МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

ГЕОМЕТРИЧНЕ МОДЕЛЮВАННЯ І КОМП'ЮТЕРНА ГРАФІКА

Методичні вказівки до виконання курсових робіт для студентів, які навчаються за напрямом підготовки 6.050101 «Комп'ютерні науки»

Київ 2011

УДК 004.92
ББК 32.973
Б19

Укладачі: Є.В. Бородавка, кандидат технічних наук
В.В. Демченко, кандидат технічних наук, доцент

Рецензент: В.М. Міхайленко, доктор технічних наук, професор

Відповідальний за випуск: В.В. Демченко, кандидат технічних наук, доцент, завідуючий кафедрою прикладної математики

Затверджено на засіданні кафедри прикладної математики, протокол №14 від 6 червня 2011 року.

Б19 Геометричне моделювання і комп'ютерна графіка: методичні вказівки до виконання курсових робіт / Уклад. Є.В. Бородавка, В.В. Демченко. – К.: КНУБА, 2011. – 16 с.

Методична розробка містить загальні положення, які визначають мету й завдання курсової роботи та основні вимоги щодо її виконання, рекомендації щодо вибору теми, формування завдань, складання плану, викладу матеріалу, подання курсової роботи викладачу, структуру курсової роботи та вимоги, щодо її технічного оформлення, рекомендації щодо захисту, теми курсових робіт, а також список рекомендованої літератури.

Призначено для студентів, які навчаються за напрямом підготовки 6.050101 «Комп'ютерні науки».

УДК 004.92
ББК 32.973

Загальні положення

Курсове проектування є логічним продовженням лекційного курсу і лабораторних занять з дисципліни «Геометричне моделювання і комп'ютерна графіка» та сполучною ланкою для переходу від виконання навчальних завдань до проведення самостійної роботи за реальною тематикою.

Метою курсового проектування є формування в майбутніх спеціалістів професійних навичок та вмінь по розробці програм автоматизованого проектування і комп'ютерної графіки з використанням алгоритмів обчислювальної геометрії та сучасних засобів графічного програмування.

В процесі виконання курсових робіт студенти повинні продемонструвати вміння застосувати на практиці теоретичні знання, отримані під час вивчення дисципліни.

Тематика курсових робіт

Тематика і зміст курсових робіт обумовлені основними розділами робочої навчальної програми дисципліни та орієнтовані на програмну реалізацію алгоритмів геометричного моделювання та графічного відображення об'єктів з використанням засобів:

- відкритої графічної бібліотеки OpenGL;
- візуальних середовищ об'єктно-орієнтованого програмування Embarcadero RAD Studio XE (мови Delphi та C++), Microsoft Visual Studio 2010 (мови C++ та C#).

Конкретну тему та опис початкових даних для розробки програми вказують у завданні на курсову роботу.

Порядок виконання курсової роботи

У відповідності з навчальним планом курсова робота передбачена на 4-му курсі у 8-му семестрі. На виконання роботи відводиться десять тижнів.

Тему курсової роботи студент вибирає самостійно чи за вказівкою викладача з урахуванням рівня підготовки студента. У випадку практичної участі студента в науково-дослідних роботах, що ведуться на кафедрі чи в університеті і мають безпосереднє відношення до дисципліни, студент може запропонувати власне формулювання теми курсової роботи, узгодивши її з викладачем.

Кожен студент виконує роботу *індивідуально*. В окремих випадках з дозволу викладача допускається об'єднання студентів у групи (2-3 студенти) для роботи над складними чи комплексними темами.

Послідовність виконання курсової роботи включає такі етапи:

- аналіз та чітка математична постановка задачі;
- розробка та опис структури даних геометричних об'єктів;
- розробка та опис структури програми;
- розробка та опис алгоритмів складових програмних функцій;
- створення файлів вхідних та контрольних даних;
- розробка та відлагодження програми геометричного моделювання та графічного виводу результатів;
- оформлення пояснювальної записки.

В процесі розробки програми рекомендується використовувати створені авторами базові програмні класи для роботи з структурами геометричних даних (текст програми подано в додатку).

Кінцевими результатами курсового проектування є файли контрольних даних, працююча програма та пояснювальна записка.

ПОЯСНЮВАЛЬНА ЗАПИСКА

Пояснювальна записка повинна включати:

- вступ;
- математичну постановку задачі;
- опис структури даних геометричних об'єктів;
- опис програми з розділами у відповідності до стандартів «Єдиної системи програмної документації (ЄСПД)»: загальні відомості, призначення та умови застосування, структура та функції програми, опис алгоритму, вхідні та вихідні дані, повідомлення оператору;
- додатки (тексти програм, графічні зображення екранних форм, файли контрольних даних);
- список використаної літератури.

Формальний опис алгоритмів програм рекомендується виконувати з використанням діаграм об'єктно-орієнтованих програмних компонентів, операторних схем чи обробних систем. В останньому випадку узагальнений алгоритм програми геометричного моделювання може бути поданий в вигляді наступної обробної системи:

$$S = (I, T, O, F, \delta),$$

де: I – множина вхідних даних; T – множина внутрішніх (проміжних) даних; O – множина вихідних даних; F – множина програмних функцій

(об'єктів) обробки даних; δ – стратегія застосування функцій, яка визначає послідовність виклику окремих $f_i \in F$.

Залежно від конкретної теми курсової роботи, до складу множини F можуть входити, наприклад, функції, що реалізують перетворення кадрування, обчислення коефіцієнтів рівняння площини за трьома точками, побудову тріангульованої моделі поверхні, визначення просторового відношення полігонів тощо.

Загальний обсяг пояснювальної записки повинен складати 10-20 аркушів формату А4.

ТЕМИ КУРСОВИХ РОБІТ

Теми курсових робіт поділені на категорії у відповідності до рівня складності виконання. Роботи найвищого рівня складності оцінюються в межах 3,75-5(С-А) в залежності від результату виконання. Роботи середнього рівня складності оцінюються в межах 3,0-4,24 (Е-С).

НАЙВИЩИЙ РІВЕНЬ СКЛАДНОСТІ

1. Розробка програми створення тріангульованої моделі поверхні (TIN-моделі) на заданій множині точок методом «жадібної» тріангуляції. *Вхідні дані*: координати тривимірних точок, що зчитані з файлу чи бази даних. *Результат роботи*: графічне зображення системи зв'язаних трикутників, що отримані в результаті роботи алгоритму. Можливість інтерактивного додавання точок вітається.
2. Розробка програми створення TIN-моделі поверхні на заданій картографічній моделі рельєфу методом жадібної тріангуляції. *Вхідні дані*: тривимірні полігони паралельні площині XOY, що зчитані з файлу чи бази даних. *Результат роботи*: графічне зображення заданих полігонів; графічне зображення системи зв'язаних трикутників, що отримані в результаті роботи алгоритму. Можливість інтерактивного додавання полігонів вітається.
3. Розробка програми створення картографічної моделі рельєфу за його TIN-моделлю. *Вхідні дані*: координати вершин системи зв'язаних трикутників, що утворюють тривимірну модель рельєфу поверхні, зчитані з файлу чи бази даних; крок деталізації картографічної моделі. *Результат роботи*: графічне зображення заданих трикутників; побудова та візуалізація картографічних ліній поверхні; можливість зміни кроку деталізації.
4. Розробка програми форматних перетворень геометричної моделі об'єктів з явним поданням в модель з поданням у вигляді списку вершин та навпаки. *Вхідні дані*: файли чи бази даних для збереження

структур опису полігональних моделей з явним поданням та поданням у вигляді списку вершин. *Результат роботи*: відображення вмісту файлів (чи таблиць БД) та перетворення даних з одного формату в інший та навпаки. Графічне відображення вмісту файлів (чи БД) вітається.

5. Розробка програми форматних перетворень геометричної моделі об'єктів з явним поданням в модель з поданням у вигляді списку ребер та навпаки. *Вхідні дані*: файли чи бази даних для збереження структур опису полігональних моделей з явним поданням та поданням у вигляді списку ребер. *Результат роботи*: відображення вмісту файлів (чи таблиць БД) та перетворення даних з одного формату в інший та навпаки. Графічне відображення вмісту файлів (чи БД) вітається.
6. Розробка програми побудови та візуалізації опуклої оболонки множини точок методом обходу Грехема. *Вхідні дані*: координати двовимірних точок, що зчитані з файлу чи бази даних. *Результат роботи*: відображення двовимірних точок; можливість інтерактивного додавання нових точок; побудова та візуалізація опуклої оболонки.
7. Розробка програми побудови та візуалізації опуклої оболонки множини точок методом обходу Ендрю. *Вхідні дані*: координати двовимірних точок, що зчитані з файлу чи бази даних. *Результат роботи*: відображення двовимірних точок; можливість інтерактивного додавання нових точок; побудова та візуалізація опуклої оболонки.
8. Розробка програми побудови та візуалізації опуклої оболонки множини точок методом обходу Джарвіса. *Вхідні дані*: координати двовимірних точок, що зчитані з файлу чи бази даних. *Результат роботи*: відображення двовимірних точок; можливість інтерактивного додавання нових точок; побудова та візуалізація опуклої оболонки.
9. Розробка програми побудови та візуалізації опуклої оболонки множини точок методом апроксимації. *Вхідні дані*: координати двовимірних точок, що зчитані з файлу чи бази даних. *Результат роботи*: відображення двовимірних точок; можливість інтерактивного додавання нових точок; можливість зміни кроку апроксимації; побудова та візуалізація опуклої оболонки.
10. Розробка програми моделювання кривих з використанням форми Без'є. *Вхідні дані*: координати двовимірних точок, що утворюють визначаючий багатокутник для кривої Без'є. *Результат роботи*: розрахунок та відображення кривої Без'є за заданими опорними точками; можливість інтерактивного введення опорних точок. Реалізація можливості зміни положення існуючих точок з інтерактивним перерахуванням форми кривої вітається.

11. Розробка програми моделювання кривих з використанням форми B-сплайну. *Вхідні дані*: координати двовимірних точок, що утворюють визначаючий багатокутник для кривої B-сплайну; порядок B-сплайну; тип вузлового вектору; значення вузлового вектору, якщо він не відкритий. *Результат роботи*: розрахунок та відображення кривої B-сплайну за заданими опорними точками; можливість інтерактивного введення опорних точок; можливість зміни порядку B-сплайну; можливість зміни типу та значень вузлового вектору. Реалізація можливості зміни положення існуючих точок з інтерактивним перерахуванням форми кривої вітається.
12. Розробка програми для визначення просторового відношення відрізка прямої та довільного полігона. *Вхідні дані*: координати відрізка та довільного плоского полігона, що зчитані з файлу чи бази даних. *Результат роботи*: відображення відрізка та полігона, що зчитані з файлу чи БД; можливість інтерактивного введення відрізка та полігона; повідомлення про їх взаємне розташування; розрахунок та візуалізація точки (чи точок) перетину відрізка та ребер полігона, якщо вони існують. Реалізація можливості зміни положення відрізка прямої та полігона, а також їх окремих вершин вітається.
13. Розробка програми для визначення просторового відношення двох довільних полігонів. *Вхідні дані*: координати двох довільних плоских полігонів, що зчитані з файлу чи бази даних. *Результат роботи*: відображення полігонів, що зчитані з файлу чи БД; можливість інтерактивного введення полігонів; повідомлення про їх взаємне розташування; розрахунок та візуалізація точок перетину ребер полігонів, якщо вони існують. Реалізація можливості зміни положення полігонів та їх окремих вершин вітається.
14. Розробка програми формування параметричної моделі конструктивного елемента з візуалізацією засобами OpenGL (варіанти завдання зображено на рис. 1).

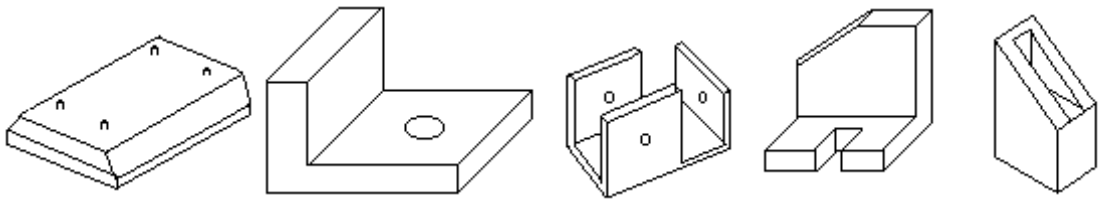

Рис. 1. Варіанти завдань до роботи №14

СЕРЕДНІЙ РІВЕНЬ СКЛАДНОСТІ

15. Розробка програми визначення висоти довільної точки тріангульованої поверхні. *Вхідні дані*: координати вершин системи зв'язаних трикутників, що утворюють тривимірну модель рельєфу поверхні, зчитані з файлу чи бази даних. *Результат роботи*: зображення проекції трикутників на площину XOY, та виведення значення координати z, вказаної мишкою точки.
16. Розробка програми формування та візуалізації перспективної проекції TIN-моделі поверхні. *Вхідні дані*: координати вершин системи зв'язаних трикутників, що утворюють тривимірну модель рельєфу поверхні, зчитані з файлу чи бази даних. *Результат роботи*: тривимірне зображення трикутників за допомогою команд OpenGL.
17. Розробка програми формування та візуалізації перспективної проекції полігональної моделі об'єктів з явним поданням. *Вхідні дані*: файл чи база даних, що описує полігональні моделі об'єктів у вигляді явного подання. *Результат роботи*: тривимірне зображення полігональних моделей об'єктів, що описані у файлі чи базі даних.
18. Розробка програми формування та візуалізації перспективної проекції полігональної моделі об'єктів з поданням у вигляді списку вершин. *Вхідні дані*: файл чи база даних, що описує полігональні моделі об'єктів у вигляді списку вершин. *Результат роботи*: тривимірне зображення полігональних моделей об'єктів, що описані у файлі чи базі даних.
19. Розробка програми формування та візуалізації перспективної проекції полігональної моделі об'єктів з поданням у вигляді списку ребер. *Вхідні дані*: файл чи база даних, що описує полігональні моделі об'єктів у вигляді списку ребер. *Результат роботи*: тривимірне зображення полігональних моделей об'єктів, що описані у файлі чи базі даних.
20. Розробка програми формування параметричної моделі конструктивного елемента з візуалізацією засобами OpenGL (варіанти завдання зображено на рис. 2).

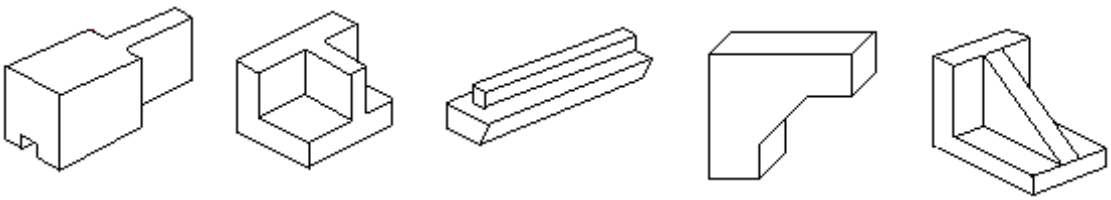

Рис. 2. Варіанти завдань до роботи №20

Додаток

```
unit GeomMod; // БАЗОВІ КЛАСИ ДЛЯ ГЕОМЕТРИЧНИХ ДАНИХ
interface
uses Classes, OpenGL;
type
// Точка на площині
     T2dPoint = class
     private
       FX,FY: double;
     public
       property X: double read FX write FX;
       property Y: double read FY write FY;
    end;
// Колекція точок на площині
    T2dPointsCollection = class(TList)
     private
     procedure SetPoint(Index: Integer; Value: T2dPoint);
     function GetPoint(Index: Integer): T2dPoint;
     public
{ Count - кількість точок (властивість успадкована від класу
TList) }
// точки колекції
     property Points[Index: integer]: T2dPoint read GetPoint
write SetPoint;
{ MaxPointsCountOrZero - задана кількість точок в колекції
або 0, якщо число точок невідоме }
     constructor Create(MaxPointsCountOrZero: integer);
{ пошук в колекції точки з мінімальною відстанню до заданої
точки: функція повертає індекс найближчої точки (>=0) або -1,
якщо в колекції немає жодної точки }
     function    SearchNearestPointIndex(Point:    T2dPoint):
integer;
{ пошук в колекції заданої точки: функція повертає індекс
точки (>=0) або -1, якщо в колекції немає такої точки }
     function FindPoint(FindingPoint: T2dPoint): integer;
     destructor Destroy; override;
    end;
// Точка в просторі
```

```
// Векторна форма задання координат точки в просторі:
type
  TGLVector = record
    x, y, z : GLfloat;    // GLfloat - тип даних OpenGL
  end;
// Координатна форма задання точки в просторі:
type T3DPoint=class
      public
      X,Y,Z: double;
      end;
// Колекція точок в просторі:
  type T3DPointsCollection=class(TList)
     private
     procedure SetPoint(Index: Integer; Value: T3DPoint);
     function GetPoint(Index: Integer): T3DPoint;
     public
// constructor Create - успадкований від класу TList
{ Count - кількість точок (властивість успадкована від класу
TList) }
// точки колекції
     property Points[Index: integer]: T3DPoint read GetPoint
write SetPoint;
{ пошук в колекції точки з мінімальною відстанню до заданої
точки: функція повертає індекс найближчої точки (>=0) або -1,
якщо в колекції немає жодної точки }
     function FindPoint(aFindingPoint: T3DPoint): integer;
{ пошук в колекції заданої точки: функція повертає індекс
точки (>=0) або -1, якщо в колекції немає такої точки }
     function    SearchNearestPointIndex(Point:    T3DPoint):
integer;
     destructor Destroy; override;
     end;
{ функція обчислення визначника 3-го порядку за умови, що
a[i,3]=1.0 }
function Det3_1(a11,a12,a21,a22,a31,a32: extended): extended;
// функція обчислення визначника 3-го порядку
function Det3(a11,a12,a13,a21,a22,a23,a31,a32,a33: extended):
extended;
```

```
{ знаходження коефіцієнтів рівняння площини за трьома точками: (за умови, що три точки не належать одній прямій) }
procedure CalcPlaneCoeff(P1,P2,P3: T3DPoint; var a,b,c,d: extended);
{ знаходження одиничної нормалі до грані за трьома точками: (задання нормалі необхідно для розрахунку параметрів освітлення сервером OpenGL) }
function Calc3PointsNormal(p1,p2,p3: T3dPoint): TGLVector;
implementation
// Колекція точок на площині
constructor T2dPointsCollection.Create(MaxPointsCountOrZero: integer);
begin
   inherited Create; Clear;
   if MaxPointsCountOrZero > 0 then Capacity:=MaxPointsCountOrZero;
end;
procedure T2dPointsCollection.SetPoint(Index: Integer; Value: T2dPoint);
begin
  if (Index >= 0) and (Index<Count) then Items[Index]:=Value;
end;
function T2dPointsCollection.GetPoint(Index: Integer): T2dPoint;
begin
  Result:=T2dPoint(Items[Index]);
end;
function T2dPointsCollection.FindPoint(FindingPoint: T2dPoint): integer;
var i: integer;
    point: T2dPoint;
begin
  Result:=-1;
  for i:=0 to Count-1 do
       begin
       point:=T2dPoint(Items[i]);
       if (Abs(point.X-FindingPoint.X) < 0.000000001) and
             (Abs(point.Y-FindingPoint.Y) < 0.000000001) then
              begin Result:=i; break; end;
```

```
      end;
end;
function  T2dPointsCollection.SearchNearestPointIndex(Point:
T2dPoint): integer;
var sqrdist,xx,yy,r: double;
    i: integer;
    PointItem: T2dPoint;
begin
    Result:=-1;
    sqrdist:=1.7*10e38;
    for i:=0 to Count-1 do
       begin
       PointItem:=Items[i];
       xx:=PointItem.X - Point.X;
       yy:=PointItem.Y - Point.Y;
       r:=xx*xx+yy*yy;
       if r < sqrdist then begin sqrdist:=r; Result:=i; end;
       end;
end;
destructor T2dPointsCollection.Destroy;
var i: integer;
begin
    if  Count  >  0  then  for  i:=0  to  Count-1  do
T2dPoint(Items[i]).Free;
    inherited Destroy;
end;
// Колекція точок в просторі:
procedure T3DPointsCollection.SetPoint(Index: Integer; Value:
T3DPoint);
begin
  if (Index >= 0) and (Index<Count) then Items[Index]:=Value;
end;
function    T3DPointsCollection.GetPoint(Index:    Integer):
T3DPoint;
begin
  Result:=T3DPoint(Items[Index]);
end;
```

```
function          T3DPointsCollection.FindPoint(aFindingPoint:
T3DPoint): integer;
var i: integer;
    point: T3DPoint;
begin
  Result:=-1;
  if Count>0 then
  for i:=0 to Count-1 do
       begin
       point:=T3DPoint(Items[i]);
       if (Abs(point.X-aFindingPoint.X) < 0.000001) and
          (Abs(point.Y-aFindingPoint.Y) < 0.000001) and
          (Abs(point.Z-aFindingPoint.Z) < 0.000001)
             then begin Result:=i; break; end;
       end;
end;
function T3DPointsCollection.SearchNearestPointIndex
               (Point: T3DPoint): integer;
var sqrdist,t,r: double;
    i: integer;
    PointItem: T3DPoint;
begin
    Result:=-1; sqrdist:=1.7*10e38;
    for i:=0 to Count-1 do
       begin
       PointItem:=Items[i];
       t:=PointItem.X - Point.X; r:=t*t;
       t:=PointItem.Y - Point.Y; r:=r+t*t;
       t:=PointItem.Z - Point.Z; r:=r+t*t;
       if r < sqrdist then begin sqrdist:=r; Result:=i; end;
       end;
end;
destructor T3DPointsCollection.Destroy;
var i: integer;
begin
  if   Count   >   0   then   for   i:=0   to   Count-1   do
T3DPoint(Items[i]).Free;
```

```
  inherited Destroy;
end;
// функція обчислення визначника 3-го порядку за умови, що a[i,3]=1.0
function Det3_1(a11,a12,a21,a22,a31,a32: extended): extended;
begin
Result:=a11*(a22-a32)-a12*(a21-a31)+a21*a32-a22*a31;
end;
// функція обчислення визначника 3-го порядку
function Det3(a11,a12,a13,a21,a22,a23,a31,a32,a33: extended): extended;
begin
Result:=a11*(a22*a33-a32*a23)-a12*(a21*a33-
a31*a23)+a13*(a21*a32-a22*a31);
end;
{ знаходження коефіцієнтів рівняння площини за трьома точками: (за умови, що три точки не належать одній прямій) }
procedure CalcPlaneCoeff(P1,P2,P3: T3DPoint; var a,b,c,d: extended);
begin
  a:=Det3_1(P1.Y,P1.Z,P2.Y,P2.Z,P3.Y,P3.Z);
  b:=Det3_1(P1.Z,P1.X,P2.Z,P2.X,P3.Z,P3.X);
  c:=Det3_1(P1.X,P1.Y,P2.X,P2.Y,P3.X,P3.Y);
  d:=-Det3(P1.X,P1.Y,P1.Z,P2.X,P2.Y,P2.Z,P3.X,P3.Y,P3.Z);
end;
// знаходження одиничної нормалі до грані за трьома точками:
function Calc3PointsNormal(p1,p2,p3: T3dPoint): TGLVector;
var wrki, vx1, vy1, vz1, vx2, vy2, vz2 : GLfloat;
  nx, ny, nz : GLfloat;
begin
  vx1 := p1.x - p2.x; vy1 := p1.y - p2.y; vz1 := p1.z - p2.z;
  vx2 := p2.x - p3.x; vy2 := p2.y - p3.y; vz2 := p2.z - p3.z;
  // вектор - перпендикуляр до центру трикутника
  nx := vy1 * vz2 - vz1 * vy2; ny := vz1 * vx2 - vx1 * vz2;
  nz := vx1 * vy2 - vy1 * vx2;
  // знаходимо вектор одиничної довжини
  wrki := sqrt (nx * nx + ny * ny + nz * nz);
  if wrki <= 0.000000001 then wrki := 1;
```

```
    Result.x := nx / wrki;
    Result.y := ny / wrki;
    Result.z := nz / wrki;
  end;
  end.
```

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. *Лященко А.А.* Геометричне моделювання у конструюванні інженерних об'єктів та систем. / А.А. Лященко, В.В. Демченко.–К.: КНУБА, 2001.–12 с.
2. *Лященко А.А*. Геометричне моделювання і комп'ютерна графіка: використання бібліотеки OpenGL. / А.А. Лященко, В.В. Демченко, Є.В. Бородавка, В.В. Смірнов. – К.: КНУБА, 2009. – 90 с.
3. *Архангельский А. Я.* C++ Builder 6. Справочное пособие. Книга 1. Язык C++ / А. Я. Архангельский. – М.: Бином-Пресс, 2002. – 544 с.
4. *Михайленко В.Е.* Геометрическое моделирование и машинная графика в САПР: учебник / В.Е. Михайленко, В.Н. Кислоокий, А.А. Лященко и др. – К.: Выща шк., 1991. – 374 с.
5. *Михайленко В.Є.* Інженерна та комп'ютерна графіка: Підручник / В.Є. Михайленко, В.М. Найдиш, А.М. Підкоритов, І.А. Скидан; За ред. В.Є. Михайленка. – К.: Вища шк., 2000. – 342 с.
6. *Краснов М.В.* OpenGL. Графика в проектах Delphi. / М.В. Краснов – СПб.: БХВ – Санкт-Петербург, 2000. – 352 с.
7. *Тихомиров Ю.* Программирование трехмерной графики. / Ю. Тихомиров – СПб.: БХВ – Санкт-Петербург, 1999. – 256 с.

Навчально-методичне видання

ГЕОМЕТРИЧНЕ МОДЕЛЮВАННЯ І КОМП'ЮТЕРНА ГРАФІКА

Методичні вказівки до виконання курсових робіт для студентів, які навчаються за напрямом підготовки 6.050101 «Комп'ютерні науки»

Укладачі: **БОРОДАВКА** Євгеній Володимирович
ДЕМЧЕНКО Віктор Вікторович

Редагування та коректура *Г.Є. Голіциної*

Комп'ютерна верстка *О.В. Кириченка*

Підписано до друку Формат 60x84$_{1/16.}$
Ум. друк. арк. 1,98. Облік.-вид. арк. 1,25.
Тираж 75 прим. Вид. № 21/І-10. Зам. №

КНУБА, Повітрофлотський проспект, 31, Київ, Україна, 03680

E-mail: red-isdat@knuba.edu.ua

Віддруковано в редакційно-видавничому відділі
Київського національного університету будівництва і архітектури

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК №808 від 13.02.2002 р.