УДК61:371.322.7

МОДЕЛІ ІНТЕРАКТИВНОЇ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В МЕДИЦИНІ: ПІДХІД НА ОСНОВІ JAVA3D - ТЕХНОЛОГІЇ. ЧАСТИНА 1

В.П. Марценюк , І.Б. Меленчук

Тернопільський державний медичний університет імені І.Я. Горбачевського
melenchuk_i@mail.ru

У статті досліджуються загальні теоретичні підходи комп'ютерної графіки. Вводиться поняття графу сцени та його основних компонент. Показане практичне використання бібліотеки Java3D на прикладі розробки переглядача .obj-файлів анатомічних зображень.

Ключові слова: віртуальна реальність, Java3D, комп'ютерна графіка.

МОДЕЛИ ИНТЕРАКТИВНОЙ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ В МЕДИЦИНЕ: ПОДХОД НА ОСНОВЕ JAVA3D - ТЕХНОЛОГИИ. ЧАСТЬ 1

В.П. Марценюк, И.Б. Меленчук

Тернопольский государственный медицинский университет имени И.Я. Горбачевского
melenchuk_i@mail.ru

В статье исследуются общие теоретические подходы компьютерной графики. Вводится понятие графа сцены и его основных компонент. Показано практическое использование библиотеки Java3D на примере разработки просмотрщика .obj-файлов анатомических изображений.

Ключевые слова: виртуальная реальность, Java3D, компьютерная графика.

MODELS OF INTERACTIVE VIRTUAL REALITY IN MEDICINE: APPROACH ON THE BASIS OF JAVA3D-TECHNOLOGIES. PART 1

V.P. Martsenyuk, I.B. Melenchuk

Ternopil State Medical University by I.Ya. Horbachevsky
melenchuk_i@mail.ru

General theoretical approaches to computer graphics are investigated in the paper. The notion of scene graph and its basic components is introduced. Practical application of Java3D library using example of development of obj-files viewer for anatomical images is shown.

Key words: virtual reality, Java3D, computer grafics.

Вступ. Комп'ютерна графіка вивчає теорію і методики моделювання, обробки і візуалізації графічних об'єктів в комп'ютерах. Головна мета комп'ютерної графіки - побудувати віртуальний світ графічних об'єктів і візуалізувати сцену віртуальної моделі на основі певного споглядання на графічному пристрої, як показано на рисунку 1.

У роботах [1-22] наведено головні програмні засоби, які використовувалися в комп'ютерній графіці - серед них були як низько- (OpenGL, DirectX), так і високорівневі (VRML, 3D Studio Max). Java3D є чи не першим некомерційним Web-інтегрованим продуктом.

Метою даної статті є дослідити основні поняття комп'ютерної графіки у їх застосуванні до віртуальної реальності, створюваної на основі API-бібліотеки Java 3D.

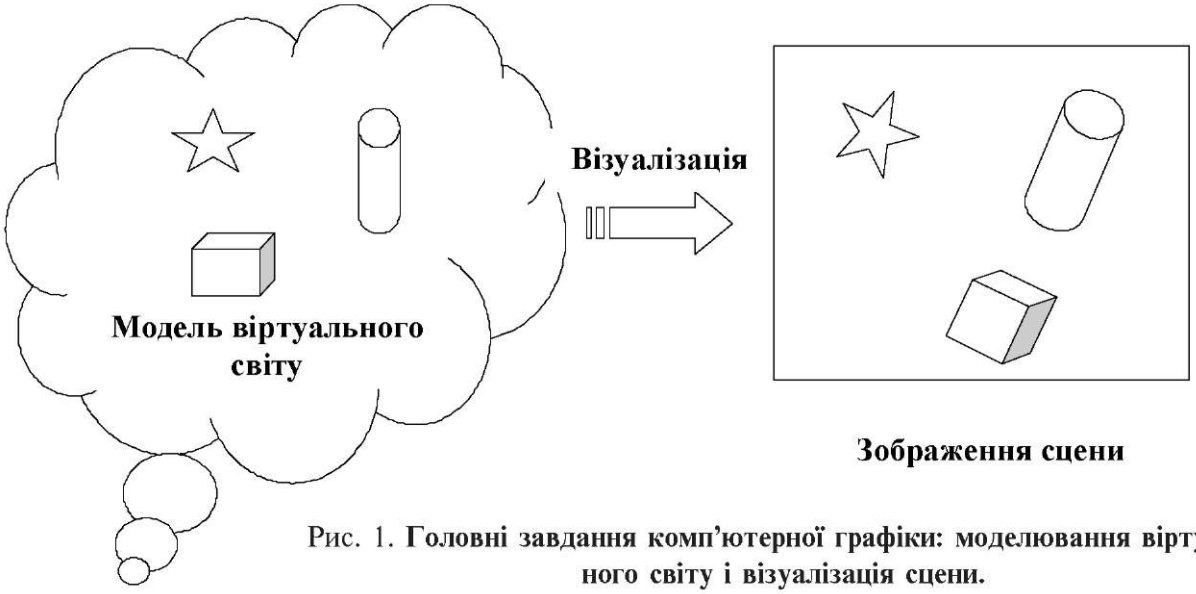


Рис. 1. **Головні завдання комп'ютерної графіки: моделювання віртуального світу і візуалізація сцени.**

Основні поняття 3-вимірної графіки в бібліотеці Java 3D. Графічна система переважно складається з двох головних компонент: модельєр та візуалізатор. Модельєр відповідає за побудову моделей віртуального світу, а візуалізатор виконує візуалізацію сцени.

Переважно графічні об'єкти, що моделюються, знаходяться у 2- або 3-вимірному просторі. Цей загальний простір, що містить усі графічні об'єкти, називається **світовим простором.** Звізуалізована сцена світового простору - головний результат графічної системи - переважно знаходиться у 2-вимірному вигляді. Отже, методики, що входять у 2- та 3-вимірні графіки, значно відрізняються. Оскільки задачі 3-вимірної графіки значно складніші, то 2- і 3-вимірні графіки часто досліджуються як окремі питання.

Графічні об'єкти, що моделюються у світовому просторі, переважно є геометричними сутностями, такими як лінії і поверхні, але вони також включають інші спеціальні об'єкти, такі як освітлення, тексти та зображення. Графічні об'єкти можуть мати багато характеристик і властивостей, таких як колір, прозорість і текстура.

Для моделювання геометричних об'єктів використовуються різноманітні математичні представлення. Прямолінійні сегменти та прості многокутники забезпечують прості та компактні представлення. Потрібно зберігати лише вершини структур і це легко реалізовується. Складніші представлення включають сплайнові криві та поверхні. Це вимагає збереження порівняно небагатьох контрольних точок.

Геометричні перетворення застосовуються до об'єктів для того, щоб отримати відповідне розміщення об'єктів у віртуальному просторі. Перетворення такого типу називаються **перетвореннями об'єктів**. Перетворення також використовуються для споглядання. Вони відомі як **перетворення споглядань.** Корисним сімейством геометричних перетворень є афінні перетворення, які включають найзагальніші їх види, такі як перенесення, обертання, масштабування і віддзеркалення. Більш загальний набір перетворень - проективні перетворення - є корисними для 3-вимірних споглядань.

Споглядання використовується для того, щоб побачити модель у віртуальному світі з певної перспективи. Процес 2-вимірних споглядань є відносно простим. Перетворення споглядання переважно не відрізняється від перетворення об'єкта. 3-вимірне споглядання значно складніше. Подібно до очей або камер 3-вимірні споглядання включають процес проекції, який відображає 3-вимірні об'єкти на 2-вимірну площину. Багато параметрів, таких як проекція, позиція споглядання, орієнтація і поле зору можуть впливати на 3-вимірну візуалізацію.

Для того, щоб отримати реалістичну візуалізацію віртуального світу потрібно вирішити ряд задач візуалізації. Потрібно коректно відобразити відносне розміщення об'єктів у візуалізовані зображення. Наприклад, об'єкт може бути прихований за іншим об'єктом. При цьому прихована частина не повинна бути показана на зображенні. Слід розглядати джерела світла з різними характеристиками. На вигляд впливатимуть властивості матеріалів об'єктів.

Можливості і характеристики апаратних пристроїв мають великий вплив на графічні системи. Найпо-

ширенішими пристроями виводу для відображення результатів графічної візуалізації є відеомонітори та принтери. До інших пристроїв виводу належать плотери та голографічні проектори. Широко розповсюдженими пристроями вводу є мишки, джойстики та табло з ручками. Також існують більш вишукані пристрої вводу і сенсори, такі як трекери з шістьма ступенями свободи.

Анімація є також важливою частиною комп'ютерної графіки. Замість непорушних зображень анімація дає динамічний графічний контент та візуалізацію. У застосуваннях, таких як візуалізація кіносцен та ігри, анімація відіграє вирішальну роль. Іншим динамічним аспектом комп'ютерної графіки є інтерактивність. У відповідь на вводи користувача графічна модель може відповідно змінюватися. Фундаментальний принцип GUI (графічного інтерфейсу користувача) грунтується на взаємодіях користувача з графічними системами.

Комп'ютерна графіка має широку область застосування. Популярність середовищ GUI зробила графіку складовою частиною програм для звичайних користувачів. CAD (дизайн за допомогою комп'ютерів) та інші інженерні пристрої залежать від графічних систем. Візуалізація даних та інші наукові застосування також широко використовують графіку. З розвитком нових комп'ютерних методик, таких як CT, PET та MRI, медичні системи все більше застосовують технології комп'ютерної графіки.

Традиційно комп'ютерна графіка повинна бути пов'язана з деталями реалізації - використовуючи алгоритми низького рівня для конвертування примітивів, таких як лінії у пікселі, для визначення поверхонь, прихованих від зору, для розрахунку значень кольору точок на поверхні та ін. Не потрібно створювати низькорівневі деталі безпосередньо, оскільки вони реалізовані в пакетах Java 2D та Java 3D.

Сприйняття фізичного світу людиною є строго тривимірним. Однак візуальні зображення, які ми бачимо очима, є двовимірними. Спеціальний тип відображення, який називається перспективними проекціями, є механізмом, який переводить 3-вимірні сцени у 2-вимірні зображення. Основна мета 3-вимірної комп'ютерної графіки - моделювати цей процес у комп'ютерах.

3-вимірна комп'ютерна графіка вивчає моделювання і візуалізацію 3-вимірного світу. Геометричні об'єкти у 3-вимірному просторі можуть мати розмірність 0 (точки), розмірність 1 (криві), розмірність 2 (поверхні) або розмірність 3 (тверді тіла). Об' єкти можуть мати різні види матеріальних властивостей. Можуть існувати джерела світла різних характеристик, що освітлюють сцену у віртуальному просторі. Віртуальні камери, що знімають сцени віртуального світу, можуть бути розміщені в різних положеннях у просторі і мати різні характеристики. 3-вимірна комп'ютерна графічна система потребує вирішення багатьох проблем представлення графічних об'єктів і їх властивостей, покращення перетворень, організації всіх компонент і візуалізації сцени.

Java 3D - це об'єктно-орієнтована API для 3-вимірноїкомп'ютерної графіки. Повна графічна модель Java 3D-програми організована в структурі, яка називається графом сцени. Кожен вузол у графі сцени є об' єктом класу, що представляє одну з багатьох графічних сутностей. Граф сцени надає систематизовану модель для механізму візуалізації Java 3D, щоб автоматично візуалізувати сцену, побудовану Java 3D-програмою.

Процес 3-вимірної візуалізації. Візуалізація 3-вимірної сцени для вироблення зображення (переважно 2-вимірного) - складний процес. На відміну від 2-вимірної графіки, візуалізоване зображення 3-вимірного об'єкта значно відрізняється від його оригінальної 3-вимірної версії. Безпосереднє моделювання об'єкта, спираючись на його візуалізоване зображення, не видається можливим. Отже, 3-вимірна графічна система незмінно включає побудову віртуального світу, в якому визначаються різні графічні об'єкти і джерела світла. Проста ілюстрація понять 3-вимірної графіки показана на рисунку 2.

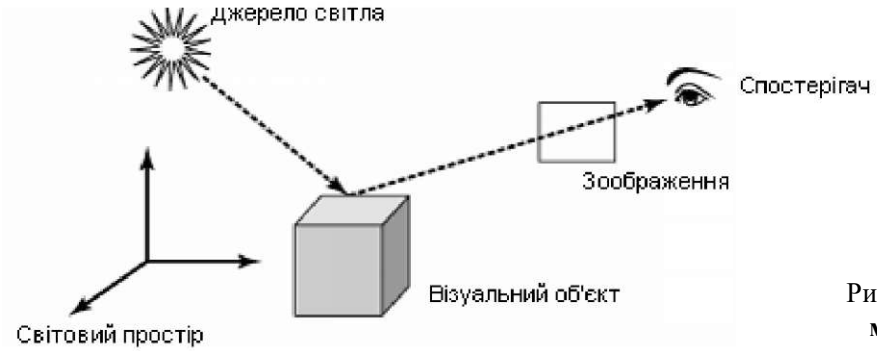


Рис. 2. **3-вимірна графічна модель і споглядання.**

Процес візуалізації статичної графічної сцени подібний до того, як реальна камера робить фото. Віртуальний світ включає візуальні об'єкти, що відбивають світло від різних джерел світла. Камера розміщується в певній точці віртуального світу і проектує видимі частини віртуального світу вздовж певного напрямку на 2-вимірну площину. Графічні об'єкти, так само як і споглядання, можуть бути динамічними. Отже, сцена і візуалізовані зображення можуть неперервно змінюватися в часі. Можуть існувати взаємодії між віртуальним світом і реальним фізичним світом.

Для того, щоб реалізувати або використати такі 3-вимірні графічні системи, потрібно розглянути багато задач, пов'язаних з моделюванням віртуального світу і візуалізацією сцени, наприклад:

- Геометрія графічних об'єктів.
- Розташування та позиціонування об'єктів.
- Геометричні перетворення, що застосовуються до об'єктів і споглядань.
- Властивості матеріалів і текстура об'єктів.
- Освітлення і його характеристики.
- Типи проекцій в спогляданні.
- Положення споглядання, поле зору і інші властивості.
- Ілюмінація і моделі затінення.
- Динамічні поведінки різних компонент.
- Реакції на вводи користувача.

Геометричні описи графічних об' єктів є найфундаментальнішими аспектами побудови віртуального світу в 3-вимірній графічній системі. Основні будівельні блоки для 3-вимірних графічних об' єктів включають точки, лінії, поверхні і тіла. Прості многокутники використовуються для апроксимації складних об'єктів. 3-вимірна графічна система переважно пропонує зручні можливості для генерації певних геометрій високого рівня, таких як 3-вимірні тексти і геометричні примітиви (наприклад сфери, конуси і паралелепіпеди). Складніші засоби моделювання включають сплайнові криві і поверхні.

Перетворення є важливим інструментом на графічній сцені. Геометричні перетворення використовуються для розміщення геометричних об'єктів у просторі віртуального світу, для зміни цих поверхонь, розмірів і, при потребі, положень. 3-вимірні афінні перетворення - сімейство перетворень у просторі віртуального світу. Інше сімейство, відоме як проективні перетворення, є більш загальним. Проективні перетворення є важливою частиною процесу 3-вимірного споглядання.

Крім геометрії, графічний об'єкт також має властивості появи, які визначають, як об'єкт буде візуалізовуватися. Ці властивості можуть включати кольори, текстури властивості матеріалу для більш витонченого затінення. Освітлення, ілюмінація або політика затінення визначають спосіб обчислення кольорів і інтенсивність світла на об'єктах. Вибір моделі ілюмінації також впливає на результат візуалізації. Певна геометрична інформація, така як нормалі поверхонь, тісно пов'язана з виглядом в деяких ілюмінаційних моделях. Нормаллю в точці поверхні є напрямок, перпендикулярний (вертикальний) до дотичної площини в точці. В деяких ілюмінаційних моделях інтенсивність світла в точці пов'язується з кутом між напрямком зору (споглядання) і напрямком відбивання світла. Вектор відбивання визначається напрямком світла і нормаллю до поверхні.

Процес 3-вимірного споглядання переважно включає проективне перетворення, яке відображає 3-вимірну сцену у 2-вимірну площину. Споглядання може мати багато параметрів, щоб керувати його характеристиками. Проекція може бути паралельною або перспективною. Для конкретного споглядання видимий об'єм віртуального світу є переважно скінченним. Просте застосування математичного перетворення проекції може бути недостатнім для візуалізації. Наприклад, відносне положення всередині об'єктів може бути також важливим для процесу візуалізації. Частина об'єкта може бути прихована за іншим об'єктом. Ці проблеми повинні бути вирішені належним чином для отримання прийнятних візуальних результатів.

Звичайно 3-вимірна візуалізація не обмежується статичною сценою. Віртуальний світ може змінюватися в часі. Система споглядання може бути пов'язана з динамічним пристроєм. Динамічні ефекти процесу візуалізації можуть включати анімацію і інтерактивність. Інтерактивність - це зміна сцени на основі зворотного зв'язку з користувачем. Анімація - це зміна, сконструйована всередині віртуального світу. Відмінність між двома типами динаміки часто стирається. Динамічні поведінки можуть походити із змін в графічних об' єктах у віртуальному світі або із змін споглядання. Споглядачі (камери або очі) можуть самі бути об'єктами, розміщеними у віртуальному світі і можуть динамічно змінювати свої положення, напрямки і інші властивості.

Java 3D API забезпечує повну реалізацію основних графічних алгоритмів, надаючи можливість концентруватися на головних поняттях і проблемах графіки замість громіздких деталей низькорівневої реалізації.

Графи сцен Java 3D. Для ефективної організації різних елементів 3-вимірної візуалізації Java 3D використовує поняття графу сцени для того, щоб побу-

дувати віртуальним простір, який включає все, що відноситься до 3-вимірної візуалізації. Граф сцени - це абстрактна математична модель для організації сцени. Це не знімок або зображення сцени. Граф сцени може бути концептуально зображений як діаграма, але його справжня реалізація виконується в програмі через інстанціацію об'єкта та виклик методу. Граф сцени дозволяє програмістам описувати складні графічні структури і дії стандартним чином. Він також дозволяє механізму візуалізації Java 3D обробляти сцену системно і ефективно.

Граф сцени - це деревовидна структура даних, відома як НАГ (напрямлений ациклічний граф). Напрямлений граф складається з набору вершин (або вузлів), з'єднаних напрямленими ребрами (або з'єднаннями). Рисунок 3 показує напрямлений граф з шістьма вершинами і вісьмома ребрами. (Напрямлений) шлях в напрямленому графі - це послідовність „ вершина-ребро", яка рухається вздовж ребер графу. Наприклад, на рисунку 3 b-c-f-e є шляхом. Циклом в напрямленому графі є замкнутий шлях - тобто, шлях, який має одну і ту ж початкову і кінцеву вершини. Наприклад, на рисунку 3 a-c-f-e-a є циклом. НАГ - це напрямлений граф без жодного циклу. Тобто рисунок 3 не є НАГ. Однак, якщо ребро e-a видалити, то він стає НАГ.

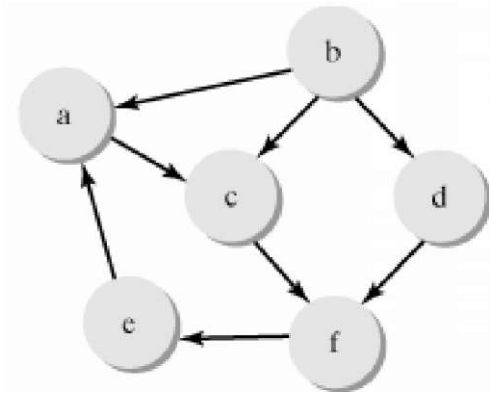


Рис. 3. **Напрямлений граф.**

(Напрямлене) дерево - це спеціальний вид НАГ. Дерево будується, починаючи з однієї вершини, яка називається коренем дерева. Може бути багато ребер, які походять з кореня і ведуть до інших вершин, які називаються дітьми кореня. Кожна дитина може мати кілька ребер, які ведуть до її дітей таким же чином. Цей процес може повторюватися довільне число разів, таким чином будуючи дерево. На рисунку 4 показано приклад дерева. В дереві вершина може мати довільне число дітей (включаючи 0). Але вона не може мати більше як одного батька. Вершина, що не має дитини, називається листком. Нелисткова вершина називається внутрішнім вузлом. На рисунку 4 вузли e, h, c, g є листками, а вузли a, b, d, f - внутрішніми вузлами.

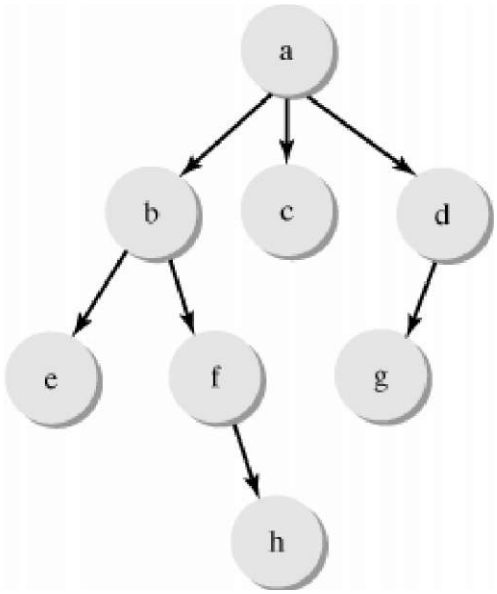


Рис. 4. **Напрямлене дерево.**

Вузли графу сцени представляють об'єкти різних класів щодо графічних функцій. Зв'язки між вузлами представляють логічні взаємозв'язки між ними. У справжній Java 3D-програмі вузли створюються інсталяцією класів, визначених в Java 3D API або класів, які походять від класів і інтерфейсів API. Зв'язки створюються викликом відповідних методів або конструктора в класах. На рисунку 5 показано дуже простий граф сцени Java 3D.

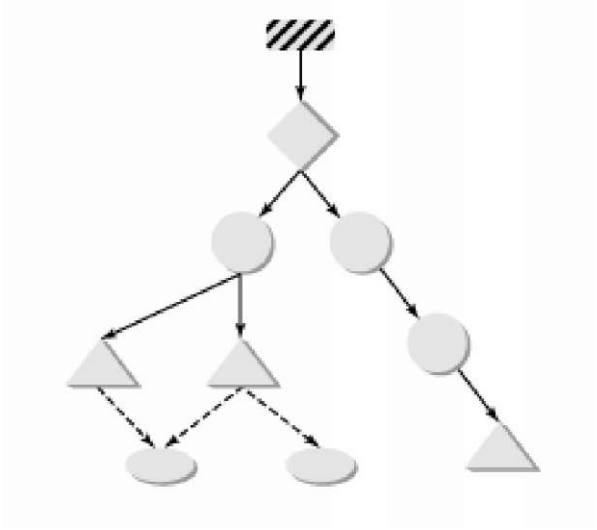


Рис. 5. **Граф сцени як НАГ.**

Різні типи вузлів і з'єднань представляються різними позначками на діаграмі графу сцени. Малюнок 6 показує позначки, які використовуються для вузлів і ребер графу сцени.

Рис. 6. **Легенда графів сцени.**

Граф сцени має три головні частини. На вершині знаходиться надструктура, яка складається з об'єктів, тобто класів VirtualUniverse і Locale. Головне тіло графу сцени - це дерево об'єктів, які належать до класу Node. Третя частина - це набір об'єктів NodeComponent. Листкові вузли в структурі дерева можуть посилатися на об' єкти вузлових компонент. На один об'єкт NodeComponent можуть посилатися кілька об'єктів Leaf. Отже, загальна структура графу сцени - це не дерево, а НАГ.

Ієрархія головних класів елементів графу сцени показана на рисунку 7. Класи VirtualUniverse і Locale - це класи для надструктури і вони не походять з абстрактного класу SceneGraphObject. Вузли дерева в графі сцени визначаються підкласами абстрактного класу Node. Абстрактний клас NodeComponent служить базовим класом для різних вузлових компонент.

Надструктура. Об'єкти VirtualUniverse і Locale - об'єкти надструктури графу сцени. Програма Java 3D переважно має лише один об'єкт VirtualUniverse. VirtualUniverse сконсутройовано для того, щоб представляти увесь простір, що може цікавити Java 3D - програму. Для того, щоб забезпечити рівень точності „ всесвіту", VirtualUniverse використовує три високорозрядні 256-бітові числа з фіксованою крапкою для представлення його координат. Високорозрядні числа з фіксованою крапкою мають свою точку посередині 256 біт. Тобто, таке число має 128 біт для цілої частини і 128 біт для дробової частини. Число 1.0 представляє одиницю 1 метр. Числа цього типу можуть представити відстань аж до 2^{127} метрів з точністю до 2^{-128} метра. Це є адекватним для вимірювання довільного реального об'єкту у всесвіті. Наприклад, відстань від Землі до Сонця становить лише 2^{37} метра, а радіусом протона вважають близько 2^{-50} метра.

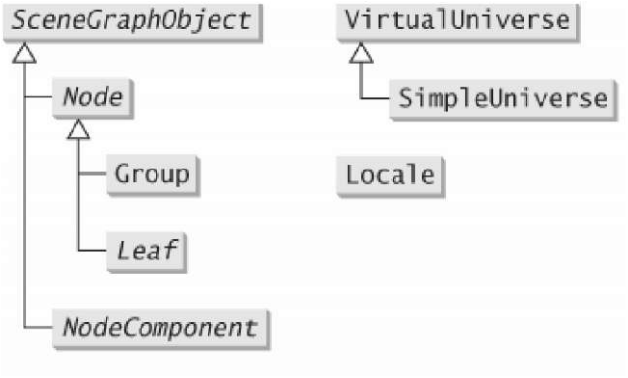


Рис. 7. **Ієрархія класів графу сцени.**

Клас HiResCoord використовується для представлення високорозрядних координат. Він включає три 256-бітні високорозрядні числа з фіксованою крапкою для представлення координат x-, y-, z.

Незважаючи на те, що VirtualUniverse в змозі моделювати увесь відомий всесвіт користуючись числами HiResCoord, проте неефективно представляє всі координати використовуючи об'єкти HiResCoord. Таким чином Java 3D використовує клас Locale для представлення менших локальних просторів і цим досягається куди більша ефективність. Об'єкт Locale визначає локальну координатну систему, закріплену на певному положенні, яке описується об' єктом HiResCoord у віртуальному всесвіті. Всередині певної локалізації координати точки представляються звичайними числами з плаваючою крапкою. VirtualUniverse включає один або більше об'єктів Locale. Locale може мати графи відгалуження, які приєднуються до нього. Коли граф відгалуження приєднується до Locale, то механізм візуалізації Java 3D розпочне візуалізовувати відгалуження, а граф таким чином „ оживає". Об'єкт Locale завжди приєднується до одного об'єкта VirtualUniverse. Цей зв'язок встановлюється в конструкторах Locale.

Locale(VirtualUniverse vu)

Locale(VirtualUniverse vu, HiResCoord location);

Locale(VirtualUniverse vu, int[] x, int[] y, int[] z).

Положення об'єкта Locale у всесвіті можна описати використовуючи об'єкт HiResCoord або три ма-

сиви з трьохіпі;, які описують високорозрядні числа. Положення за припущенням - це (0, 0, 0). Наступні рядки створюють надструктуру для графу сцени:

VirtualUniverse universe = new VirtualUniverse();

Locale locale = new Locale(universe);

Відгалуження графу сцени, які вказуються об'єктами BranchGroup, можуть бути приєднані до об'єкта Locale використовуючи такий метод в Locale:

void addBranchGraph(BranchGroup branch)

Відгалуження можуть редагуватися наступними методами:

void replaceBranchGraph(BranchGroup oldBranch, BranchGroup newBranch);

void removeBranchGraph(BranchGroup branch).

Наступні методи повертають число відгалужень і всі відгалуження в Locale:

int numBranchGraphs()

Enumeration getAllBranchGraphs()

Клас SimpleUniverse - це допоміжний клас, який походить з VirtualUniverse. Він включає об'єкт Locale і набір об'єктів для визначення стандартного споглядання. Об'єкт SimpleUniverse може поєднуватися з відгалуженням візуального вмісту для того, щоб швидко сформувати повний граф сцени. Світова система координат в Java 3D - прямокутна система згідно правої руки. За припущенням, положення споглядання розміщене на осі z в напрямку від'ємної осі z. З перспективи споглядача вісь x вказує праворуч, а вісь y - вгору.

Приклад. Програма-переглядач obj-файлів. Граф сцени, який відповідає Java 3D-додатку, наведено на рисунку 8.

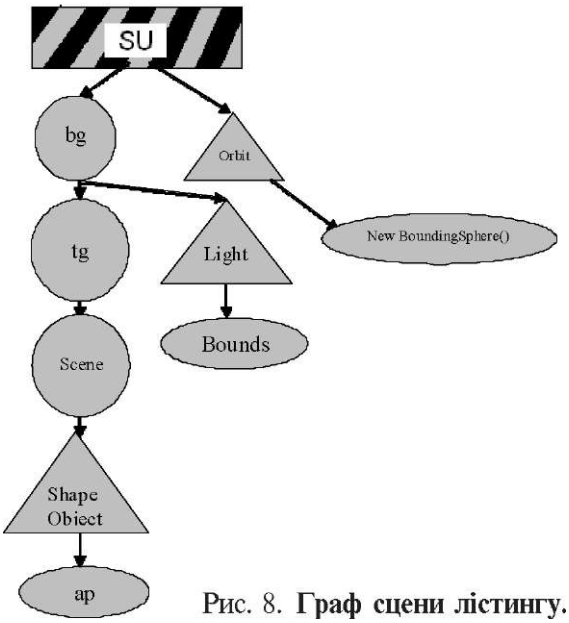
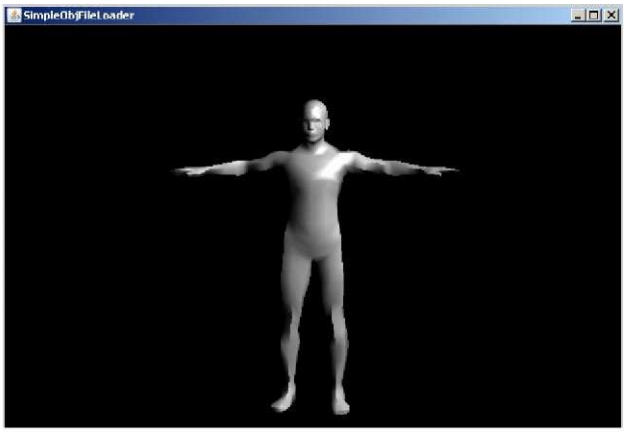


Рис. 8. **Граф сцени лістингу.**

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.applet.MainFrame;
// important classes for this application
import com.sun.j3d.loaders.objectfile.ObjectFile;
import com.sun.j 3d.loaders.Scene;
// important class for viewing platform motion
import com.sun.j3d.utils.behaviors.vp.*;
public class SimpleObjFileLoader extends Applet {
  public static void main(String s[]) {
   new MainFrame(newSimpleObjFileLoader(), 640, 480);
   }
  public void init() {
   GraphicsConfiguration gc =
   SimpleUniverse.getPreferredConfiguration();
    Canvas3D cv = new Canvas3D(gc);
   setLayout(new BorderLayout());
   add(cv, BorderLayout.CENTER);
   BranchGroup bg = createSceneGraph();
   bg.compile();
   SimpleUniverse su = new SimpleUniverse(cv);
  su.getViewingPlatform().setNominalViewingTransform();

   // viewplatform motion
   OrbitBehavior orbit = new OrbitBehavior(cv);
   orbit.setSchedulingBounds(new BoundingSphere());
  su.getViewingPlatform().setViewPlatformBehavior(orbit);

   su.addBranchGraph(bg);
   }
  private BranchGroup createSceneGraph() {
   BranchGroup root = new BranchGroup();
    // object
   Appearance ap = new Appearance();
   ap.setMaterial(new Material());

 // load the object file
  Scene scene = null;
  Shape3D shapeObject = null;
  // read in the geometry information from the data file
   ObjectFile objFileloader = new ObjectFile(
ObjectFile.RESIZE );
  try
   {
  scene = objFileloader.load( "samples/human.obj" );
```

```
// тут вказується ім'я .obj-файлу для завантаження
 }
catch ( Exception e )
 {
 scene = null;
 System.err.println( e );
 }
if( scene == null )
 System.exit( 1 );
// retrieve the Shape3D object from the scene
BranchGroup branchGroup=scene.getSceneGroup();
shapeObject = (Shape3D) branchGroup.getChild( 0 );
// assign the appearance to the Shape
shapeObject.setAppearance( ap );

  // transformation
  Transform3D tr = new Transform3D();
   tr.setScale(0.5);
  tr.setTranslation(new Vector3f(0f, 0f, 0f));
  TransformGroup tg = new TransformGroup(tr);
  root.addChild(tg);
  // add object to the transform group
tg.addChild(scene.getSceneGroup());
   // light
      PointLight light = new PointLight(new
Color3f(Color.white),
                        new Point3f(1f,1f,1f),
                        new Point3f(1f,0.1f,0f));
  BoundingSphere bounds = new BoundingSphere();
  light.setInfluencingBounds(bounds);
  root.addChild(light);
  return root;
  }
 }
```

Вікно виконання програми:



Висновки. В роботі введено основні поняття комп'ютерної графіки в термінах API-бібліотеки Java3D, яка, на нашу думку, на сьогодні є найприйнятнішою для створення Web-додатків інтерактивної віртуальної реальності в медицині, оскільки:

1. Бібліотека Java3D є Web-інтегрованою та кросплатформовою.

2. Тут використана ієрархічна графічна модель, яка грунтується на графах сцен. Це має вкрай важливе значення для візуалізації складних біологічних об'єктів.

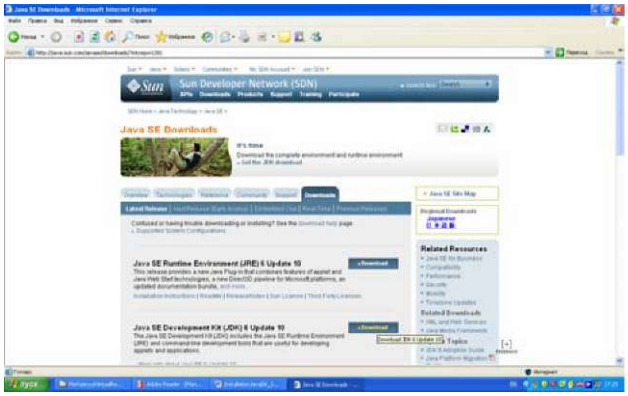
3. Бібліотека є безкоштовною з відкритим вихідним кодом.

1.Angle, E. (2006). Interactive computer graphics: A top-down approach with OpenGL. Addison Wesley
2. Bannach, D., Amft, O., Kunze, K. S., Heinz, E. A., Troster, G., & Lukowicz, P. (2007).Waving real hand gestures recorded by wearable motion sensors to a virtual car and driver in a mixed-reality parking game. Proceedings of the IEEE Symposium on Computational Intelligence and Games (pp. 32-39).
3. Bekins, D., Yost, S., Garrett, M., Deutsch, J., Htay, W. M., Xu, D., & Aliaga, D. (2006). Mixed reality tabletop (MRT): A low-cost teleconferencing framework for mixed-reality applications. Proceedings of the IEEE Virtual Reality Conference (pp 34). Washington, DC: IEEE Computer Society
4. Benko, H., Ishak, E. W., & Feiner, S. (2004). Collaborative mixed reality visualization of an archaeological excavation. Proceedings of the 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (pp 132-140). Washington, DC: IEEE Computer Society.
5. Blanchebarbe, P., & Diehl, S. (2001). A framework for component based model acquisition and presentation using Java3D. Proceedings of the 6th International Conference on 3D Web Technology (pp 117-125). New York: ACM.
6. Bouvier, D. J. (2006). Getting started with the Java 3D API, Sun Microsystems. Retrieved from http://java.sun.com/products/java-media/3D/collateral/j3d_tutorial_ch7.pdf
7.Burrows, A. L., & England, D. (2002). Java 3D, 3D graphical environments and behaviour. Software—Practice and Experience, 32(4), 359-376.
8.Myron W Krueger, Artificial Reality (1983), Artificial Reality II (1991) Wellner, P, Mackay, W. & Gold, R. Eds. Special issue on computer augmented environments: back to the real world. Communications of the ACM, Volume 36, Issue 7.

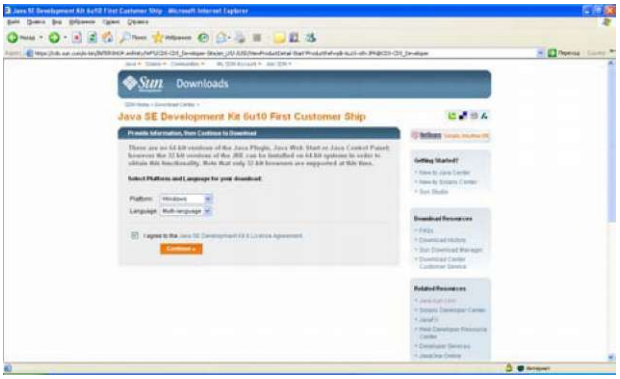
13±ttp//www.ospru/pcworld/2001/12/162603/_p2.html
14.http://cs.usu.edu.ru/home/kost/Java%203D%20p1.ppt.
15http://wwwrenderru/books/show_bookphp?book_id=521
16http://vrml.about.com/computE/vrml/
17http://wwwvrmlorg/
18^ttp://wwwuriitru/conf_erohin_50/Part_02_04.pdf
19http://sun.com/
20.http://coding;derkeiler.^m/Archive/Ja\a/comj:)lang^javahelp/2004-08/0241html
21http://www.cspm/pcworld/2002/01/162727/
22. http://www.java2s.com/Code/Java/Swing-JFC/FileSystemTreehtm

Додаток. Інсталяція інструментів розробки Java в стандартній редакції (Java Standard Edition Developer Kit або Java SE Developer Kit або Java SE). На сайті http://java.sun.com/ знаходиться посилання на найостаннішу версію інструментів розробки Java в стандатній редакції (як правило, текст посилання Java SE):

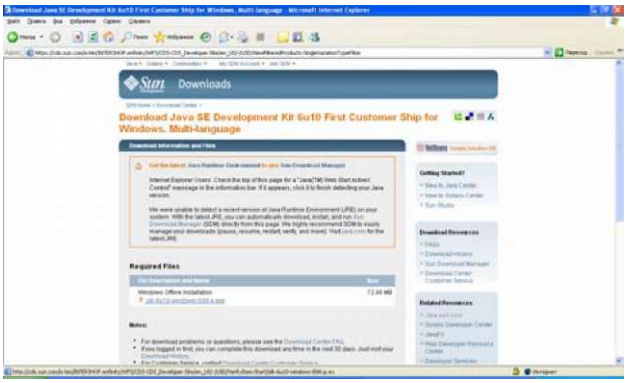
У вікні, що з'явилося, виберіть саме Developer Kit і натисніть кнопку "Download":



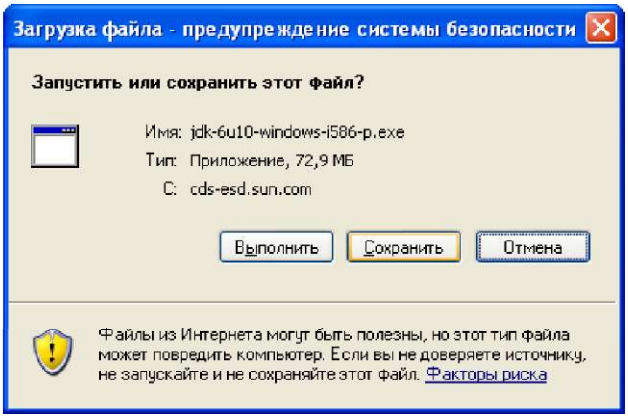
У вікні, що з'явилося, правильно вкажіть платформу, на якій буде встановлюватися JDK SE, багатомовність, дотримання умов ліцензії та натисніть "Continue":



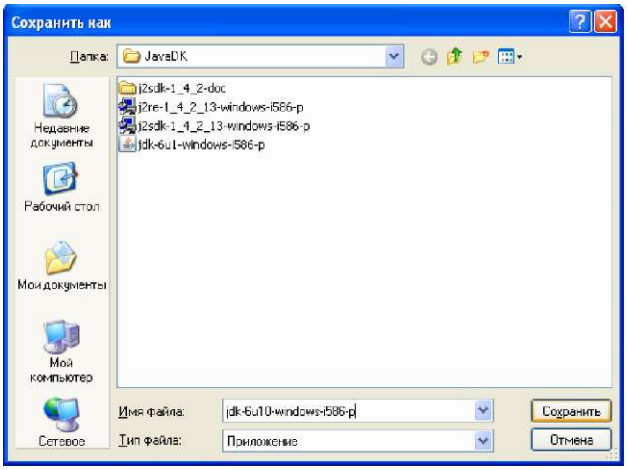
У вікні, що з'явилося, натисніть назву файлу інсталяції:



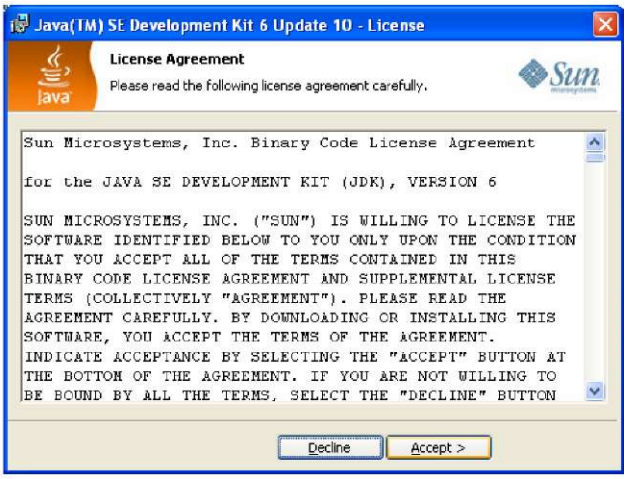
У вікні, що з'явилося, натисніть «Сохранить»:



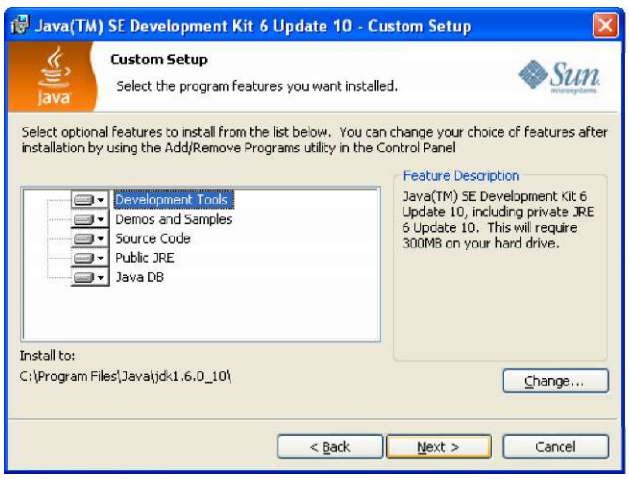
і виберіть бажане розташування файлу на Вашій системі:



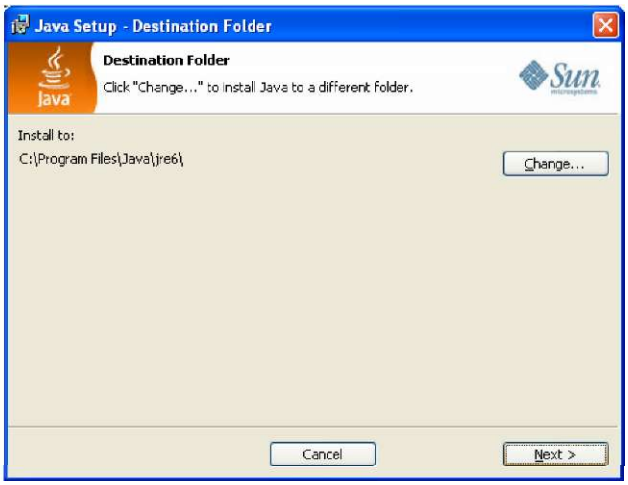
Після того, як інсталяційний файл JDK SE завантажено, запустіть його на виконання. Перечитавши умови ліцензійної угоди прийміть їх, натиснувши "Accept":



При виборі складових інсталяційного пакету рекомендуємо вибрати усе, що пропонується:



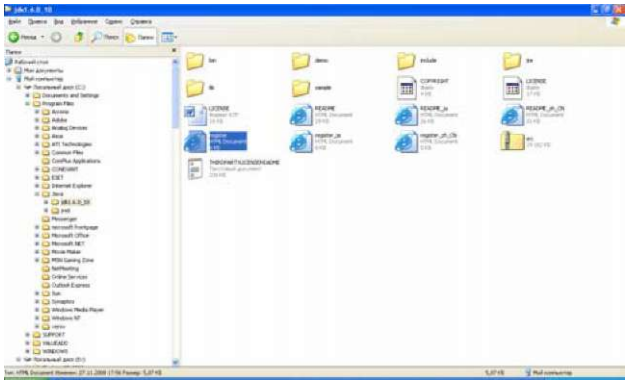
і вибравши каталог для інсталяції, натиснути "Next". Далі слідує процес інсталяції, під час якого буде можливість вказати розташування бібліотек JRE (Java Runtime Environment). Можна прийняти шлях за припущенням:



і натиснути "Next". Натиснувши в кінці "Finish" ми завершуємо інсталяцію:

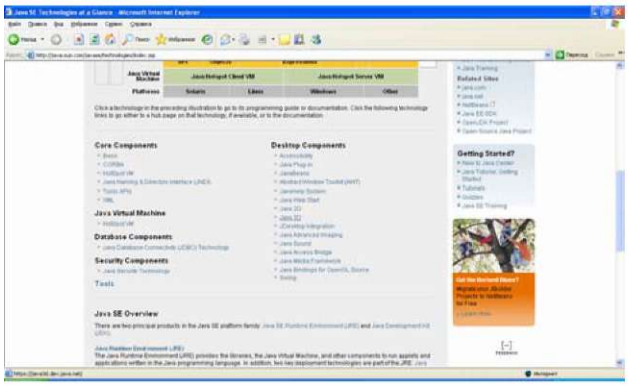
а сайт Sun Microsystems запропонує нам зареєструватися:

Реєстрація гарантуватиме Вам приєднання до широкої спільноти користувачів Java - розсипки повідомлень про нові версії та поновлення, знижки на продукти та тренінги, доступ до документації і ін. Зареєструватися можна і пізніше, вибравши в папці JDK SE файл register.html:

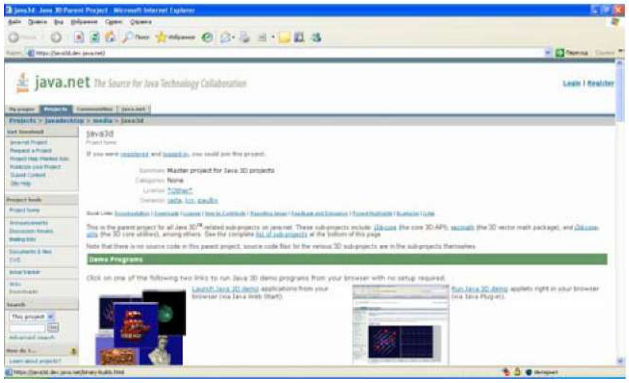


Інсталяція бібліотеки Java 3D. На сайті http://java.sun.com знову зайдіть на посилання Java SE. Перейдіть на закладку Technologies:

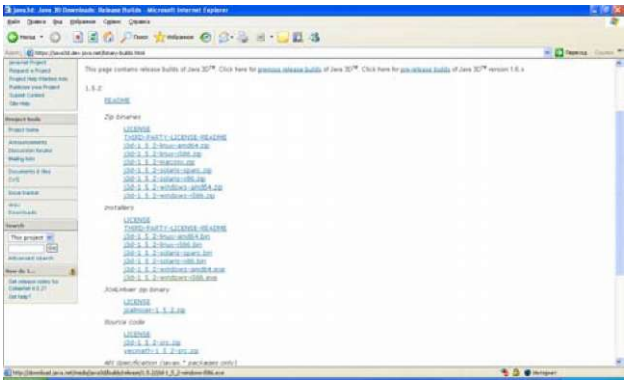
Серед Desktop Components відшукайте Java 3D:



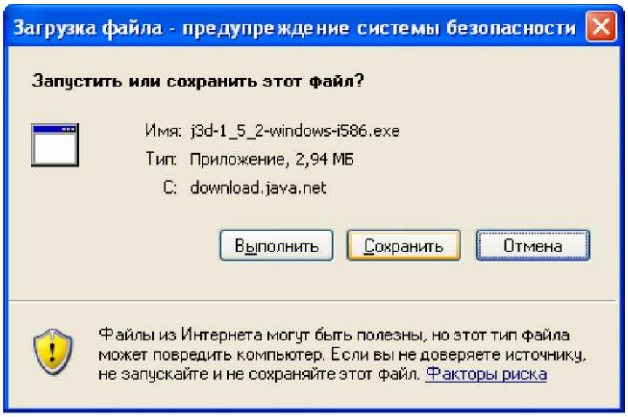
У вікні, що відкрилося:



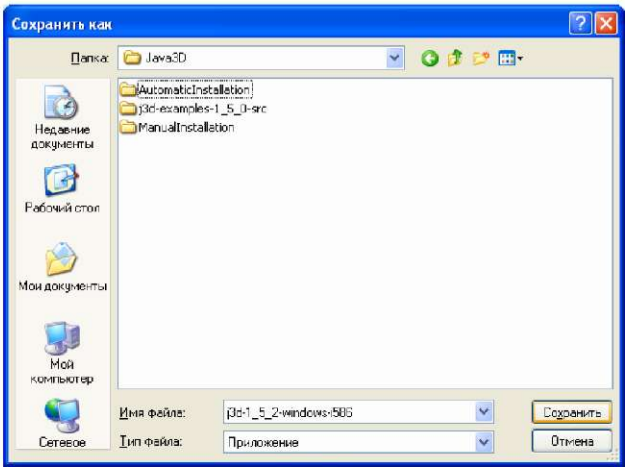
в головному меню виберемо "Downloads". Виберемо файл ддя інсталяції, який ми бажаємо завантажити - він повинен відповідати нашій платформі та способу інсталяції - ми скористаємося готовим інсталяційним виконуваним файлом (Installer):



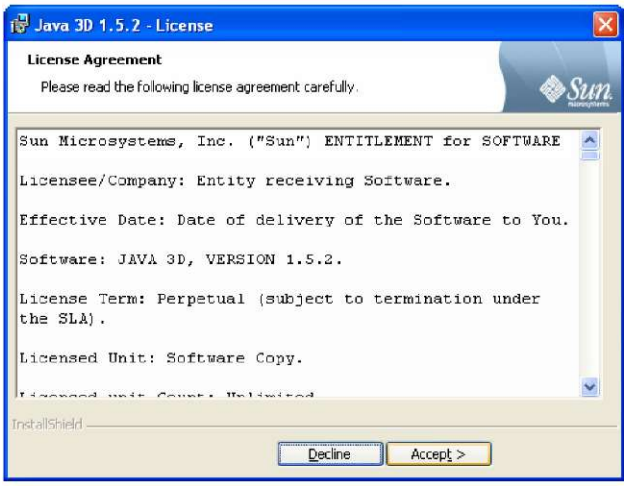
Вибравши файл, вкажемо «Сохранить»:



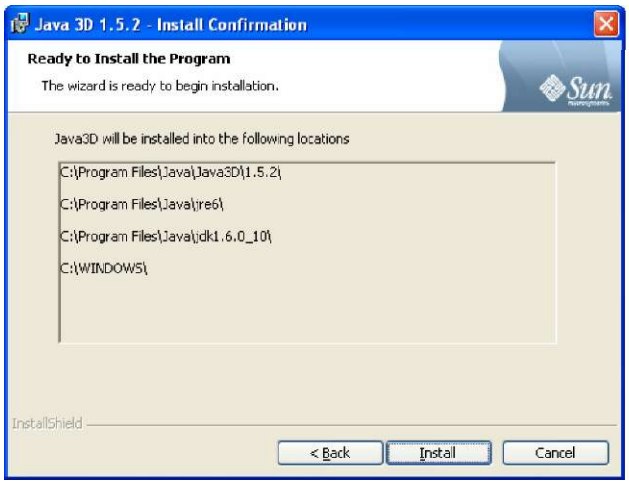
та місцерозташування:



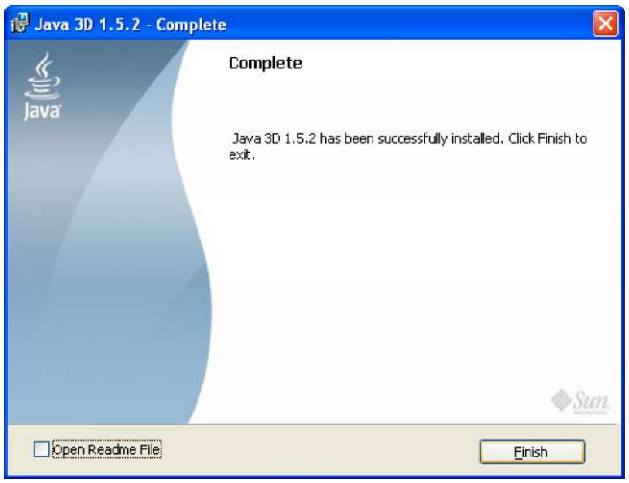
Запускаємо завантажений файл на виконання. Прочитавши, приймаємо ліцензійні домовленості:



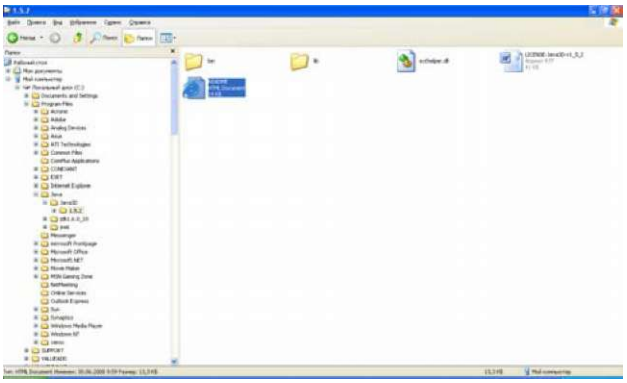
Далі буде показано місця, де буде інстальовано бібліотеку:



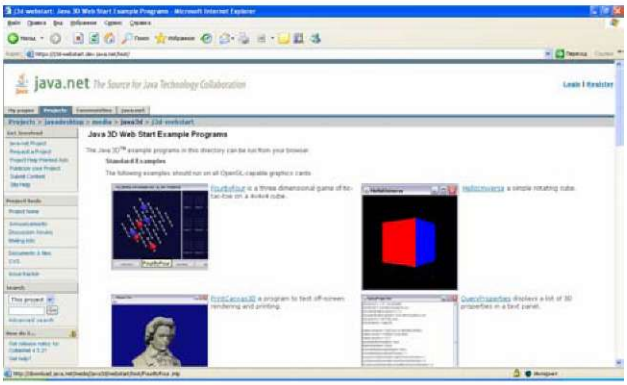
На завершення натискаємо "Finish":



Для того, щоб пересвідчитися, що інсталяція проведена успішно, можна переглянути роботу демонстраційних проектів, на які є посилання у файлі Readme інсталяційного каталогу Java 3D:



Спробуємо запустити один із демонстраційних додатків:



а саме, гру «Чотири з чотирьох»: