Konstruksi Pemrograman 1 + Sylabus

Pemrograman Game

Content:

- Sylabus
- Konstruksi Pemrograman 1

Sylabus

Kode Mata Kuliah	: **IF330325/IF430325**
Nama Mata Kuliah	: **Pemrograman Game**
Semester/TA	: **6/2015-2016**
Jumlah pertemuan (T/P)	: **10/20**
SKS	: **3 SKS (1SKS Teori dan 2 SKS Praktikum)**

Deskripsi Mata Kuliah

- Kuliah ini menggunakan bahasa pemrograman berbasis objek oleh karena dibutuhkan konsep-konsep pemrograman berbasis objek yang diperoleh pada kuliah IF21319 Object Oriented Programming.

- Kuliah ini memberikan kemampuan untuk memprogram game menggunakan bahasa pemrograman berbasis objek dengan menggunakan game engine. Kuliah ini akan memberikan wawasan tentang game design, penerapan kecerdasan buatan pada game.

- Pada kuliah ini mahasiswa akan diberikan tugas-tugas untuk membuat game dengan menerapkan teknik pemrograman game sehingga mahasiswa memiliki pengalaman membuat game karya sendiri.

Capaian Pembelajaran

Setelah mengikut kuliah ini mahasiswa diharapkan mampu untuk:

- Pemrograman game menggunakan bahasa C# (*High Level Techniques in Game Programming*)
- Menerapkan Artificial Intelegentia pada pembuatan Game
- Mengunakan Game Engine Unity untuk membangun Game

Komponen Penilaian

Komponen	**Bobot**
Kuis	10%
Ujian Tengah Semester	25%
Ujian Akhir Semester	30%
Proyek 1	15%
Proyek 2	20%

Topik Bahasan

Kuliah terdiri dari beberapa topik (modul) utama yaitu:

- Module 1: Konstruksi C# Programming in Unity
- Module 2: Movement
- Module 3: Path Finding
- Module 4: Decision Making
- Module 5: Learning
- Module 6: Strategy
- Module 7: Unity

Buku Pegangan Utama dan Referensi

1) Ian Millington, Artificial Intelligence For Games, Elseiver 2006
2) Aung Sithu Kyaw and Thet Naing Swe, Unity 4.x Game AI Programming, Packt Publishing 2013
3) Jeremy Gibson, Introduction to Game Design, Protoyping, and Developmen (From Concept to Playable Game – with Unity and C#), Addison Wesley – 2015
4) Stuart Russel and Peter Norvig, Artificial Inteligence-A Modern Approach, Pearson 2010
5) John Patrick Flynt and Danny Kodicek, Mathematic and Physics for Programmer
6) Venita Pereira, Learning Unity 2D Game Development by Example, Packt Publishing
7) Simon Jacson, Mastering Unity 2D Game Development, Packt Publishing
8) Alan Thorn, Mastering Unity Script, Packt Publishing
9) Claudio Scholastici, Unity 2D Game Development Cookbook, Packt Publishing
10) Curtis Bennet, Unity AI Programming Essentials, Packt Publishing
11) Carnegie Mellon, Game Programming (http://www.cs.cmu.edu/~maxim/classes/CIS15466_Fall11/)

Konstruksi Pemrograman 1

VARIABEL

Strongly Typed Variables in C#

Instead of being able to be assigned any kind of value, C# variables are *strongly typed*, meaning that they can only accept a specific type of value.

This is necessary because the computer needs to know how much space in memory to allocate to each variable

VARIABEL

Important C# Variable Types

- bool: A 1-Bit True or False Value
- int: A 32-Bit Integer
- float: A 32-bit Decimal Number
- char: A 16-Bit Single Character

VARIABEL

Naming Conventions

1. Use *camelCase* for pretty much everything
2. Variable names should start with a lowercase letter (e.g., someVariableName).
3. Function names should start with an uppercase letter (e.g., Start(), Update()).
4. Class names should start with an uppercase letter (e.g., GameObject, ScopeExample).
5. Private variable names can start with an underscore (e.g., _hiddenVariable).
6. Static variable names can be all caps with snake_case (e.g., NUM_INSTANCES). As you can see, snake_case combines multiple words with an underscore in between them.

BOOLEAN OPERATIONS

! (The NOT Operator)

&& (The AND Operator)

|| (The OR Operator)

BOOLEAN OPERATIONS

Shorting Versus Non-Shorting Boolean Operators

The standard forms of AND and OR (&& and ||) are *shorting* operators, which means that if the shorting operator can determine its return value from the first argument, it will not bother to evaluate the second argument.

By contrast, a non-shorting operator (& and |) will always evaluate both arguments completely.

BOOLEAN OPERATIONS

Logical Equivalence of Boolean Operations

- (a & b) is the same as !(!a | !b)
- (a | b) is the same as !(!a & !b)
- **Associativity:** (a & b) & c is the same as a & (b & c)
- **Commutativity:** (a & b) is the same as (b & a)
- **Distributivity of AND over OR**: a & (b | c) is the same as (a & b) | (a & c)
- **Distributivity of OR over AND**: a | (b & c) is the same as (a | b) & (a | c)

BOOLEAN OPERATIONS

Comparison Operators

- == (Is Equal To)
- != (Not Equal To)
- > (Greater Than) and < (Less Than)
- >= (Greater Than or Equal To) and <= (Less Than or Equal To)

CONDITIONAL STATEMENT

- if Statements
- if...else
- if...else if...else
- Nesting if Statements
- switch Statements

LOOPS

- Computer programs are usually designed to do the same thing repeatedly.
- A loop in C# code causes the computer to repeat a certain behavior several times. This could be anything from looping over every enemy in the scene and considering the AI of each to looping over all the physics objects in a scene and checking for collisions

LOOPS

- **while loop:** The most basic type of loop. Checks a condition before each loop to determine whether to continue looping.

- **do...while loop:** Similar to the while loop, but checks a condition *after* each loop to determine whether to continue looping.

LOOPS

- **for loop:** A loop statement that includes an initial statement, a variable that increments with each iteration, and an end condition. The most commonly used loop structure.

- **foreach loop:** A loop statement that automatically iterates over every element of an enumerable object or collection.

LOOPS

Jump Statements within Loops

break

Example:

```
void Start() {
        for ( int i=0; i<10; i++ ) {
            print( i );
            if ( i==3 ) {
                break;
            }
        }
    }
```

List and Array (Collection)

These collections enable you to act on several things as a group.

For example, you could loop over a List of GameObjects each frame to update all of their positions and states

List and Array (Collection)

A collection is a group of objects that are referenced by a single variable. In regular life, collections would be things like a group of people, a pride of lions, a parliament of rooks, or a murder of crows. In C#, there are two important types of collections for you to understand:

- Array
- List

List and Array (Collection)

Array: Arrays are the most primitive but fastest collection type. Arrays can only hold data of a single type, and their length must be set when they are defined.

List and Array (Collection)

List: Lists are more flexible than arrays but are still strongly typed (meaning that they can only hold one type of data). Lists are flexible in length, making them useful when you don’t know exactly how many objects will be in the collection.

Daftar Pustaka

Jeremy Gibson, Introduction to Game Design, Protoyping, and Developmen (From Concept to Playable Game – with Unity and C#), Addison Wesley – 2015