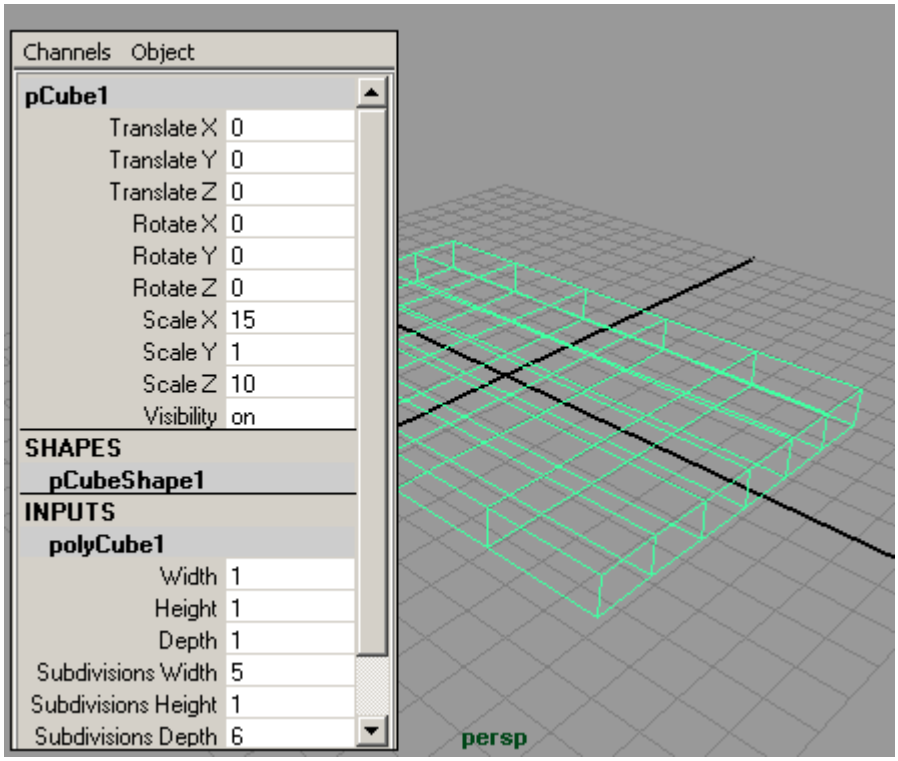


Рис. 4.1. Початковий куб

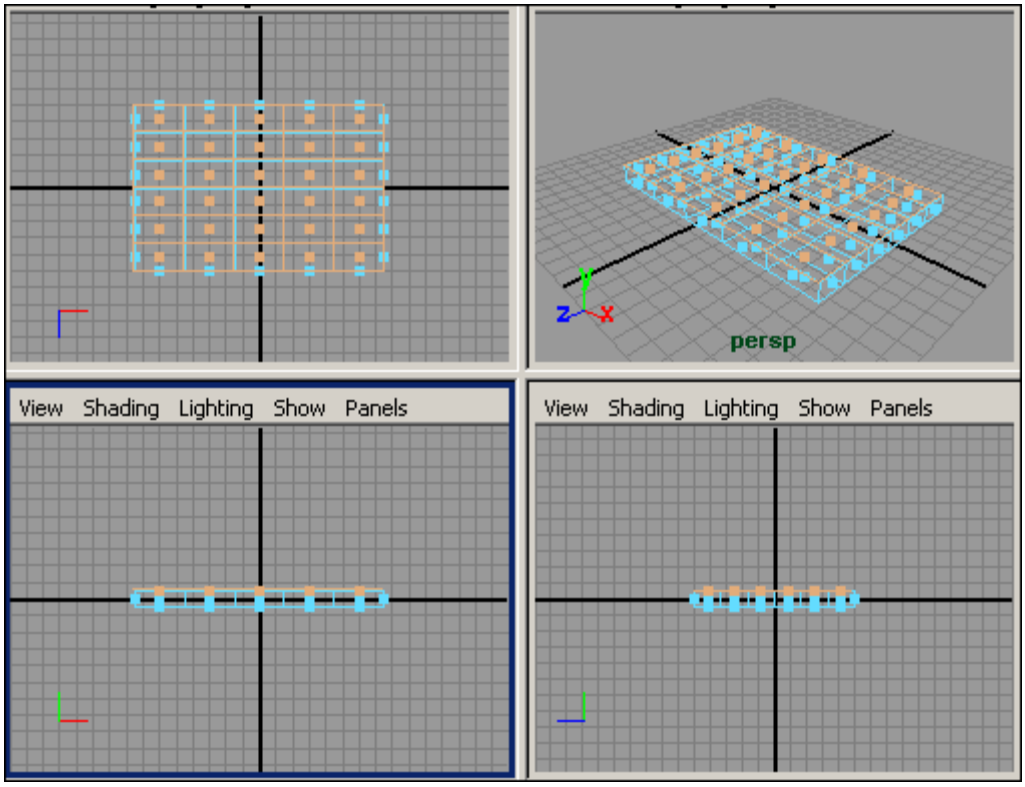


Рис. 4.2. Виділення граней

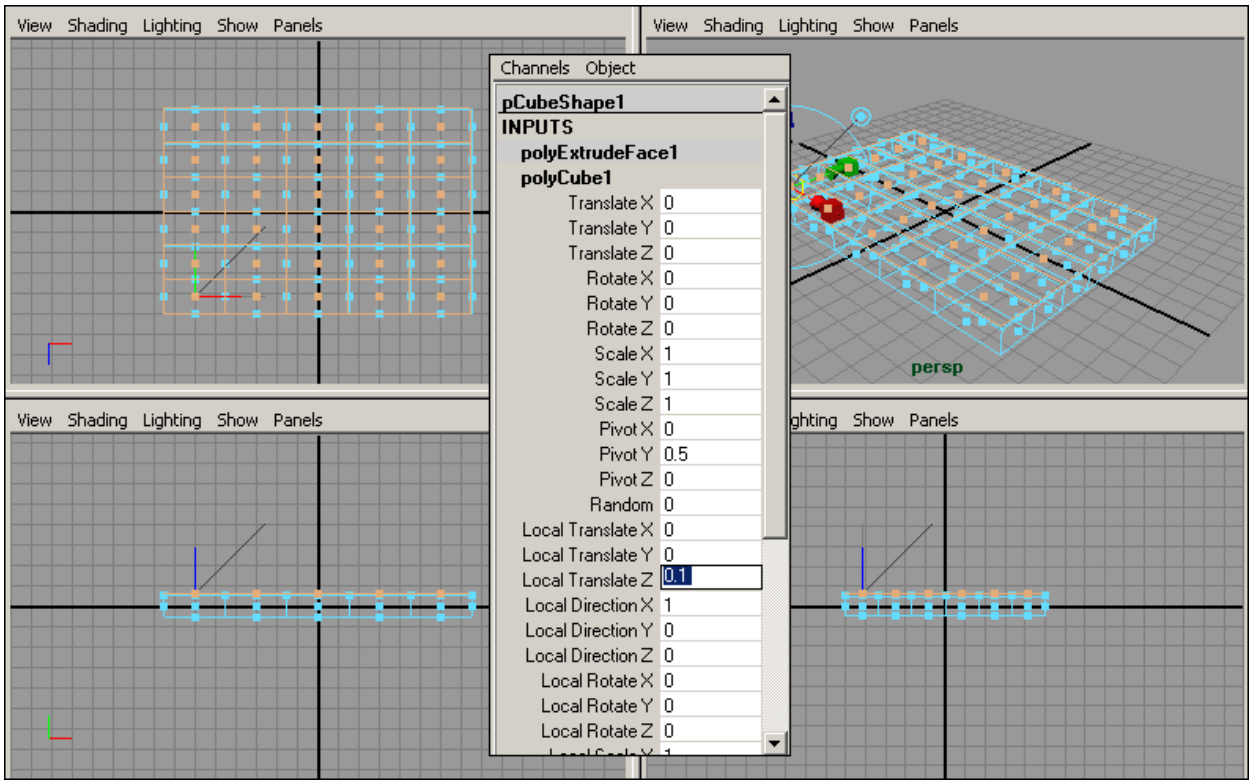


Рис. 4.3. Корекція параметрів операції ***Extrude Face***

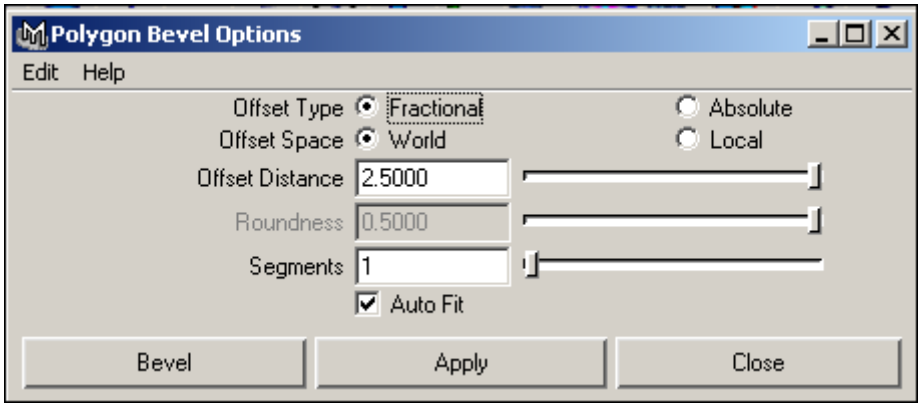


Рис. 4.4. Настройка параметрів операції ***Bevel***

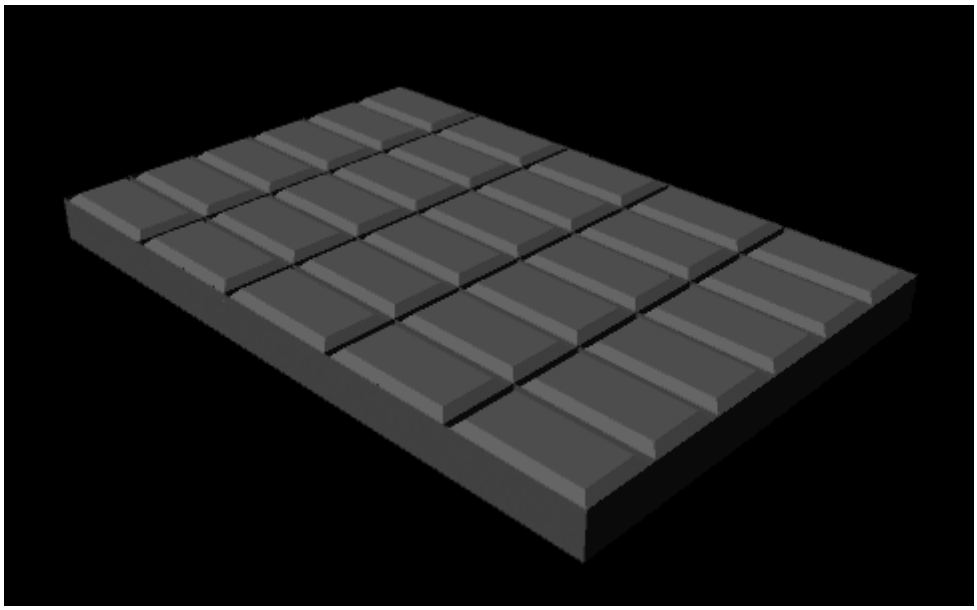


Рис. 4.5. Плитка шоколаду

5. Шестерня з торуса

Створіть торус з 36 сторонами і 7 сегментами висоти (рис. 5.1). Перейдіть в режим редагування граней (клавіша F11) і в проекції **Top** виділіть половину граней на зовнішній стороні торуса, вибираючи їх через одну (рис. 5.2). Застосуйте до них команду **Edit Poligons => Extrude Face** (Редагування полігонів => Видавлювання граней), встановивши параметри операції відповідно до рис. 5.3. Об'єкт стане нагадувати представлений на рис. 5.4. У

проекції **Side** виділіть зазначені на рис. 5.5 верхні межі торуса і шляхом масштабування трохи перемістіть їх по осі Y вниз. Аналогічним чином виділіть симетричні тільки що обробленим нижні поверхні торуса і перемістити їх трохи вгору - об'єкт стане більш плоским (рис. 5.6). У проекції **Top** виділіть грані внутрішньої поверхні (рис. 5.7) і шляхом їх масштабування добийтеся, щоб внутрішня поверхня об'єкта стала рівною. Верніться в режим редагування об'єкта, натиснувши клавішу F8, і для більшої природності застосуйте до всього об'єкту операцію **Bevel**, встановивши значення параметра **Offset Distance** рівним 0,05. Отримана в результаті шестерня представлена на рис. 5.8.

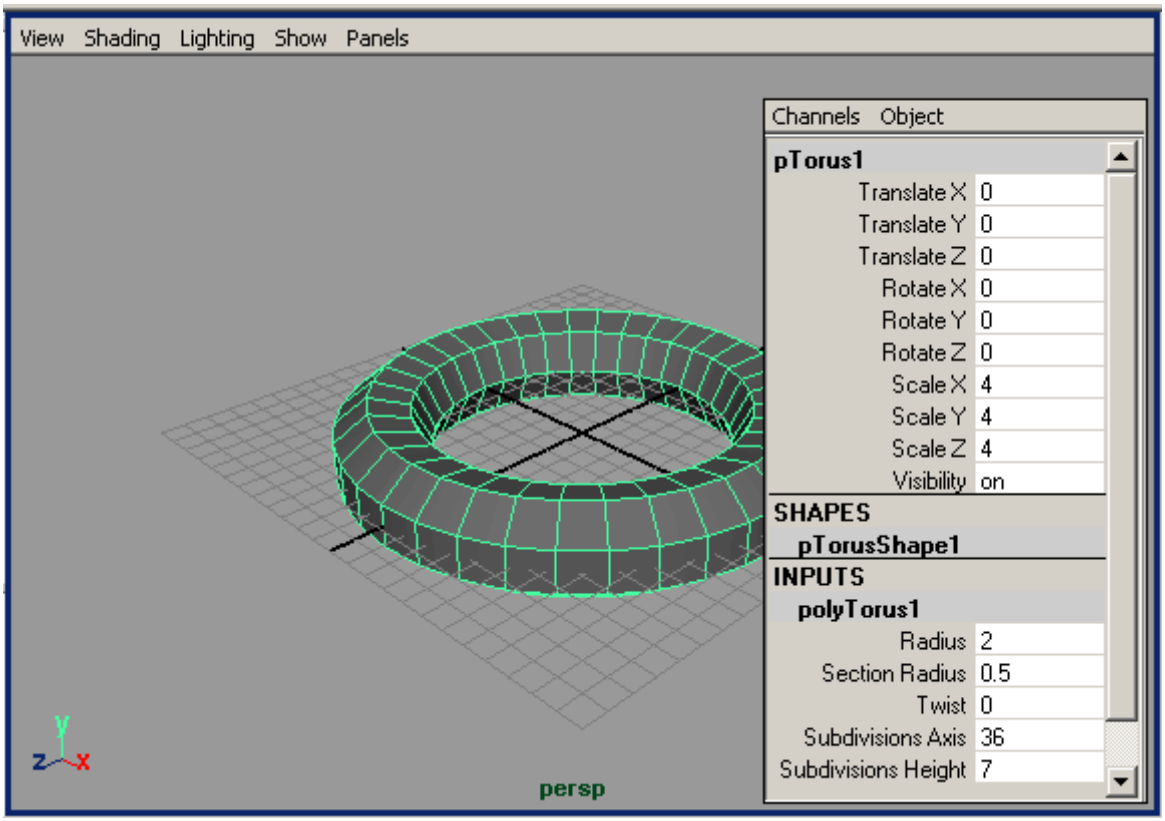


Рис. 5.1. Початковий торус

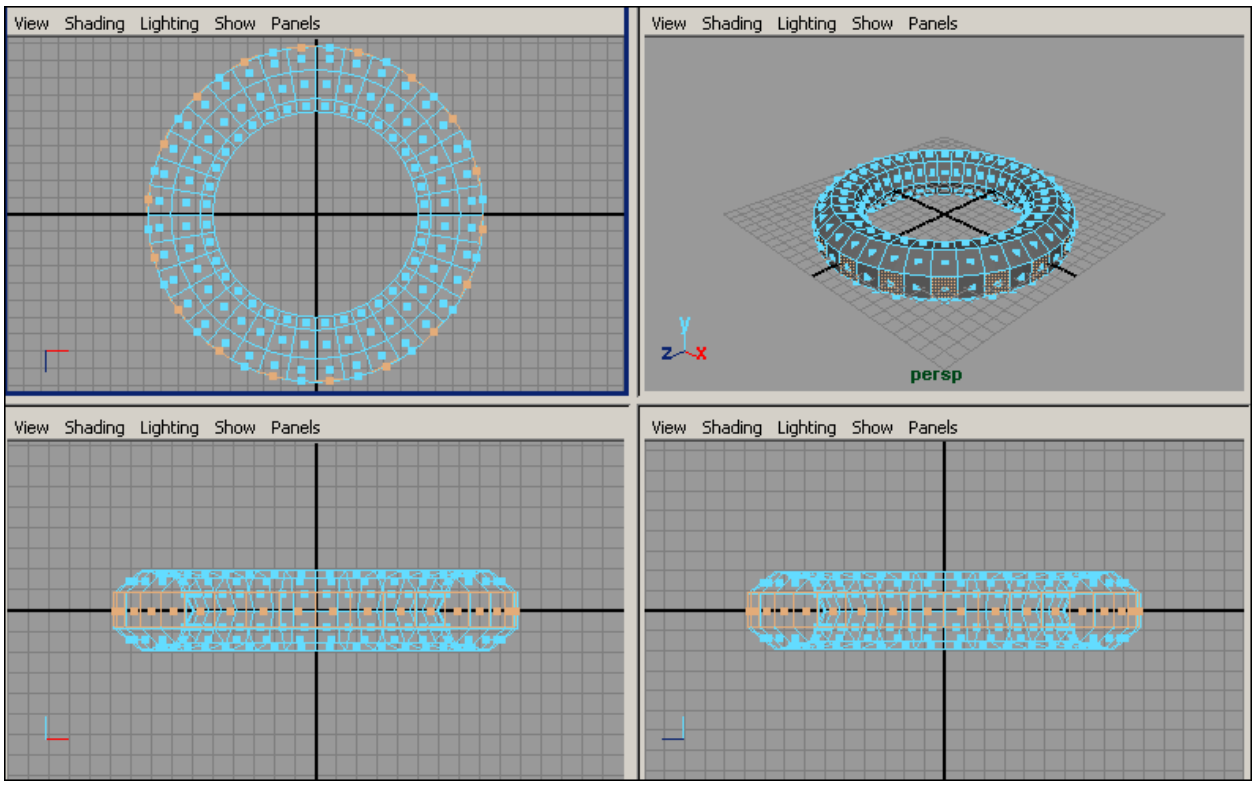


Рис. 5.2. Виділення граней на зовнішній поверхні тора

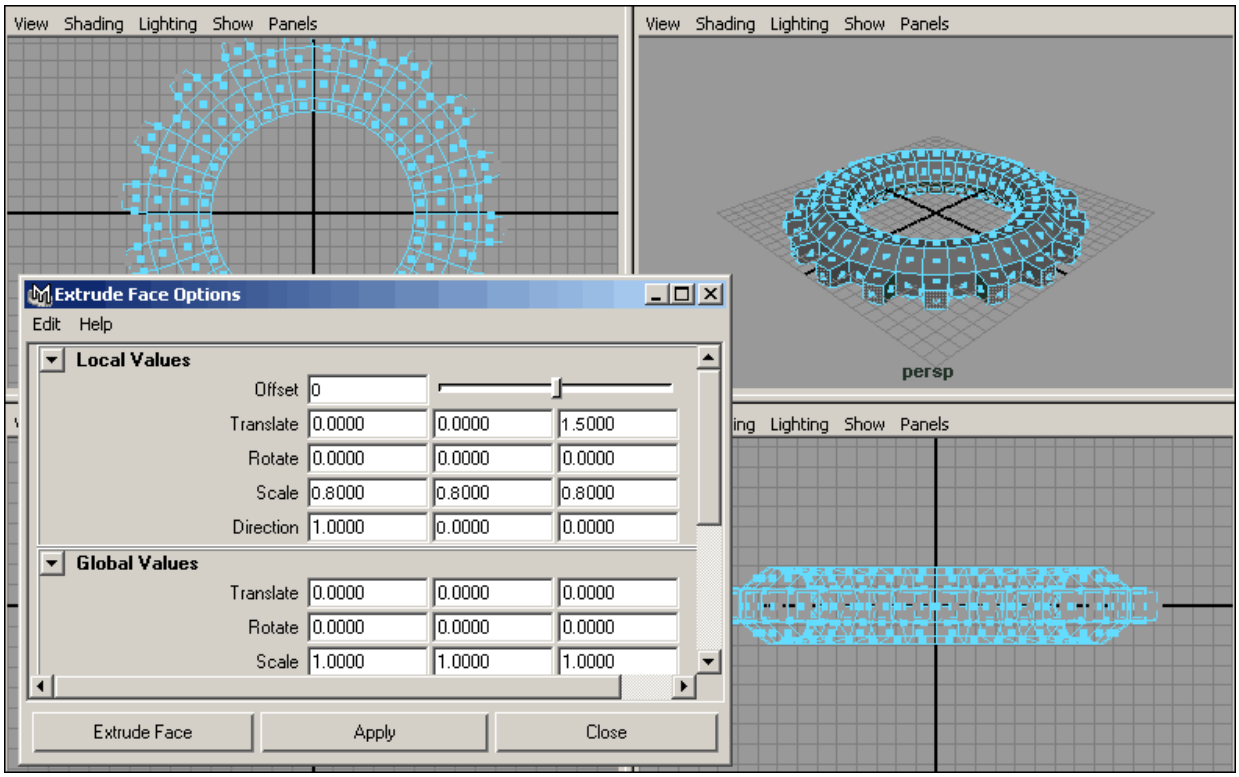


Рис. 5.3. Настройка операції ***Extrude Face***

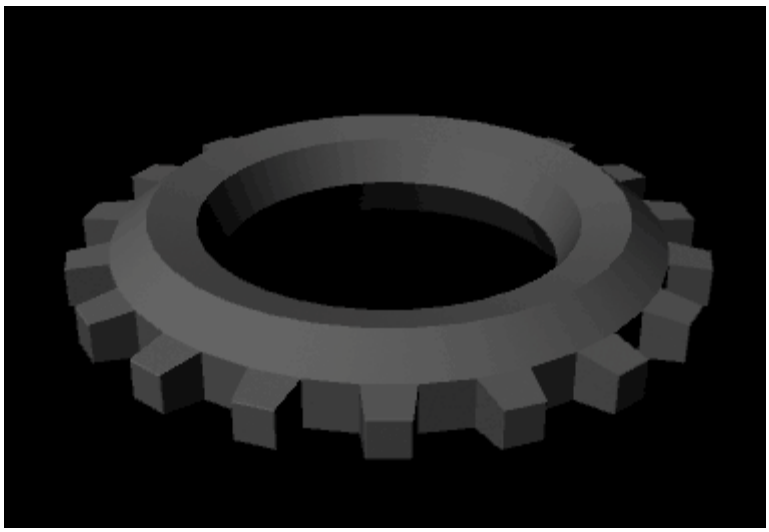


Рис. 5.4. Результат застосування операції ***Extrude Face***

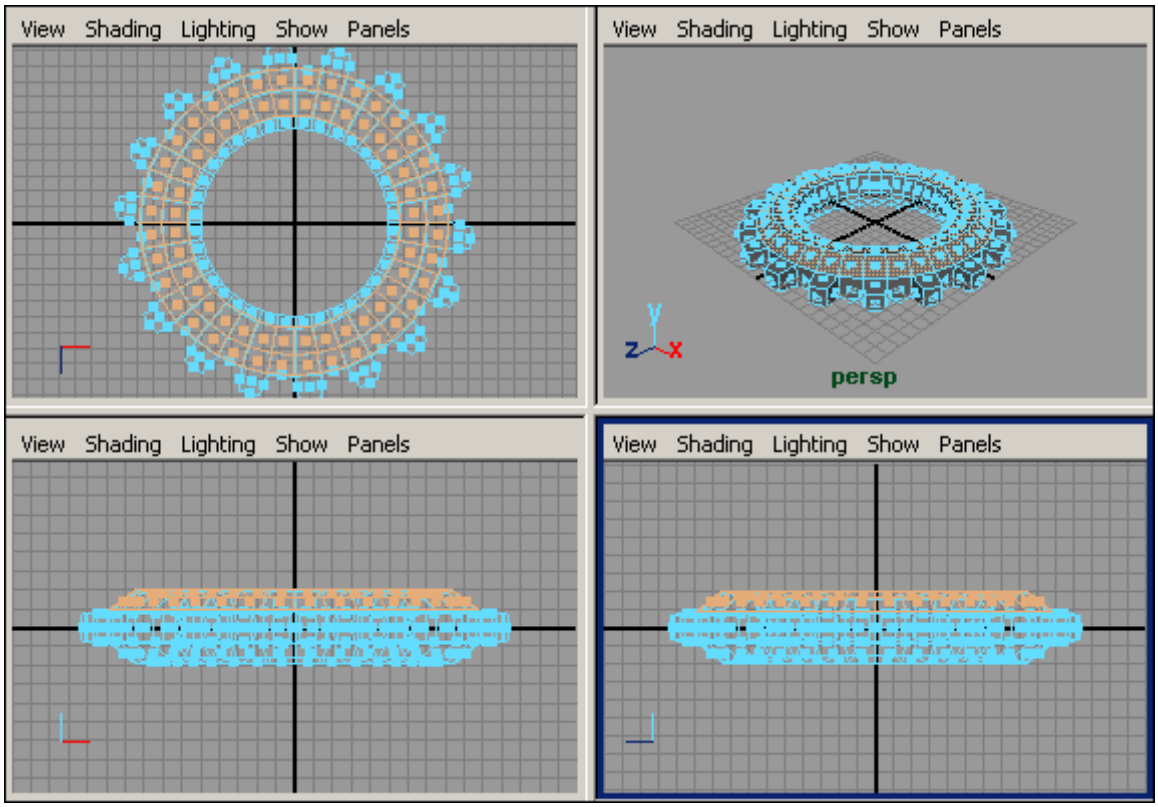


Рис. 5.5. Виділення граней на верхній поверхні тора

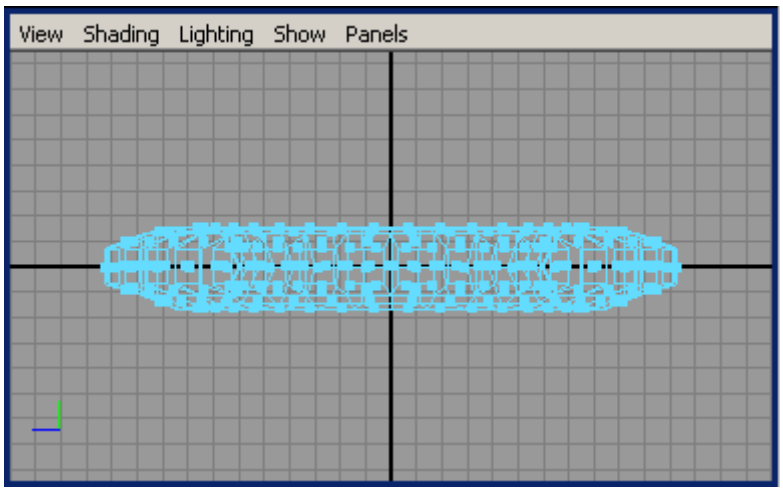


Рис. 5.6. Вид проекції ***Side*** *після масштабування*

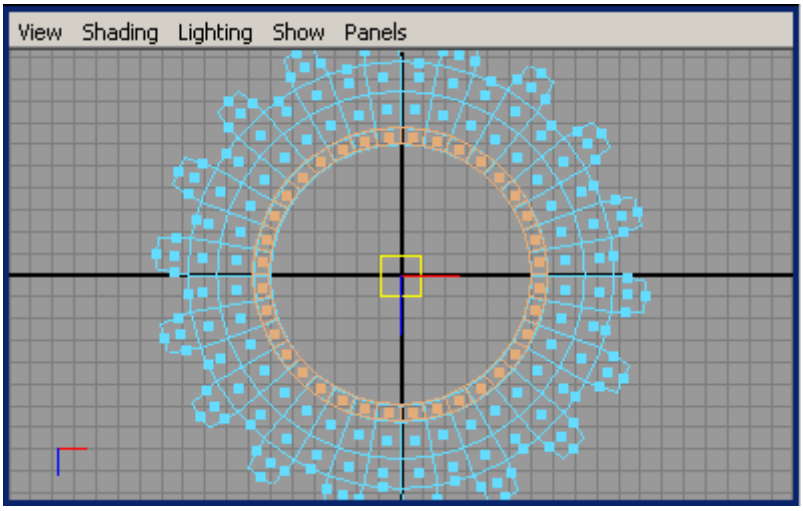


Рис. 5.7. Виділення граней на внутрішній поверхні тора

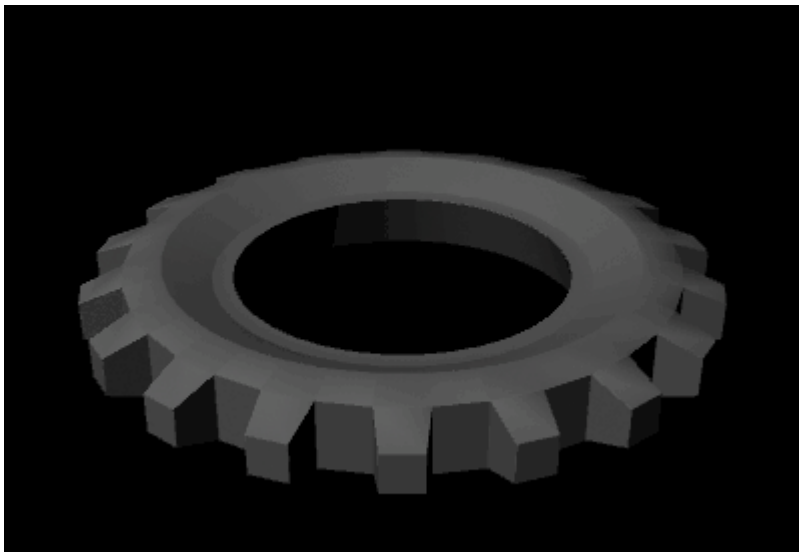


Рис. 5.8. Шестерня

6. Ваза з циліндру

Створіть циліндр з такими параметрами, як показано на рис. 6.1. Насамперед видаліть верхню площину циліндра: перейдіть в режим роботи з чотирма проекціями, натисніть клавішу F11 для переходу до редагування граней, виділіть грані самого верхнього перерізу (рис. 6.2) і натисніть клавішу **Del** (рис. 6.3). Тепер потрібно надати об'єкту форму вази. Виділіть грані 2-го і 3-го зверху перерізів і масштабуйте їх так, щоб радіус перерізів зменшився (рис. 6.4). Приблизно так само зменшіть радіус самого нижнього перерізу (рис. 6.5). Потім виділіть грані трьох верхніх перерізів і перетягніть їх вгору, щоб надати вазі витягнуту форму (рис. 6.6). Потім підкоректуйте форму вази по своєму бажанню, схожим чином масштабуючи, витягаючи і стискаючи грані тих чи інших перерізів, а після закінчення перейдіть в режим редагування на рівні об'єкта, натиснувши клавішу F8, виділіть вазу і проведіть її згладжування, застосувавши команду **Poligons => Smooth** (полігони => Згладжування). Отриманий в результаті об'єкт представлений на рис. 6.7 і 6.8.

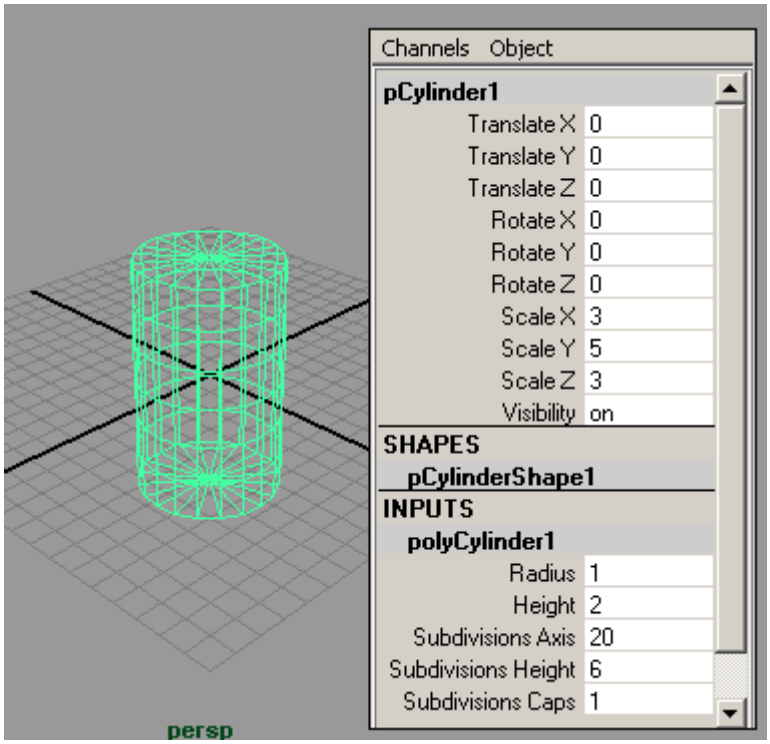


Рис. 6.1. Початковий циліндр

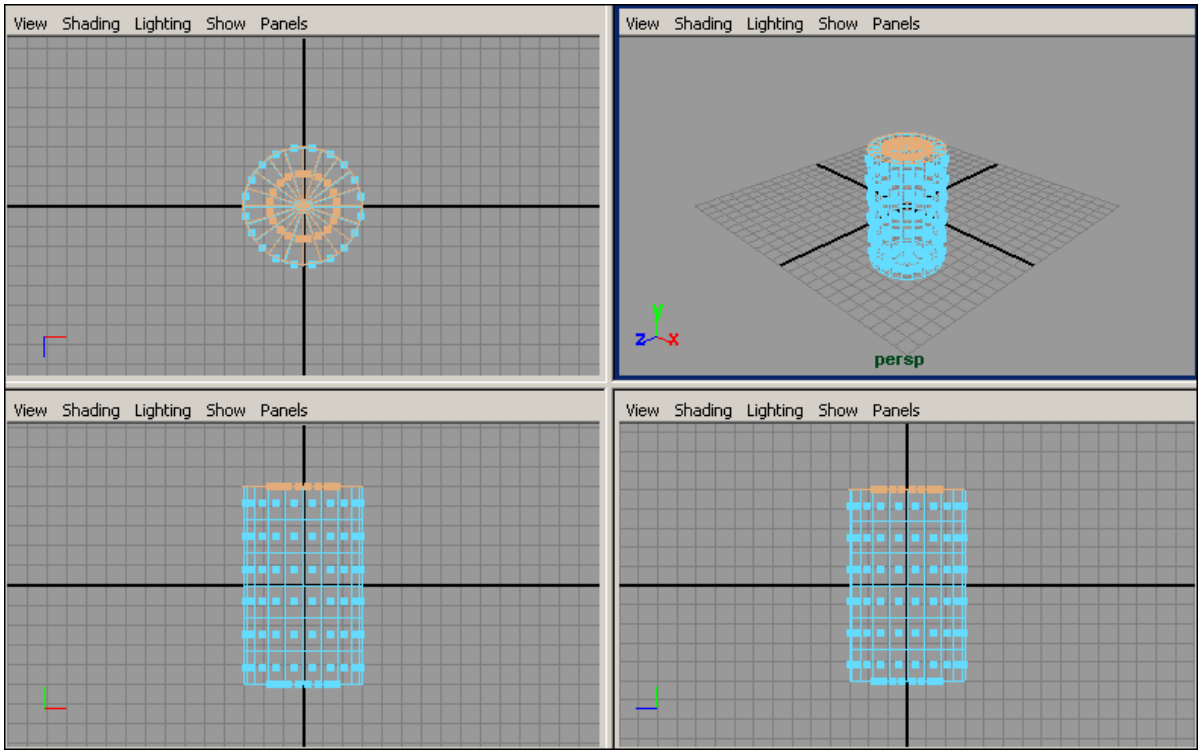


Рис. 6.2. Виділення граней верхнього перерізу

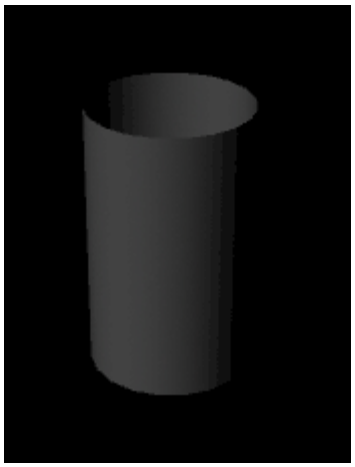


Рис. 6.3. Результат видалення верхніх граней

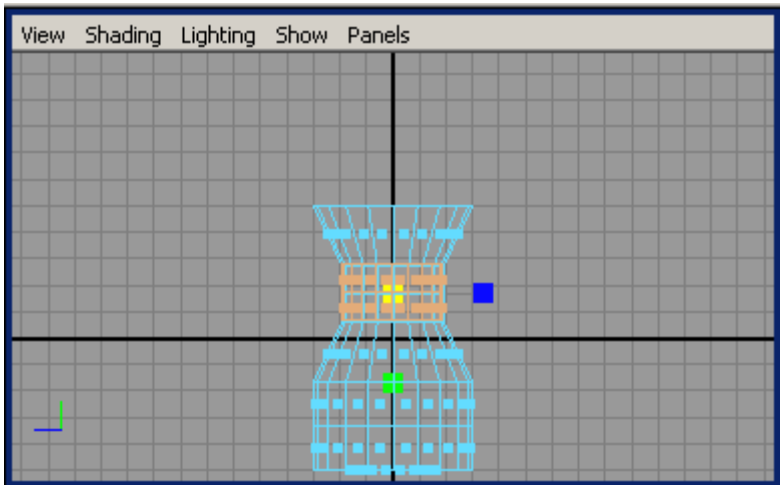


Рис. 6.4. Об'єкт після першого масштабування перерізів

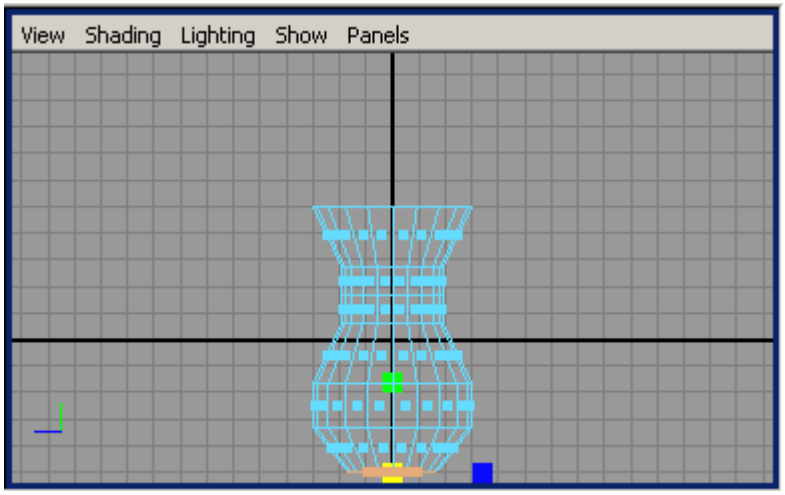


Рис. 6.5. Результат другого масштабування перерізів

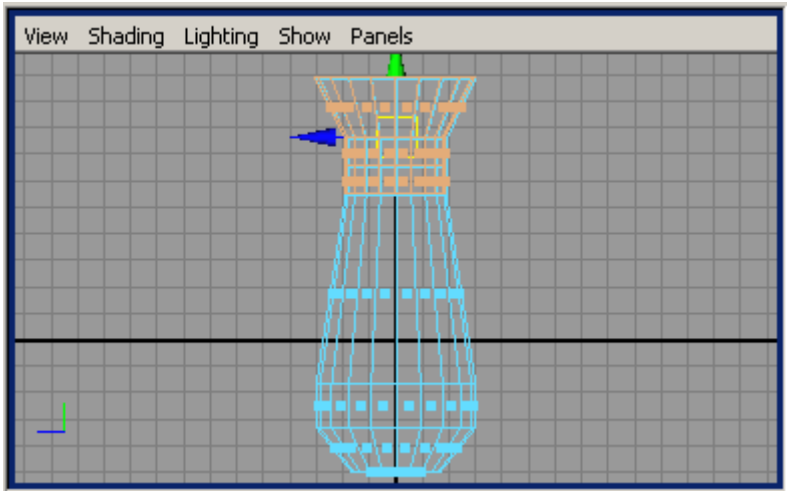


Рис. 6.6. Об'єкт після переміщення перерізів

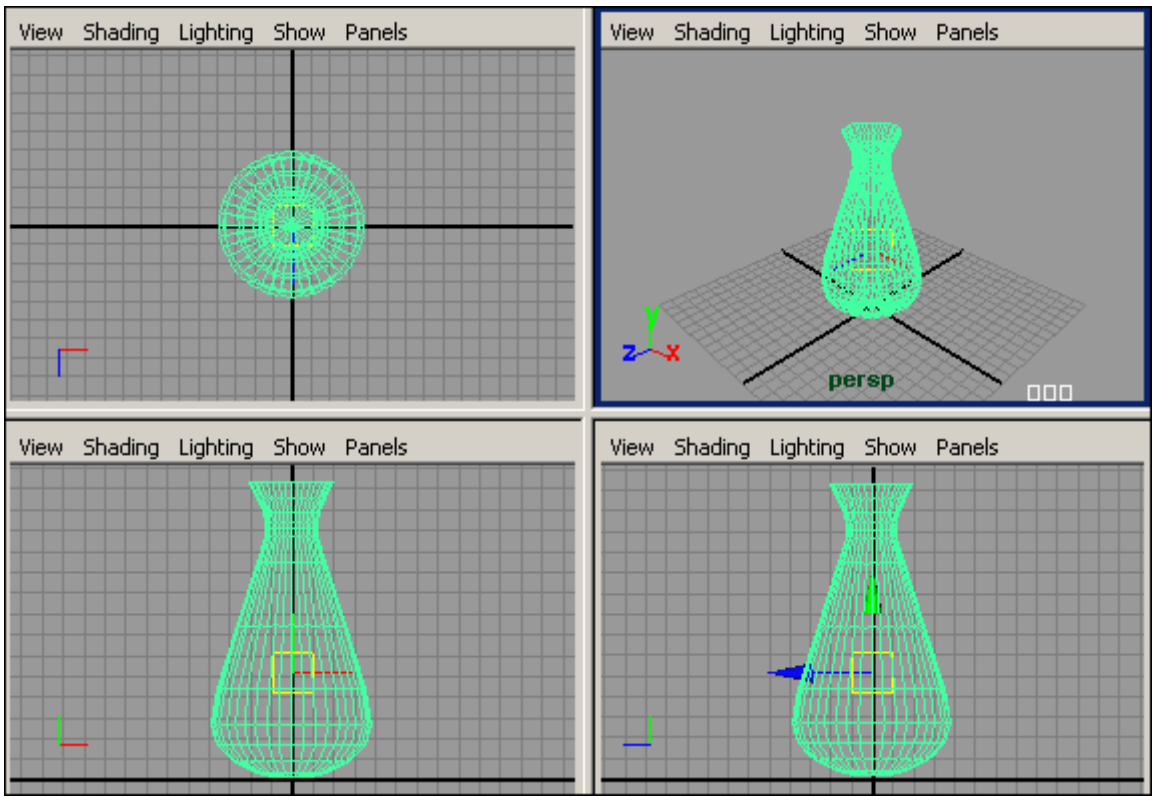


Рис. 6.7. Остаточний вигляд об'єкта у вікнах проекцій

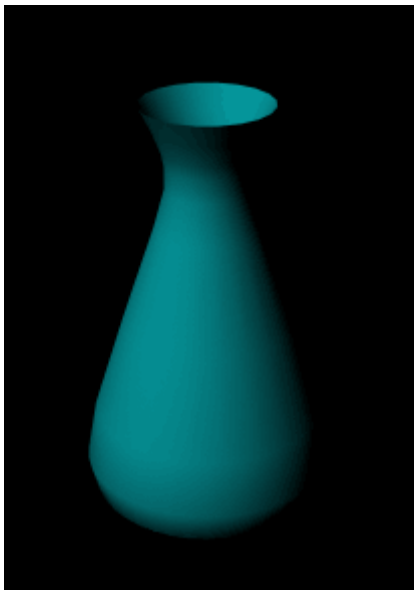


Рис. 6.8 . Ваза

7. Огранений кристал з куба

Створіть полігональний куб (рис. 7.1), перейдіть в режим редагування граней (клавіша F11), виділіть всі грані і застосуйте до них операцію **Bevel** (Фаска) - рис. 7.2. Перейдіть в режим роботи з вершинами (клавіша F9), виділіть всі чотири вершини верхньої грані (рис. 7.3) і зваріть їх в одну точку, скориставшись командою **Edit Poligons => Merge Vertices** (Правка полігонів => З'єднати вершини) - рис. 7.4. Перейдіть в режим роботи з гранями, виділіть грані нижніх перерізів (рис. 7.5) і також зведіть їх в одну точку за допомогою тієї ж самої команди (рис. 7.6), а потім перемістіть виступаючу вершину вгору по осі Y так, щоб всі вершини нижніх перерізів стали розташовуватися на одній площині (рис. 7.7). Перейдіть до редагування вершин (клавіша F9) і підкоригуйте їх положення так, щоб об'єкт дійсно став нагадувати огранений кристал (рис. 7.8). В результаті буде отримано приблизно такий кристал, як на рис. 7.9

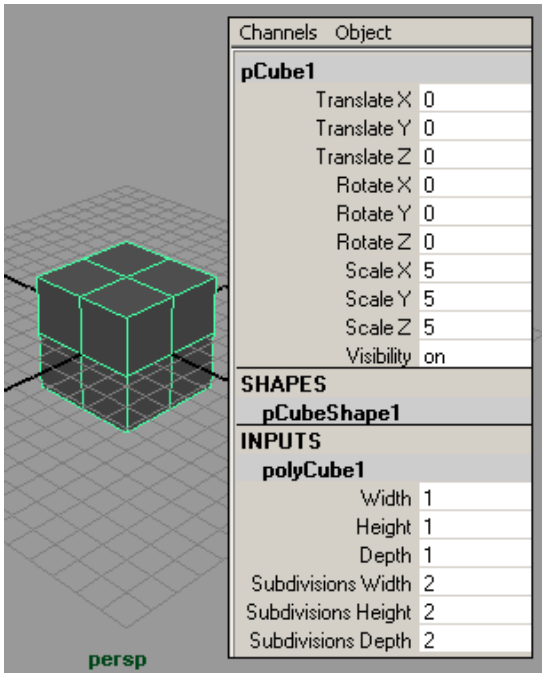


Рис. 7.1. Початковий куб

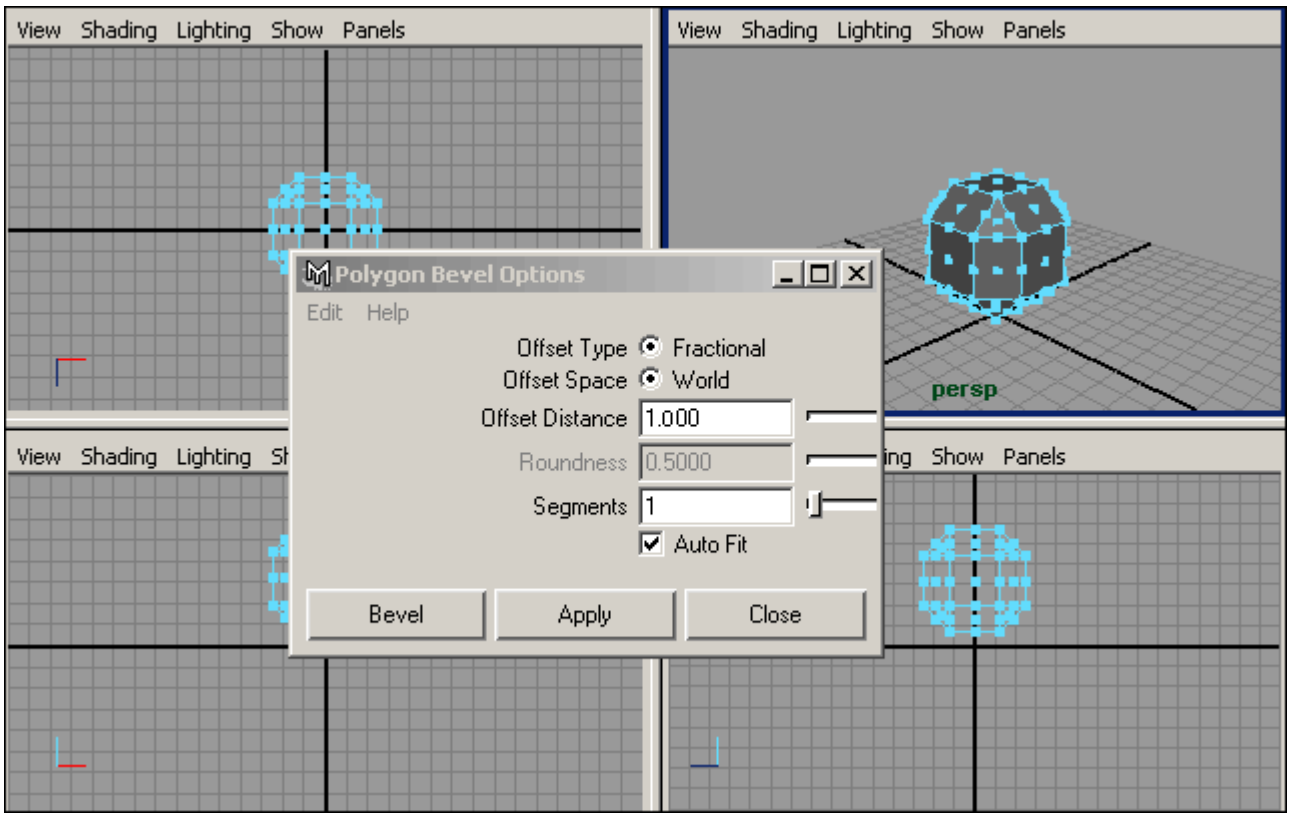


Рис. 7.2. Зкошування граней

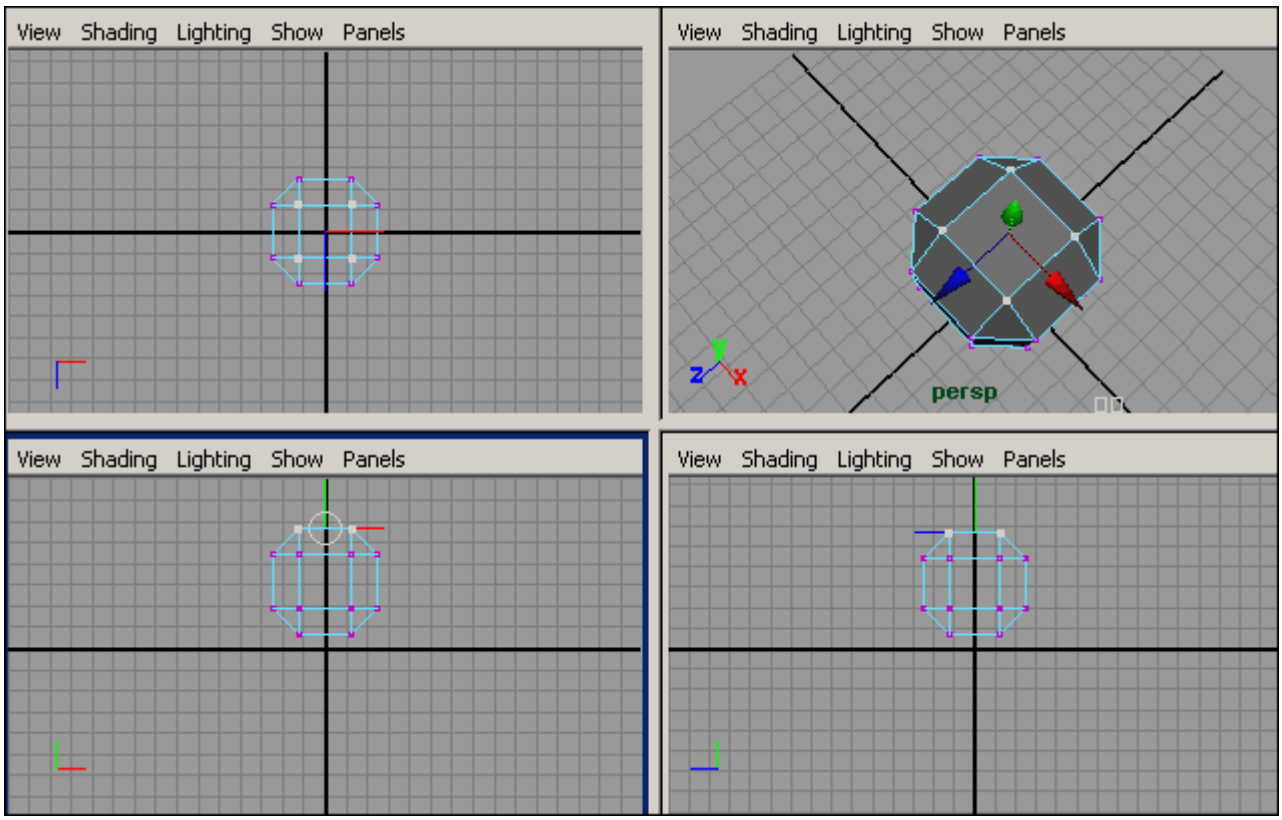


Рис. 7.3. Виділення вершин верхньої грані

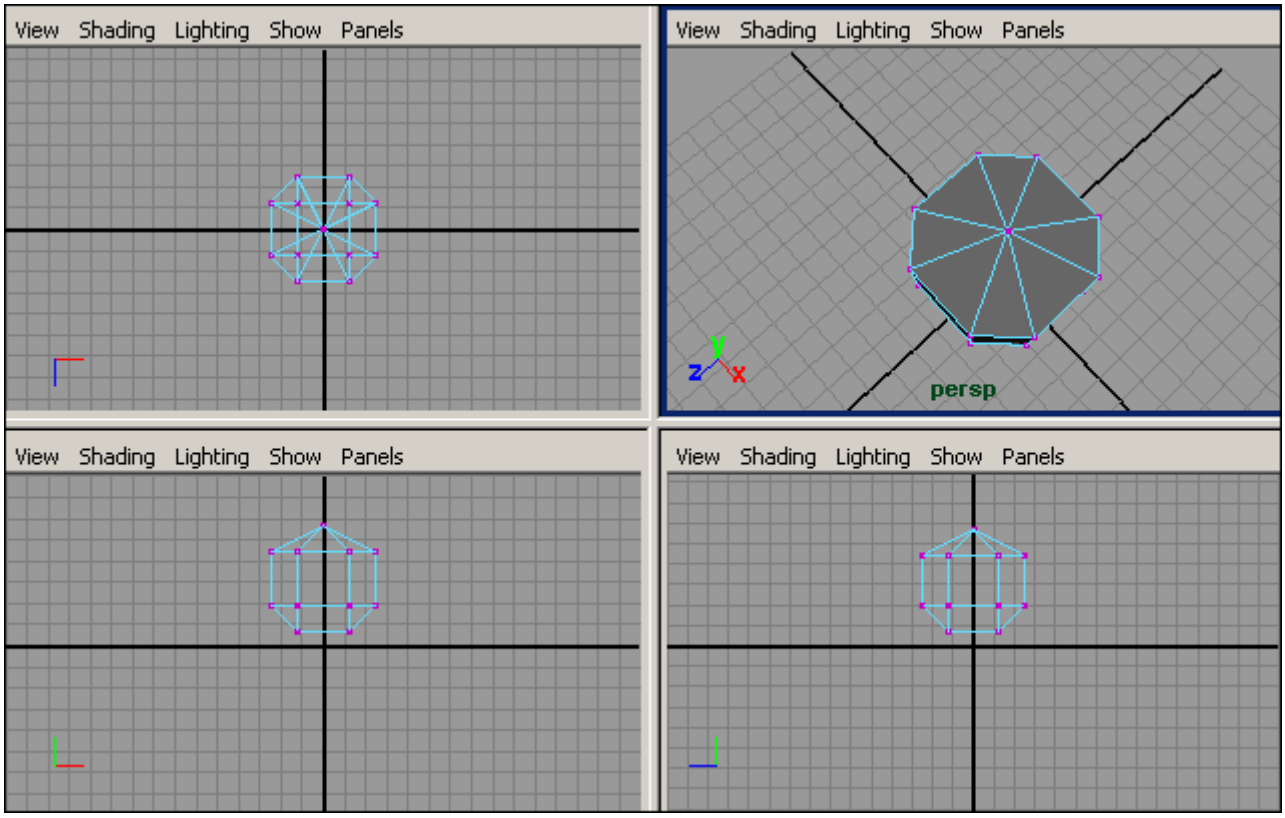


Рис. 7.4. Результат з'єднання вершин верхньої грані

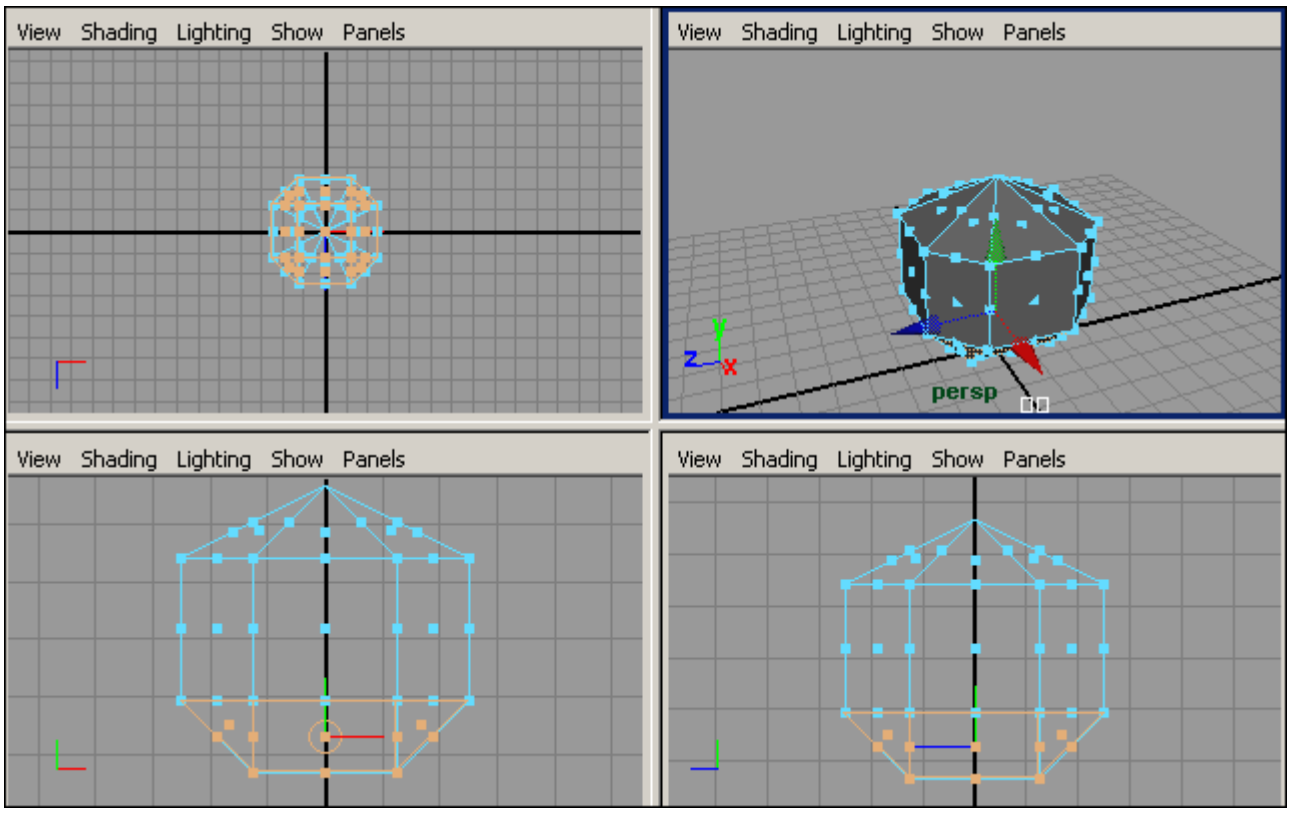


Рис. 7.5. Виділення граней нижніх перерізів

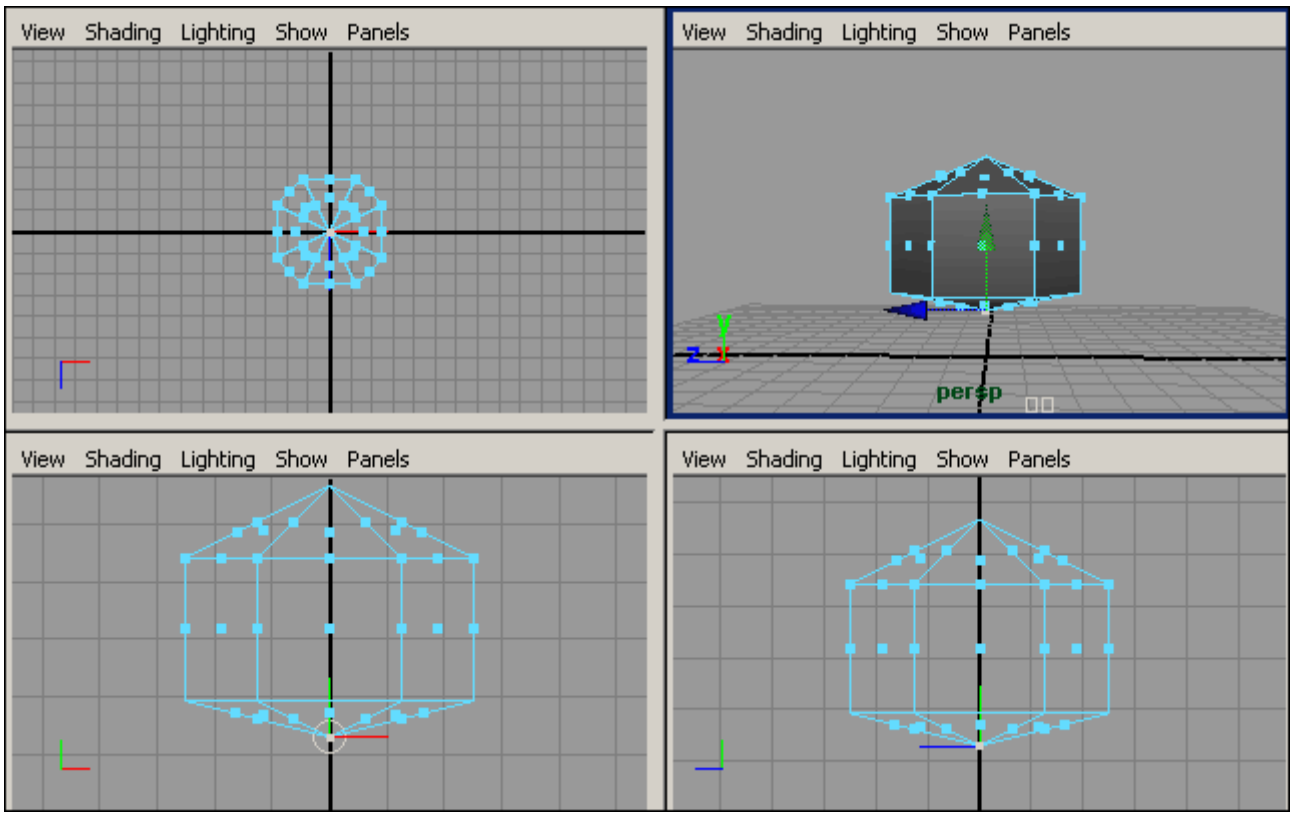


Рис. 7.6. Результат з'єднання вершин нижніх граней

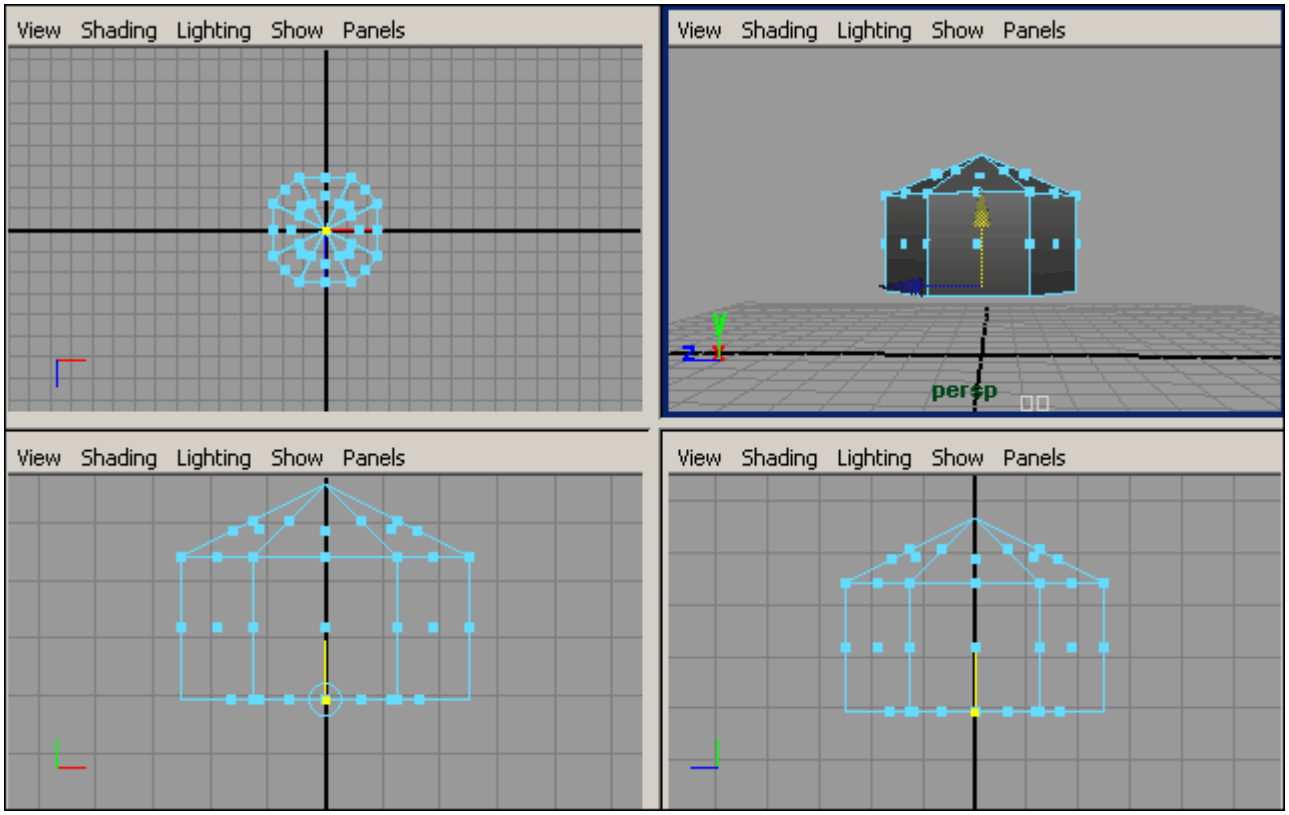


Рис. 7.7. Вигляд об'єкту після зведення всіх вершин нижніх граней на одну площину

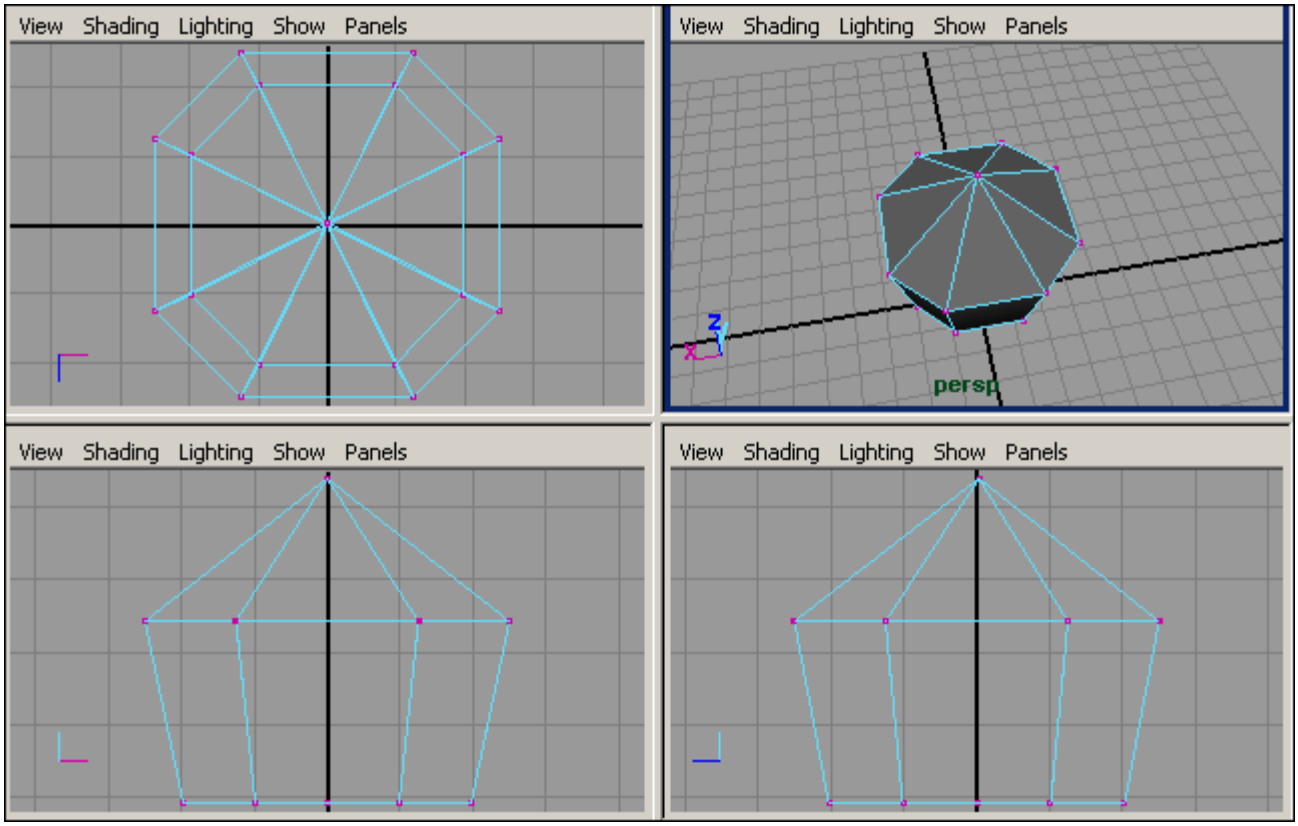
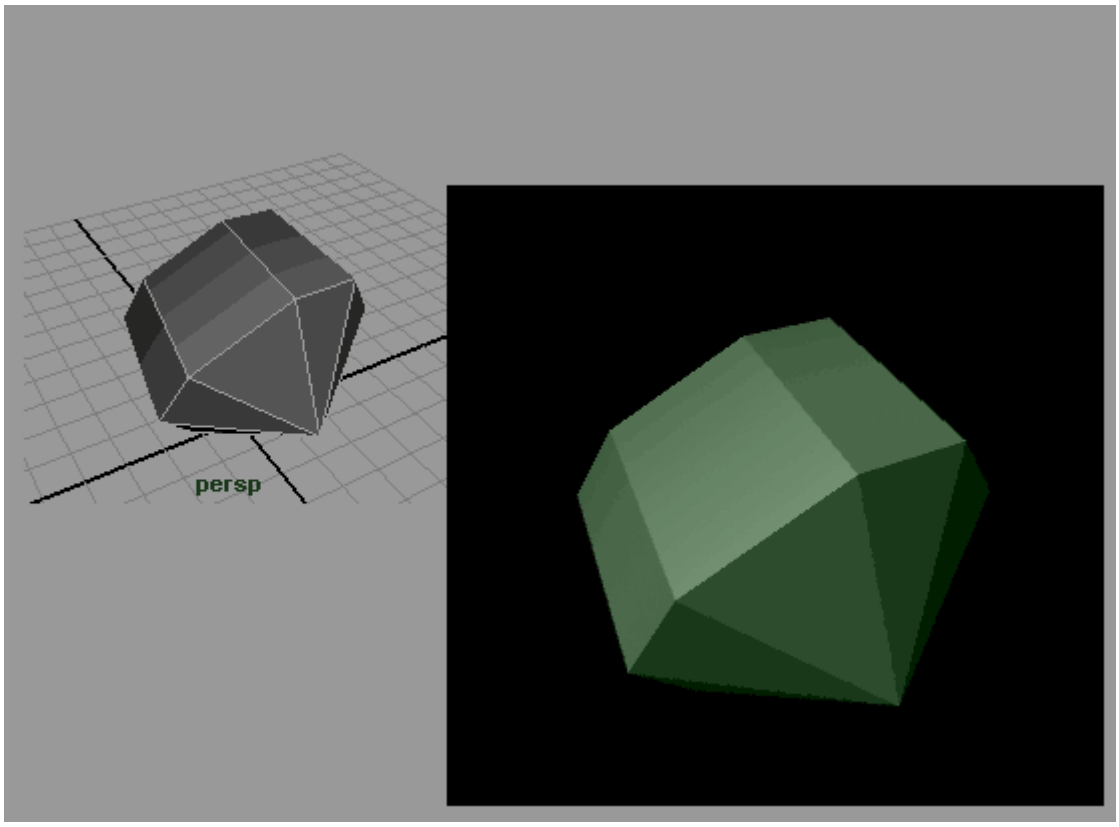


Рис. 7.8. Кінцевий вид об'єкта



8. Дерево з куба

Розглянемо найпростіший варіант «витягування» і спробуємо витягнути зі звичайного куба (рис. 8.1) стовбур і гілки дерева.

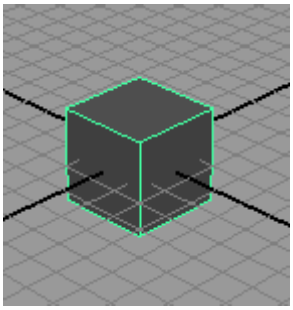


Рис. 8.1. Початковий куб

Перейдіть в режим редагування граней, виділіть верхню грань і витягніть з неї нову грань трохи меншого розміру, відкривши з меню **Edit Poligons** команду **Extrude Face** (Видавлювання граней) і скорегувавши параметри грані стрілками маніпулятора інструменту - рис. 8.2. Застосуйте дану операцію багато разів, в якості початкової використовуйте завжди тільки що створену грань, зменшуючи розміри новостворюваної грані і коригуючи маніпуляторами напрямок її зміщення. У результаті повинна вийти перша головна гілка полігонального дерева, яка може нагадувати представлену на рис. 8.3. Перейдемо до створення другої основної гілки - виділіть на стовбурі дерева відповідний полігон (з якого теоретично може рости така гілка) і почніть той же самий процес багаторазового застосування операції **Extrude Face**, послідовно формуючи все нові фрагменти гілки. Процес отримання першого фрагмента другої гілки показаний на рис. 8.4, а вся гілка повністю - на рис. 8.5.

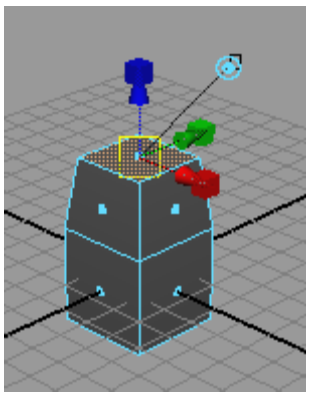


Рис. 8.2. Поява першої грані першої гілки

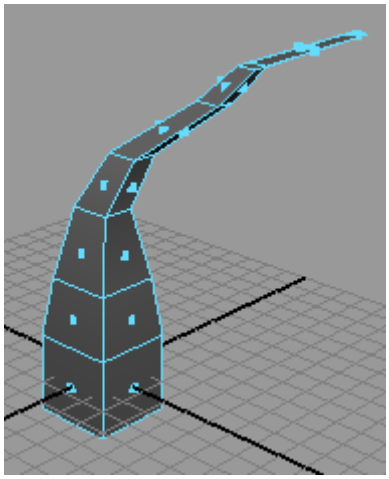


Рис. 8.3. Перша гілка дерева

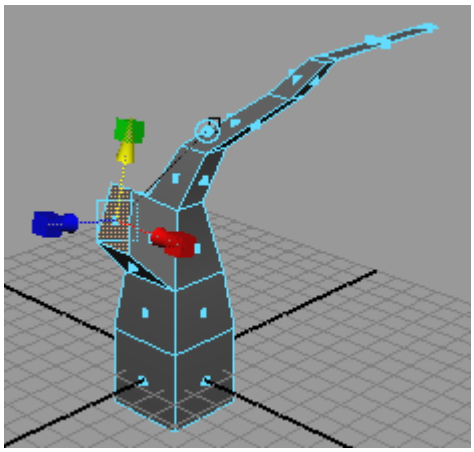


Рис. 8.4. Поява першої грані другої гілки

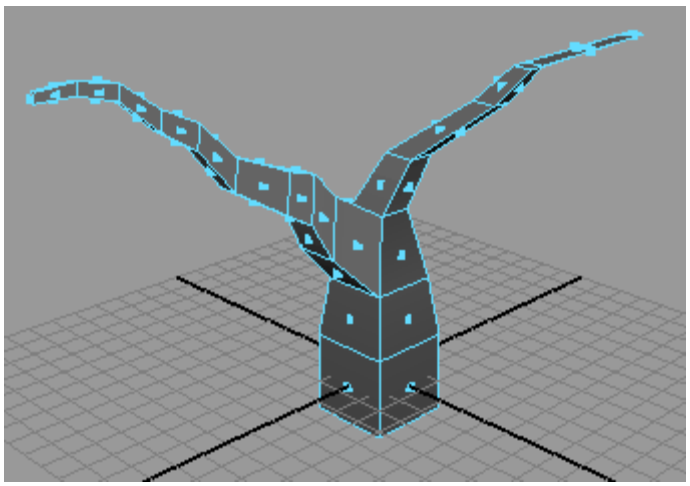


Рис. 8.5. Друга гілка дерева

Далі на першій основній гілці сформуйте нову гілку наступного порядку - виділіть вказаний на рис. 8.6 полігон і витягніть з нього задуману гілка. Точно таким же чином змоделюймо ще кілька гілок, наприклад відповідно до рис. 8.7. Тепер перейдемо до моделювання нижньої частини стовбура, яка, як правило, буває значно ширше, ніж основна частина стовбура. Оскільки потрібний для роботи полігон зараз прихований, змініть кут огляду сцени шляхом переміщення миші у вікні проекції при натиснутих лівої кнопці миші і клавіші **Alt**, а потім виділіть полігон, що знаходиться в передбачуваній основі дерева (рис. 8.8). За допомогою операції **Extrude Face** витягніть з даного полігону кілька полігонів, не забуваючи кожного разу збільшувати їх розміри. Після закінчення оцініте результат і при необхідності скорегуйте положення окремих вершин, ребер і граней, уважно оглянувши модель з усіх боків. Можливо, отримане в результаті дерево буде схоже на представлене на рис. 8.9. Натисніть клавішу F8, щоб повернутися до режиму редагування на рівні об'єкта, виділіть об'єкт і двічі застосуйте до нього операцію згладжування (команда **Poligons => Smooth** - Полігони => Згладжування). Після цього дерево буде виглядати досить реалістично (рис. 8.10).

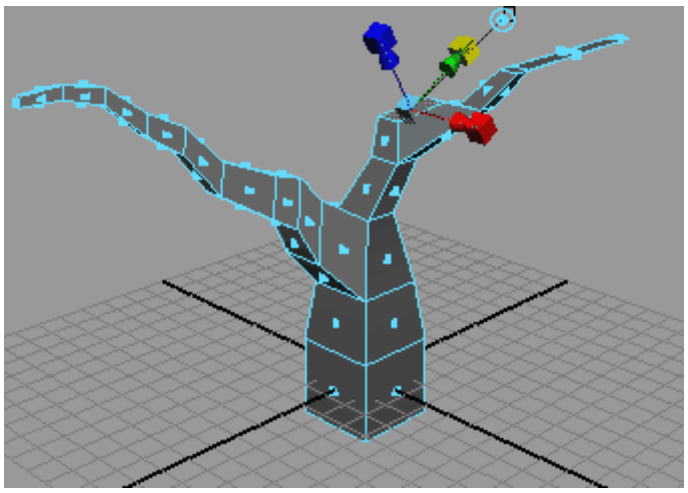


Рис. 8.6. Виділення полігону для створення додаткової гілки на першій основній гілці

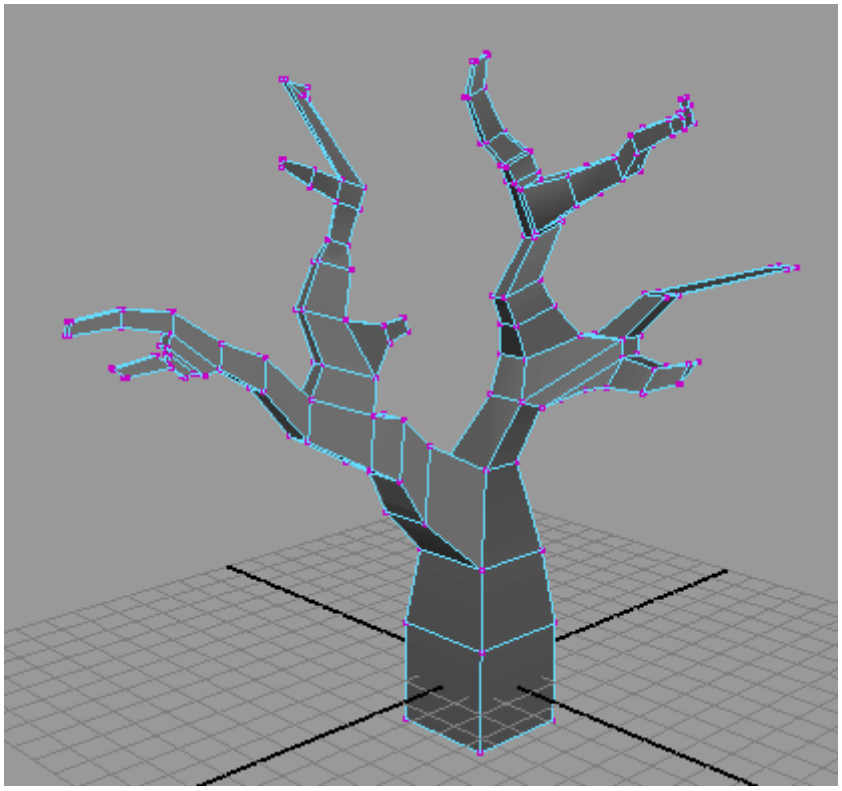


Рис. 8.7. Вид дерева після створення всіх гілок

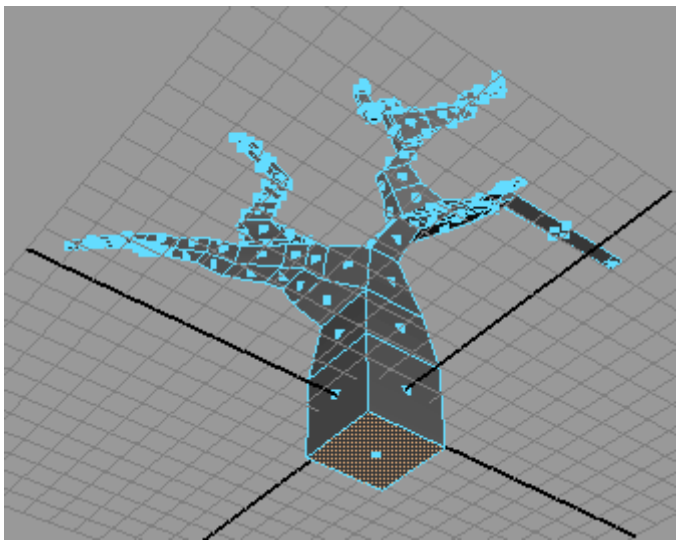


Рис. 8.8. Виділення полігону в основі дерева

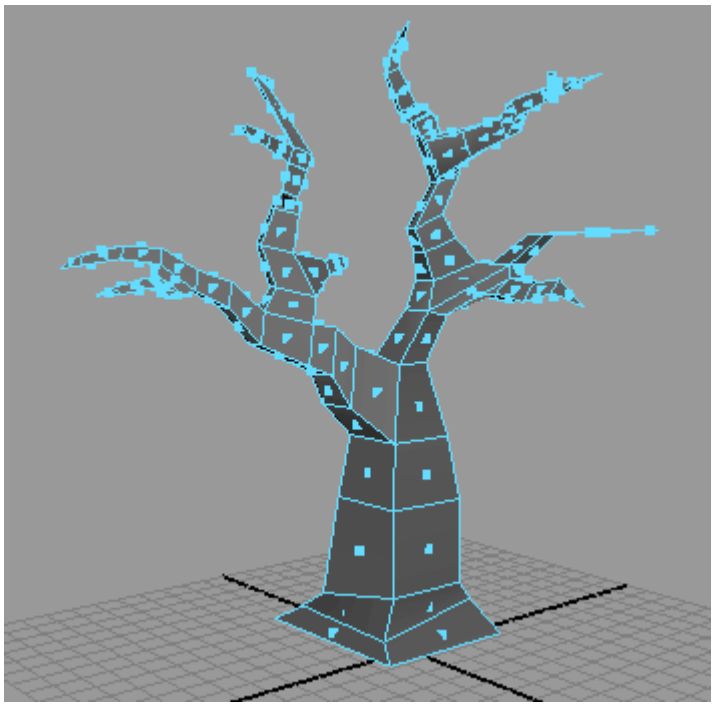


Рис. 8.9. Полігональне дерево без згладжування

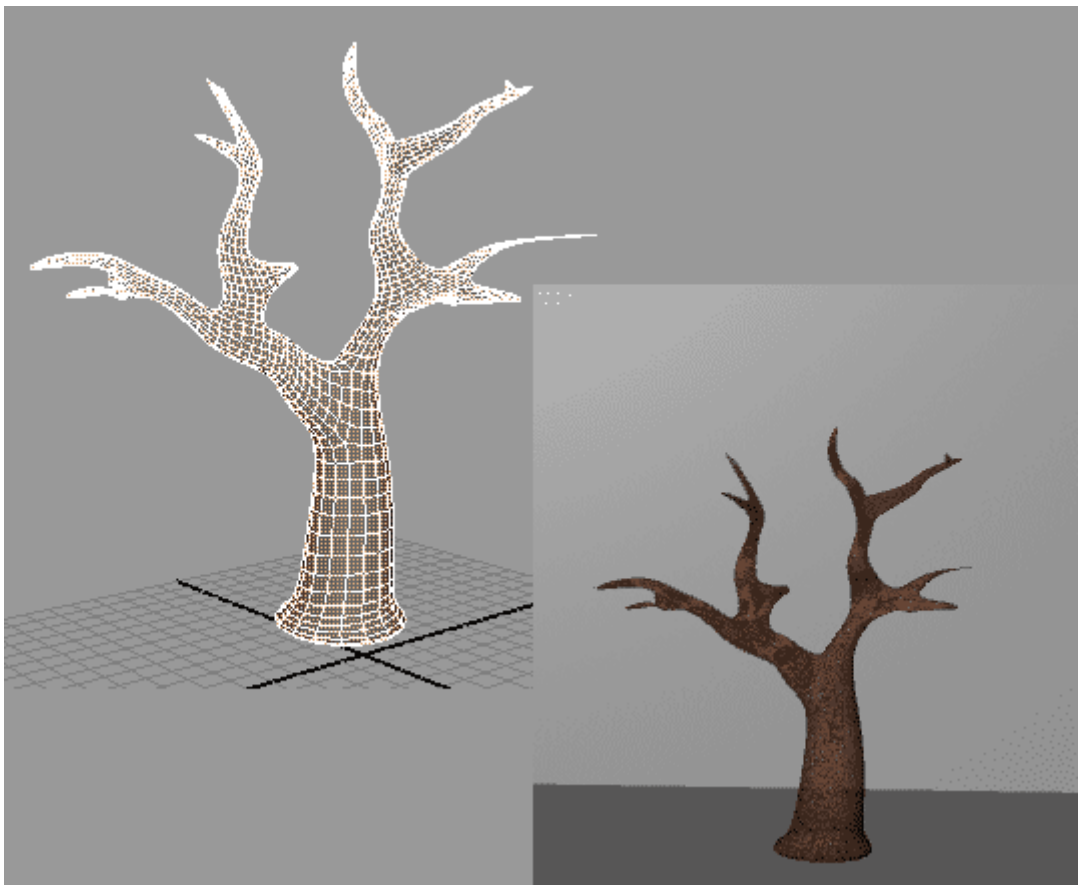


Рис. 8.10. Дерево

9. Кофейна чашка з циліндра

Створіть полігональний циліндр (рис. 9.1), натисніть клавішу F11 для переходу в режим редагування граней, виділіть всі грані верхнього перерізу (рис. 9.2) і видаліть їх, натиснувши клавішу **Del**. Потім послідовно виділяючи грані того чи іншого перерізу і застосовуючи операції масштабування (**Scale Tool**), переміщення (**Move Tool**) і видавлювання граней (**Edit Poligons => Extrude Face**), змоделюймо контури кавової чашки. Можливий варіант показаний на рис. 9.3.

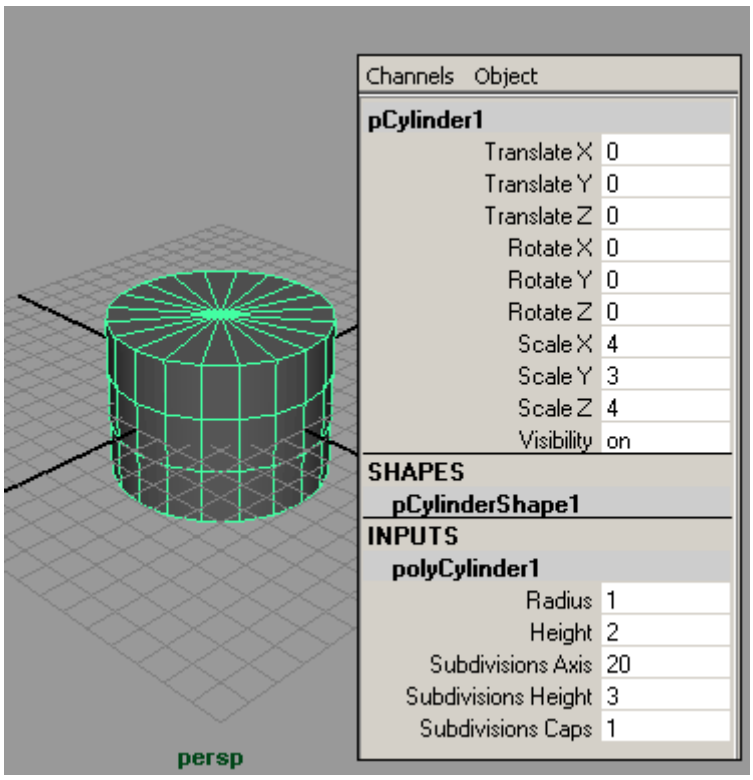


Рис. 9.1. Початковий циліндр

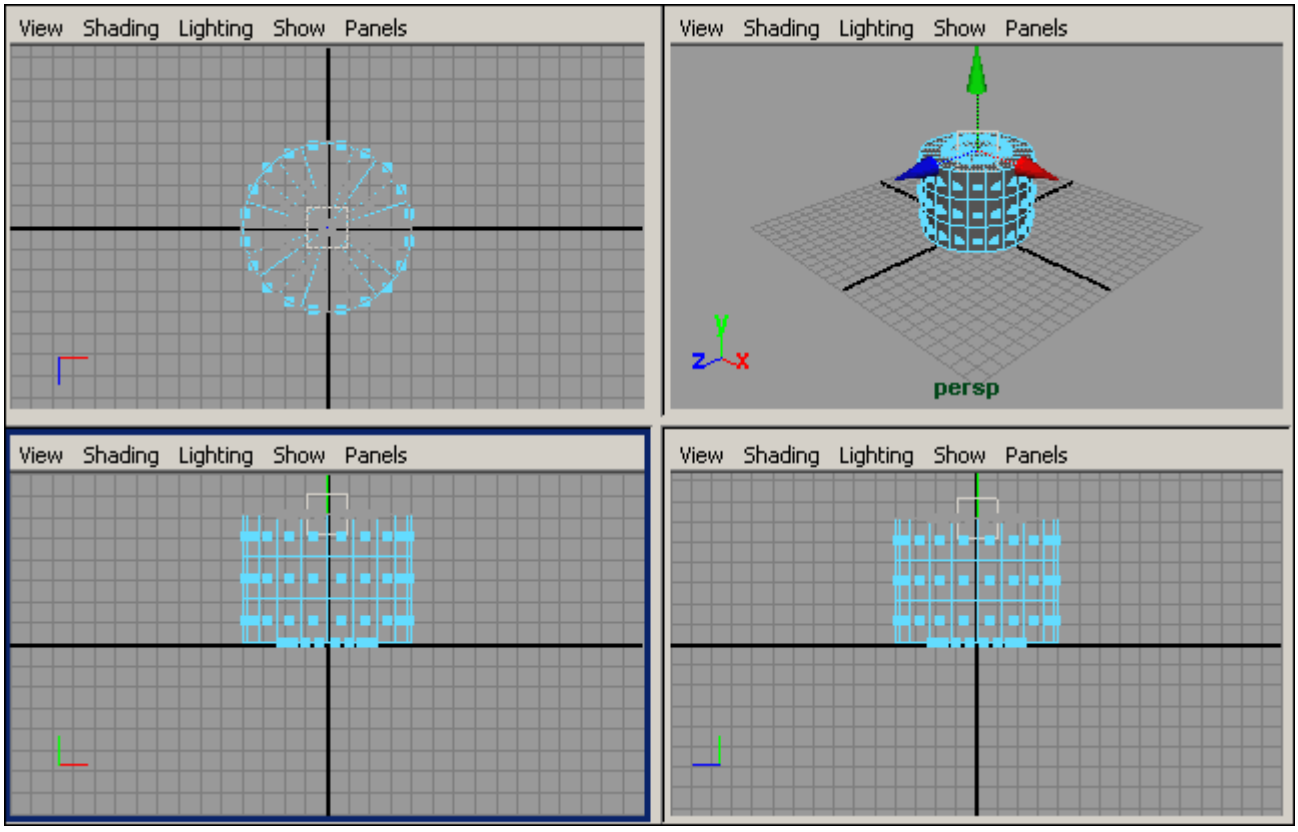


Рис. 9.2. Виділення граней верхнього перерізу

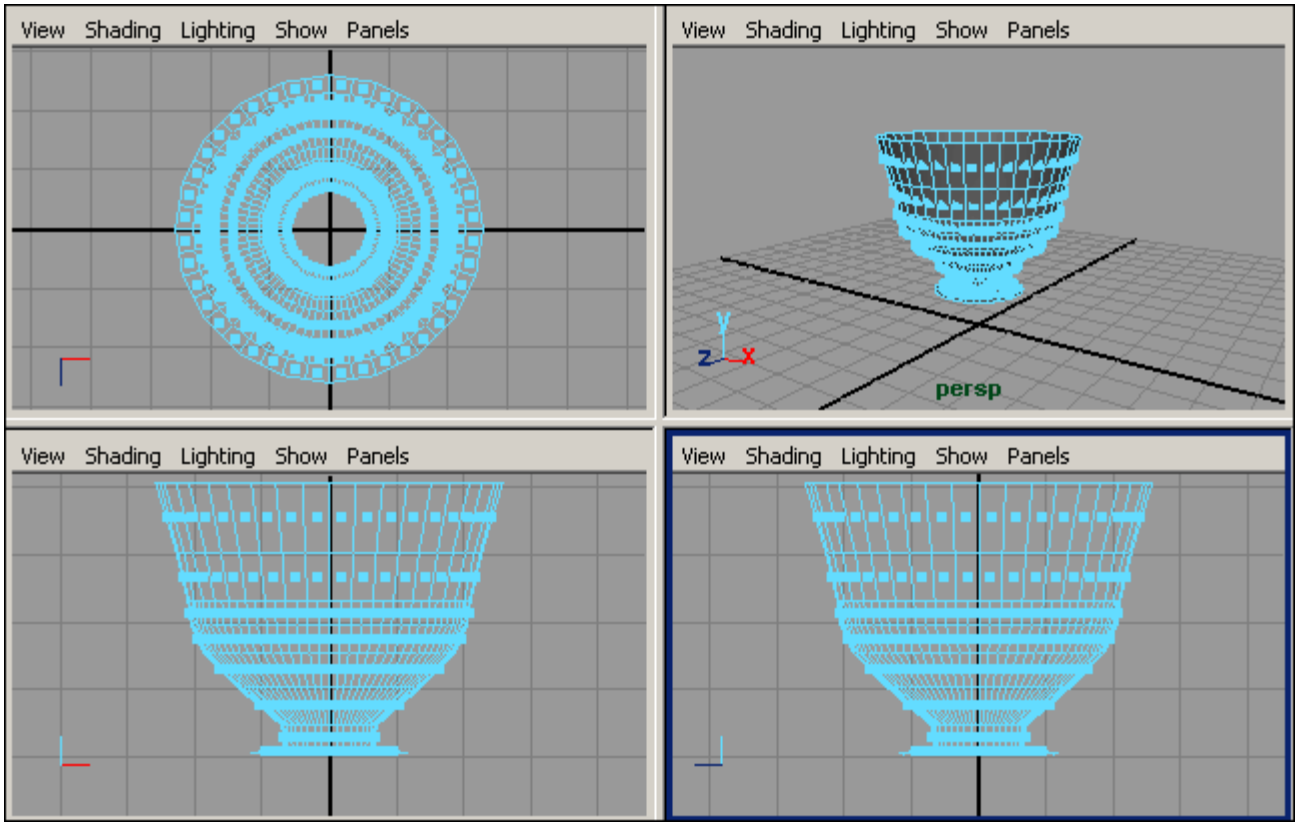


Рис. 9.3. Результат генерування основних контурів чашки

Тепер потрібно перейти до створення ручки, що можна зробити шляхом витягування граней прямо з контуру чашки. Спочатку в тому місці контуру, де повинна кріпитися ручка, доведеться розбити дві грані приблизно навпіл - для цього з меню **Edit Polygons** відкрийте команду **Split Polygon Tool** (Розбити полігон) і послідовно вкажіть точки, які повинні з'єднуватися новими ребрами. Грані, що розбиваються показані на рис. 9.4, а результат їх розбиття - на рис. 9.5. Приступимо до витягування ручки. Виділіть зазначені на рис. 9.6 грані і за допомогою багаторазового застосування команди **Edit Poligons => Extrude Face** (Редагування полігонів => Видавлювання граней) сформуйте верхню частину ручки (рис. 9.7). А потім просто перемістіть крайні полігони ручки потрібним чином, наприклад як показано на рис. 9.8.

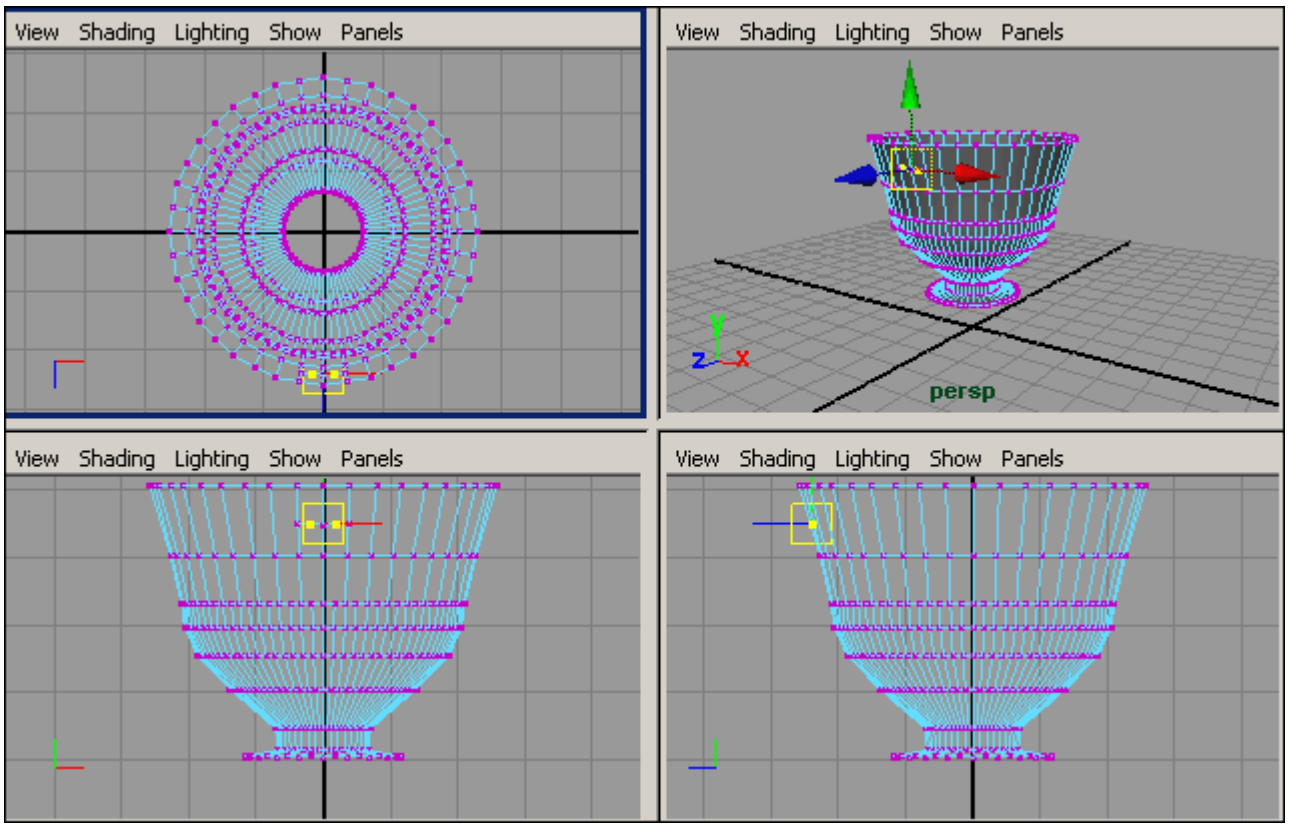


Рис. 9.4. Грані, що розбиваються

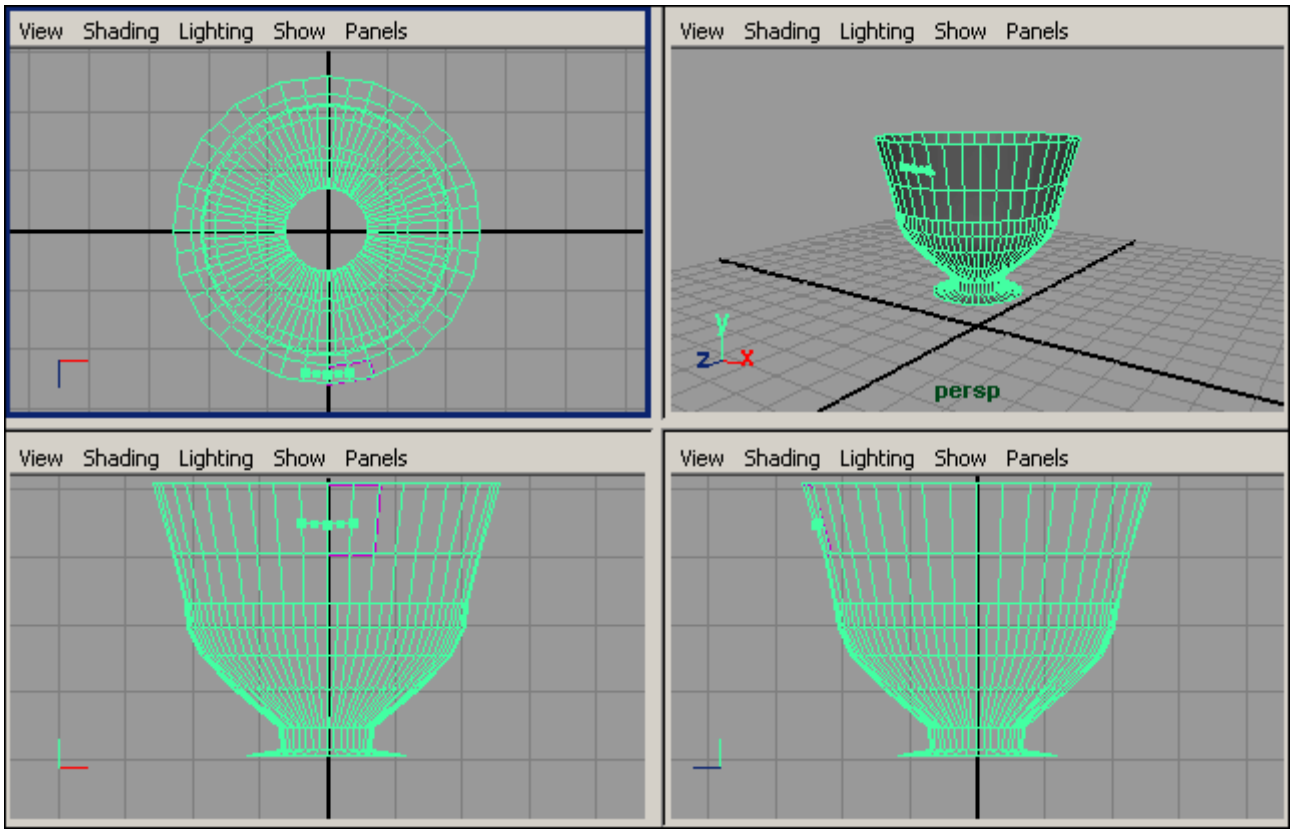


Рис. 9.5. Поява нових ребер в місці кріплення ручки

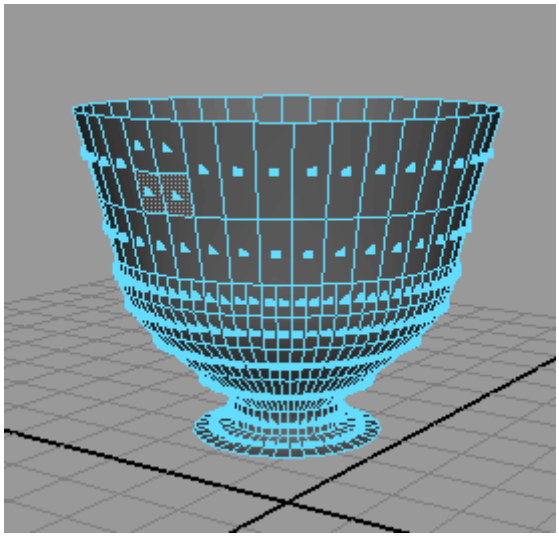


Рис. 9.6. Грані, що витягуються

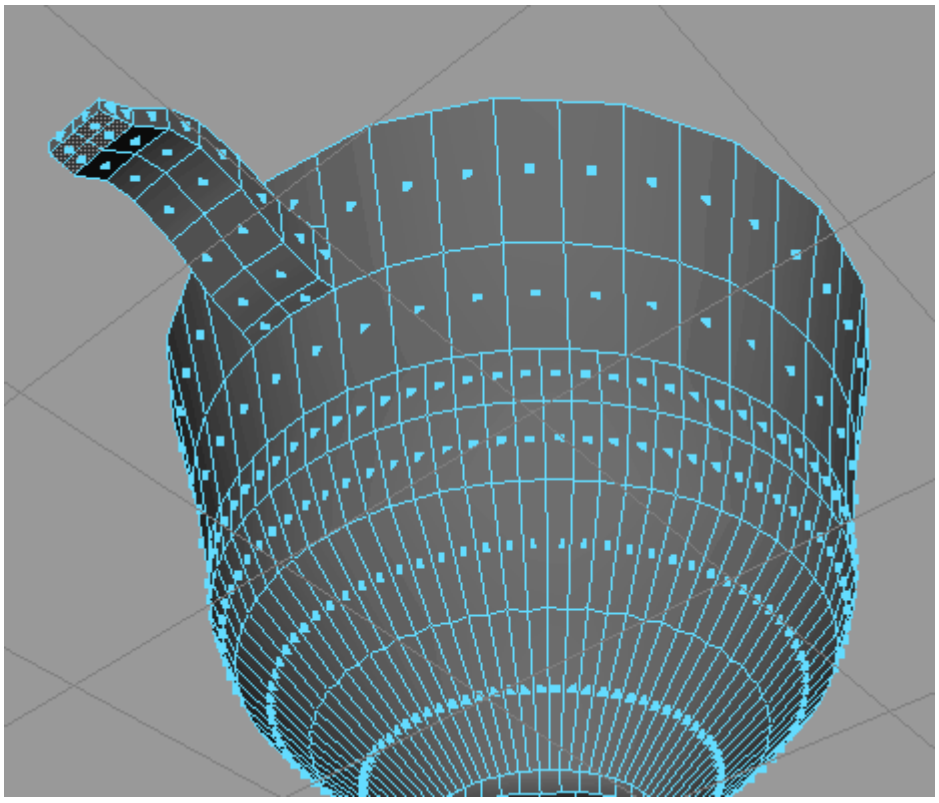


Рис. 9.7. Поява верхньої частини ручки

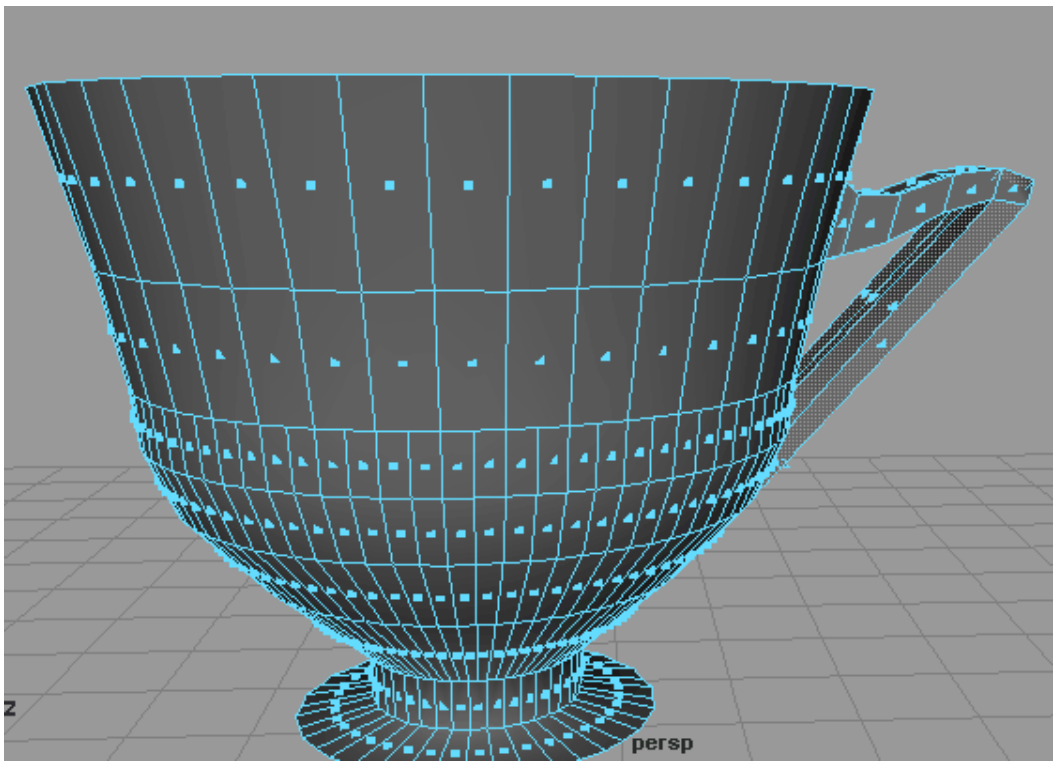


Рис. 9.8. Чашка с повністю готовою ручкою

Поверніться в режим редагування моделі на рівні об'єктів (клавіша F8) і застосуєте до неї операцію згладжування (команда **Poligons => Smooth** - Полігони => Згладжування) - отримаєте приблизно таку ж чашку, як представлена на рис. 9.9.

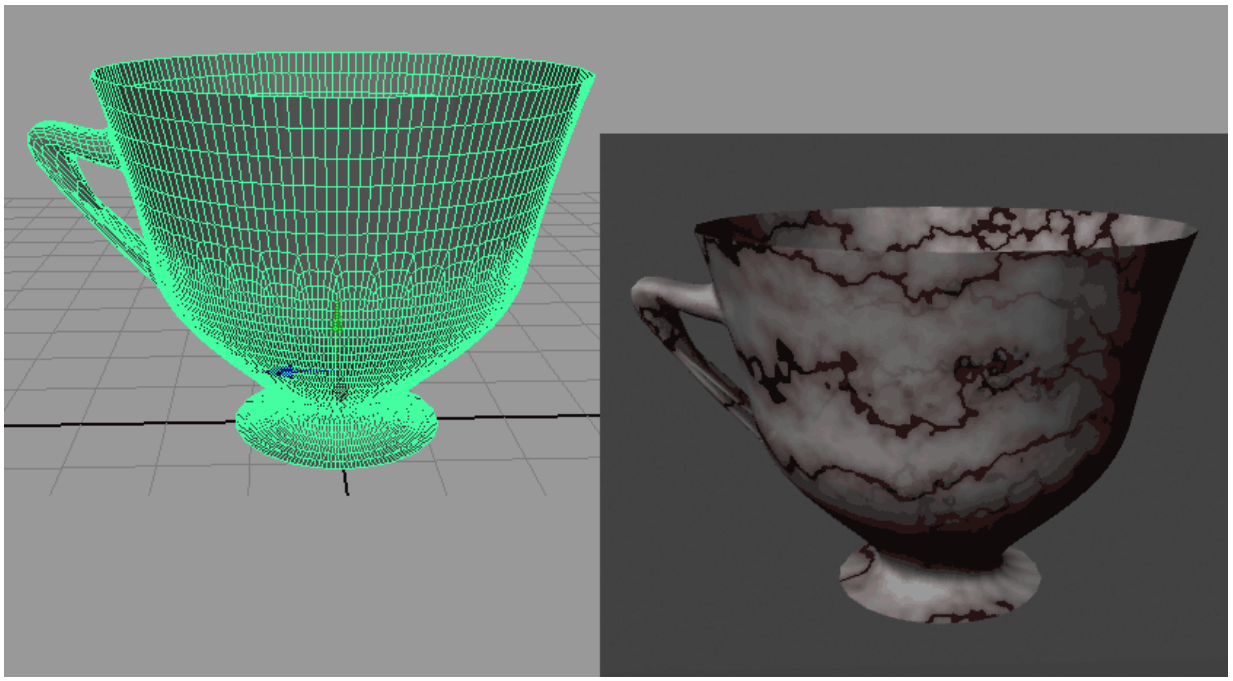


Рис. 9.9. Чашка

6.6. Завдання до самостійної роботи по темі «Основи NURBS моделювання»

1. Спадаюча тканина

– створіть звичайну NURBS-площину і перейдіть в режим редагування вершин (клавіша F9) - спочатку кількість її керуючих вершин невелика, рис.1.1;

– поверніться в режим роботи на рівні об'єкта (клавіша F8) і додайте ще групу вершин, застосувавши команду **Insert Knot** (Вставити вузли), рис.1.2.

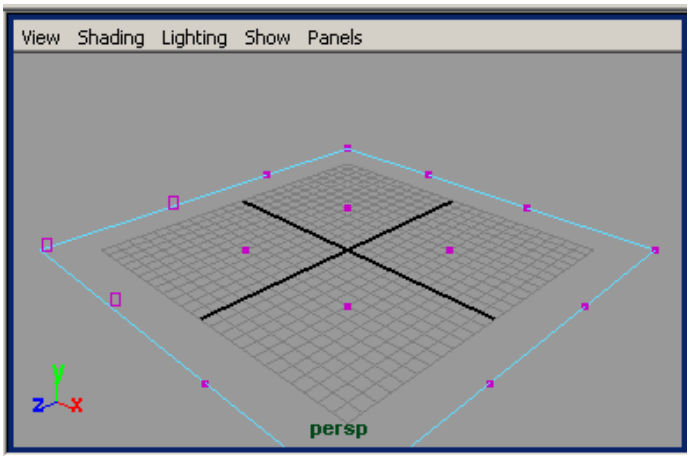


Рис. 1.1. Вихідна площина

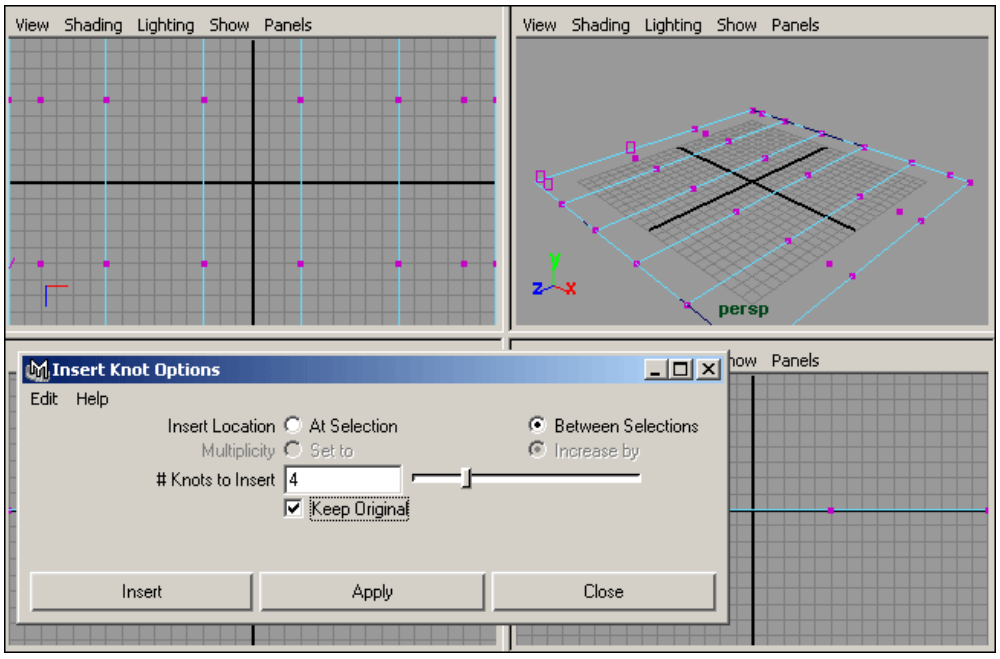


Рис. 1.2. Додавання нових вершин

– відрегулюйте положення вершин за допомогою інструментів **Move Tool** і **Scale Tool**, керуючись видом наявної текстури тканини зі складками і тим, як плануєте розкласти цю тканину, рис.1.3 - 1.4.

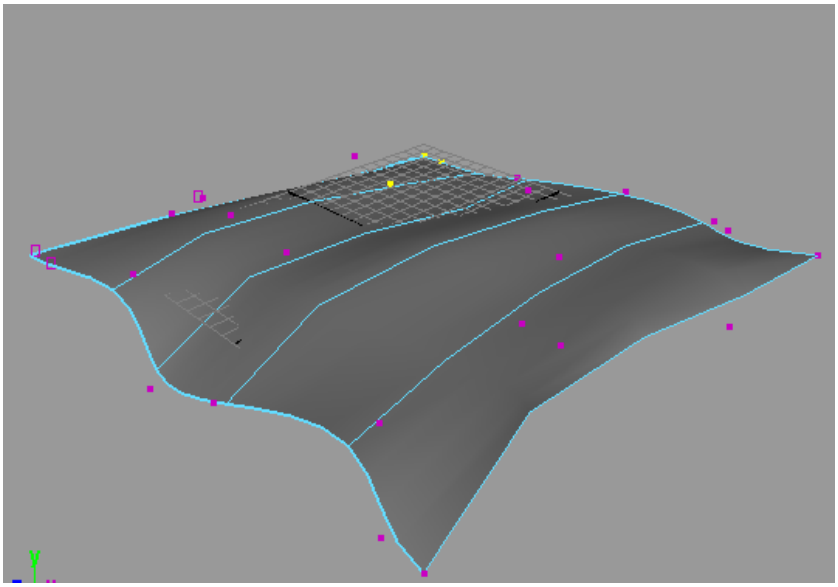


Рис. 1.3. Результат деформації площини

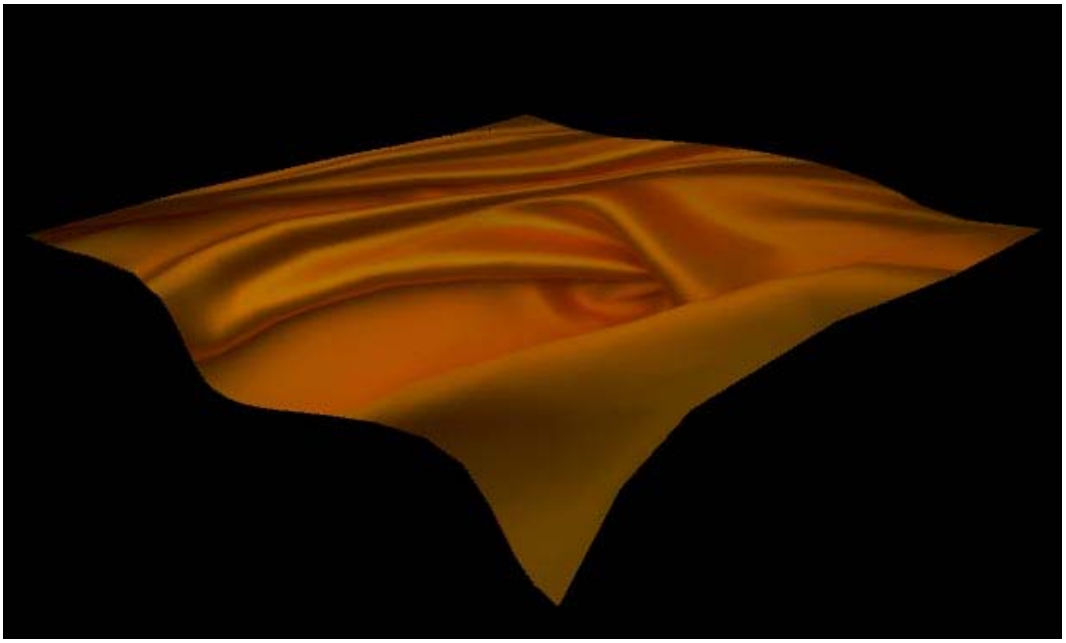


Рис. 1.4. Візуалізація деформованої площини

– накиньте тканину на деяку поверхню, наприклад на звичайний куб. Створіть куб, відкоригуйте в сцені положення обох об'єктів, а після відрегулюйте положення керуючих точок площини тканини так, щоб створювалося відчуття, що тканина спадає з поверхні куба, рис.1.5 - 1.7.

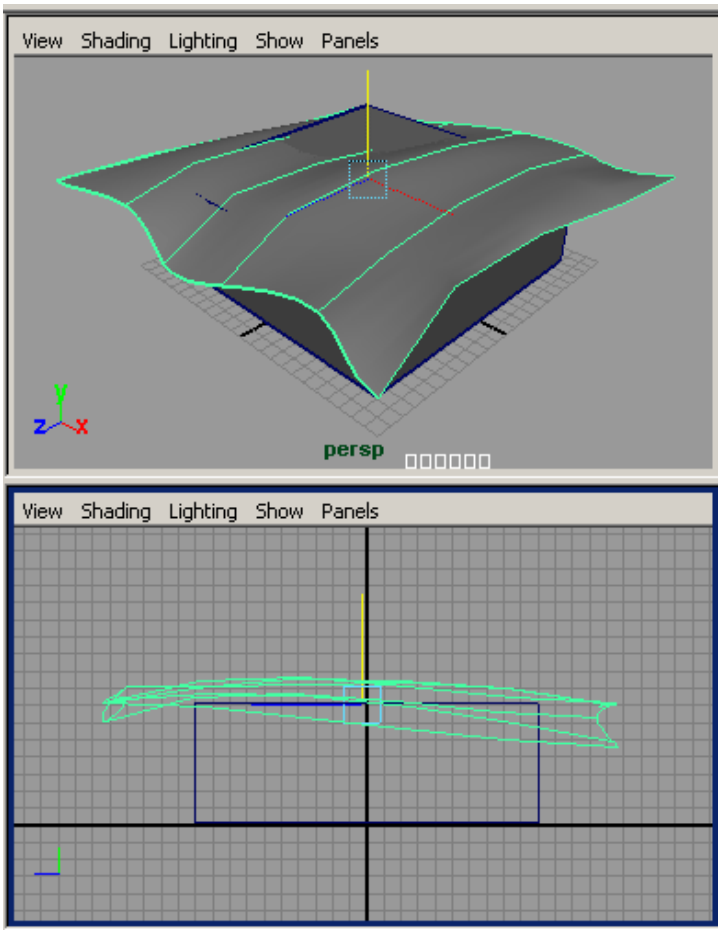


Рис. 1.5. Поява куба

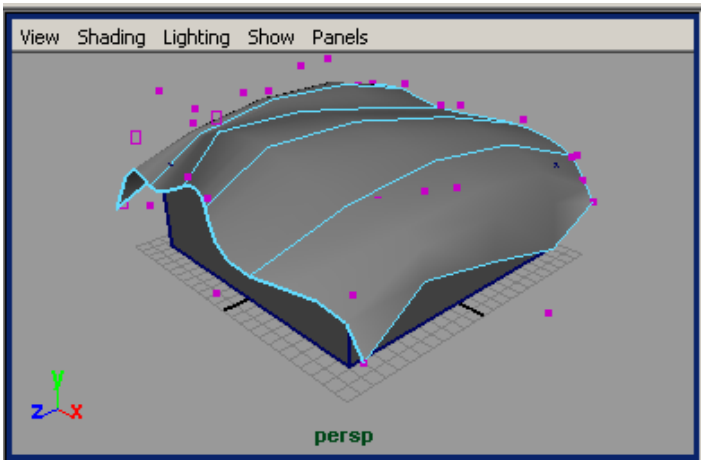


Рис. 1.6. Остаточний вигляд деформованої площини

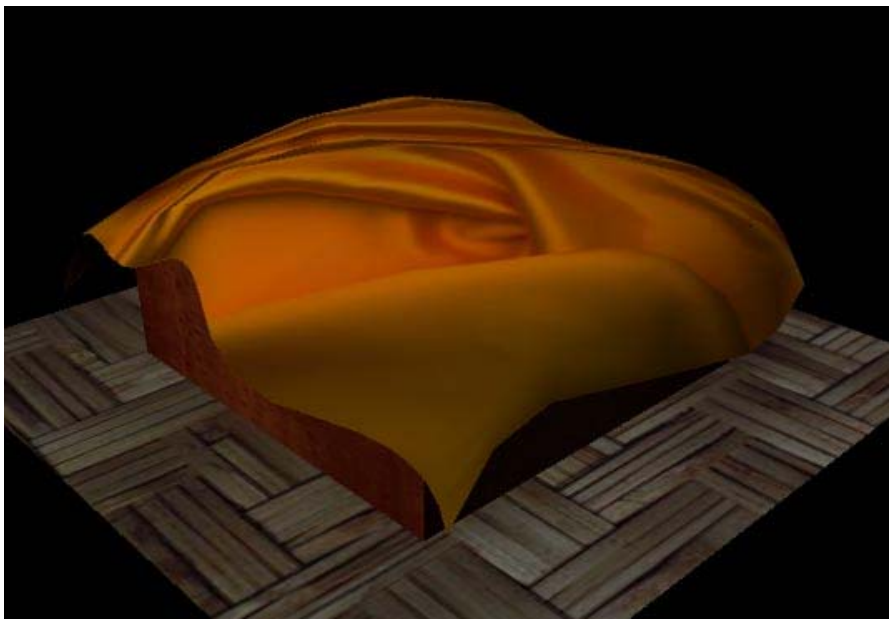


Рис. 1.7. Спадаюча тканина

2. Фужер для шампанського

– створіть нову сцену і сформуйте криву типу **CV Curve Tool** (рис. 2.1). Якщо крива вийшла не зовсім такою, як була задумана, то перейдіть в режим редагування вершин, натиснувши клавішу F9, і відредагуйте положення вершин. Поверніться в режим роботи на рівні об'єктів (клавіша F8), виділіть криву і застосуйте до неї операцію **Revolve** (Обертання). В результаті отримаєте приблизно такий же фужер, як на рис. 2.2.

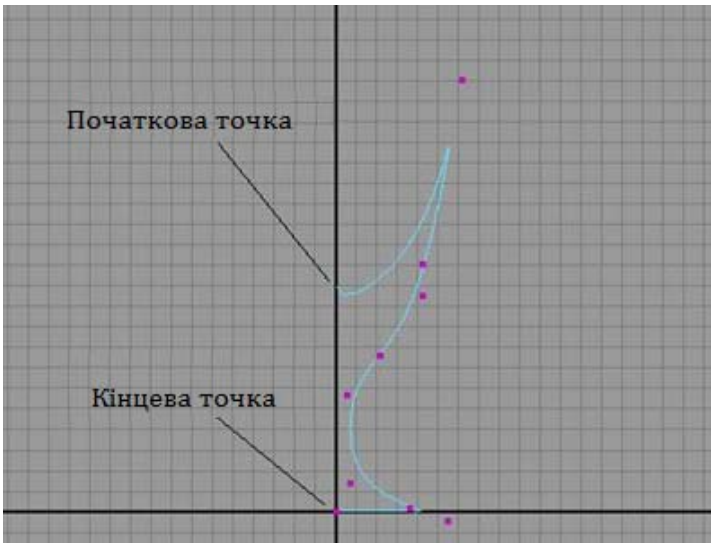


Рис. 2.1. Вихідна крива

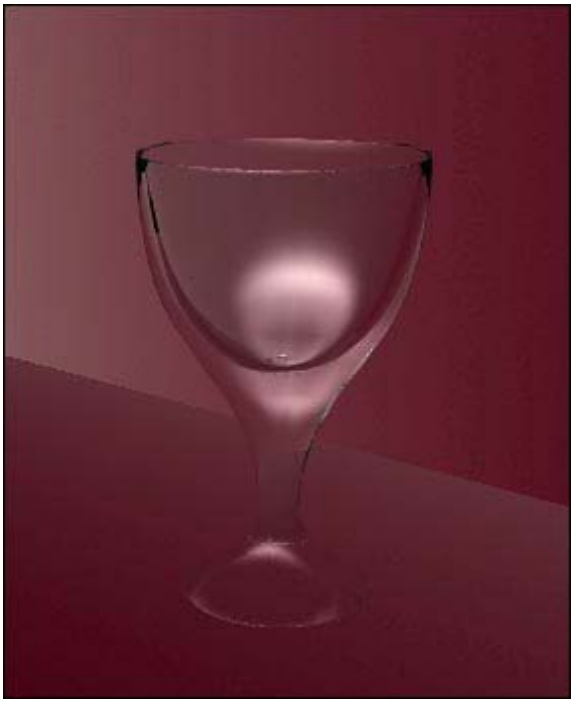


Рис. 2.2. Фужер

Враховуючи, що за замовчуванням історія створення будь-якого об'єкта в Maya зберігається (оскільки кнопка **Construction History** (Історія конструювання), розташована в рядку стану, натиснута), то будь-яка модель обертання може бути відредагована після створення, для цього достатньо змінити вигляд відповідної кривої. Це можна зробити в режимі роботи з підоб'єктами, наприклад перетягнувши окремі точки кривої (керуючі вершини або точки редагування), видаливши деякі з них або створивши нові.

– скопіюйте криву, натисніть клавішу F9 і спробуйте змінити форму кривої, відрегулювавши положення вершин, відповідно до рис. 2.3 - і округлий фужер перетвориться в вузьку чарку (рис. 2.4).

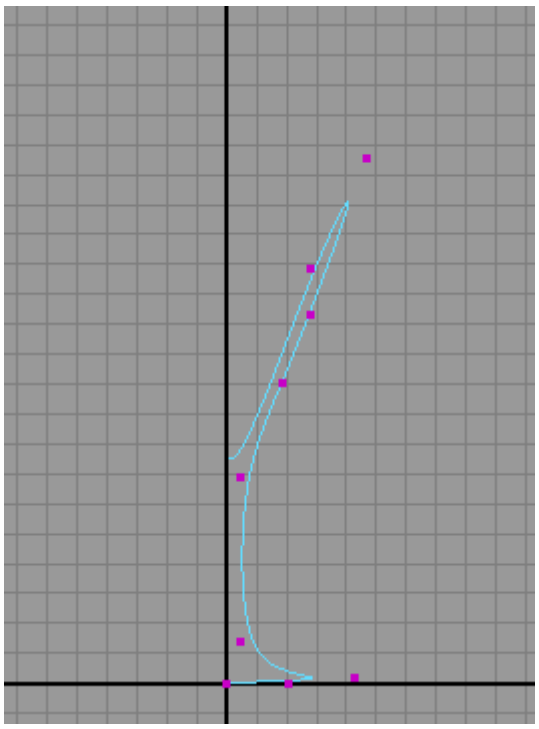


Рис. 2.3. Вихідна крива

Рис. 2.4. Чарка

3. Підставка

Створіть NURBS-криву у вигляді багатокутника (рис. 3.1). В даному випадку зручніше скористатися інструментом **Create=>CV Curve Tool**, але тільки натиснути потрібно на квадратик праворуч від інструменту, щоб встановити для параметра **Degree** варіант **Linear** - рис. 3.2. Виділіть криву, зробіть її замкнутою, застосувавши команду **Open/Close Curves** (Відкрити/Закрити криву), а після здійсніть видавлювання поверхні із даної кривої, скориставшись командою **Surfaces=>Bevel Plus** (Поверхні=>Покращений скіс) при параметрах як на рис. 3.3 (не забувайте, що для встановлення параметрів команди необхідно натискати на квадратик, розташований праворуч від команди). В результаті буде створена поверхня з фаскою, перетином якої і буде вихідна крива (рис. 3.4). Можливий вигляд візуалізованої сцени з створеною поверхнею представлено на рис. 3.5.

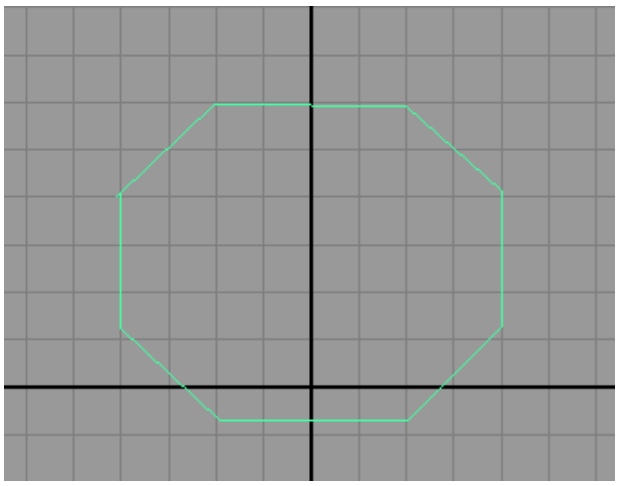


Рис. 3.1. Вихідна крива

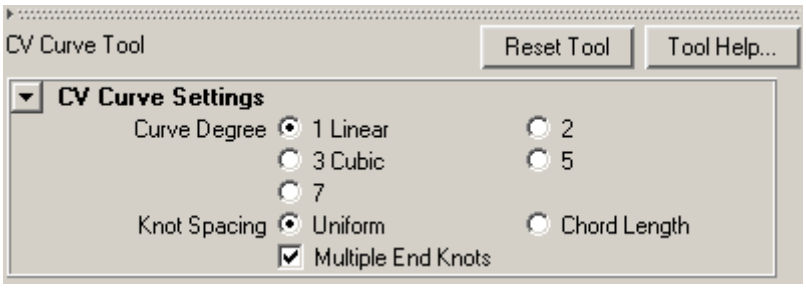


*Рис. 3.2. Визначення параметрів інструмента **Create=>CV Curve Tool***

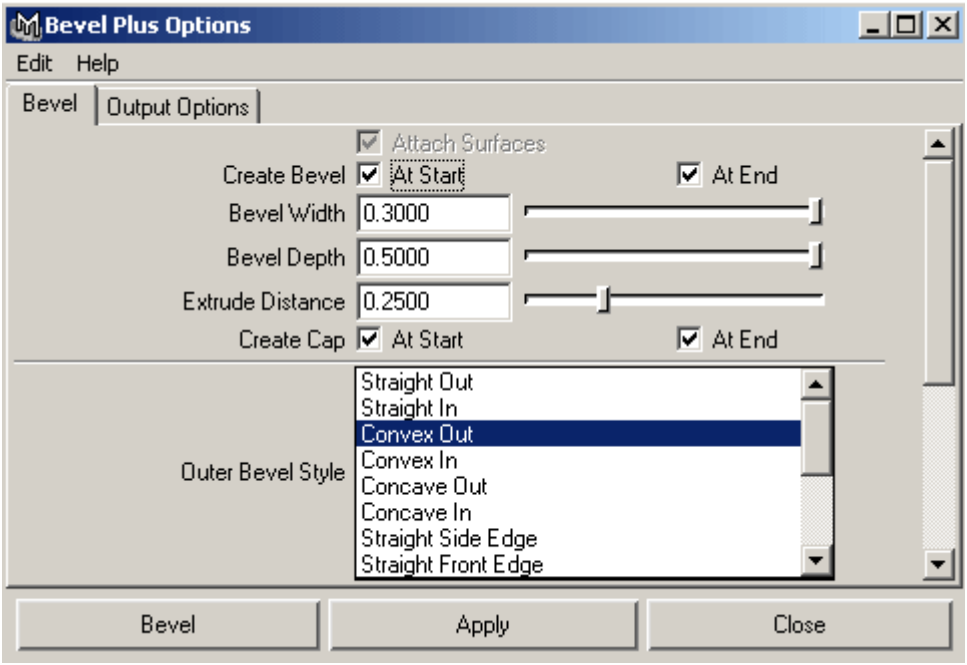


Рис. 3.3. Налаштування параметрів команди ***Bevel Plus***

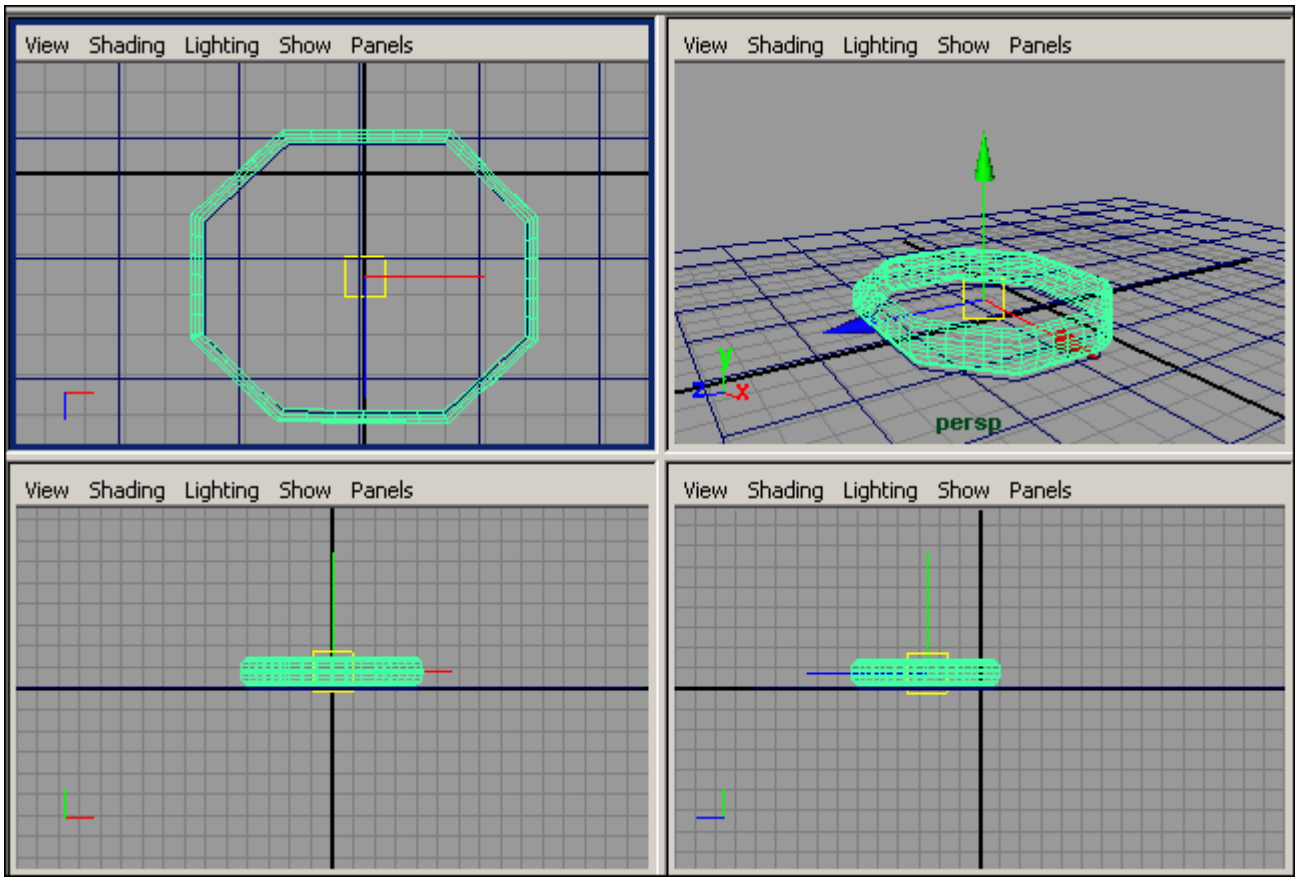


Рис. 3.4. Поява поверхні з фаскою

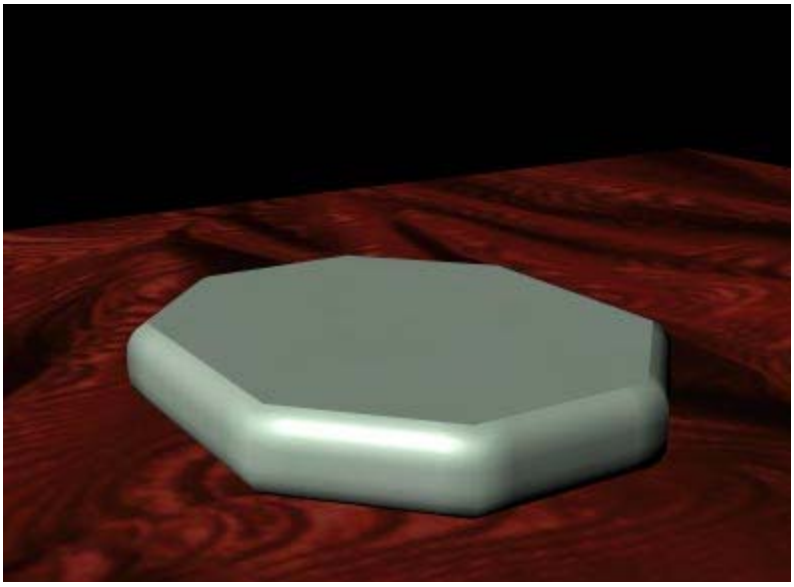


Рис. 3.5. Візуалізована поверхня з фаскою

4. Покрівельний лист

Візьміть в якості кривої-профілю криву, створену інструментом **CV Curve Tool**, а в якості кривої-шляху - відрізок прямої, отриманий інструментом **EP Curve Tool**, попередньо встановивши для параметра **Curve Degree** варіант **1 Linear** (рис. 4.1). Виділіть криву, а після, утримуючи клавішу Shift - відрізок прямої і створіть Extrude-об’єкт (рис. 4.2), який після накладання матеріалу може виглядати приблизно так, як показано на рис. 4.3.

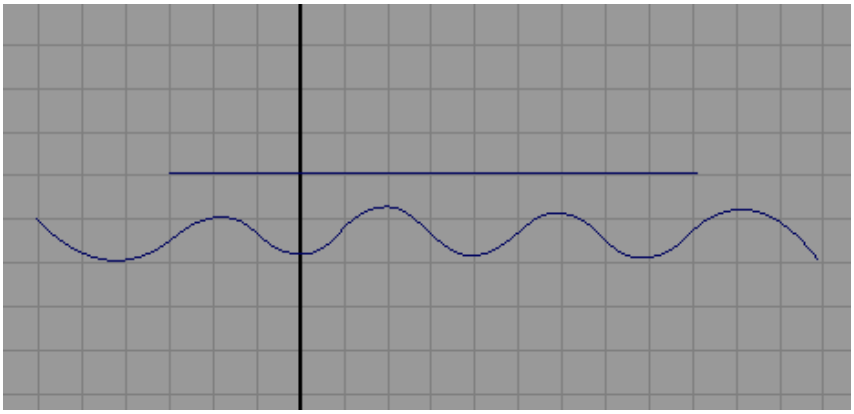


Рис. 4.1. Вихідні криві

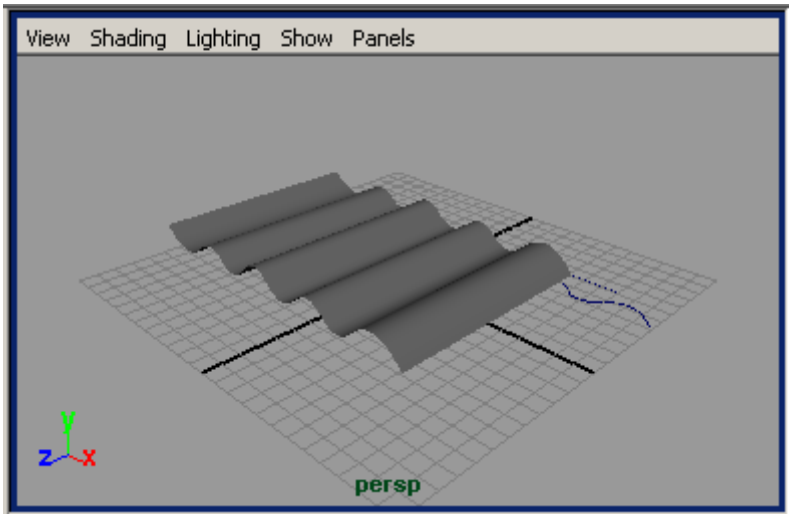


Рис. 4.2. Поява об'єкта, отриманого в результаті операції ***Extrude***

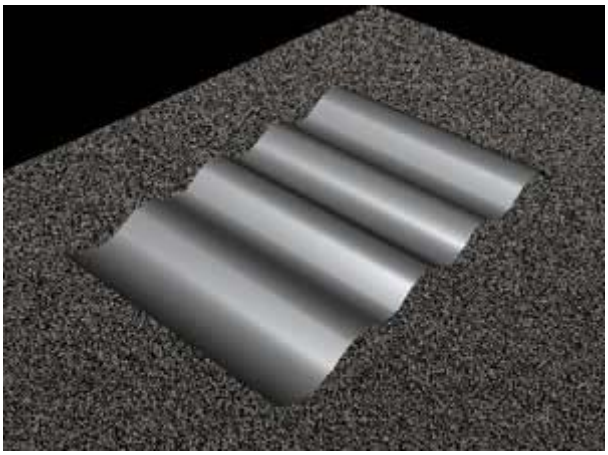


Рис. 4.3. Покрівельний лист

5. Спадаюча складками тканина

Підготуйте два криволінійних контура з великою кількістю вузлів і згенеруйте на їх основі лофт-об'єкт. Створіть криві, застосувавши команду **Create**=>**EP Curve Tool** (Створити=>Крива по точках редагування), - рис. 5.1, виділіть їх, натисніть на квадратик праворуч від команди **Surfaces**=>**Loft** (Поверхні=>Лофтинг), встановіть для параметра **Degree** варіант **Cubic** (оскільки складки повинні бути гладкими) і створіть лофт-об'єкт (рис. 5.2). Накладіть відповідний матеріал, враховуючи структуру тканини, і проведіть рендеринг (рис. 5.3).

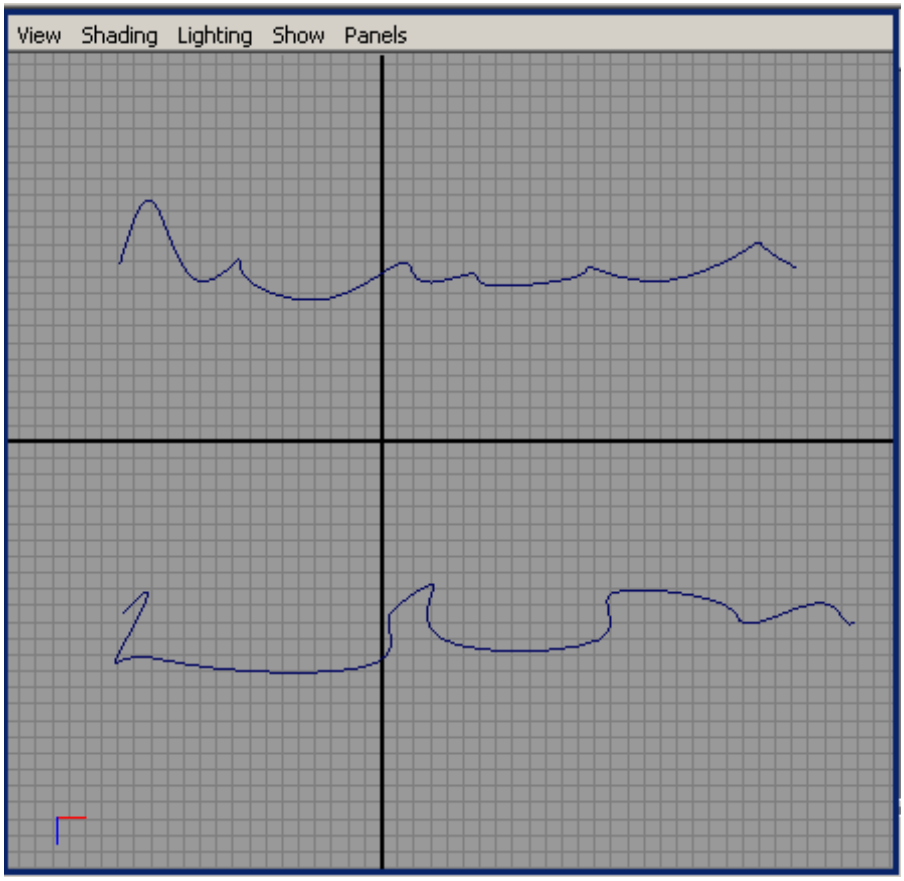


Рис. 5.1. Вихідні елементи для loft-об’єкта

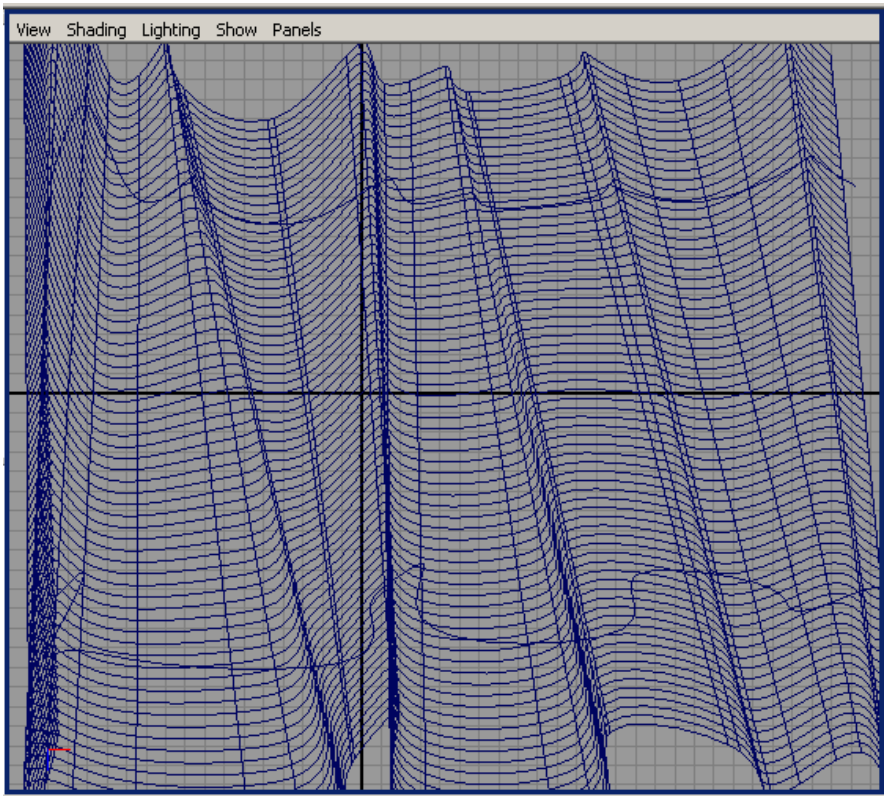


Рис. 5.2. Спадаюча складками тканина

Рис. 5.3. Спадаюча складками тканина

6. Прикраса для ялинки

Створіть кулю і циліндр із групи NURBS Primitives (рис. 6.1) і у вікні каналів збільште для них кількість вертикальних і горизонтальних ізопарм. Виділіть кулю, натисніть праву кнопку і перейдіть в режим виділення ізопарм, клацнувши на команді **Isoparm**. Виділіть верхню ізопарму кулі, натисніть на циліндр, утримуючи клавішу Shift, перейдіть в режим виділення ізопарм і виділіть нижню ізопарму циліндра - це будуть базові криві для проведення лофтингу (рис. 6.2). Застосуйте команду **Surfaces => Loft** (Поверхні => Лофтинг) - куля і циліндр виявляться пов'язаними проміжною поверхнею (рис. 6.3). Налаштуйте розміри циліндра і положення його нижньої ізопарми так, щоб домогтися бажаного виду кулі. Створіть дугу по двом базовим точкам, застосувавши команду **Create => Arc Tools => Two Point Circular ARC** і відрізок прямої (інструмент **EP Curve Tool**), - рис. 6.4. Тепер з дуги і відрізка за допомогою операції **Extrude** змоделюйте дугоподібний фрагмент кріплення. Після закінчення регулювання положення об'єктів сцени, призначте їм матеріали і візуалізуйте сцену, можливий вид якої представлено на рис. 6.5.

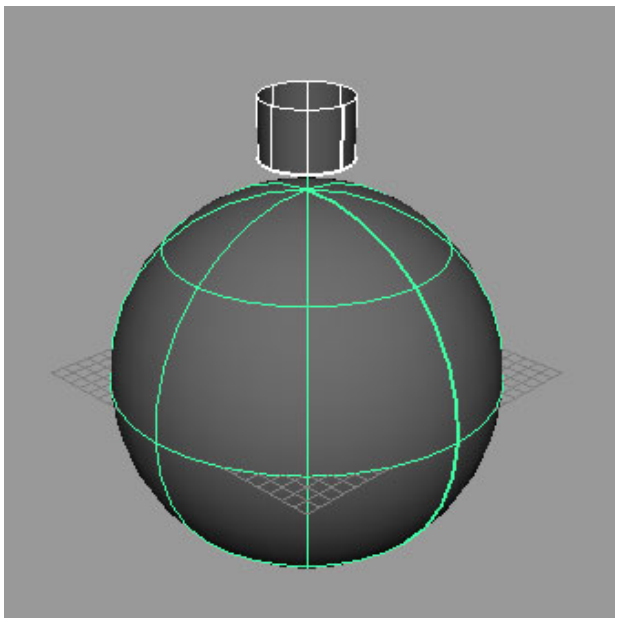


Рис. 6.1. Вихідні об’єкти

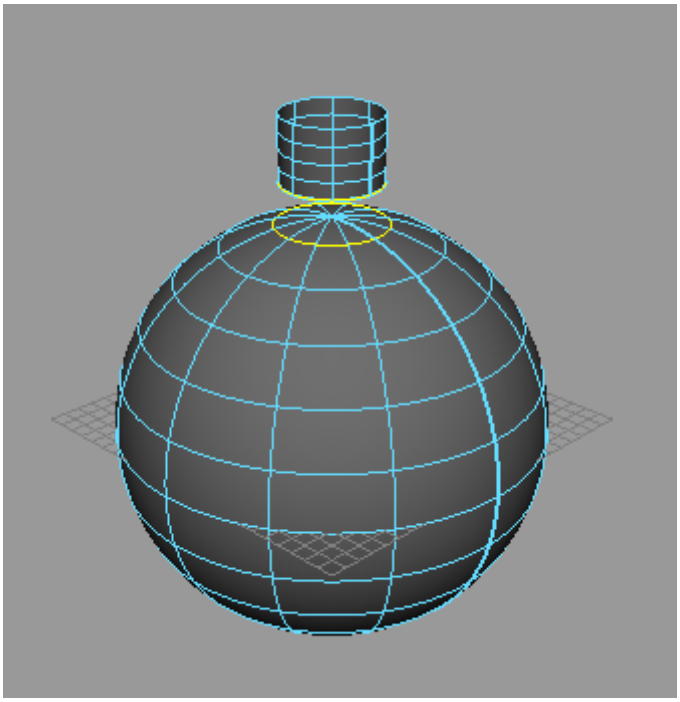


Рис. 6.2. Виділення ізопарм для операції ***Loft***

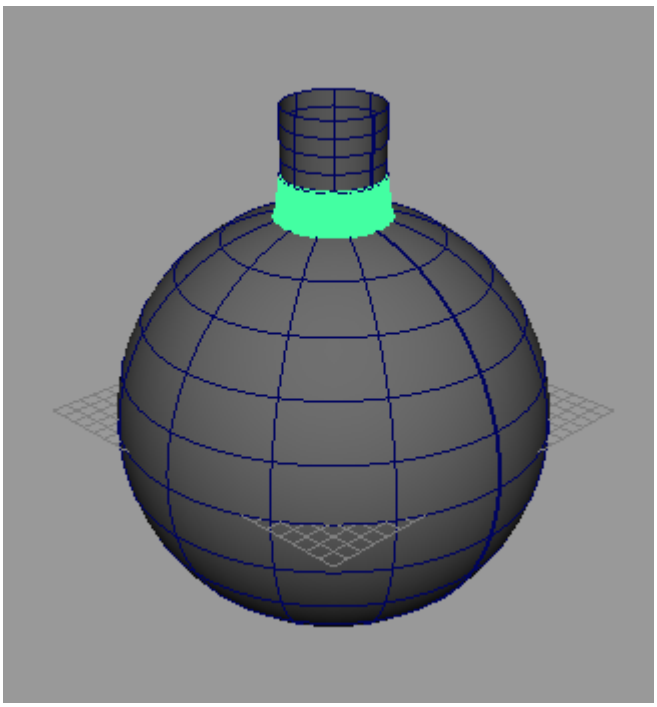


Рис. 6.3. Поява лофт-об’єкта

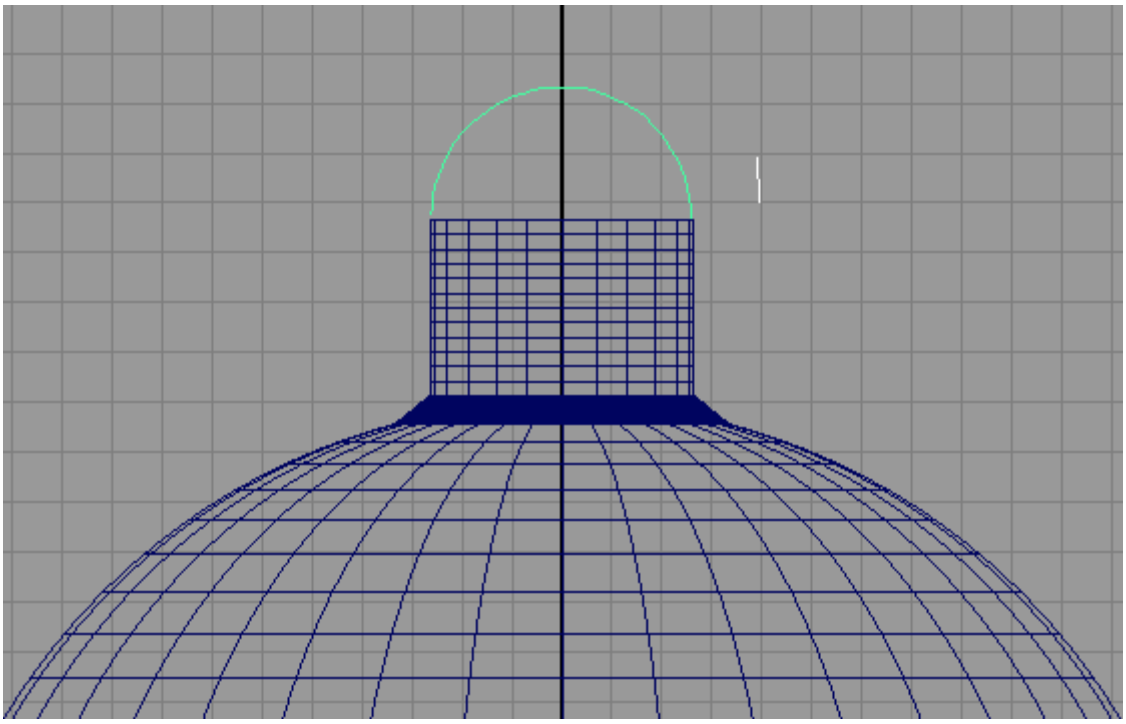


Рис. 6.4. Поява дуги і відрізка

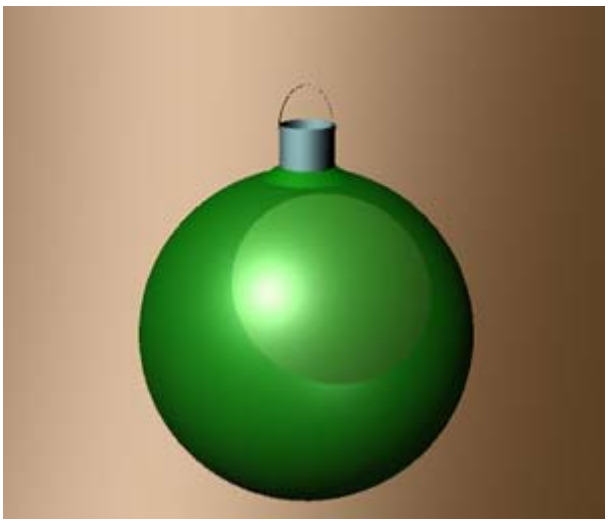


Рис. 6.5. Прикраса для ялинки

7. Дах для будиночка

Створіть NURBS-прямокутник, застосувавши команду **Create => NURBS Primitives => Square**, і проведіть по його середині пряму лінію інструментом **EP Curve Tool** (рис. 7.1). Зніміть виділення з кривих, відкрийте в меню **Surfaces** команду **Birail => Birail 3 Tool**, послідовно вкажіть три профільні криві, натисніть клавішу Enter і виберіть дві напряючі. З'явиться плоска поверхня (рис. 7.2). З огляду на, що число керуючих вершин на вихідних кривих при даному варіанті їх формування мінімально (а тому при спробі їх переміщення поверхня зникне), збільште їх кількість. Для цього виділіть всі криві і застосуєте команду **Edit Curves => Rebuild Curve** (Редагувати криві => Перебудувати криву), вказавши у вікні її параметрів необхідне число вершин - в нашому випадку 20 (рис. 7.3). На центральній профільній кривій виділіть середні вершини і перемістіть їх вгору так, щоб поверхня прийняла бажану криволінійну форму (рис. 7.4). Отриману поверхню можна використовувати, наприклад, в якості даху для будиночка (рис. 7.5).

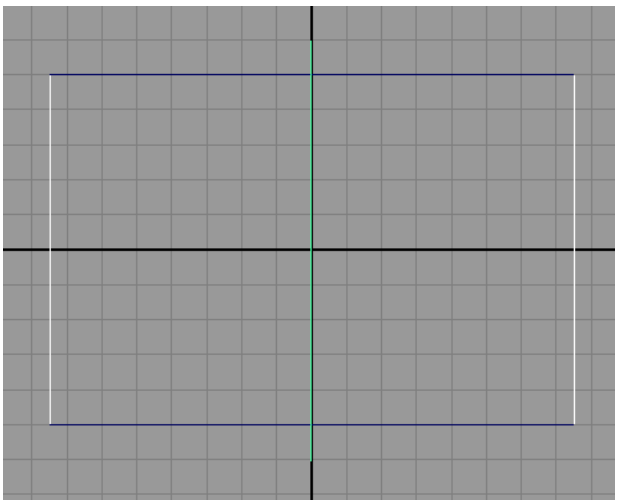


Рис. 7.1. Криві для Birail-об’єкта — профільні криві виділені

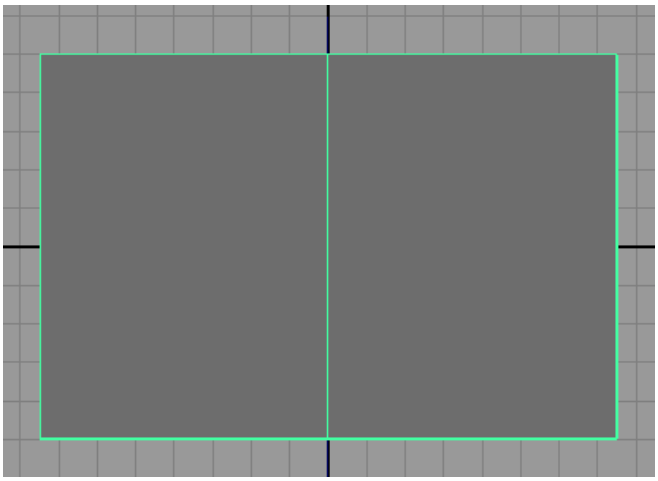


Рис. 7.2. Плоска Birail-поверхня

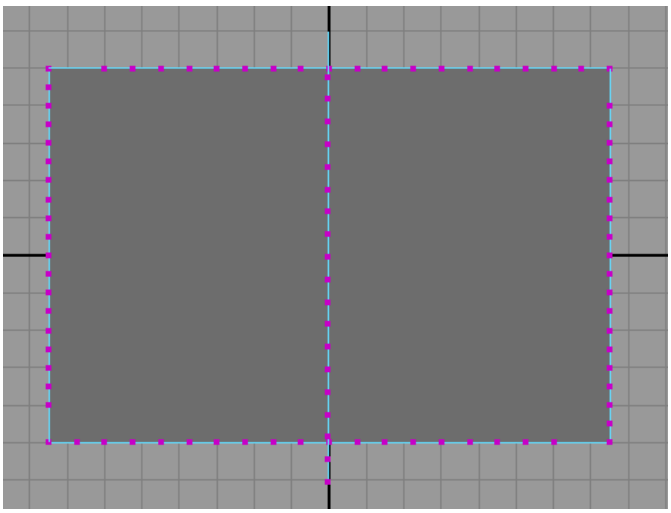


Рис. 7.3. Результат збільшення кількості керуючих вершин

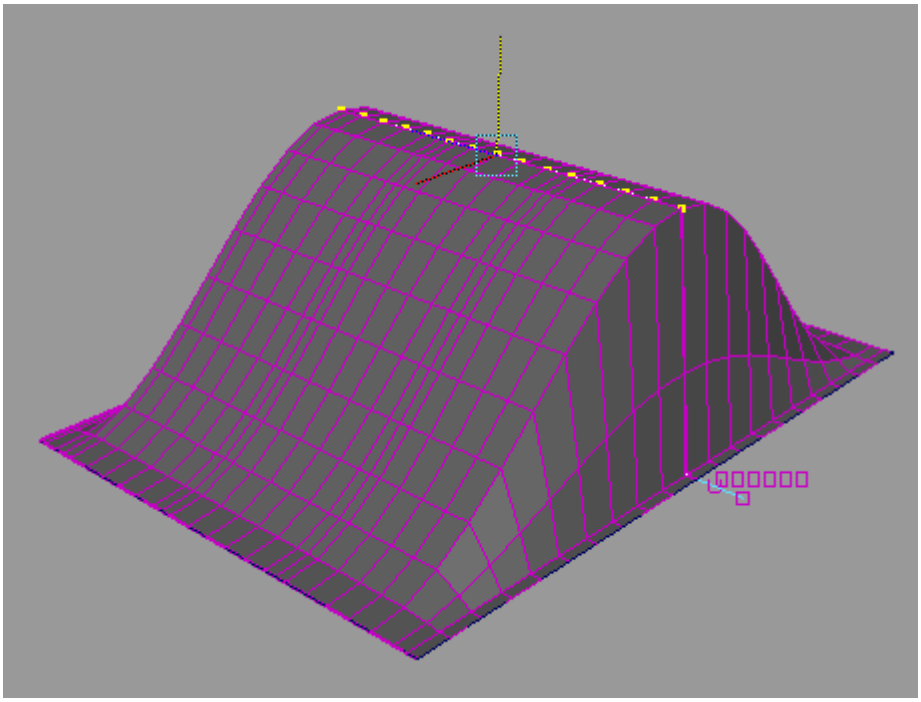


Рис. 7.4. Об’ємна Birail-поверхня

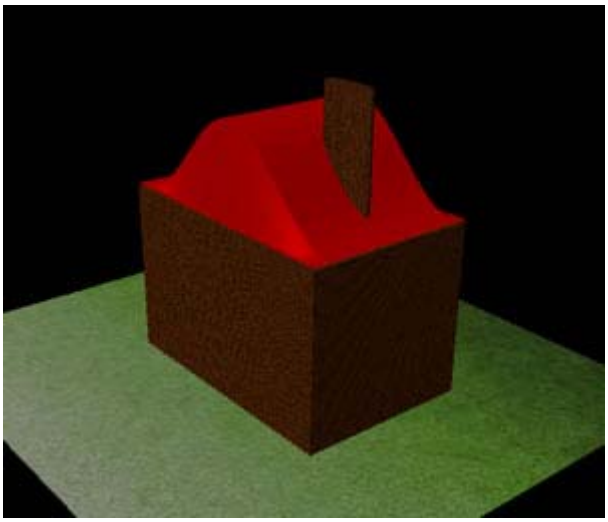


Рис. 7.5. Будинок з дахом

8. Булижники

Створіть NURBS-сферу, в вікні каналів **Channel Box** встановіть розмір рівним 5 (параметри **Scale X, Scale Y** і **Scale Z**) і збільште кількість вертикальних (**Sections**) і горизонтальних ізопарм (**Spans**) до 12. Виділіть сферу і клацніть на квадратику праворуч від команди **Edit NURBS => Sculpt Surface Tool** (Редагування NURBS-поверхонь => Створення рельєфу). Встановіть режим роботи **Pull**, збільште максимальний радіус (**Radius U**) і максимальне зміщення (**Max Displacement**) і трохи витягніть сферу в одному з напрямків, переміщаючи мишку при натиснутій лівій клавіші (рис. 8.1). Перейдіть з режим **Push**, зменшіть глибину впливу інструменту і поекспериментуйте з формуванням вм'ятин, потім знову поверніться в режим **Pull** і остаточно доопрацюйте об'єкт, щоб він дійсно став нагадувати камінь (рис. 8.2). Створіть другу сферу і перетворіть її в булижник аналогічним чином, розмістіть обидва булижники на площині (рис. 8.3), а потім призначте їм відповідні матеріали - можливий результат представлено на рис. 8.4.

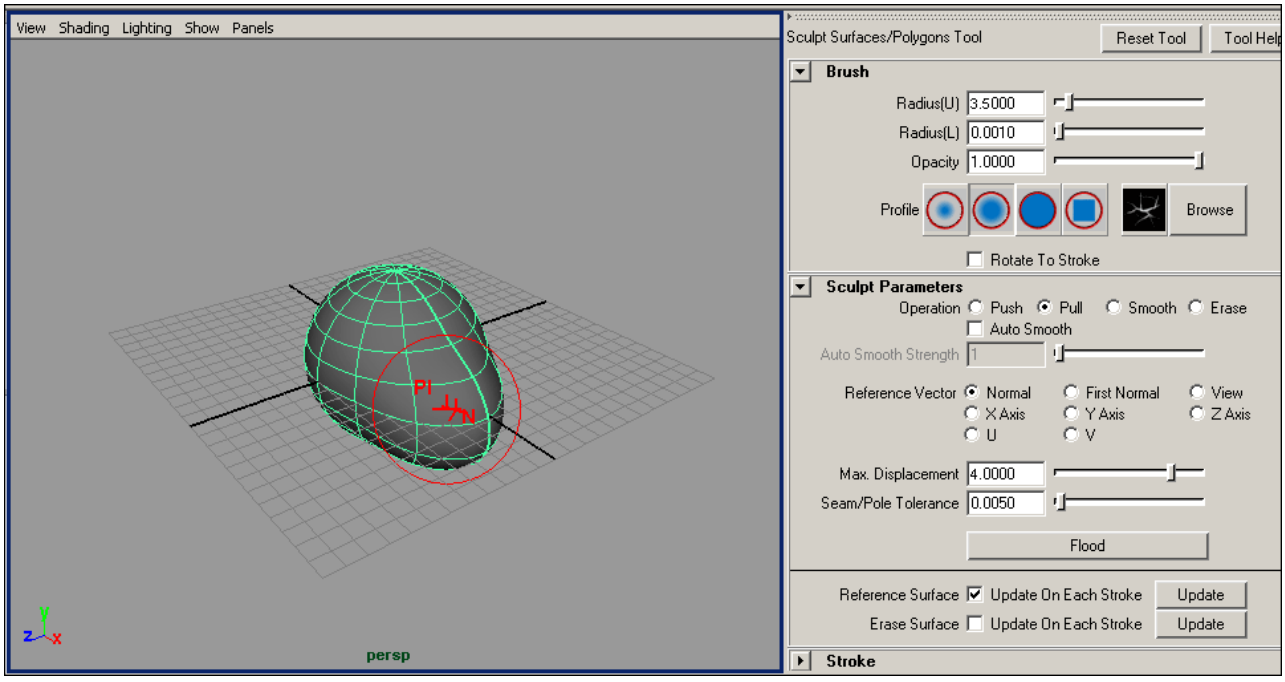


Рис. 8.1. Витягування сфери

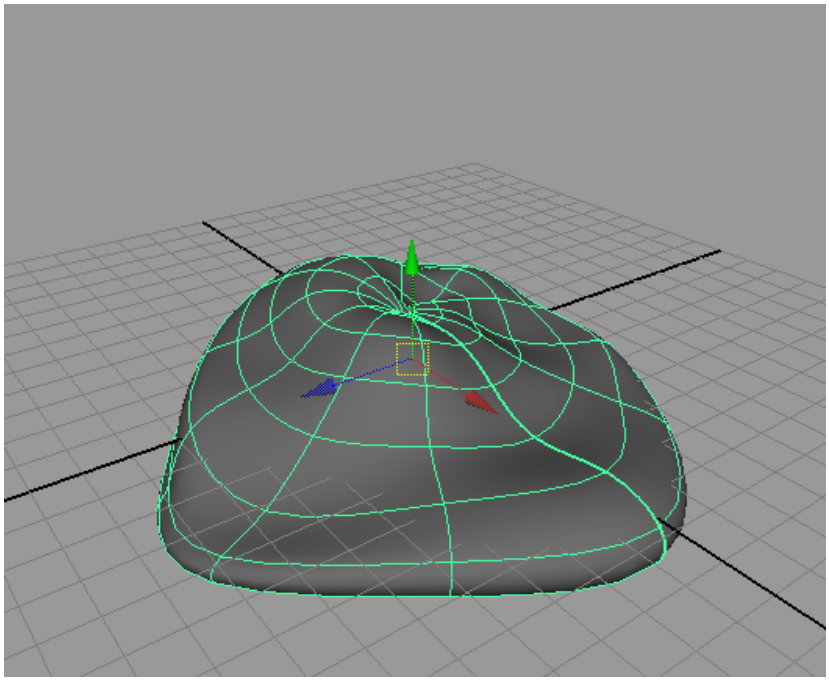


Рис. 8.2. Можливий варіант першого булижника

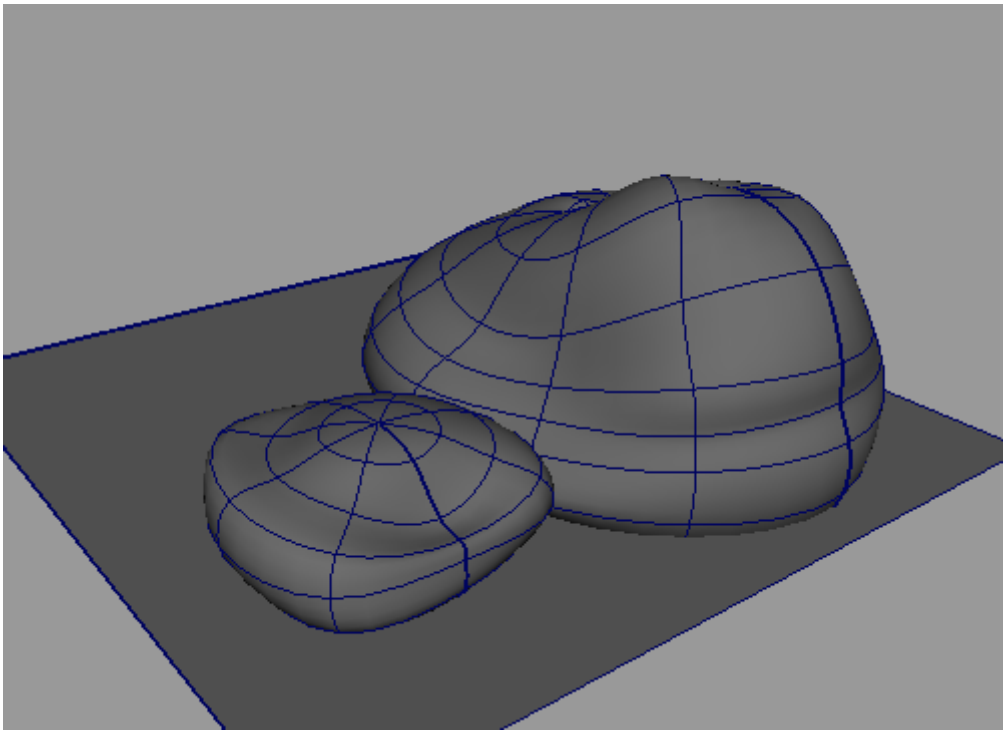


Рис. 8.3. Поява другого булижника на площині

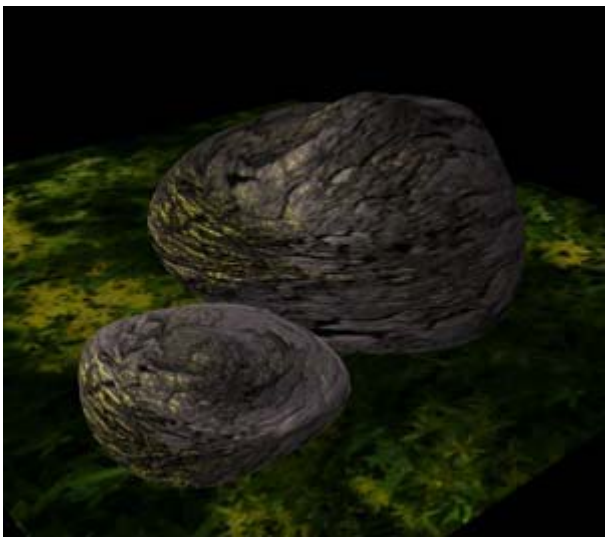


Рис. 8.4. Булижники

9. Скелясте узбережжя

Основою ландшафту стануть дві NURBS-площини і NURBS-сфера. Для початку створіть площини (обидві з 30 вертикальними і горизонтальними ізопармами) і розмістіть одну над іншою (рис. 9.1) - з верхньої будемо генерувати гірський ландшафт, а нижню перетворимо в водну поверхню.

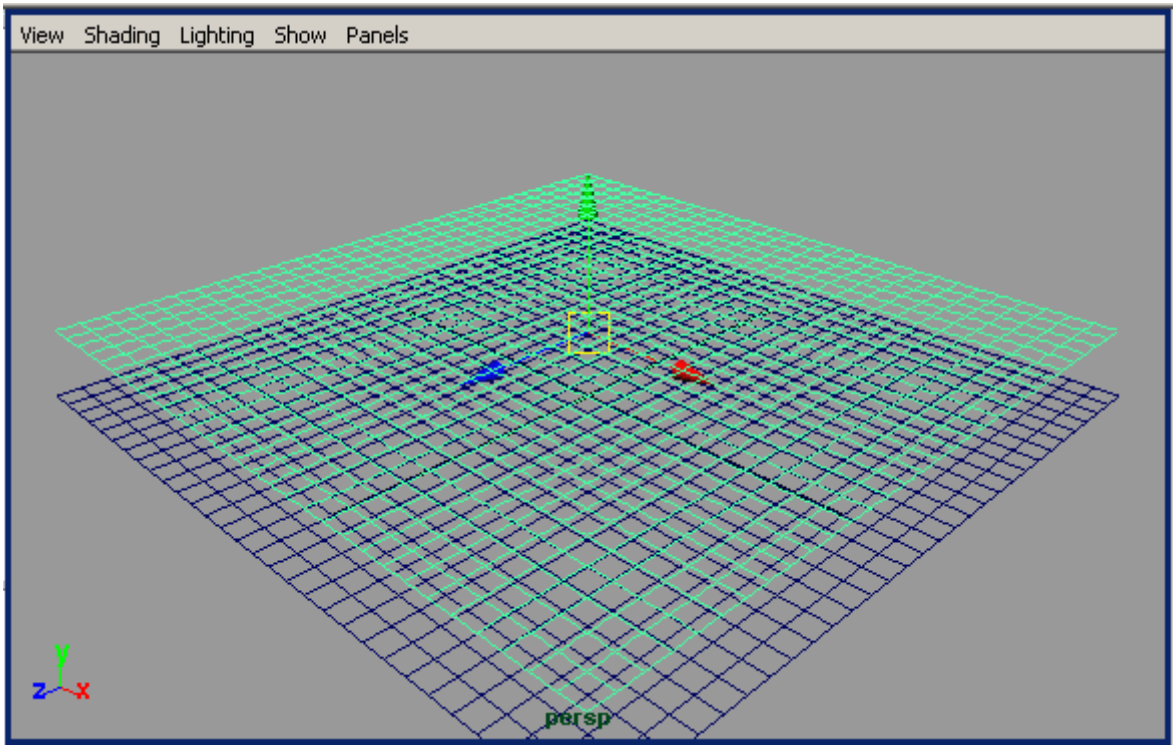


Рис. 9.1. Вихідні площини

Виділіть верхню площину, відкрийте вікно **Sculpt Surface**, встановіть режим **Pull** з великим значенням параметрів **Radius U** і **Max Displacement**, завантажте кисть **Follage** і змоделюйте перший скелястий масив (рис. 9.2), додайте в сцену ще один-два масиви приблизно того ж розміру (рис. 9.3). Зменшіть значення максимального радіусу і максимального зсуву, встановіть кисть **SkinBump** і доповніть картину високою скелею (рис. 9.4). Відмасштабуйте деформовану площину на свій розсуд (рис. 9.5).

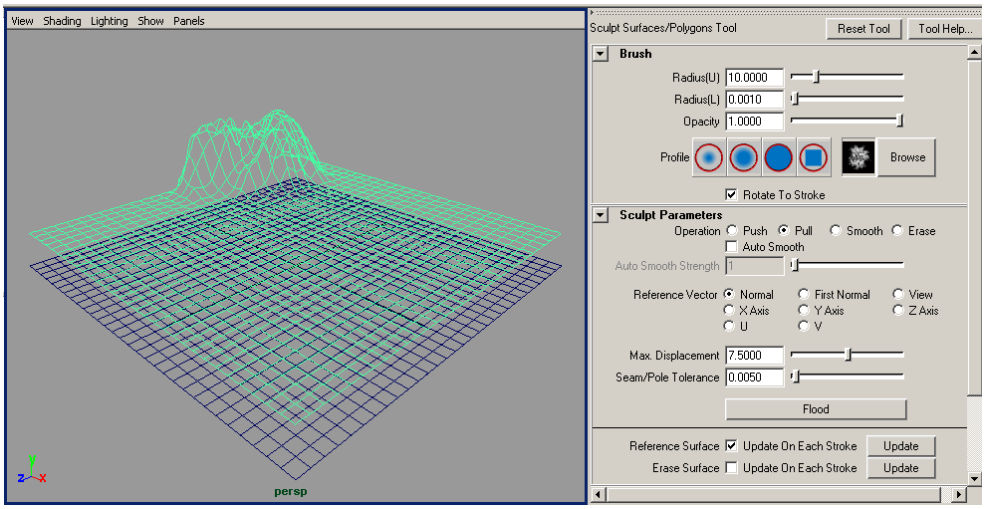


Рис. 9.2. Перший скелястий масив

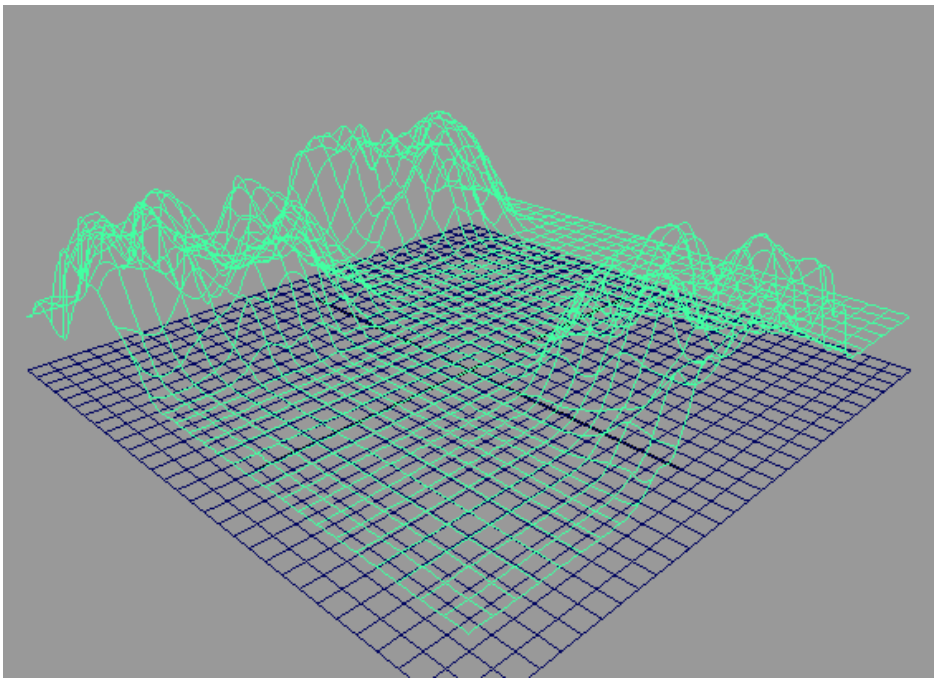


Рис. 9.3. Поява додаткових скелястих масивів

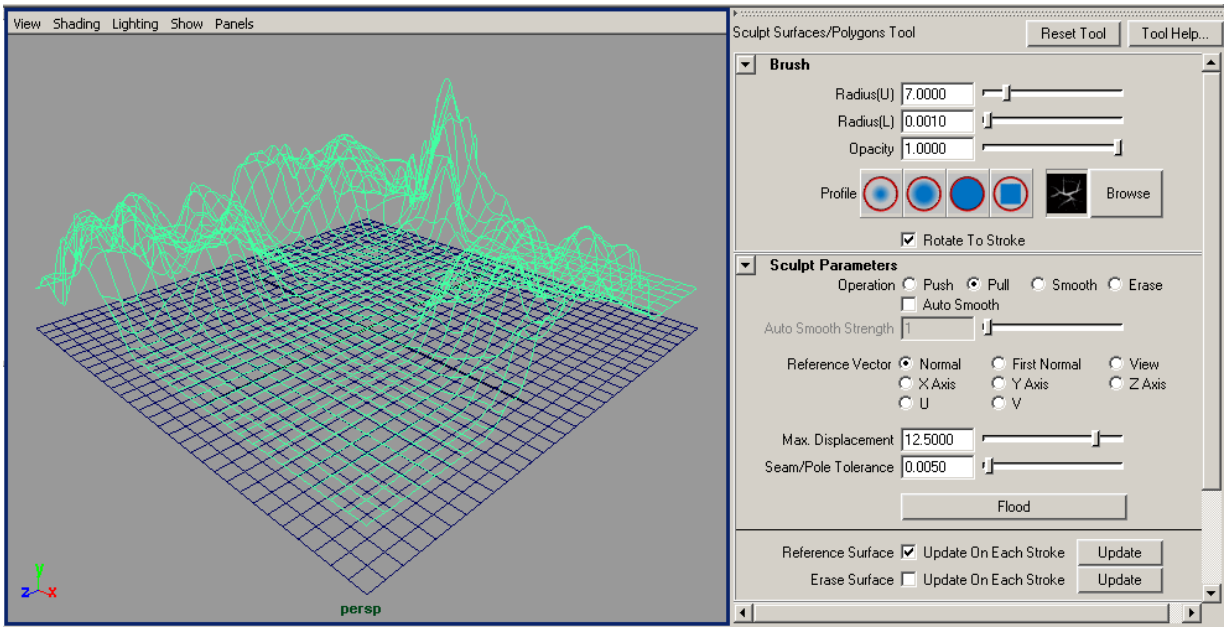


Рис. 9.4. Додавання високої скелі

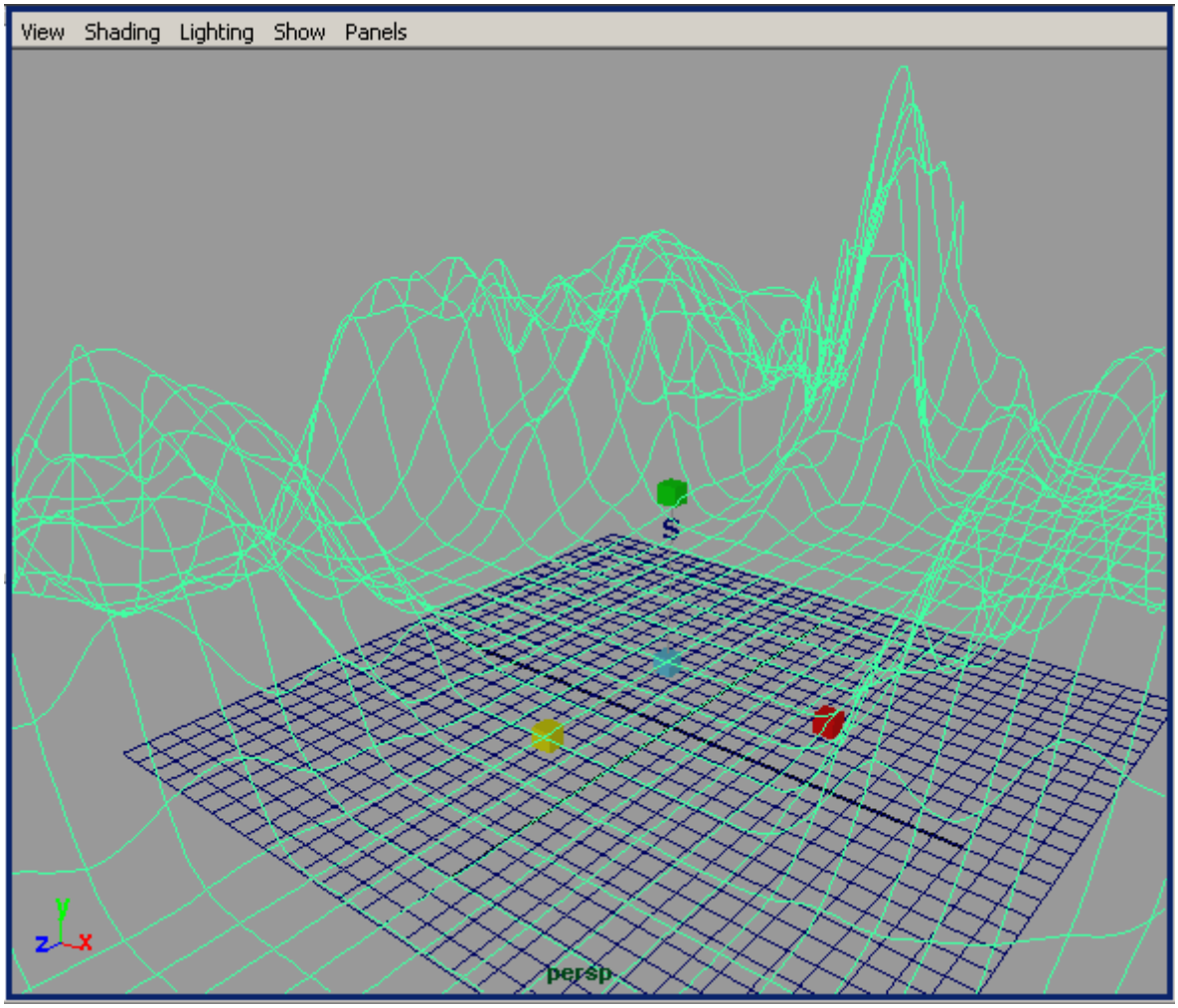


Рис. 9.5. Результат масштабування деформованої площини

Виділіть нижню площину, активуйте інструмент **Sculpt Surfaces Tool**, встановіть максимальний радіус рівним 4, а максимальне зміщення рівним 0,5, завантажте кисть **Splat** і в режимі **Pull** обробіть всю поверхню площини до бажаного ефекту. Аналогічну операцію виконайте в режимі **Push** (рис. 9.6). Розмістіть площини одна відносно одної таким чином, щоб створювалася ілюзія єдиного простору (рис. 9.7), і після рендерингу побачите приблизно такий результат, як показано на рис. 9.8. Тепер створимо небо - для цього краще скористатися NURBS-сферою, за рахунок викривлення поверхні якої простір неба виглядає більш правдоподібно. Створіть сферу дуже великого діаметра і помістіть її на задньому плані сцени (рис. 9.9). І нарешті, залишилося призначити матеріали. Як матеріал для води був використаний Ocean Shader, для неба - Cloud (у обох матеріалів були змінені відтінки кольорів), а для оформлення скель - растрове зображення (рис. 9.10). Отриманий в результаті пейзаж показано на рис. 9.11.

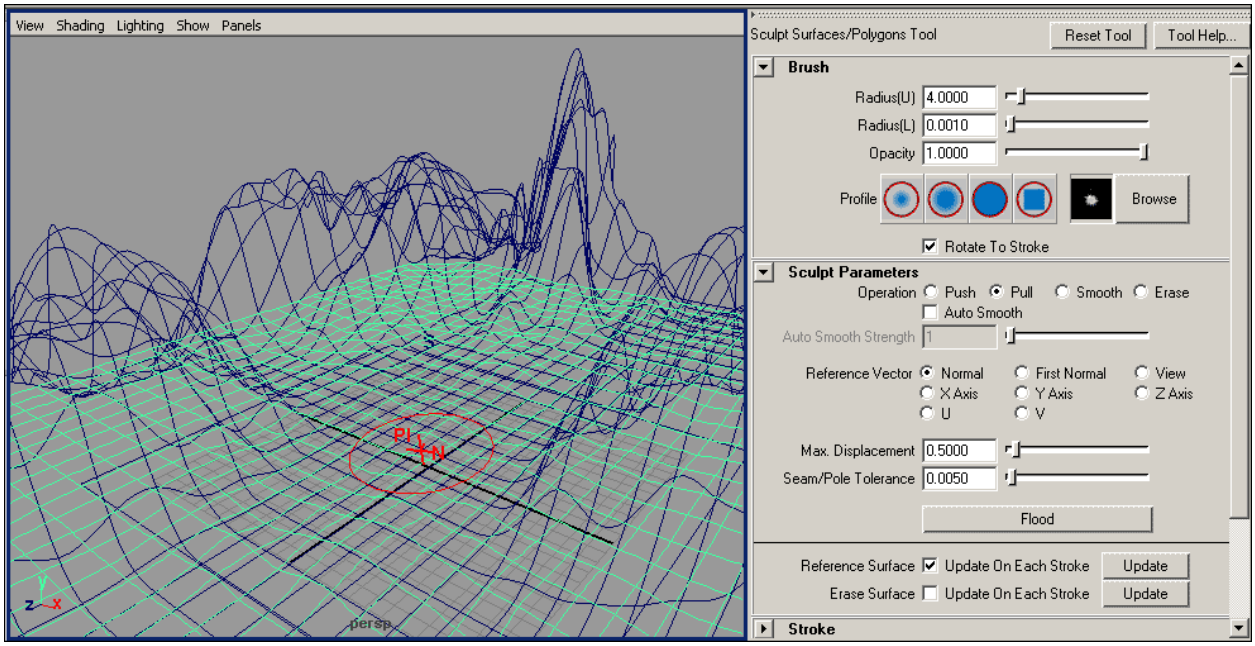


Рис. 9.6. Деформування другої площини

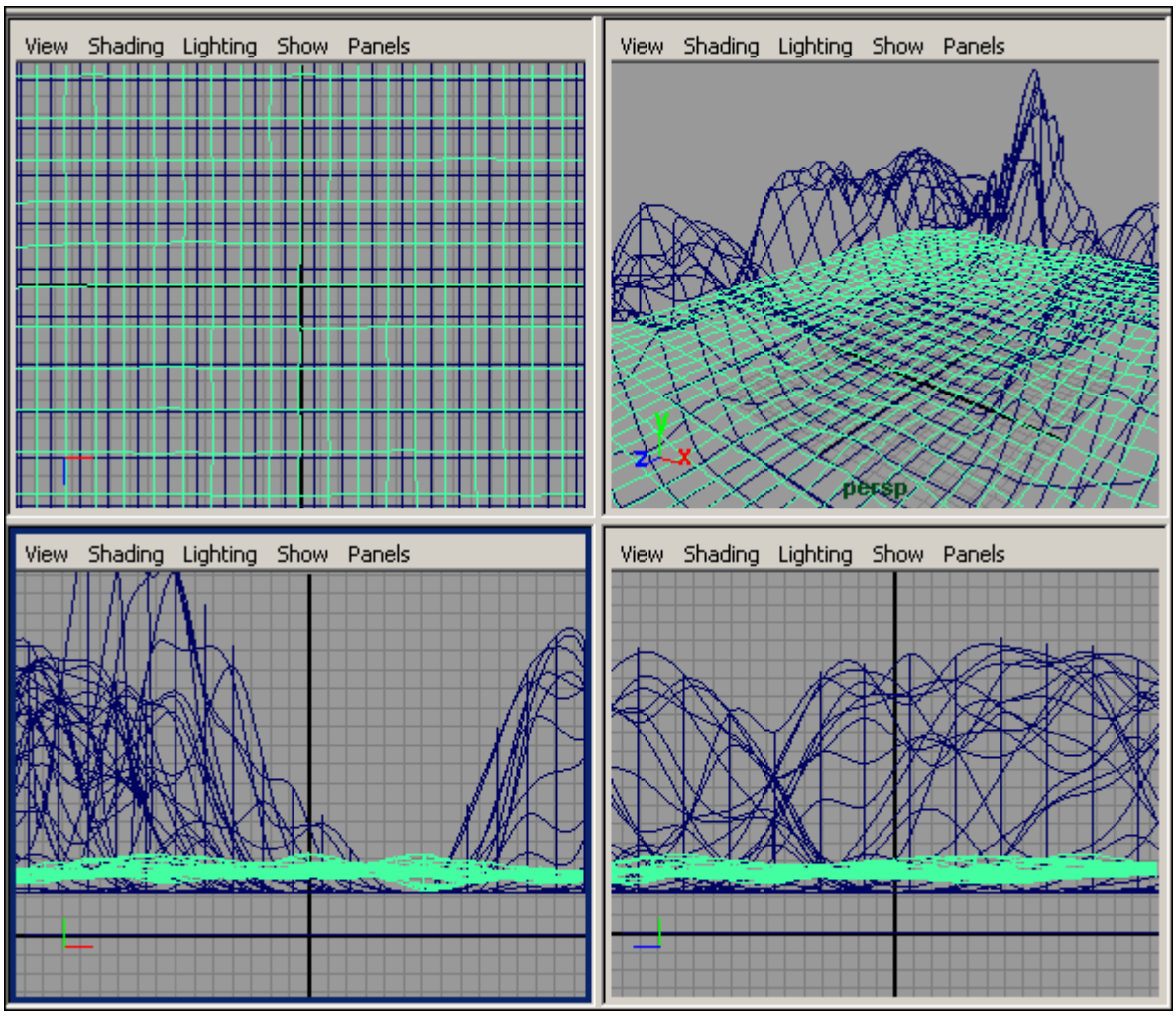


Рис. 9.7. Розміщення площин

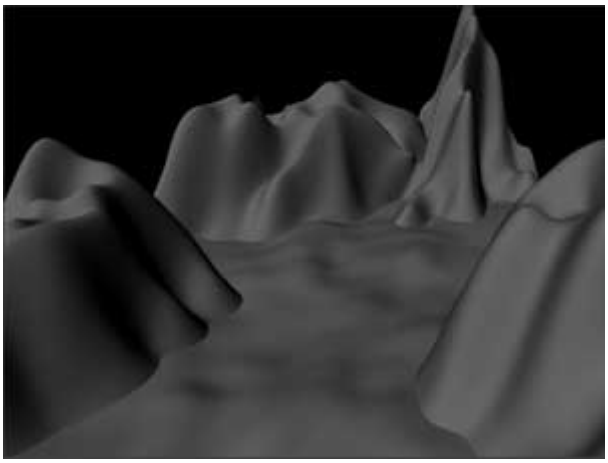


Рис. 9.8. Рендеринг нетекстурованої сцени

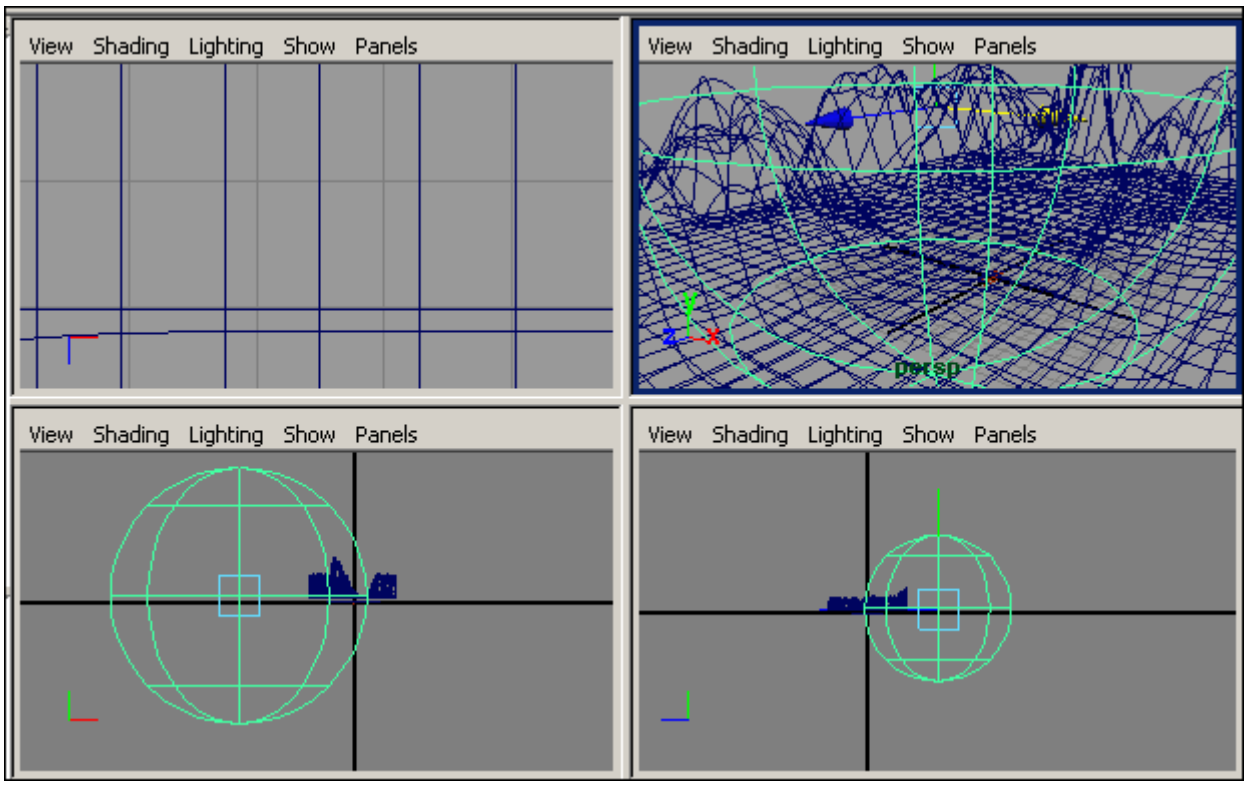


Рис. 9.9. Додання сфери

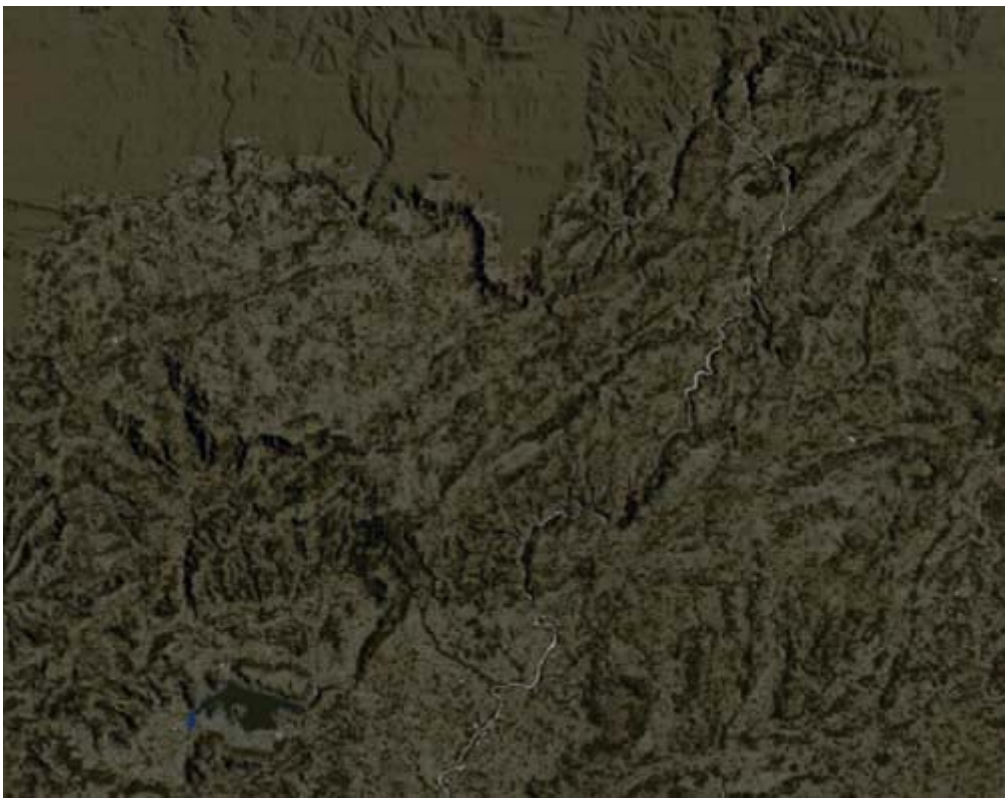


Рис. 9.10. Растрове зображення

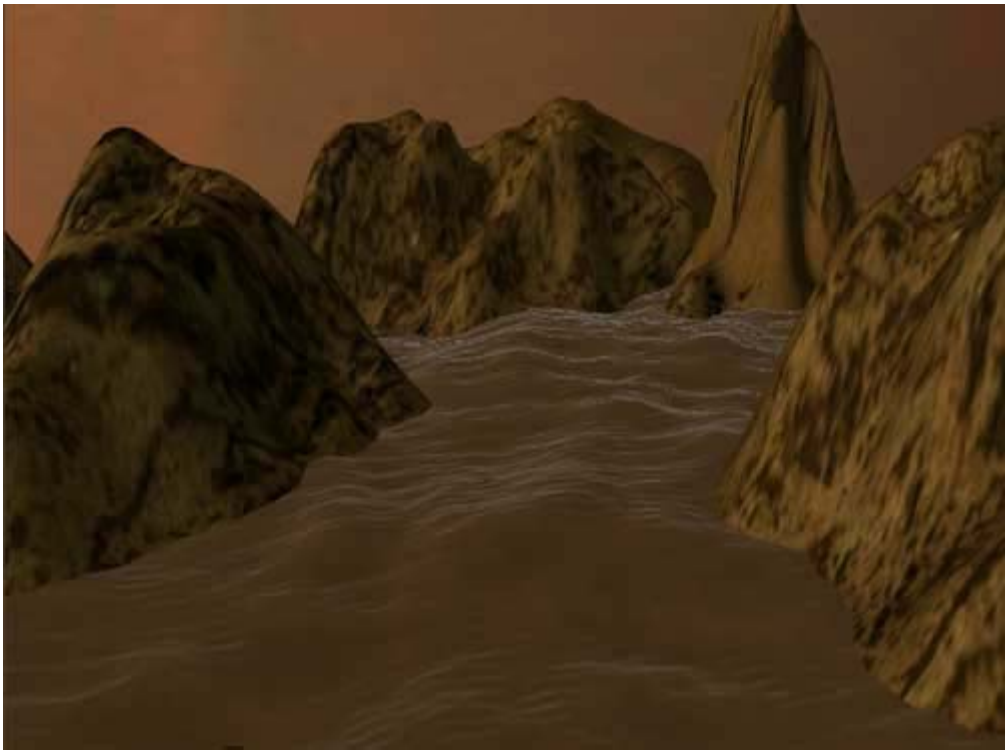


Рис. 9.11. Скелясте узбережжя

Список літератури до розділів

До I, II розділу

1. Власій О.О Комп’ютерна графіка. Обробка растрових зображень: Навчально-методичний посібник / О. О. Власій, О. М. Дудка. – ІваноФранківськ: ДВНЗ «Прикарпатський національний університет імені Василя Стефаника», 2015. – 72 с.

2. Комп’ютерна графіка: AutoCAD: навчальний посібник / М.М. Козяр, Ю.В. Фещук. − Херсон: Грінь Д.С., 2015. − 304 с.

3. Веселовська Г.В., Ходакова В.Є.: Комп'ютерна графіка: Навч. пос. - К.: Кондор, 2015. - 584 с.

4. Шкіца Л. Є., Корнута О. В., Бекіш І. О., Павлик І. В. Інженерна графіка. Навчальний посібник. – Івано-Франківськ, 2015. – 301 с.

5. Шкіца Л. Є., Бекіш І. О. Нарисна геометрія, інженерна та комп’ютерна графіка. Електронний курс для дистанційного навчання. - 2017

6. Корнута О. В., Пригоровська Т. О. Інженерна і комп’ютерна графіка: практикум. – Івано-Франківськ: ІФНТУНГ, 2016. - 61 с.

7. Тарас І. П. Комп’ютерна графіка. Навчальний посібник. – Івано-Франківськ, 2017. – 60с.

До III, IV розділу

1. Комп’ютерна графіка : конспект лекцій для студентів усіх форм навчання спеціальностей 122 «Комп’ютерні науки» та 123 «Комп’ютерна інженерія» з курсу «Комп’ютерна графіка» / Укладач: Скиба О.П. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2019. – 88 с.

2. Компьютерная графика. КОМПАС и AutoCAD: учебное пособие / И.П. Конакова, И. И. Пирогова. – Екатеринбург: Изд-во Урал. ун-та, 2015. – 148 с.

3. Жарков, В.А. Visual Basic 2005, DirectX 9.0 и Microsoft Agent в компьютерной графике, мультимедиа и играх (+ CD-ROM) / В.А. Жарков. - М.: Жарков Пресс, 2017. - 717 с.

4. Кэмпбелл, Марк Компьютерная графика / Марк Кэмпбелл. - М.: АСТ, Lingua, Астрель, 2016. - 384 с.

5. Рассел, Джесси Компьютерная графика / Джесси Рассел. - М.: VSD, 2015. - 298 с.

6. Хейфец, Александр Инженерная компьютерная графика. AutoCAD / Александр Хейфец. - М.: БХВ-Петербург, 2015. - 813 с.

До V розділу

1. Кувшинов, Н.С. Инженерная и компьютерная графика (для бакалавров) / Н.С. Кувшинов, Т.Н. Скоцкая. - М.: КноРус, 2017. - 208 с.

2. Максимова, И.А. Приёмы изобразительного языка в современной архитектуре (ручная и компьютерная графика): Учебное пособие / И.А. Максимова, А.Е. Винокурова, А.В. Пивоварова. - М.: Инфра-М, 2018. - 264 с.

3. Немцова, Т.И. Компьютерная графика и Web-дизайн. Практикум. Практикум по информатике: Учебное пособие / Т.И. Немцова, Ю.В. Назарова. - М.: Форум, 2018. - 144 с.

4. Немцова, Т.И. Компьютерная графика и web-дизайн: Уч.пос / Т.И. Немцова, Т.В. Казанкова, А.В. Шнякин и др. - М.: Форум, 2015. - 144 с.

До VI розділу

1. Аббасов, И.Б. Двухмерное и трехмерное моделирование в 3ds MAX / И.Б. Аббасов. - М.: ДМК, 2012. - 176 с.

2. Ганеев, Р.М. 3D-моделирование персонажей в Maya: Учебное пособие для вузов / Р.М. Ганеев. - М.: ГЛТ, 2015. - 284 с.

3. Зеньковский, В. 3D-моделирование на базе Vue xStream: Учебное пособие / В. Зеньковский. - М.: Форум, 2011. - 384 с.

4. Зеньковский, В.А. 3D моделирование на базе Vue xStream: Учебное пособие / В.А. Зеньковский. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013. - 384 с.

5. Инженерная 3D компьютерная графика / А.Л. Хейфец и др. - Москва: Машиностроение, 2016. - 464 с.

6. Миловская О.С. 3ds Max 2017. Дизайн интерьеров и архитектуры, – Питер, 2017, – 416стр.

7. Миловская О.С. 3ds Max 2018 и 2019. Дизайн интерьеров и архитектуры, – Питер, 2018, – 416стр.

8. Максимова, И.А. Приёмы изобразительного языка в современной архитектуре (ручная и компьютерная графика): Учебное пособие / И.А. Максимова, А.Е. Винокурова, А.В. Пивоварова. - М.: Инфра-М, 2018. - 264 с.

9. Немцова, Т.И. Компьютерная графика и Web-дизайн. Практикум. Практикум по информатике: Учебное пособие / Т.И. Немцова, Ю.В. Назарова. - М.: Форум, 2018. - 144 с.

Навчальне видання

О. С. Булгакова,

В. В. Зосімов,

Г. В. Ходякова

Комп'ютерна графіка (2D/3D): теорія

Навчальний посібник

для дистанційної форми навчання

Формат 60×841/16. Ум. друк. арк. 8,7. Тираж 100 пр. Зам. № 684-658.

ВИГОТОВЛЮВАЧ

СПД Румянцева Г. В.

54038, м. Миколаїв, вул. Бузника, 5/1.