

Spelutveckling

med folk från TGJ





Karlstad Innovation Park
(och Stora Enso)

Spelutveckling

- Spelutvecklings-Community
- Håller evenemang i Kronoparken varje månad

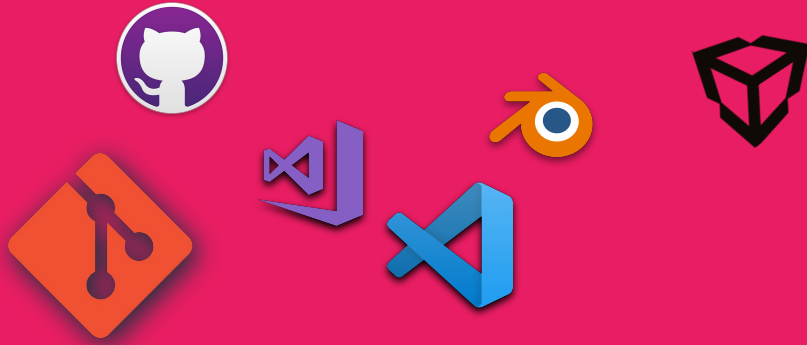
- Spelutvecklare sitter där
- Folk- och Yrkehögskola inom spel där också
- [TheGreatJourney.se](https://www.thegreatjourney.se)

Vad vi ska göra

idag och framtida
tillfällen

- få se hur spelutveckling ser ut från början till slut
- få lära er allt ni (minimalt) behöver för att **arbeta** med, **samarbeta** med, och även **lansera** ert **eget spel**
- komma igång med spelutveckling i Unity

— — —



Verktyg

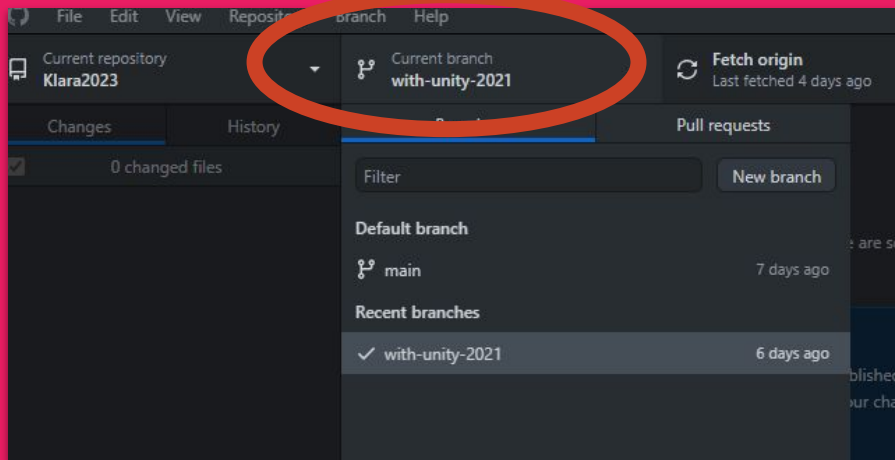
- Unity
- Visual Studio
 - eller Visual Studio Code, Rider, MonoDevelop, etc.
 - Helst med någon Unity plugin
- Blender
- Github Desktop
- Audacity eller Garage Band

Filerna igen:

[https://github.com/
ErikHogberg/Klara2023](https://github.com/ErikHogberg/Klara2023)

OBS: Lägg inte filerna i
OneDrive, Google Drive, etc.

Då går saker sönder i unity



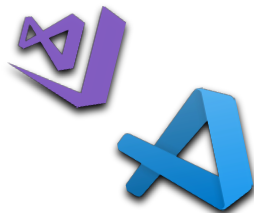
Branching

Git
multi-dimensional
time travel

- Jag har gjort en ny “branch” med hur långt vi kom förra gången
- ni som är nya kan hoppa direkt in i den
 - eller om ni inte vill använda det ni tidigare gjort

Dagens uppgifter

- Få animationerna ni gjort att spela vid knapptryckning
- Få spelkaraktären att vända sig mot dit den går



.NET

Lite ord (igen)

— — —

- **Klass/Class**
 - **Basklass/Base class**
- **Metod/Method**
 - en typ av **Funktion/Function**
- (Funktions/Metods-)anrop
- **Scope**

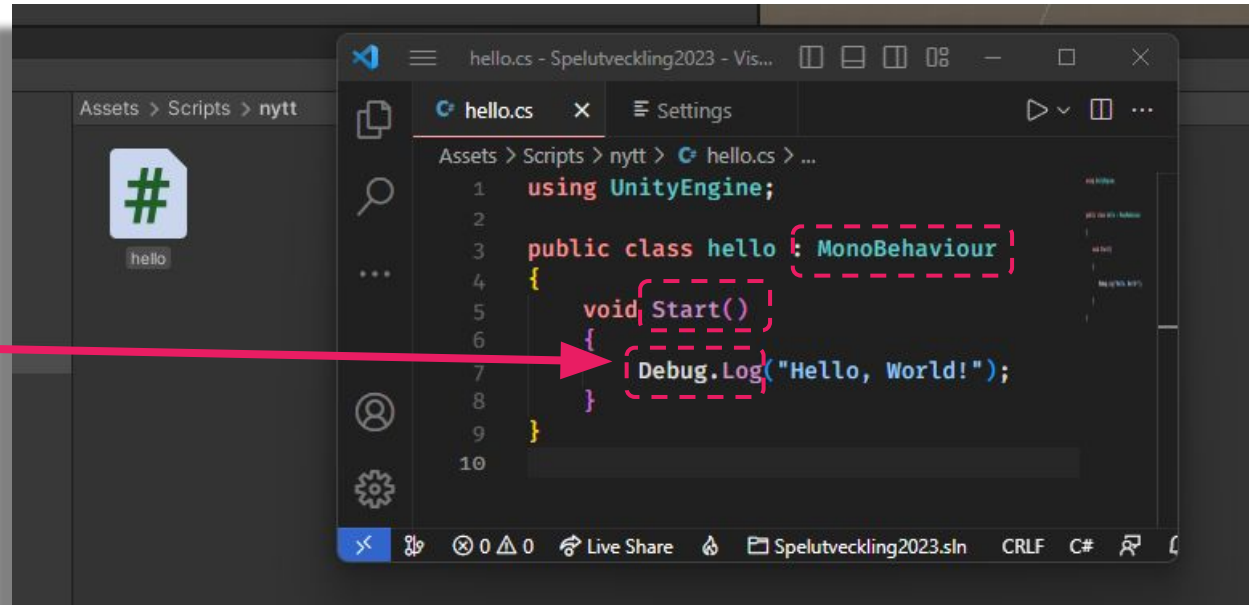
The diagram illustrates the relationship between programming concepts and their application in code. On the left, a list of concepts is provided: 'Klass/Class' (with a sub-item 'Basklass/Base class'), 'Metod/Method' (with a sub-item 'en typ av Funktion/Function'), '(Funktions/Metods-)anrop', and 'Scope'. On the right, a code snippet is shown with various parts enclosed in dashed boxes of different colors. Arrows connect the concepts to these boxed parts: a pink arrow from 'Klass/Class' points to the 'public class hello' box; an orange arrow from 'Basklass/Base class' points to the ': MonoBehaviour' box; a blue arrow from 'Metod/Method' points to the 'void Start()' box; a green arrow from 'en typ av Funktion/Function' points to the 'Debug.Log' call; a light pink arrow from 'Scope' points to the opening curly brace of the class; and a grey arrow from '(Funktions/Metods-)anrop' points to the closing curly brace of the class.

```
public class hello : MonoBehaviour
{
    void Start()
    {
        Debug.Log("Hello, World!");
    }
}
```


“Hello World”

— — —

Ett “MonoBehaviour”
i Unity med C#



“Hello World”

— — —

Ett nytt
Unity-skript med
“Hello, World!”,
innan onödigt
tagits bort

Assets > Scripts > nytt > hello.cs > hello

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  0 references
   public class hello : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      0 references
       void Start()
9      {
10         Debug.Log("Hello, World!");
11     }
12
13     // Update is called once per frame
14     0 references
       void Update()
15     {
16
17     }
18 }
```

“Hello World”

Vid knapptryckning

med denna bit kod så
sker “hello” när man
trycker “e”

det går att byta “e”
mot andra tangenter

```
public class hello : MonoBehaviour
{
    public int nummer;
    public float flyttal;
    public string text;
    public UnityEngine.Events.UnityEvent OnHello;

    void Start()
    {
        Hello();
    }

    public void Hello()
    {
        Debug.Log("Hello, World!");
        OnHello.Invoke();
    }

    private void Update()
    {
        if (Input.GetKeyDown("e"))
        {
            Hello();
        }
    }
}
```

Rotera spelaren

— — —

Rotera spelaren

— — —

nu krävs matte!

öppna “PlayerInput.cs”

```
// Update is called once per frame
void Update()
{
    Vector3 moveDirection = Vector3.zero;

    //Basic walk input
    moveDirection.x = Input.GetAxis("Horizontal") * walkSpeed;
    moveDirection.z = Input.GetAxis("Vertical") * walkSpeed;

    bool isMoving = moveDirection.magnitude > 0;

    // rotera spelaren mot rörelseriktning, bara om den rör sig
    if (isMoving)
    {
        // beräkna vinkel från X/Y riktning
        float signedAngle = Vector3.SignedAngle(Vector3.forward, moveDirection, Vector3.up);
        // vänd spelaren gradvis över tid mot den riktningen
        targetAngle = Mathf.MoveTowards(targetAngle, signedAngle, TurnSpeed * Time.deltaTime);
        transform.rotation = Quaternion.AngleAxis(targetAngle, Vector3.up);
    }

    // Jump or gravity
}
```

Rotera spelaren

— — —

Först, lägg till några fält utanför “Update()”

Dessa sparar var vi var förra gången vi uppdaterade

```
// värden för att spara data mellan varje uppdatering, för jämförelser och gradvisa förändringar  
bool wasMoving = false;  
bool wasGrounded = false;  
float targetAngle = 0;
```

Rotera spelaren

— — —

Lägg också till en inställning (som är public) så att vi kan ändra hastigheten på vändningen i Unity

Även den utanför “Update()”, t.ex. uppe vid dom andra inställningarna

```
public float jumpStrength = 7f;  
You, 2 months ago • added assets  
[Space]  
public float TurnSpeed = 1f;  
//Other variables
```

Rotera spelaren

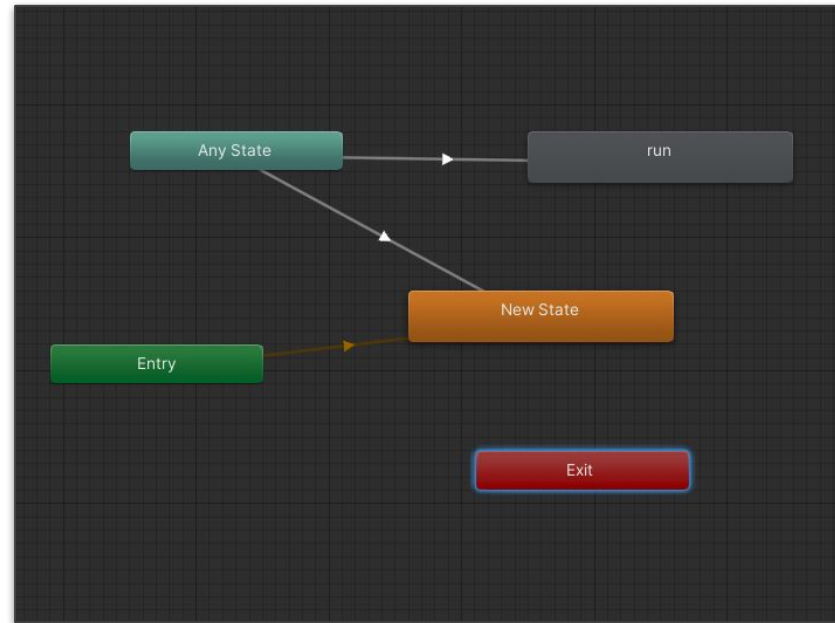
tillbaka till matten

```
// rotera spelaren mot rörelseriktning, bara om den rör sig
if (isMoving)
{
    // beräkna vinkel från X/Y riktning
    float signedAngle = Vector3.SignedAngle(Vector3.forward, moveDirection, Vector3.up);
    // vänd spelaren gradvis över tid mot den riktningen
    targetAngle = Mathf.MoveTowards(targetAngle, signedAngle, TurnSpeed * Time.deltaTime);
    transform.rotation = Quaternion.AngleAxis(targetAngle, Vector3.up);
}
```

Rotationer ändras i Unity genom en Transform

Efter rasten

Användning av animationer
i skript





The Great Journey



Karlstad Innovation Park
(och Stora Enso)

TGJ

- Spelutvecklings-Community
- Håller evenemang i Kronoparken varje månad
- kalender i vår discord och web

- Spelutvecklare
- Även Folk- och Yrkeshögskola inom spel
- TheGreatJourney.se