Ohjelmointikurssi – Jakso 1: Godotin perusteet ja ohjelmoinnin alkeet

1. Jakson tiedot

Tavoitteet:

- Ymmärtää, mitä ohjelmointi ja pelinkehitys tarkoittavat.
- Tutustua Godotin käyttöliittymään ja peruskäsitteisiin.
- Oppia ohjelmoinnin perussanastoa ja koodaamisen alkeet.
- Tehdä ensimmäinen oma yksinkertainen peli tai sovellus.

---

2. Ohjelmointikurssi – Syventyminen aiheeseen

Miksi oppia ohjelmointia?
Ohjelmointi on luova tapa ratkaista ongelmia ja rakentaa asioita, kuten pelejä, sovelluksia ja verkkosivustoja. Pelinkehityksessä yhdistyvät koodaus, suunnittelu ja tarinankerronta.

Godotin vahvuudet:

- Helppo oppia aloittelijalle.
- Ilmainen ja avoimen lähdekoodin.
- Tukee 2D- ja 3D-pelejä.
- Nopea koodata **GDScriptin** avulla.

Pelinkehityksen perusvaiheet:

1. **Suunnittelu:** Määrittele, millainen peli on ja miten sitä pelataan.
2. **Rakentaminen:** Luo hahmot, kohtaukset ja logiikka.
3. **Testaus:** Pelaa ja korjaa mahdolliset virheet.

---

3. Tutustuminen Godotiin

Godotin osat:

- **Scene (Kohtaus):** Pelin perusyksikkö, kuten huone, esine tai hahmo.
- **Node (Solmu):** Rakennuspalikka, jolla luodaan kohtaus (esim. Sprite, Kamera).
- **Script (Skripti):** Koodi, joka ohjaa solmujen toimintaa.

Godotin käyttöliittymän osat:

- **Scene-välilehti:** Näyttää pelin rakenteen.
- **Inspector:** Muokkaa valitun solmun ominaisuuksia.
- **FileSystem:** Kaikki projektin tiedostot.
- **Play-painike:** Testaa pelisi.

Harjoitus: Godotin käyttöliittymän tutkiminen

1. Avaa uusi projekti.
2. Tunnista Godotin eri osat (Scene, Inspector, FileSystem).
3. Lisää uusi kohtaus ja tutki, miten solmuja lisätään (esim. Sprite).

---

4. Sanasto ja harjoitukset

Sanasto (laajennettu):

Sana	Selitys
Node	Solmu, joka edustaa yksittäistä elementtiä kohtauksessa.
Scene	Kohtaus, joka koostuu yhdestä tai useammasta solmusta.
Script	Kooditiedosto, joka määrittää solmujen toiminnan.
Variable (Var)	Muuttuja, joka tallentaa tietoa.
Function (Func)	Koodipätkä, joka suorittaa tietyn tehtävän.

If-else	Ehtolause: tarkistaa, mikä toimintatapa valitaan.
Array	Lista, joka sisältää useita arvoja.
Position	Sijainti kohtauksessa (x- ja y-koordinaatit).
Sprite	2D-kuva tai hahmo.
CollisionShape	Määrittelee alueen, jossa törmäys tunnistetaan.
Input	Käyttäjän syöte, kuten näppäimistön painallus.
Process	Godotin funktio, joka päivittää joka ruudun (frame).

Sanaston tehtävät

1. **Täydennä lauseet:**
   - Solmu on pelin _____.
   - Sijainti määritetään muuttujalla _____.
   - Koodi, joka tekee tarkistuksen, on nimeltään _____.
2. **Paritehtävä:**
   Yhdistä oikeat parit numeroimalla:

Termi	Selitys
1 Node	__Lista useista arvoista.
2 Variable	_____Kooditiedosto, joka ohjaa toimintaa.
3 Array	___Pelin rakennuspalikka.
4 Script	_____Tallentaa tietoa.

3. **Kirjoita koodi:**
   Kirjoita koodi, joka:

- Luo muuttujan nimeltä "pisteet".
- Tulostaa konsoliin tekstin: *"Pelaajan pisteet: 10"*.

---

5. Harjoituksia Godotissa

Harjoitus 1: Kuution luominen (3D)

1. Avaa uusi projekti ja valitse **3D Scene**.
2. Lisää kohtaukseen **MeshInstance** ja valitse Mesh-kohdasta **CubeMesh**.
3. Lisää kuutioon väri:
   - **Material** → **New Spatial Material** → **Albedo** ja valitse väri.

Harjoitus 2: 2D-hahmon luominen ja liikuttaminen

1. Avaa uusi **2D Scene**.
2. Lisää **KinematicBody2D** ja sen alle **Sprite** sekä **CollisionShape2D**.
3. Lisää hahmolle koodi:

gdscript
Copy code

```
extends KinematicBody2D

var speed = 200

func _process(delta):
    var direction = Vector2()
    if Input.is_action_pressed("ui_right"):
        direction.x += 1
    if Input.is_action_pressed("ui_left"):
        direction.x -= 1
    move_and_slide(direction * speed)
```

4. Lisää Input Mapissa näppäimet "ui_right" ja "ui_left".

Harjoitus 3: Hahmon törmäys esineen kanssa

1. Lisää kohtaukseen toinen **StaticBody2D** ja törmäysmuoto.
2. Koodaa hahmo niin, että se pysähtyy törmätessään esineeseen.
