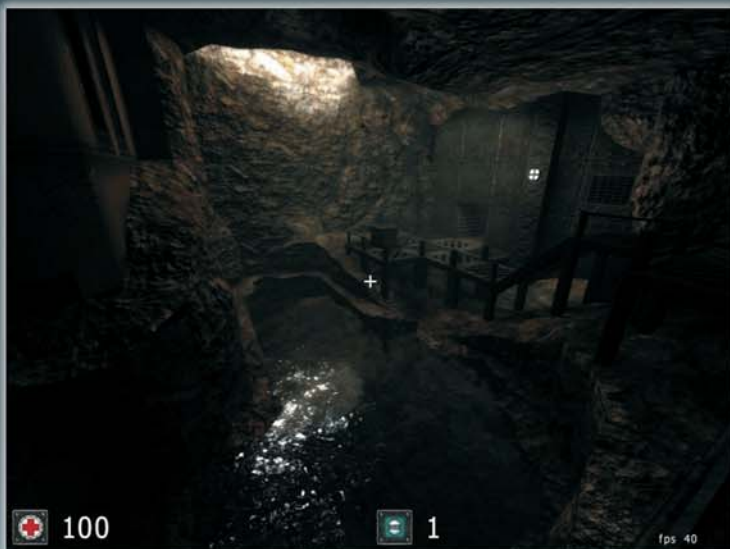


Advanced Diploma in Game Development

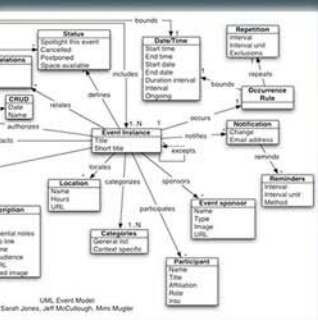
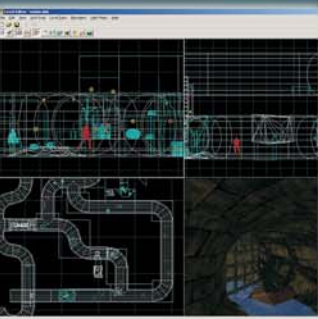
Developing a production quality 3D Game can be a mammoth undertaking. The skill-set required to master the subject spans a wide gamut of interdependent disciplines - from low-level memory management to high-level software architecture. But that's not all; game developers need to be proficient not only with many different aspects of computer science and engineering viz. artificial intelligence, mathematics, physics, graphics etc., but also most importantly the software engineering skills required to orchestrate such wide and varied subjects in the most efficient manner possible.

KODE|academy offers world-class training designed to create true-blue game programmers through its on-line as well as on-site courses. On-line courses are provided through India's first ever game programming portal www.kodeacademy.com. The modules of online courses are not just another pre-packaged course material. They have been crafted with utmost diligence to be interactive and a form of "live education". "Advanced Diploma in Game Development" is an on-site course on the fundamentals of game development. Our course materials are an outcome of intensive research on the latest real-time technologies in game programming as well as training methodologies delivered by experienced faculty. Industry inputs have also been incorporated to the greatest possible extent in our pedagogy to make it relevant to students and the industry alike.

Needless to say that mastering anything worthwhile involves a steep learning curve and an uncanny dedication to see it through. At KODE|academy, we help you to flatten the learning curve by providing the most complete and industry relevant courses to ease your transition to the exciting and rewarding realm of computer games. Our mission is to guide our students in understanding the intricacies involved in making a "software" fun by weaving the most elegant computer algorithms and design patterns leveraging state-of-the-art technology.



The KODE|academy also provides an online learning portal based on the learning methodology of "social constructionism". The portal utilizes all the major web 2.0 features such as forums, blogs, instant messaging and news to provide an incredible learning experience that is changing the way people interact and learn all around the world.



```
// Right hand arm...
// ...a better class to derive, so to make shorter classes...
class Starter {
public:
    Starter() {}
    Starter(int x, int y) {}
    Starter(int x, int y, int z) {}
    Starter(int x, int y, int z, int w) {}
    Starter(int x, int y, int z, int w, int v) {}
    Starter(int x, int y, int z, int w, int v, int u) {}
    Starter(int x, int y, int z, int w, int v, int u, int t) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c, int b) {}
    Starter(int x, int y, int z, int w, int v, int u, int t, int s, int r, int q, int p, int o, int n, int m, int l, int k, int j, int i, int h, int g, int f, int e, int d, int c, int b, int a) {}
};
```

• Software Architecture and Design Patterns

A modern computer game requires all of its components to co-exist in harmony and this is made possible by the architecture of the game. Not only that, but each component of the game also requires its own architecture. For example, different graphics engines may have different architectures. The architecture of a computer game to a large extent defines the efficiency of the entire work-flow of game creation and therefore requires much thought and consideration from the get go.

Also, in any discipline, there are some frequently recurring problems. It may be the challenge of designing the suspension structure for a bridge in engineering, or laying the foundation of a multistory building in architecture. In programming too, there are some scenarios that often arise when solving many different types of design problems. Design patterns provide template solutions to such problems.

Key Highlights:

- Refresher of Object Oriented Programming techniques
- Windows programming
- Windows GUI programming
- Working with COM
- Memory management for large scale software applications
- Design patterns
- Plugin based software architecture
- UML (Universal Markup Language)
- Collaborative software development

• Graphics

The task of a graphics programmer is to faithfully represent the game world on the two dimensional computer screen. A graphics engine is primarily responsible for loading and rendering 3D data created by artists. This area of game programming can be further sub-classified to include disciplines such as shader programming. Shaders are programs written for your graphics card that allow you to define the "look" of an object.

Key Highlights:

- Programming the DirectX fixed function pipeline
- Introduction to Programmable Rendering pipeline using Cg [used in games such as Quake Wars on Playstation 3 and Xbox 360]

• Mathematics

Mathematics is the heart of game development. It is a tool that allow us to implement logic within game code and like it or not, if you want to become a great game programmer, you need to have a command over mathematics. But luckily for us, the math required in games are fun to learn and easy to implement.

Key Highlights:

- Cartesian coordinate system
- Multiple coordinate spaces
- Vector and linear algebra
- Matrices and linear transformations
- Calculus
- Trigonometry
- Functions and set theory

• Data Structures and Algorithms

A computer program can be thought of as data [memory] and operations we perform upon that data. Different components of a game requires their own specialized data structures and algorithms. This module will cover some of the most frequently used data structures and algorithms in a game.

Key Highlights:

- Vectors, Deques, Linked Lists, Sets, Maps etc.

• Physics

We live in an interactive world where all physical actions have some reaction or another. The job of a physics programmer is to simulate Newtonian physics within a computer. Having a virtual game world react in a way consistent with the real world adds to the players level of immersion within a game.

Key Highlights:

- Primer on Newtonian Physics

...the most complete and comprehensive introduction to

GAME DEVELOPMENT

• Artificial Intelligence

Artificial Intelligence or AI in short, is one of the most esoteric branches of computer science. Ever since the advent of computers, man has striven to create a digital counterpart that can live and behave much like us. Game AI programming is all about simulating intelligence within a computer. AI is used in games to give the player a sense of interacting with entities that think.

Key Highlights:

- State driven agents
- Autonomous moving agents
- Group Behavior

• Scripting

Game development requires the marriage of art and technology. Artists are not expected to know low level game engine development details. Therefore, programmers often implement a scripting engine to allow artists and end users to modify the engine in order to create new content. Almost all commercial games these days contain some support for scripting or another. Scripts also allow for the modification of a game without the need to re-compile it therefore aiding in rapid prototyping and gameplay adjustments. Most games use an off-the-shelf scripting solution like Lua. Lua has been used in games like World of Warcraft, Far Cry and Heroes of Might and Magic V. In this module you will learn how to program with Lua and integrate it within your game engine.

Key Highlights:

- Creating embedded script-able Finite State Machines

• Input

A game is an interactive software. Input programming is all about programming the physical devices that the gamers use to play the game. DirectInput, which is one of the components of the DirectX SDK simplifies the process of programming for input devices like keyboards, mice and joysticks.

Key Highlights:

- Using DirectInput to program keyboards, mouse and joysticks

• Audio

Still considered a black art by many, audio/sound programming is all about programming the auditory responses of a game.

Key Highlights:

- Using Direct Audio to add sound and music to a game

• Projects

The course will require the students to complete two game projects:

- A 2D Casual Game
- A Case study of a Quake II like First Person Shooter

The projects will utilize all the knowledge accumulated in different modules throughout the course and will put all the theory into perspective. Before attempting to create a 3D game, it is not only advisable but mandatory to gain some experience creating simple 2D games. 2D games are not only essential stepping stones to the more complex 3D games, but are an entire genre into themselves.

The case study of a 3D first person shooter will dissect a Quake II like 3D engine that supports 6 DOF [degrees of freedom.] Although Quake II utilized procedural programming techniques, this course will focus on a 3D engine that utilizes Object Oriented paradigm in its architecture.

Log in...

www.kodeacademy.com

For more information visit:
www.KodeInfotainment.net

KODE|academy Telephone: 080 - 41276688
 II Floor, K R Towers 9243400833
 100 Feet Ring Road 9880223323
 BTM Layout, 1st Stage Email: info@kodeinfotainment.net
 Bangalore - 560068

© 2008 KODE Infotainment Pvt. Ltd. All rights reserved. KODE and KODE|academy are registered trademarks of KODE Infotainment, the New Delhi based company. All other trademarks contained herein are the property of their respective owners.



ENTERTAINMENT | TECHNOLOGY | SIMPLIFIED™