Computer Graphics

Dr./ Ahmed Mohamed Rabie

طريقة الدخول للمقرر الدراسي عبر صفحة الانترنت


كليات الجامعة
كلية التربية
كلية العلوم
كلية التجارة
كلية الآداب
كلية الزراعة
كلية التربية النوعية
كلية التربية الرياضية
كلية الفنون التطبيقية
كلية الهندسة
كلية الحقوق
كلية الآثار
كلية التمريض
كلية الحاسبات والمعلومات

7		د / أحمد محمد ربيع سيد	تكنولوجيا المعلومات	مدرس

أحمد محمد ربيع سيد
كلية الحاسبات والمعلومات

البيانات الشخصية

الوظيفة : مدرس
القسم : تكنولوجيا المعلومات
التخصص :
هاتف العمل :
البريد الإلكتروني : amrabie@du.edu.eg

المقررات الدراسية

الأبحاث العلمية علي موقع الجامعه

There is no research

السيرة الذاتية

الموقع الشخصي

Course Description

المرفقات: 1 - تحميل الملف

0 26-02-2022 06:27

اقرأ المزيد

Lecture 1

0 26-02-2022 06:28

اقرأ المزيد

عدد الصفحات (1):

1

رابط الموقع الشخصي

http://staff.du.edu.eg/1058

Course Objectives

By the end of this course the student must be able to:

1. Introduce the essentials background of the visible surface detection algorithms.
2. Study the reflection and illumination models.
3. Study rendering algorithms for 3-D objects.
4. Demonstrate parametric representation of 3-D objects.
5. Apply shadows algorithms, 2-D texture mapping, 3-D texture mapping, Ray tracing, volume rendering, and anti-aliasing.
6. Introduce fractals and 3-D computer animation.
7. Color space in computer graphics.

Course Contents

Introduction to Visible surface detection algorithms.
The reflection and illumination models.
Rendering algorithms for 3-D objects.
Parametric representation of 3-D objects.
3-D computer animation.
3-D computer animation.
Color space in computer graphics.
Shadows algorithms.
2-D texture mapping.
3-D texture mapping
Ray tracing
Volume rendering. Anti-aliasing; Fractals and 3-D computer animation

Course References

1. Buss, Samuel R. *3D Computer Graphics: A Mathematical Introduction with OpenGL*. 2003. ISBN: 9780521821032

2. Shirley, Peter, Michael Ashikhmin, Steve Marschner. *Fundamentals of Computer Graphics*. 3rd ed. A K Peters/CRC Press, 2009. ISBN: 9781568814698.

Chapter 1

Introduction

The power and utility of **computer graphics** is widely recognized, and a broad range of graphics hardware and software systems is now available for applications in virtually all fields. Graphics capabilities for **both two-dimensional and three dimensional applications** are now common, even on general-purpose computers.

There are a number of approaches we can take to accomplish this, and numerous **algorithms** have been devised for efficient identification and display of visible objects for different types of applications.

Some methods **require more memory**, some involve more processing time, and some apply only to special types of objects. Which method we select for a particular application can depend on such factors as the complexity of the scene, type of objects to be displayed, available equipment, and whether **static or animated** displays are to be generated.

The various algorithms are referred to as **visible-surface detection methods.** Sometimes these methods are also referred to as hidden-surface elimination methods, although there can be subtle differences between identifying visible surfaces and eliminating hidden surfaces.

We can broadly classify visible-surface detection algorithms according to whether they deal with the object definitions or with their projected images. These two approaches are **called object-space methods and image-space methods**, respectively.

An **object-space method** compares objects and parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.

In an **image-space algorithm**, visibility is decided point by point at each pixel position on the projection plane. Most visible-surface algorithms use image-space methods, although object-space methods can be used effectively to locate visible surfaces in some cases.

There are two broad classifications for computer-graphics software: **special-purpose packages and general programming packages.**

Special-purpose packages are designed for **nonprogrammers** who want to generate pictures, graphs, or charts in some application area without worrying about the graphics procedures that might be needed to produce such displays.

The interface to a special-purpose package is typically a **set of menus that allow users to communicate with the programs** in their own terms. Examples of such applications include artists' painting programs and various architectural, business, medical, and engineering CAD systems.

By contrast, a **general programming package** provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or Fortran.

Basic functions in a typical graphics library include those for specifying picture components (straight lines, polygons, spheres, and other objects), setting color values, selecting views of a scene, and applying rotations or other transformations.

Some examples of general graphics programming packages are **GL (Graphics Library), OpenGL**, VRML (Virtual-Reality Modeling Language), Java 2D, and Java 3D.

A set of graphics functions is often called a computer-graphics application programming interface (CG API) because the library provides a software **interface between a programming language (such as C++) and the hardware**. So when we write an application program in C++, the graphics routines allow us to construct and display a picture on an output device.

Graphics functions in any package are typically defined as a set of specifications independent of any programming language. **A language binding** is then defined for a particular high-level programming language. This binding gives the syntax for accessing the various graphics functions from that language.

Each language binding is defined to make best use of the corresponding language capabilities and to **handle various syntax issues**, such as data types, parameter passing, and errors. Specifications for implementing a graphics package in a particular language are set by the ISO.

The **OpenGL** bindings for the C and C++ languages are the same. Other OpenGL bindings are also available, such as those for Java and Python.

A basic library of functions is provided in OpenGL for specifying graphics primitives, attributes, geometric transformations, viewing transformations, and many other operations. **OpenGL is designed to be hardware independent**, so many operations, such as input and output routines, are not included in the basic library.