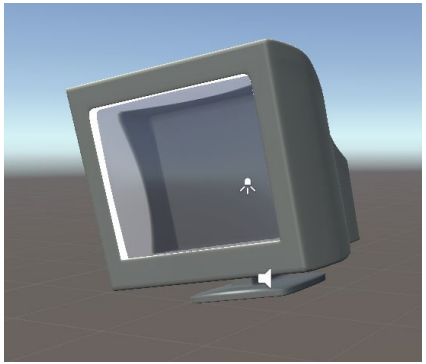
Peliohjelmoinnin alkeet Unitylla

Ohjelmointi

Sisällysluettelo

Ohjelmoinnin perusajatus



- Paljon erilaisia kieliä, mutta perusajatus pysyy samana
- Tietokone on tyhmä laite:
  - Se ei osaa tehdä mitään, jollei sille kerrota mitä sen pitää tehdä
- Meillä on joukko käskyjä, joita kone osaa suorittaa
  - Yhdistelemällä näitä käskyjä saadaan rakennettua isompia kokonaisuuksia
  - Jos käskyt ovat oikeanlaisia, niin tietokone tekee jotain mielekästä
- Kaikki käskyt ovat täysin yksiselitteisiä, eli toimivat samassa tilanteessa aina samalla tavalla
  - Tietokone ei siis kykene minkäänlaiseen “luovaan ajatteluun”, vaan toimii aina ennaltamäärätyn logiikan mukaisesti

C# ohjelmointikieli

- C# on niin sanottu olio-ohjelmointikieli (englanniksi object-oriented programming)
- Tämä tarkoittaa, että koodia kirjoitetaan luokkien sisään
  - Mutta luokat eivät vielä itsestään tee mitään
  - Sen sijaan luokista tehdään ns. olioita (objects) käyttämällä “new” sanaa
  - Luokat toimivat pohjina, joiden mukaan olioita tehdään
- Olio-ohjelmointi sisältää paljon muitakin mielenkiintoisia toimintoja
- Tämä lähinnä selityksenä siitä minkä takia kaikki kurssin koodi kirjoitetaan erilaisten luokkien sisään

C# ohjelmointikieli

- Koodirivit päättyvät puolipisteeseen “ ; “
- Koodilohkot jäsennetään aaltosuluillua “ { } “
  - Muotoilulla (rivinvaihdoilla, sisennyksillä, välilyönneillä) ei ole merkitystä koodin toiminnan kannalta C#:ssa
  - Periaatteessa on siis mahdollista kirjoittaa koko ohjelma yhdelle riville

- Luettavuuden takia kuitenkin käytetään rivinvaihtoja ja sisennyksiä
  - Visual Studio hoitaa automaattisesti suuren osan koodin muotoilusta
  - Perusperiaatteena on käyttää sisennyksiä aina kun joku osa koodista tapahtuu toisen osan sisällä, kuten esimerkiksi ehtolauseiden tai looppien sisällä

Muuttujat

- Tunnuksia/sanoja, joille voidaan antaa arvoja ja arvoja voidaan muokata ohjelman suorituksen aikana
  - Esimerkiksi: x = 10;
- Muuttujien arvoja voidaan vaihtaa esimerkiksi:
  - Käyttäjän syötteen perusteella
  - toisen muuttujan perusteella
  - laskutoimituksen seurauksena
- C#:ssa muuttujat ovat tyypitettyjä
  - Tämä tarkoittaa, että yksi muuttuja voi sisältää vain yhden tyyppistä tietoa

Muuttujatyypit

- Eri tietotyyppejä ovat esimerkiksi
  - Kokonaisluvut (integer)
    - int x = 2
  - Liukuluvut (double, float)
    - Eli tavallisemmin desimaaliluvut
    - double y = 2.6
    - float z = 0.324f
      - C#:ssa float-luvut loppuvat f-merkkiin
  - Merkkijonot (string)
    - string s = “Tämä on jono merkkejä”
  - Totuusarvot (boolean)
    - bool b = true
    - saavat vain arvoja true/false

Huom. desimaalit erotetaan pisteellä, ei pilkulla

Varatut sanat

- Muuttujia saa nimetä mielensä mukaan melko vapaasti
  - Mutta ne eivät saa olla varattuja sanoja
    - Eli sanoja jotka tarkoittavat jo jotain muuta, kuten
      - Muuttujatyypit: int, string, bool
      - Määreet: class, public, static
      - Muut komennot: if, for, new, return
      - Täysi lista varatuista sanoista: Linkki
  - Muuttujat eivät myöskään saa alkaa numerolla
    - Mutta saavat sisältää numeroita nimen lopussa, tai vaikka keskellä nimeä
  - Muuttujat eivät saa sisältää erikoismerkkejä (“_” lukuunottamatta)

Matemaattiset operaatiot C#:ssa

- C#:sta löytyy kaikki tavalliset matemaattiset toimenpiteet (+, -, /, *)
  - Tämän lisäksi % antaa jakojäännöksen (modulo)
    - Esimerkiksi: 15 % 6 = 3
  - Matemaattiset lausekkeet lasketaan noudattaen perus laskujärjestyssääntöjä, suluilla voi muuttaa järjestystä
    - 1 + 3 * 2 = 7
    - (1 + 3) * 2 = 8
- Muita matemaattisia toimintoja varten Unityssa on Mathf-kirjasto
  - Potenssit, juuret, sinit, cosinit, logaritmit, pyöristykset
  - Esimerkiksi yhdeksän neliöjuuri saadaan käskyllä: Mathf.Sqrt(9)
    - (Sqrt = squareroot = neliöjuuri)

Muuttujan arvon kasvattaminen

- Usein (etenkin loopeissa) on hyödyllistä kasvattaa muuttujan arvoa yhdellä
  - Tai jollain muulla arvolla
- Tämä voidaan saavuttaa kolmella eri syntaksilla:

`i = i + 1;`	i:n arvo kasvaa yhdellä
`i += 1;`	i:n arvo kasvaa yhdellä
`i++;`	i:n arvo kasvaa yhdellä

Muuttujan arvon muuttaminen

- Toimii myös muilla matemaattisilla operaattoreilla ja lukuarvoilla:

`i = i * 4;`	i:n arvo kerrotaan neljällä
`i /= 3;`	i:n arvo jaetaan kolmella
`i -= 2;`	i:n arvo pienenee kahdella
`i--;`	i:n arvo pienenee yhdellä

Numeeriset muuttujien tietotyypit

- Joitain kokonaislukuja
  - int
  - long
  - uint
- Joitain liukulukuja (desimaalilukuja)
  - double
  - float
  - decimal
- Eri kokonaisluku- ja liukulukutyyppien välillä on eroja, mutta käytännössä niillä ei ole vielä suurta merkitystä tässä vaiheessa ohjelmointiuraa
- Unityssa tärkeimpinä int ja float
  - C#:ssa float ilmaistaan numeron lopussa olevalla “f”:llä
  - esimerkiksi: float x = 2.5f

Mahdollisia ongelmia liittyen tietotyyppeihin

- int x = 5 / 2
  - antaa meille x = 2, ei x = 2.5
  - Tämä johtuu siitä int ei voi sisältää desimaaleja, joten kaikki desimaalipisteen jälkeinen data hylätään
  - Toisin sanottu luku pyöristetään aina alaspäin (Mathf kirjasto sisältää oikeita pyöristyksiä)
- int x = 2.5
  - Antaa virheen, älä käytä kokonaislukua (int), vaan floatia tai doublea

Kommentointi

- Yleensä on hyödyllistä kirjoittaa selkokielistä tekstiä koodin sekaan
  - Kirjoitetaan ylös mitä funktio tekee, että jonkun muun on helppo ymmärtää mitä tapahtuu
  - Tai itse ymmärrät mitä tapahtuu, jos palaat koodin äärelle vuosien jälkeen
  - Joitain muita merkintöjä jotka selventävät koodia
- Jos kuitenkin kirjoitat suomea/englantia koodin sekaan, niin ohjelma todennäköisesti ei edes käänny
- Tämän takia käytetään kommentteja
  - Kääntäjä ei käännä kommentteja ollenkaan vaan hyppää suoraan niiden yli
- C#:ssa voidaan kommentoida kahdella tavalla
  - // kommentoi tämän rivin loppuun asti
  - /* nämä merkit kommentoivat kaiken niiden välissä, voidaan kommentoida useampi rivi */
    - Hyödyllinen esimerkiksi jos halutaan nopeasti poistaa osa koodista tilapäisesti

Kommentointi - esimerkki

```
/* funktio joka palauttaa sinne annetun listan kokonaislukuja
   keskiarvon floatina */
static float KeskiArvo(List<int> numbers)
{
    //alustetaan tarvittavat muuttujat
    int summa = 0;
    float keskiArvo = 0.0f;

    //käydään läpi listan jokainen numero ja lisätään se summaan
    foreach (int number in numbers)
    {
        summa += number;
    }

    //lasketaan keskiarvo jakamalla summa numeroiden määrällä
    //koska meillä on int / int, niin se pitää castaa floatiksi
    keskiArvo = (float) summa / numbers.Count();
    return keskiArvo;
}
```

```
/* funktio joka palauttaa sinne annetun listan kokonaislukuja
   keskiarvon floatina */
static float KeskiArvo(List<int> numbers)
{
    //alustetaan tarvittavat muuttujat
    int summa = 0;
    float keskiArvo = 0.0f;

    /*
    foreach (int number in numbers)
    {
        summa += number;
    }
    */

    //hyödynnetään valmista listojen Sum() metodia
    summa = numbers.Sum();
    //lasketaan keskiarvo jakamalla summa numeroiden määrällä
    //koska meillä on int / int, niin se pitää castaa floatiksi
    keskiArvo = (float) summa / numbers.Count();
    return keskiArvo;
}
```

Esimerkeissä on tarpeettoman paljon kommentteja esimerkin vuoksi

Kommentointi - esimerkki

```
/* funktio joka palauttaa sinne annetun listan kokonaislukuja
   keskiarvon floatina */
static float KeskiArvo(List<int> numbers)
{
    int summa = 0;
    float keskiArvo = 0.0f;

    summa = numbers.Sum();
    keskiArvo = (float) summa / numbers.Count();
    return keskiArvo;
}
```

- Yleensä kommentoidaan vain oleellisimmat asiat, eli mitä funktio tekee
- Tämän lisäksi muuttujille annetaan kuvaavat nimet eli kirjoitetaan ns. “itsestään kommentoivaa koodia”

```
/* funktio joka palauttaa sinne annetun listan kokonaislukuja
   keskiarvon floatina */
static float Ka(List<int> n)
{
    int s = 0;
    float x = 0.0f;

    s = n.Sum();
    x = (float) s / n.Count();
    return x;
}
```

- Jos muuttujia ei nimetä selvästi, niin näinkin lyhyestä funktiosta voidaan saada melko sekava

Ohjelmointi - uusi taito

- Ohjelmoinnin opettelu tarkoittaa siis uuden kielen opettelua
  - Jotkin sanat muistuttavat tunnettuja kieliä (englantia)
  - Kielen oppiminen edellyttää sanojen ja peruskonseptien opettelua
  - Ohjelmointikielet perustuvat yleensä hyvin määriteltyyn syntaksiin (matematiikka ja logiikka)
- Teorian osaaminen ja ymmärtäminen
  - Ohjelmoinnissa tarvitaan sekä teorian osaamista, että käytännön ohjelmointitaitoa
    - Eli koodia pitää pystyä kirjoittamaan ja sen lisäksi on hyvä ymmärtää mitä se tekee ja miksi
  - Parhaiten oppii itse tekemällä

Tehtävästä suunnitelmaksi

- Yleensä ohjelmoinnissa lähdetään liikkeelle siitä, että meillä on ongelma, joka pitää ratkaista
- Ongelman tunnistamisen jälkeen sitä lähdetään pilkkomaan osiin, kunnes päästään niin matalalle tasolle, että se voidaan ratkaista koodaamalla
- Esimerkiksi numeroiden keskiarvon laskeminen:
  - Alusta *summa* nollaksi
  - Käy läpi jokainen luku a1, …, an ja lisää se *summaan*
  - jaa *summa* lukujen määrällä n

Suunnitelmasta ohjelmaksi

- Kun suunnitelma on tehty riittävällä tarkkuudella, niin sen kirjoittaminen ohjelmaksi on melko suoraviivaista
- Sen sijaan jos suunnitelma ei ole tarkka, niin tässä kohtaa pitää tehdä paljon työtä ja virheiden riski kasvaa