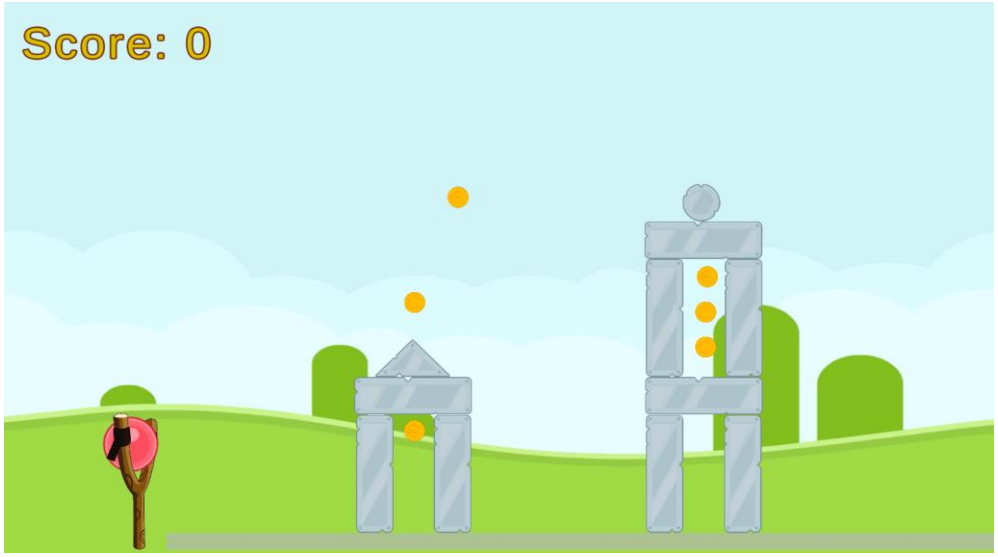
Unity Super Slingshot Guide

Workshop Coding Pirates, 2023, version 1.01
Af: Michael Hansen

Dokument og kode ligger her: https://github.com/mhfalken/unity/



Dette er en guide i, hvordan man laver et simpelt Super Slingshot spil (Angry Bird klon) i Unity, som vist på billedet.

Materialet er lavet til Børne-IT konferencen 2023 med fokus på at man kun har ca. 2 timer. Det er en forudsætning at Unity er installeret.

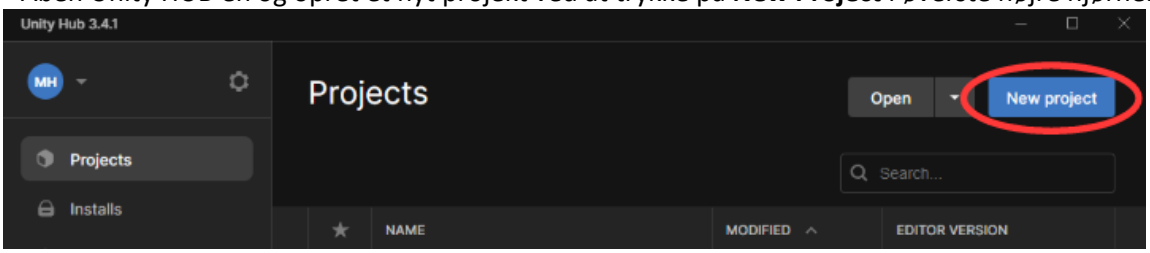
Dokumentet og koden er lavet i Unity version **2021.3**. Den burde også virker i andre versioner, men der kan være små forskelle.

1 Tutorial

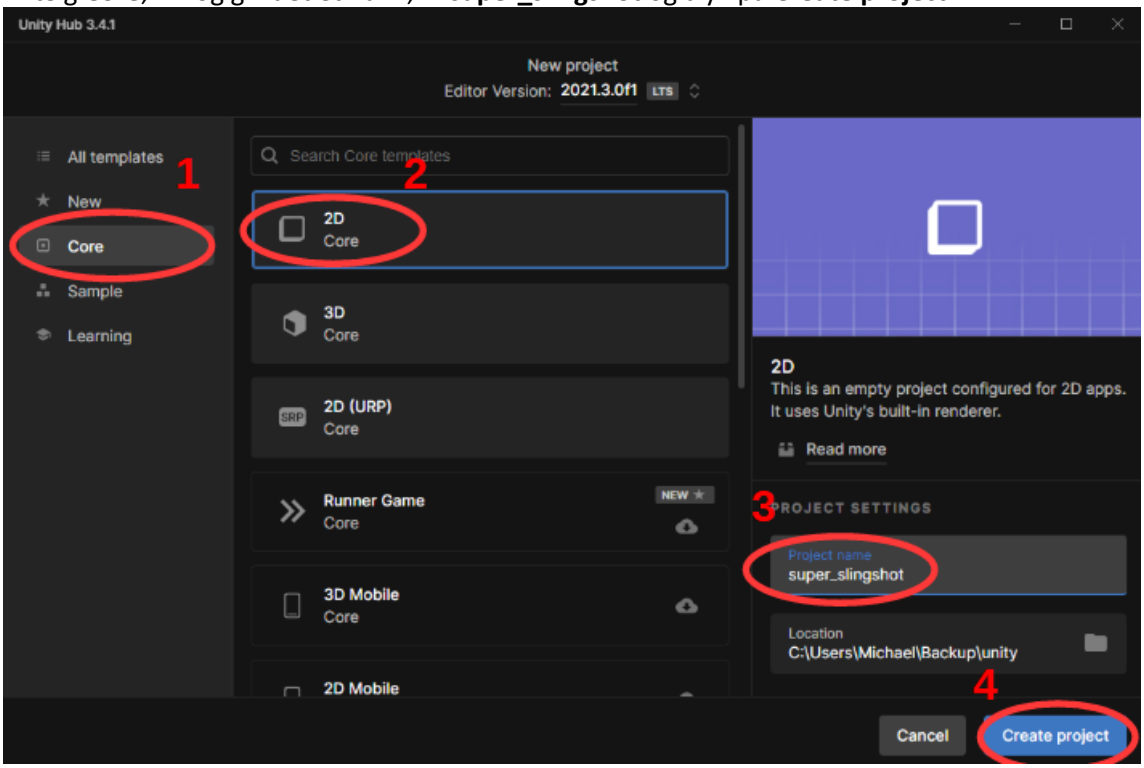
1.1 Intro

Generelt er al `C# kode markeret med denne font`, og alle referencer til **Unity GUI'en markeret med denne font**. På den måde skulle det være lidt nemmere at følge guiden.

1. Åben Unity HUB'en og opret et nyt projekt ved at trykke på **New Project** i øverste højre hjørne.



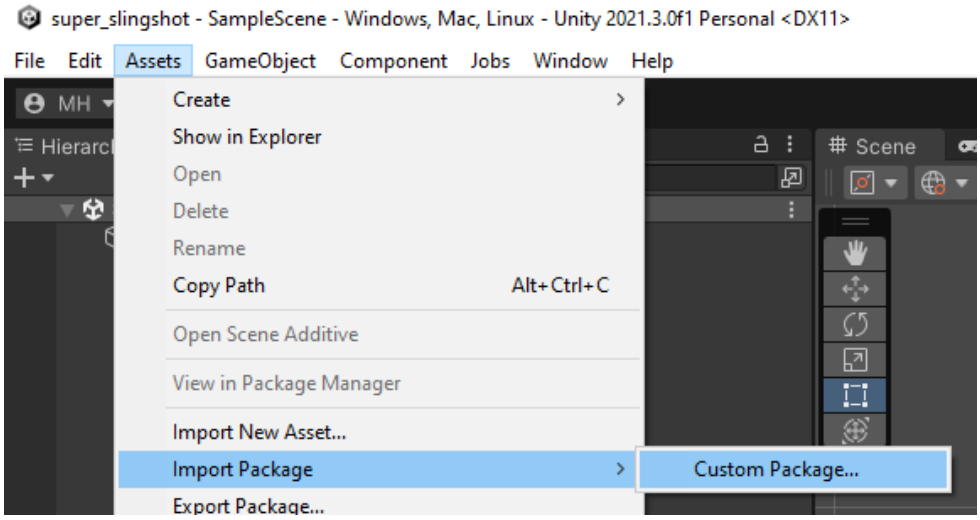
2. Vælg **Core**, **2D** og giv det et navn, fx **super_slingshot** og tryk på **Create project**.



Det tager lidt tid … Det program som kommer frem vil fremover blive kaldt **GUI'en**.

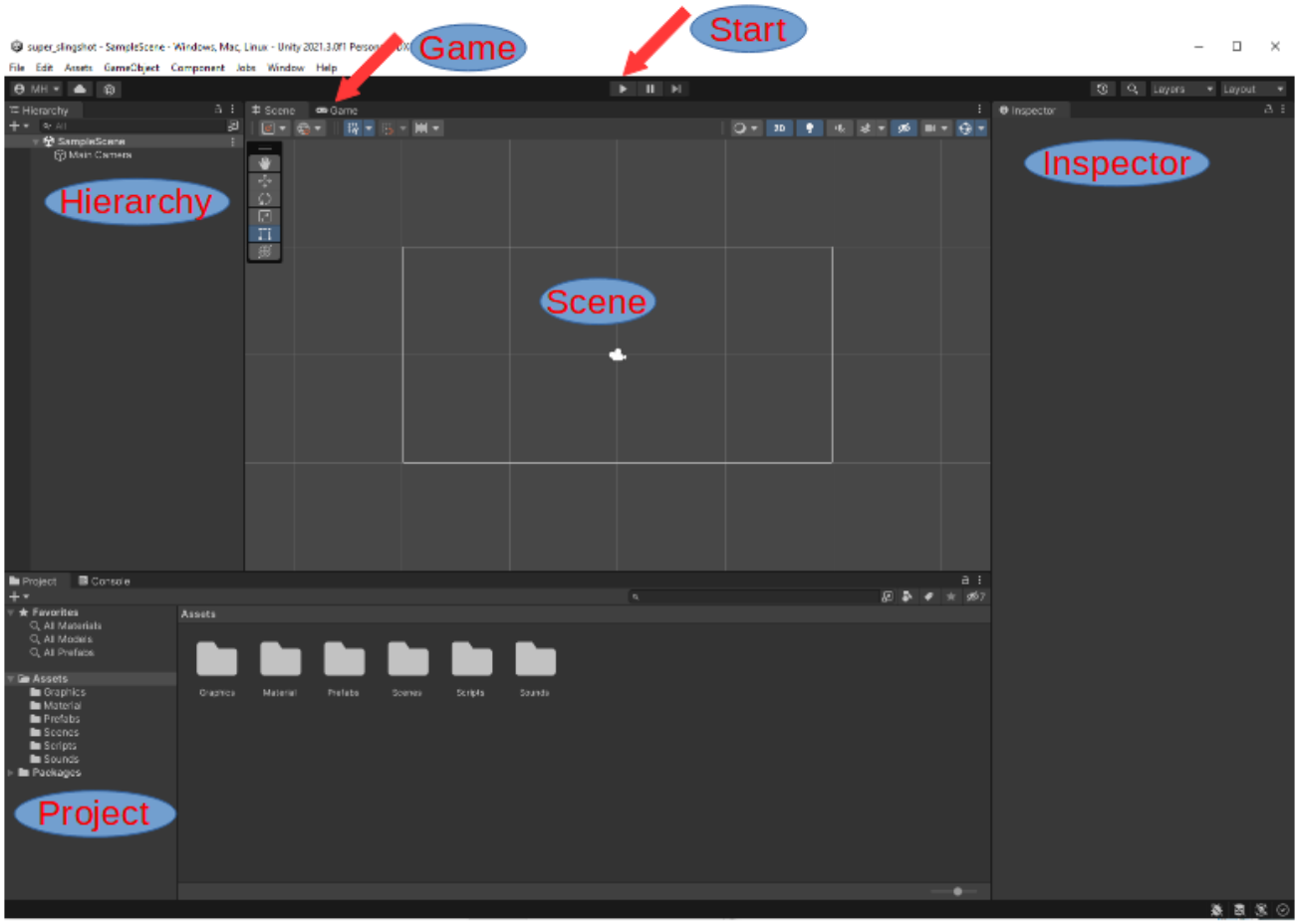
3. Det først man skal gøre er at importere den start pakke vi skal bruge. Den ligger her: https://github.com/mhfalken/unity/blob/main/super_slingshot.unitypackage
   Vælg download.

4. I GUI'en: **Assets->Import Package->Custom package** og vælg ovenstående fil (ligger typisk i Downloads/Overførsler).



5. I det vindue som kommer frem trykkes på **Import** i nederst højre hjørne. Pakken er nu importeret og ligger under **Assets**.

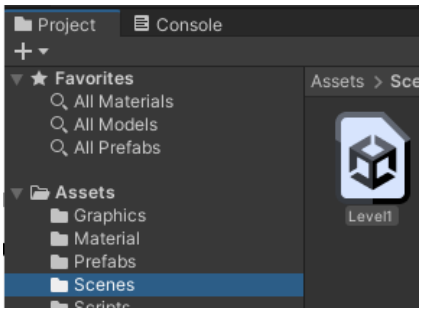
Det skal gerne se sådan ud nu.



Vindue	Beskrivelse
Scene	Her designer vi vores spil
Game	Her kan vi se hvordan spillet kommer til at se ud, med kameraets synsfelt osv (Her kan man **ikke** lave nogen rettelser)
Project	Svarende til "Windows Stifinder" for vores spil, dvs. alle filer som spillet består af kan findes her
Hierarchy	Game objekter i den åbne scene. Et spil kan bestå af flere scener, f.eks. forskellige levels
Inspector	Her vises komponenterne ("egenskaber") for et valgt game objekt

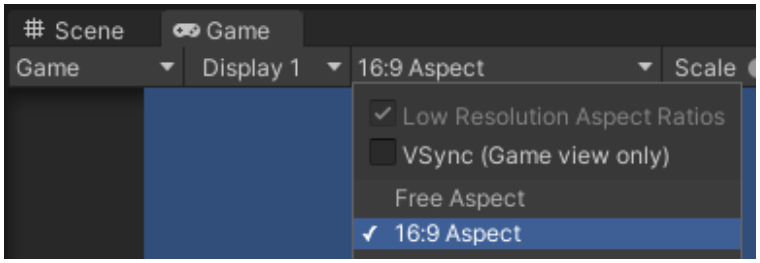
Start	Her starter man sit spil

6. Det første man skal gøre er under **Assets->Scenes** at vælge *Level1* (dobbelt klik på Level1) (se billede til højre).



7. Under **Game**, vælg **16:9 Aspect** (se billede nedenunder).

8. Prøv nu at starte spillet og prøv at skyde kuglen afsted.

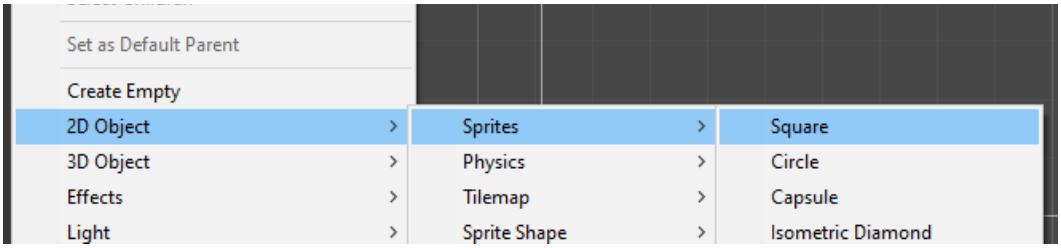


Unity har IKKE auto-save, så det er vigtigt at gemme spillet en gang imellem!

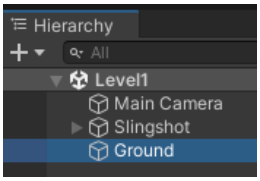
1.2 Grund

Vi skal nu til at lave en bane, hvor der kan stilles nogle klodser på, som vi kan vælte med kuglen.

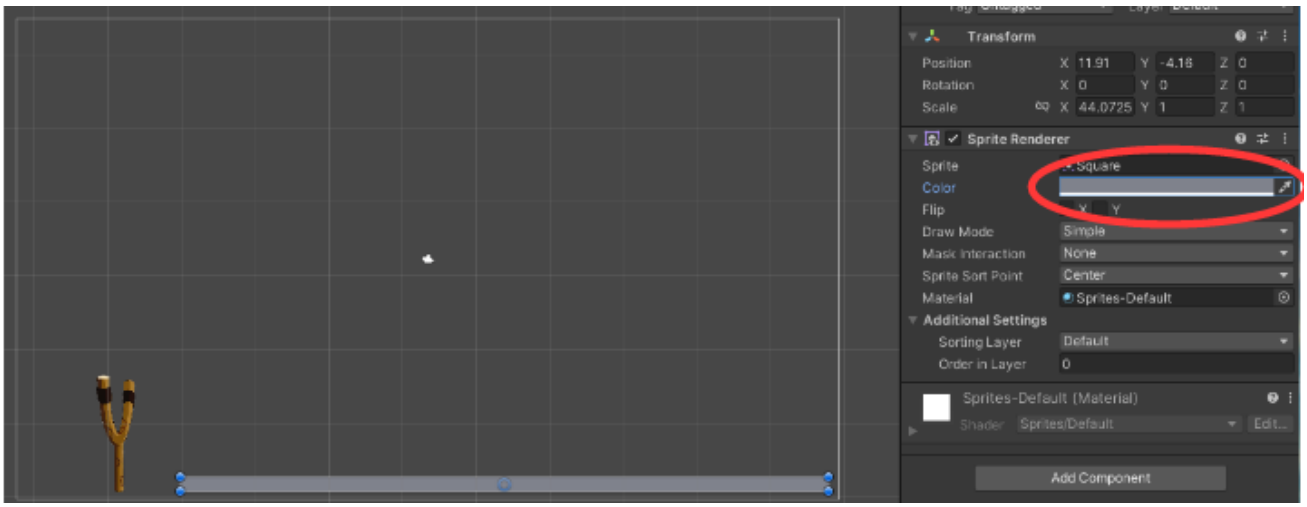
1. Højreklik i **Hierarchy**'et og vælg: **2D Object->Sprites->Square**.



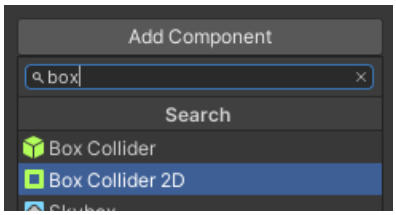
2. Ret navnet i **Hierarchy**'et fra *Square* til *Ground* (højreklik på navnet og vælg **Rename**).



3. Tag fat i firkanten (*Ground*) og flyt den ned i bunden af den hvide firkant og træk i den så den fylder hele længden (se bilede).
4. I **Inspector**'en (ude til højre) tryk på **Color** og vælg en farve til *Ground*.



Prøv spillet og få kuglen til at ramme Ground - den ryger lige igennem! Det er fordi vi ikke har fortalt Unity at kuglen skal kunne 'ramme' *Ground*.



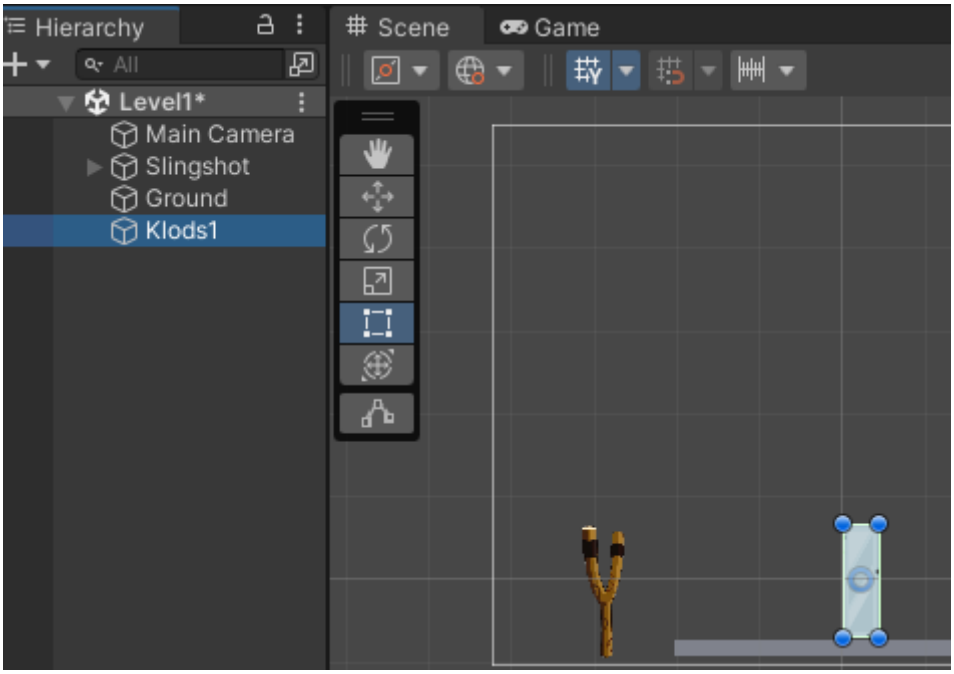
1. Vælg *Ground* i **Hierarchy**'et
2. I **Inspector**'en (ude til højre i bunden) tryk på **Add Component**
3. Skriv *box* og vælg **Box Collider 2D**

Start spillet og se at kuglen nu kan ramme *Ground.*

1.3 Klodser

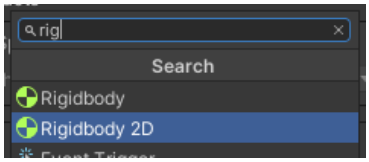
Vi skal nu lave nogle klodser som vi kan stable og lave en bane ud af, som vi så senere kan vælte. Under **Graphics** ligger der forskellige billeder af klodser som man kan bruge.

1. Under **Projects**, Vælg **Assets->Graphics**
2. Vælg en klods og træk den ind i **Scene**n så den 'står' på *Ground*.
3. Ret navnet i **Inspector**'en til *Klods1*.
4. I **Inspector**'en vælg **Add Component** og vælg **Box Collider 2D** (som for *Ground*).

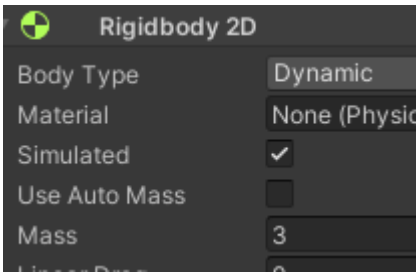


Prøv spillet og se om man kan vælte klodsen. Det kan man ikke, da vi ikke har fortalt Unity at den skal 'opføre' sig som en 'normal' fysisk klods.

5. Vælg *Klods1* og i **Inspector**'en, tryk **Add Component** og vælg **Rigidbody 2D**

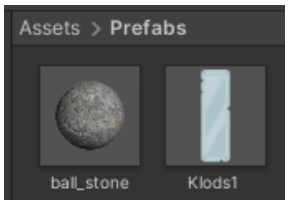


Prøv spillet og se at man nu kan vælte klodsen.
Hvis man synes klodsen virker for 'let' i det kan man under **Ridigbody 2D** (ude til højre) ændre **Mass**.



Vi skal nu lave en Prefab af klodsen, så vi kan genbruge klodsen uden at skulle lave en ny hver gang.

6. I **Hierarchy**'et, træk *Klods1* ned i **Assets->Prefabs** folderen.



Man kan nu trække en 'ny' klods fra **Prefabs** folderen ind i **Scenen** og på den måde bygge en bane. Det er dog lidt kedeligt kun at have en type klods.

7. Lav nogle flere forskellige klodser på samme måde som vi lavede *Klods1*.
   Husk at lave Prefabs af dem når de er færdige.
8. Byg en lille bane (noget som kan vælte) og se at det hele virker som det skal.
   Vær opmærksom på at så snart spillet starter, så begynder Unity at regne på fysikken, dvs. at hvis det man har bygget ikke er i balance, så kan det godt vælte af sig selv.

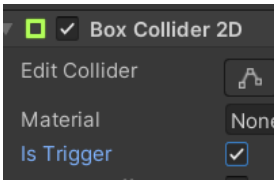
Ekstra: Man kan for hver klods under **Inspector->Ridgetbody 2D** ændre **Mass**, så klodserne opfører sig forskelligt.

1.4 Score

1.4.1 Mønter

Vi skal nu have tilføjet nogle penge så vi kan få nogle point.

1. Under **Assets->Graphics** ligger der nogle animeret mønter – *EUR_6x4*.
   Træk dem ind i **Scene**n og i den dialogboks som kommer frem, kald dem *Euro* og tryk **Save**.
2. I **Inspector**'en tryk **Add Component** og vælg **Circle Collider 2D**
3. I **Inspector**'en under **Circle Collider 2D**, skal man vælge **Is Trigger**.



Hvis man kører spillet nu, så kan man se mønten, men der sker ikke noget når man rammer den. Vi skal derfor nu tilføje noget kode, så der sker noget når man rammer mønten.

4. Vælg folderen **Assets->Scripts**
5. Højreklik i folderen og vælg **Create->C# Script** og kald det *Items*
   (Det er vigtigt at man kalder filen det rigtige navn første gang, da man **IKKE MÅ RENAME** filen senere!)
6. Træk *Items* scriptet op over *EUR_6x4* i **Hierarchy**'et

Vi skal nu til at skrive noget kode. Her er to muligheder: Man kan enten selv taste koden ind eller man kan 'bare' kopiere den fra eksemplet.
Her er lidt genvejstaster som man kan få brug for – specielt hvis man har en MAC, hvor tasterne kan være lidt svære at finde.

Gode genvejstaster: (CTRL = CMD på MAC)

Specielle MAC taster:

Tast	Funktion
CTRL-S	Gem filen
CTRL-Z	Undo
CTRL-C	Kopier
CTRL-V	Indsæt

Tast	Funktion
ALT-8	[
ALT-9	]
SHIFT-ALT-8	{
SHIFT-ALT-9	}

7. Åben det nye *Item* script (dobbelt klip på det – nu skal Visual Studio gerne starte op) og tilføj følgende linjer i bunden af filen lige inden den sidste '}':

```
private void OnTriggerEnter2D(Collider2D collision)
{
  Destroy(gameObject);
}
```

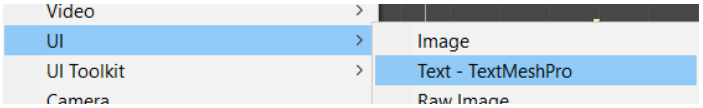
8. Gem filen

Prøv og kør spillet nu, og se at mønten forsvinder når noget rammer den.

9. Lav en **Prefab** af mønten (træk den fra **Hierarchy**'et ned i **Prefabs** folderen).

1.4.2 Tekst

Vi skal nu have tilføjet en score tekst på skærmen, så vi kan se hvor mange point vi har.



1. I **Hierarchy**'et, højreklik og vælg **UI->Text - TextMeshPro**
2. I pop op-vinduet, tryk på **Import TMP Essentiel** og luk derefter vinduet
3. I **Hierarchy**'et rename *Text (TMP)* til *Score*
4. Flyt teksten op i øverste venstre hjørne (se billede)

5. I **Inspector**'en ret teksten til *Score*
6. Ret farven, størrelsen mv. så det ser godt ud

Kør spillet og se at der står *Score* i øverste venstre hjørne.

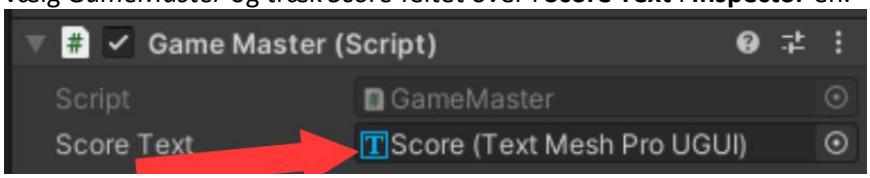
1.4.3 Point

Vi skal nu lave en Gamemaster til at holder styr på vores point.

1. I **Hierarchy**'et, højreklik og vælg **Create Empty** og kald det *GameMaster*
2. I **Scripts** folderen, højreklik og lav et **C# Script** og kald det *GameMaster*
3. Træk *GameMaster* scriptet op over *GameMaster* i **Hierarchy**'et
4. Åben *GameMaster* scriptet og ret det så det ser sådan ud (Linjerne med **fed** er de nye linjer):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class GameMaster : MonoBehaviour
{
    [System.NonSerialized] public int score = 0;
    [SerializeField] TMP_Text scoreText;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        scoreText.text = "Score: " + score;
    }
}
```

5. Gem filen.
6. I **Hierarchy**'et vælg *GameMaster* og træk *Score* feltet over i **Score Text** i **Inspector**'en.



Nu er alt klar til at vi kan score nogle point, når vi tager en mønt.

7. Åben *Items* scriptet og tilføj følgende linje i toppen af **OnTriggerEnter2D()** funktionen (se billedet):

```
GameObject.Find("GameMaster").GetComponent<GameMaster>().score += 100;
```

```
private void OnTriggerEnter2D(Collider2D collision)
{
    GameObject.Find("GameMaster").GetComponent<GameMaster>().score += 100;
    Destroy(gameObject);
}
```

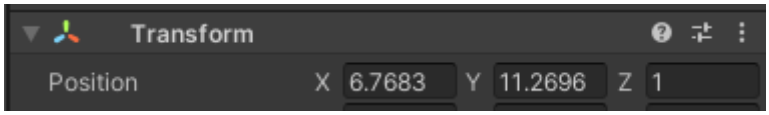
Kør spillet og se at man scorer point når noget rammer mønten. Tilføj nogle flere mønter til spillet – husk at bruge dem fra **Prefabs** folderen.

8. Gem spillet (Ctrl+S)

1.5 Baggrund

Vi skal nu have vores spil til at se lide bedre ud ved at tilføje et baggrundsbillede.

1. Tag et billede fra **Assets->Graphics** og træk det ind i **Scene**n
2. Ret størrelsen så det fylder hele skærmen (når spillet kører). Kameraet kan 'se' alt i den 'lille' hvide firkant.
3. I **Inspector**'en skal man rettet **Position Z** til 1.



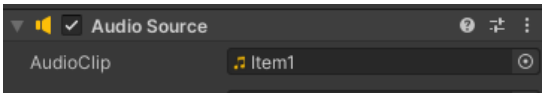
Man kan også finde et billede selv, hvis man har lyst til det. Det skal bare trækkes ind i **Asset->Graphics**, så kan det bruges i spillet.

1.6 Lyd

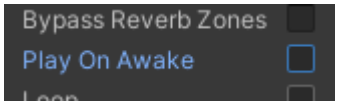
Vi skal nu have lidt lyd på spillet, når man scorer nogle point, dvs. tager nogle mønter.

1. Åben vores mønt i **Prefabs** folderen. (Vælg **Assets->Prefabs** og dobbelt klik på *EUR_6x4*)

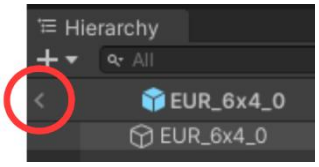
2. I **inspector**'en tryk **Add Component** og vælg **Audio Source**.
3. I folderen *Sounds*, træk lydfilen *Item1* over i **Inspector**'en i **AudioClip** (se billede)



4. Fjern fluebenet i **Play On Awake**.



5. I **Hierarchy**'et tryk på '<' (se billedet) for at komme tilbage.



6. I *Items* scriptet, ret koden så den ser sådan ud (**fed** er nye linjer):

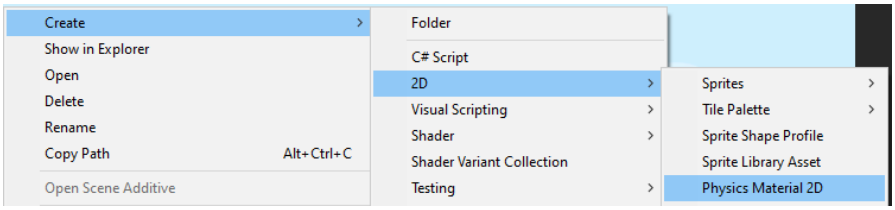
```
AudioSource audioSrc;
// Start is called before the first frame update
void Start()
{
    audioSrc = GetComponent<AudioSource>();
}
// Update is called once per frame
void Update()
{
}

private void OnTriggerEnter2D(Collider2D collision)
{
    GameObject.Find("GameMaster").GetComponent<GameMaster>().score +=
100;
    audioSrc.Play();
    GetComponent<SpriteRenderer>().enabled = false;
    GetComponent<CircleCollider2D>().enabled = false;
    Destroy(gameObject, 1);
}
```

Læg mærke til at der i sidste linje også er tilføjet et **1** tal. Det er vigtigt da lyden ellers bliver 'slået' ihjel for hurtigt.
Prøv spillet og hør at lyden virker, når man scorer point.

1.7 Ekstra kugle

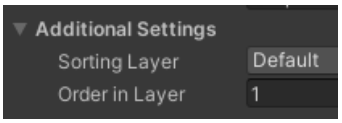
Lige nu er det den samme kugle man har hver gang. Vi skal nu lave en ny kugle som ser anderledes ud og som også har nogle andre egenskaber.
Vi skal først have lavet en fysisk egenskab til vores nye kugle.

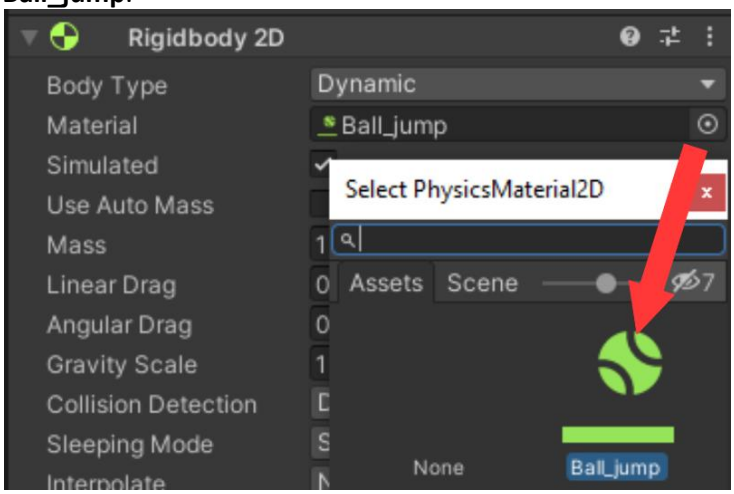
1. Åben folderen **Assets->Material**
2. Højreklik og vælg **Create->2D->Physical Material 2D** (se billede), og kald den Ball_jump



3. Vælg *Ball_jump* og i **Inspector**'en ret **Bounciness** til 0.8.

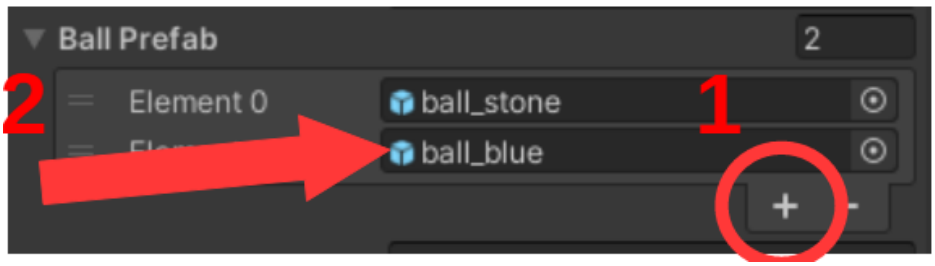
4. I **Assets->Graphics** find den blå kugle og træk den ind i **Scene** og placer den lidt over *Ground* linjen.
5. I **Inspector**'en ret **Order in Layer** til 1. (se billede)
6. Brug **Add Component** og tilføj **Ridgetbody 2D** og **Circle Collider 2D**.
7. I **Inspector**'en under **Ridgetbody 2D** tryk på 'cirklen ud for **Material** (se billede) og vælg **Ball_jump**.





Kør spillet og se at bolden hopper.

8. Træk scriptet *Ball* op over *ball_blue*. (**Assets->Scripts**)
9. Vælg *ball_blue* i **Hierarchy**'et og træk den ned i **Prefabs** folderen.
10. Slet derefter *ball_blue* i **Hierarchy**'et, da vi kun skal bruge Prefab'en.
11. I **Hierarchy**'et vælg *Slingshot* og i **Inspector**'en under **Ball Prefab** tryk på **+** (se billede - 1)
12. Træk derefter den *ball_blue* vi lige har lavet fra **Prefab** folderen over i **Element1**. (se billede - 2)



Prøv nu spillet og se at kuglerne skifter efter hvert skud.

1.8 Lyd – ekstra

Hvis man har lyst kan man også tilføje lyd til klodserne når det vælter. Det kræver at man laver et nyt script, som regner på om klodserne bevæger sig og så afspiller en lyd.

1. Opret et nyt script under **Assets->Scripts** og kald det *Blocks*
2. Åben scriptet og ændre koden til:

```
public class Blocks : MonoBehaviour
{
    AudioSource audioPlayer;
    Rigidbody2D rb;
    bool hit;

    // Start is called before the first frame update
    void Start()
    {
        audioPlayer = GetComponent<AudioSource>();
        rb = GetComponent<Rigidbody2D>();
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        float power = rb.velocity.magnitude;
        if (power > 3 && (hit == false))
        {
            audioPlayer.Play();
            hit = true;
        }
        if (power < 1 && hit)
        {
            audioPlayer.Play();
            hit= false;
        }
    }
}
```

3. Gem filen
4. Under **Assets->Prefabs**, åben en af klodserne og tilføj komponenten **Audio Source**.
5. Under **Assets->Sounds**, træk *Plastic-Hit* lyden over i **AudioClip**
6. Fjern fluebenet i **Play On Awake**.
7. Træk scriptet *Blocks* op over klodsen i **Hierarchy**'et.

(Kik eventuelt under Lyd afsnittet tidligere for mere info om hvordan man gør)
Prøv spillet og hør om det virker. Step 3-6 skal gøres for alle klodserne.
Husk at gemme spillet en gang imellem.

2 Næste steps

Her er en kort beskrivelse af hvordan man kan forsætte spillet. Her skal man selv lave mere og vejledningen er derfor noget mere overordnet.

2.1 Ekstra baner

Vi skal nu lave spillet, så det kan have flere baner.

1. Åben **File->Build Settings** og tryk på **Add Open Scenes**
2. Fjern 'fluebenet' ud for **Scenes/SampleScene** og luk derefter vinduet.
3. I **Scenes** folderen kopier *Level1* til *Level2* og åben Level2.
4. Åben **File->Build Settings** og tryk på **Add Open Scenes** og luk derefter vinduet.
5. Ændre baggrunden i Level2, så det er let at se forskel på *Level1* og *Level2*. Gem og skift til *Level1*.

Vi skal nu lave den kode som afgør hvornår man skal skifte fra *Level1* til *Level2*. Det kan gøres på mange forskellige måder – nogle mere avanceret end andre. Vi starter med en meget simpel måde, som så senere gøres lidt mere avanceret.

6. I **Scripts** folderen åben *Slingshot* scriptet. Det er det script som styrer kuglen.

Vi skal lave en simpel tæller, så man har 3 skud pr. bane og så skifter man til næste bane uanset om man er færdig eller ej.

7. Lav en integer variabel som hedder *ballCnt*. (`int ballCnt`)
8. Indsæt følgende linjer (linjerne med **fed** er de nye):

```
using UnityEngine;
using UnityEngine.SceneManagement;

...

void NextLevel()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
}

void Createball()
{
    if (ballCnt == 3)
        Invoke("NextLevel", 0);
    ballCnt++;
...
```

(Grunden til at der er brugt `Invoke(…)` er, at koden senere bliver lettere at lave.)

Prøv spillet og se at man skifter til næste bane efter 3 skud.

2.1.1 Overfør point

Som det ses, så taber man alle sine point, når man skifter bane. Hvis man ikke ønsker det kan det løses på følgende måde.

9. For *Level1* tilføjes et tomt objekt, som man kalder *Init* og laver et C# script som hedder *Init* som trækkes op over *Init* objektet.
10. I *Init* scriptet skriver man følgende (det sletter 'alt' første gang):

```
private void Awake()
{
    PlayerPrefs.DeleteAll();
}
```

I *GameMaster* scriptet skriver man følgende:

```
// i Start()
score = PlayerPrefs.GetInt("Score");

// i Update()
PlayerPrefs.SetInt("Score", score);
```

Den sidste linje gemmer *Score* og den første linje henter *Score*.

Test at ens point overlever fra *Level1* til *Level2*.

2.1.2 Bonus hvis bolde tilbage

Vi skal nu udvide vores betingelse for at gå til næste bane, så man også skifter til næste bane når der ikke er flere mønter at tage.

11. Åben **Prefab**en for *EUR_6x4* og tilføj et tag som hedder *Coin*. (Toppen af **Inspector**'en)
    Vi skal bruge tag'et til at se om der er flere mønter tilbage.
12. Udvid koden i *Slingshot* `Createball()` med følgende linjer (**fed** er nye linjer)

```
if (ballCnt == 3)
    Invoke("NextLevel", 0);
if (!GameObject.FindGameObjectWithTag("Coin"))
{
    Invoke("NextLevel", 2);
}
ballCnt++;
```

OBS: Hvis man skriver koden i "hånden", så husk **!** i starten af anden `if` sætning.

Prøv koden og se at man går videre til næste bane, når der ikke er flere mønter tilbage.

Husk, at når man tilføjer flere baner skal man huske at også tilføre dem i **Build Settings**, som vist tidligere.

13. Lav nu nogle flere baner så det ligner mere et rigtig spil.

Forslag til flere features:

1. Giv ekstra point (bonus) for hver bold som er tilbage.
2. Afspil en speciel lyd, når man får bonus.

3 Links

Tips videoer	TOP 10 UNITY TIPS - 2017 TOP 10 UNITY TIPS #2
Unity manual	https://docs.unity3d.com/2021.3/Documentation/ScriptReference/index.html
Unity assets store	https://assetstore.unity.com
Dansk C#/Unity document	https://github.com/Grailas/CodingPiratesAalborg/blob/master/Guides/Hj%C3%A6lpeguide.pdf