

Unity Basketball Guide

Workshop Coding Pirates, 2022, version 0.30

Af: Michael Hansen, Ide: Rasmus Selsmark

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



Dette er en tutorial i, hvordan man laver et lille 2D basketball spil i Unity, som vist på billedet.

Materialet er blevet lavet til Børne-IT konferencen 2022 med fokus på at man kun har 2 timer. Det er en forudsætning at Unity er installeret. På GitHub ligger den fulde "løsning" som en pakket fil

https://github.com/mhfalken/unity/blob/main/basket_final.zip

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

1	Tutorial	2
1.1	Intro	2
1.2	Basketball bold	3
1.3	Gulv	3
1.4	Fysik og tyngdekraft	4
1.5	Fysisk materiale	4
1.6	Boldkontrol	5
1.7	Kurv	7
1.8	Score	9
1.9	Tidsbegrænsning	12
1.10	Lyd	13
2	Links	14

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.

Åben Unity HUB'en og opret et nyt projekt ved at trykke på **New Project** i øverste højre hjørne. Vælg **2D** og giv det et navn, fx **basketball** og tryk **Create project**. Det tager lidt tid ... Det program som kommer frem vil fremover blive kaldt **GUI'en**.

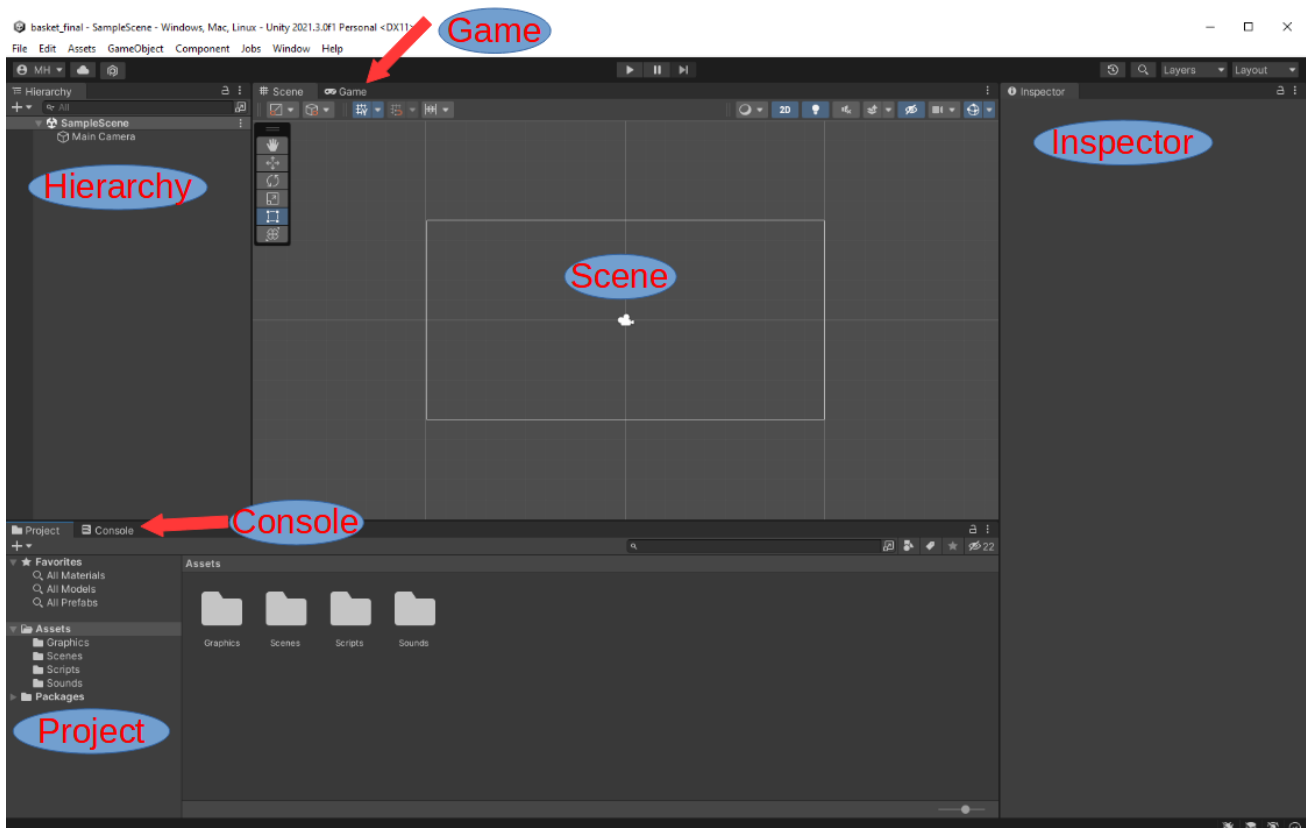
Det først man skal gøre er at importere den start pakke vi skal bruge. Den ligger her:

https://github.com/mhfalken/unity/blob/main/basket_start_pakke.unitypackage

Vælg download og gem filen et sted du kan finde igen.

I GUI'en: **Assets->Import Package->Custom package** og vælg ovenstående fil. I det vindue som kommer frem trykkes på **Import** i nederst højre hjørne. Pakken er nu importeret og ligger under **Assets**.

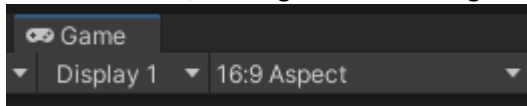
Det skal gerne se sådan ud nu.



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
Console	Hvis der er fx fejl i koden, vises fejlbeskeder her

1.2 Basketball bold

Inden vi starter, så vælg **Game** fanen og vælg **16:9 Aspect** – se billede.

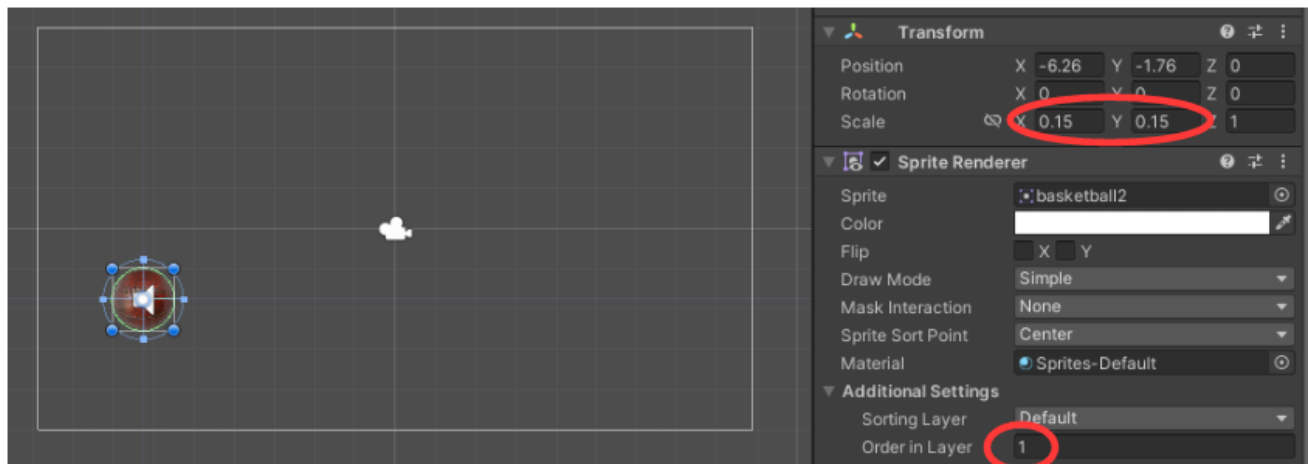


Skift derefter tilbage til **Scene** vinduet.

I **Assets/Graphics** ligger der nogle billeder af basketballbolde.

1. Tag basketball2 og træk det ind i **Scene** vinduet. Den hvide ramme i **Scene** vinduet angiver kameraets synsfelt, dvs. hvad der er synligt i spillet.
2. Skaler basketballen til 0.15 for X og Y så den bedre passer i størrelsesforholdet.
3. Set **Order in Layer** til 1.

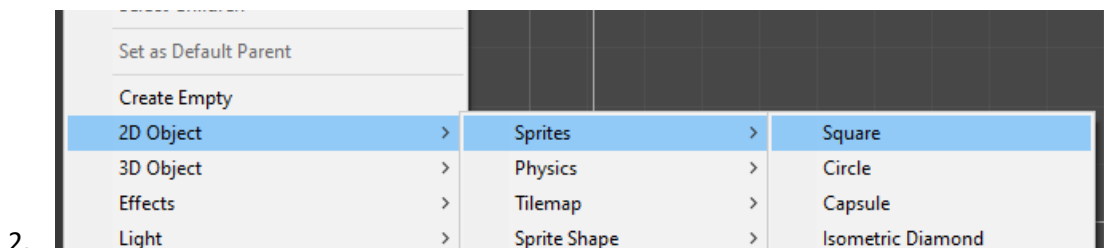
Det skal se ca. sådan ud:



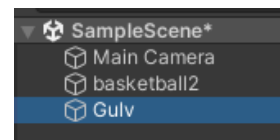
1.3 Gulv

Vi skal nu lave et gulv, som bolden kan hoppe på.

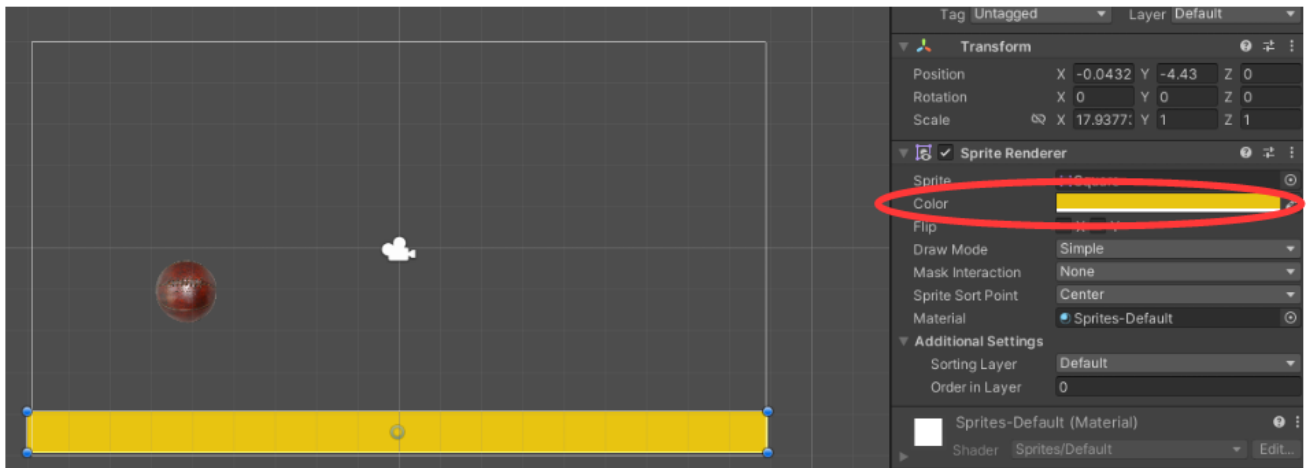
1. Højreklik i **Hierarchy**'et og vælg: **2D Object->Sprites->Square**.



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



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



1.4 Fysik og tyngdekraft

Vi skal nu have vores bold til at bevæge sig realistisk, dvs. efter normale fysiske regler.

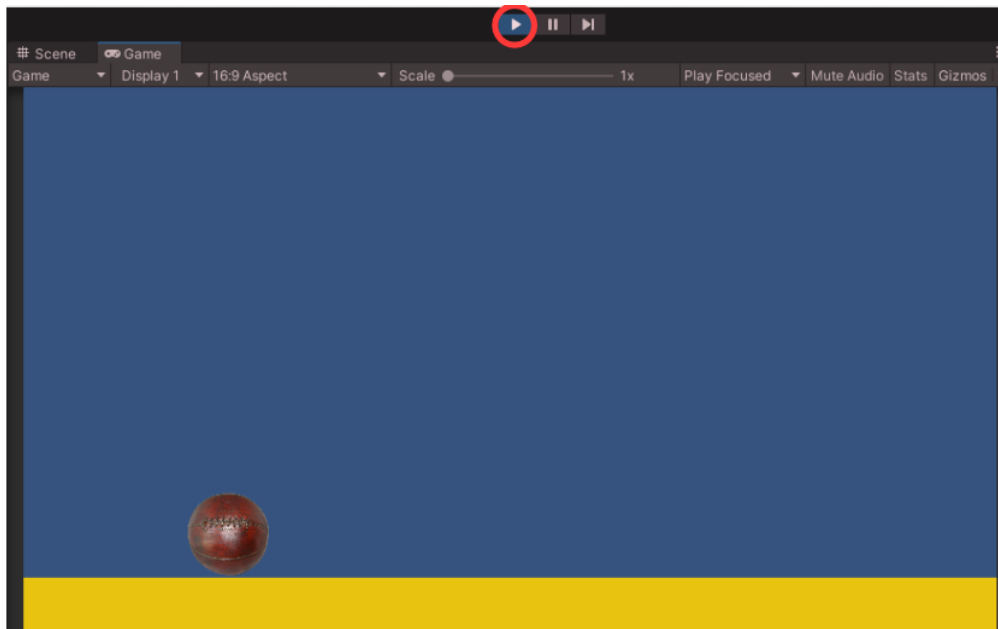
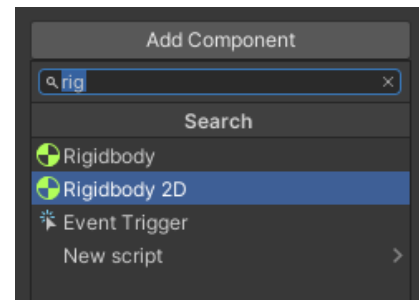
I **Hierarchy**'et vælg *basketball* objektet og i **Inspector**'en tryk på **Add Component** og skriv 'rig' og vælg **Rigidbody 2D**.

Herefter vil bolden opføre sig som 'fysisk', dvs. blive påvirket af tyngdekraften.

Derefter skal man tilføje en **Circle Collider 2D** (**Add Component**). Den gør at objektet kan kollidere med andre objekter.

Vælg *Gulv* og tilføj en **Box Collider 2D** på samme måde.

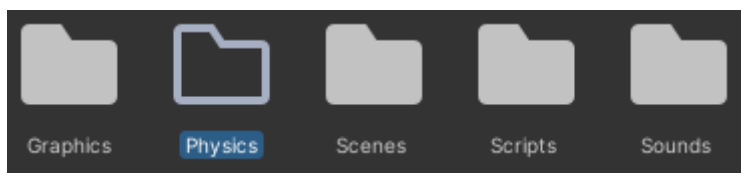
Tryk på **Play** knappen i toppen af skærmen og se at bolden falder ned og lægger sig på gulvet.



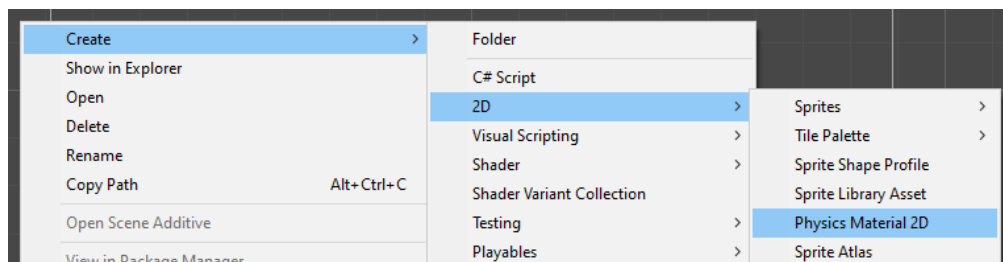
1.5 Fysisk materiale

For at få bolden til at virke mere realistisk, så den kan hoppe når den rammer gulvet, skal vi arbejde med fysisk materiale.

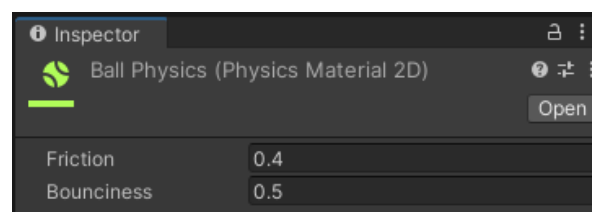
I **Project** vinduet vælg **Asset** folderen og højreklik i vinduet til højre og tilføj en folder som hedder *Physics*.



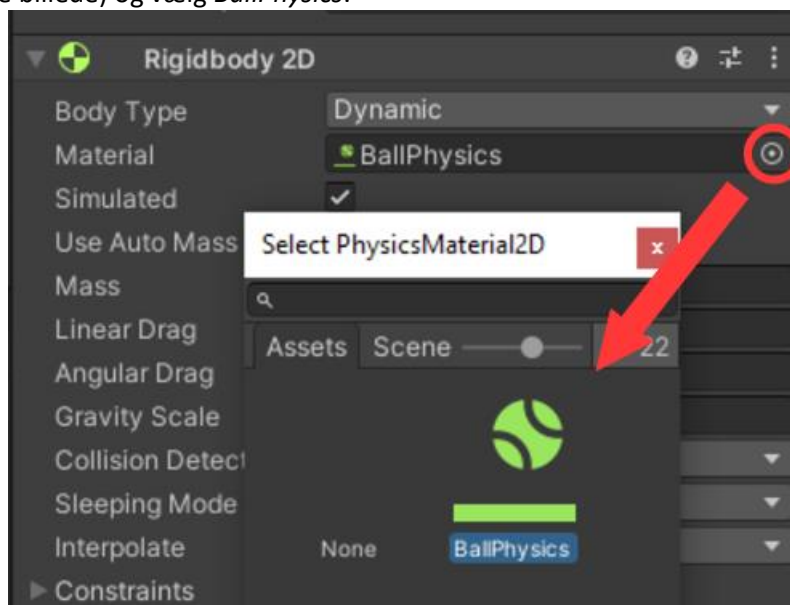
Højreklik i mappen *Physics* og vælg **Create->2D->Physics Material 2D** og giv det navnet *BallPhysics*.



I **Inspector**'en ret **Bounciness** til 0.5, hvilken gør at bolden hopper 50% af den højde den havde før den blev sluppet.



I **Hierarchy**'et vælg *basketball* objektet. I **Inspector**'en find feltet **Rigidbody 2D->Material** og tryk på den lille cirkel ude til højre (se billede) og vælg *BallPhysics*.



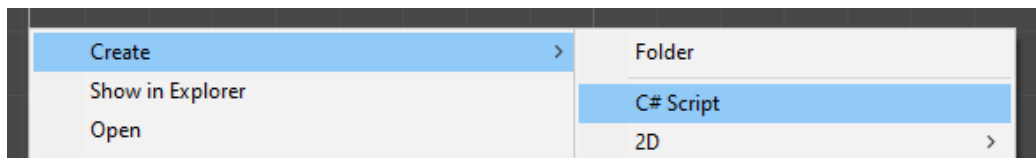
Når man kører spillet nu, kan man se at bolden hopper lidt på gulvet.

1.6 Boldkontrol

Vi skal nu tilføje noget kode som gør at vi kan styre bolden vha. musen.

Under **Assets** folderen lav en folder som hedder *Scripts*.

I **Assets/Scripts** folderen højreklik og opret en **C#** fil og kald den *Basketball*.



(Det er vigtigt at man kalder filen det rigtige navn første gang, da man **IKKE MÅ RENAME** filen senere!)

Træk filen op over *basketball* i **Hierarchy**'et.

Åben filen ved at dobbelt klikke på den. (Den åbner i den ekstern editor.)

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)

Tast	Funktion
CTRL-S	Gem filen
CTRL-Z	Undo
CTRL-C	Kopier
CTRL-V	Indsæt
CTRL-H	Søg/erstat

Specielle MAC taster:

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

Filen indeholder allerede noget kode, som vi ikke skal bruge nu, så det meste kan slettes. Koden skal se sådan ud:

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

public class Basketball : MonoBehaviour
{
    private void OnMouseUp()
    {
        GetComponent<Rigidbody2D>().AddForce(Vector2.up * 10.0f,
            ForceMode2D.Impulse);
    }
}
```

Prøv at køre spillet og tryk på bolden med musen og se at den hopper op.

Vi skal nu have bolden til at bevæge sig i den retning vi bevæger musen, så vi ændrer lidt i koden så den ser sådan ud:

```
public class Basketball : MonoBehaviour
{
    Vector3 startOfMovement;

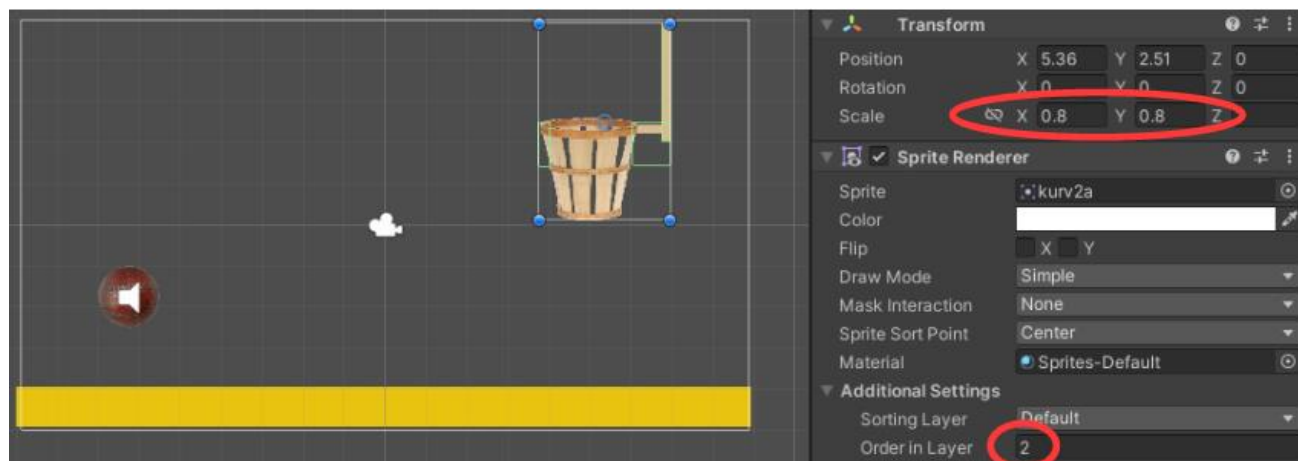
    private void OnMouseDown()
    {
        startOfMovement = Input.mousePosition;
    }


    private void OnMouseUp()
    {
        Vector2 movement = Input.mousePosition - startOfMovement;
        GetComponent<Rigidbody2D>().AddForce(movement / 20.0f,
            ForceMode2D.Impulse);
    }
}
```

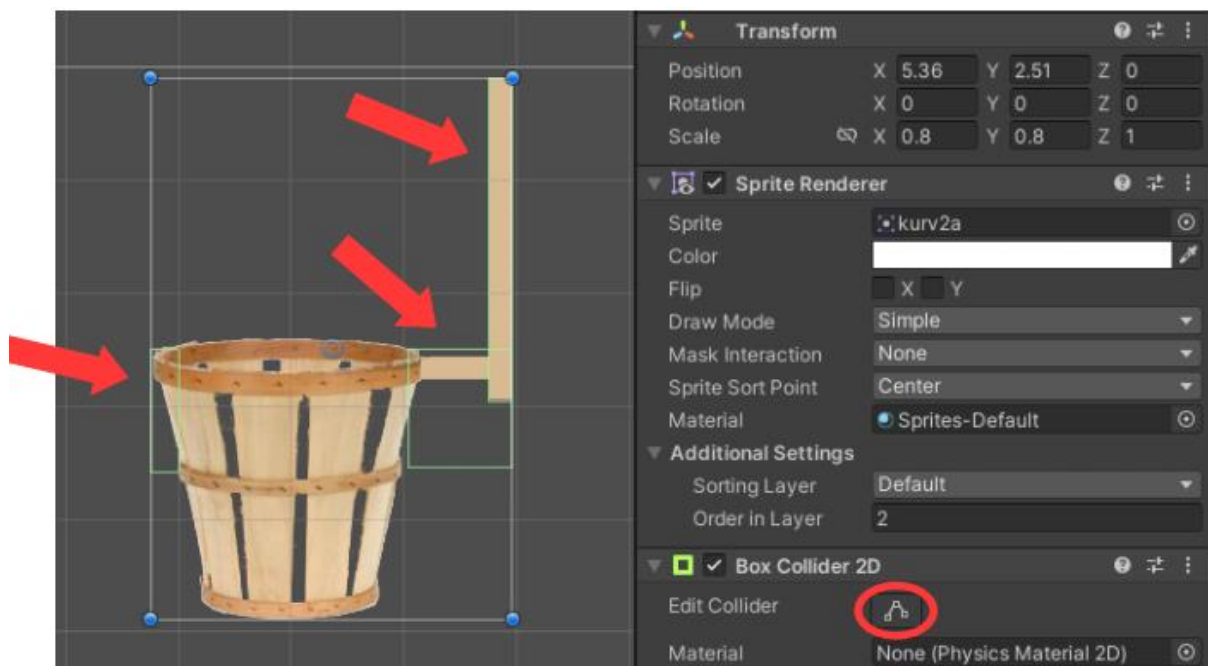
Prøv spillet. Tryk på bolden med musen og så træk og slip musen for at få bolden til at hoppe i en bestemt retning.

1.7 Kurv

Vi skal nu have tilføjet en kurv som vi kan få bolden ned i. I folderen *Assets/Graphics* ligger der nogle forskellige kurve man kan vælge imellem. Vælg *kurv2* og træk det ind i **Scene** vinduet oppe i højre side. Ændre størrelsen til 0.8 så den passer til boldens størrelse og sæt **Order in Layer** til 2.



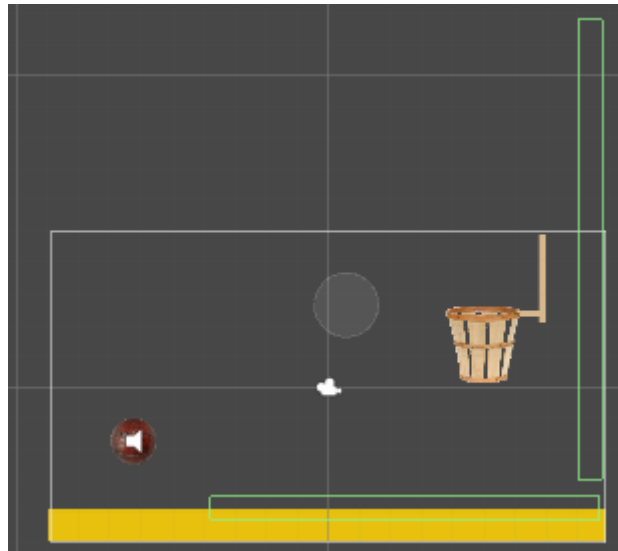
Tilføj 3 **Box Collider 2D** til kurv objektet og tryk på **Edit Collider**  og ret dem så det ligner billedet nedenfor (de 3 grønne firkanter).



Kør spillet og se om du kan ramme kurven. Det er lidt irriterende at man skal starte og stoppe spillet efter hvert forsøg.

Vi tilføjer derfor en 'reset' collider, som skal resette spillet hver gang man rammer den.

1. Højreklik i **Hierarchy**'et og vælg **Create Empty**.
2. Omdøb objektet til **Reset**.
3. Tilføj 2 stk **Box Collider 2D**.
4. Rediger de 2 colliders så det kommer til at se sådan ud (de 2 grønne firkanter):



Ret koden så den kommer til at se sådan ud (koden med **fed** er det nye):

```
public class Basketball : MonoBehaviour
{
    Vector3 startOfMovement;
    Vector3 originalPosition;

    private void Start()
    {
        originalPosition = transform.position;
    }

    private void OnMouseDown()
    {
        startOfMovement = Input.mousePosition;
    }

    private void OnMouseUp()
    {
        Vector2 movement = Input.mousePosition - startOfMovement;
        GetComponent<Rigidbody2D>().AddForce(movement / 20.0f,
            ForceMode2D.Impulse);
    }
    private void Reset()
    {
        GetComponent<Rigidbody2D>().velocity = Vector2.zero;
        GetComponent<Rigidbody2D>().angularVelocity = 0.0f;
        transform.position = originalPosition;
    }

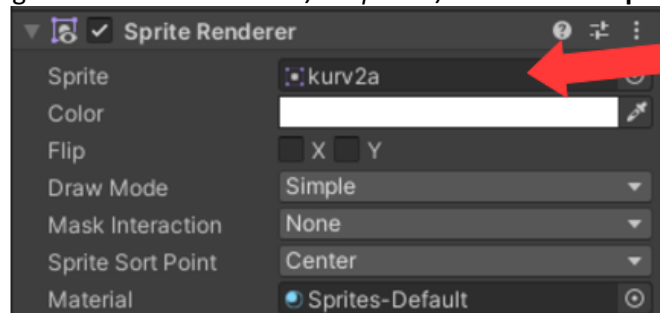
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.name == "Reset")
        {
            Reset();
        }
    }
}
```

Prøv spillet.

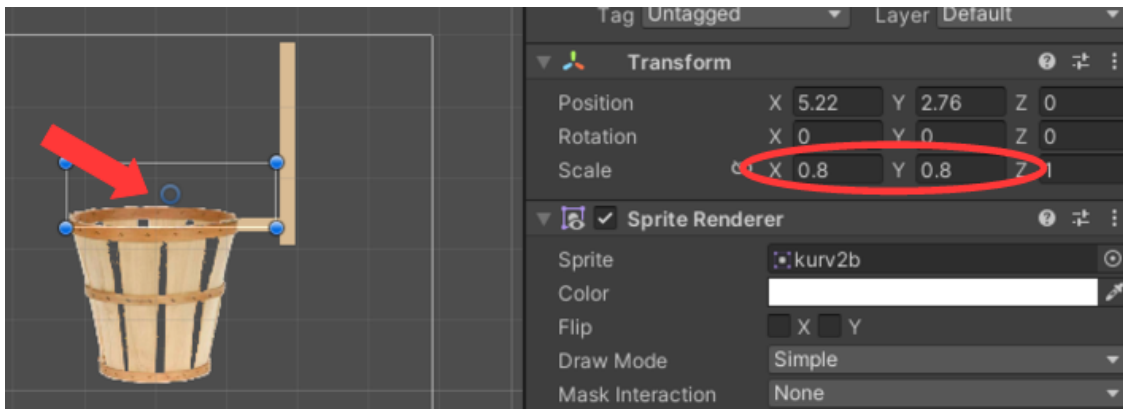
1.7.1 Bedre kurv

Det ser lidt mærkeligt ud når bolden falder 'ned' i kurven, ligesom om den ikke rigtigt kommer 'igennem' den. Det skyldes at billedet er taget lidt fra oven og at man derfor kan 'se' den øverste bagerste kant af kurven. Det kan ændres, hvis man i stedet bruger to billeder for kurven, et som det der er forrest og et som det der er bagerst.

I **Hierarchy**'et vælg *kurv2* og træk billedet fra *Assets/Graphics/kurv2a* over i **Sprite Renderer->Sprite**.



Træk billedet *Assets/Graphics/kurv2b* ind i **Scene** vinduet og skaler det til 0.8 og placer det så det passer til kurven.



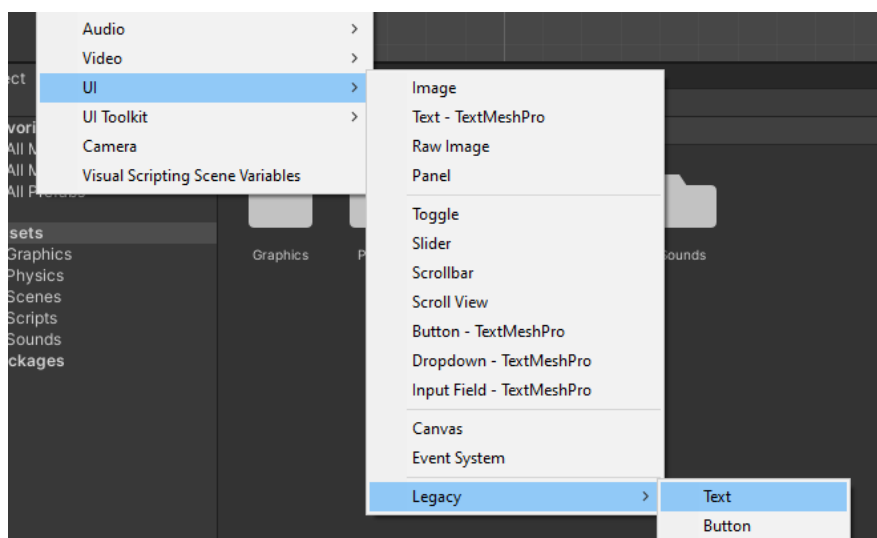
Prøv spillet – nu skulle det gerne så meget bedre ud når bolden falder gennem kurven.

1.8 Score

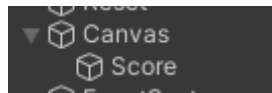
Vi skal nu have en score tæller, som tæller hver gang vi scorer et mål.

Først skal vi have en tekst på skærmen.

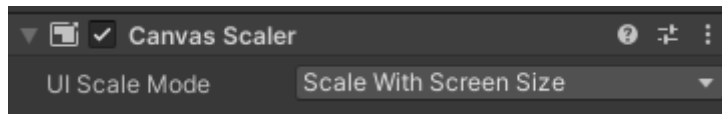
Højreklik i **Hierarchy**'et og vælg **UI->Legacy->Text** og kald den *Score*.



Den laver automatisk et objekt som hedder *Canvas*.

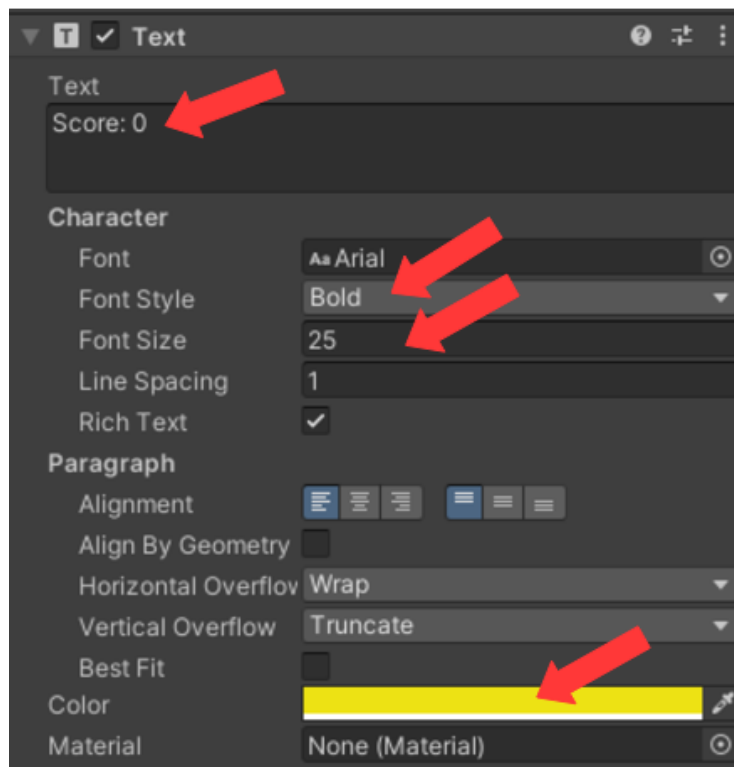


Vælg *Canvas* og set **UI Scale Mode** til **Scale With Screen Size**.

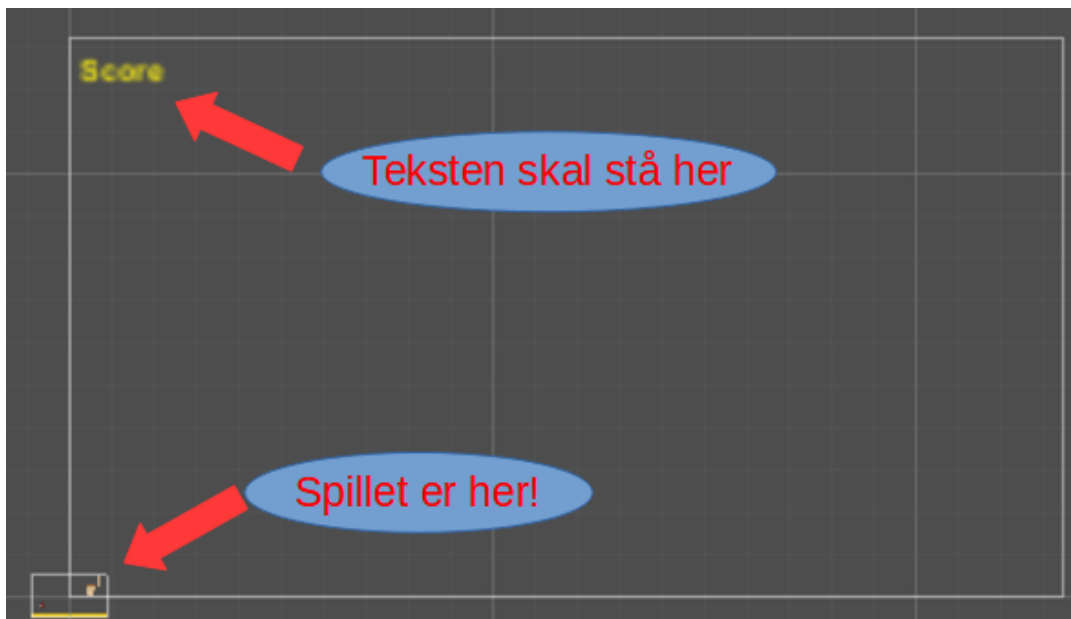


Vælg *Score* og set følgende i **Inspector**'en:

1. Set **Text** til *Score: 0*
2. Set **Font Style** til *Bold*
3. Set **Font Size** til 25
4. Set **Color** til en lys farve



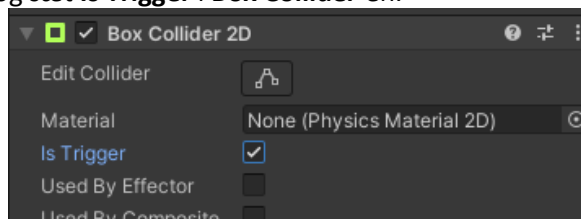
Zoom meget ud og flyt teksten op til venstre i den 'nye' hvide firkant som kommer frem – se billede nedenfor:




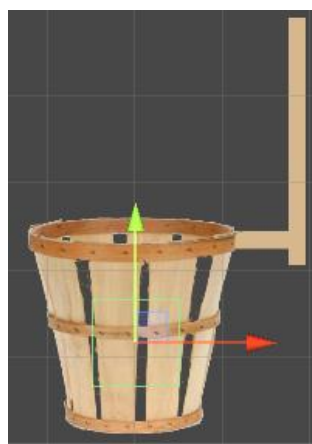
Prøv at køre spillet og se at teksten står pænt.

Vi skal nu have scoren til at tælle.

1. I **Hierarchy**'et højreklik og **Create Empty** objekt og kald det *KurvHit*.
2. Tilføj en **Box Collider 2D** og sæt **Is Trigger** i **Box Collider**'en.



3. Flyt den så den ligger 'inde' i kurven, så den kun kan blive ramt hvis man scorer et point. (Man flytter den ved at trykke på  og 'hiver' i pilene).



Ret koden ved at tilføje følgende linjer i toppen af koden (kun det med **fed**):

```
public class Basketball : MonoBehaviour
{
    [SerializeField] Text score;
    Vector3 startOfMovement;
    Vector3 originalPosition;
    int point;
}
```

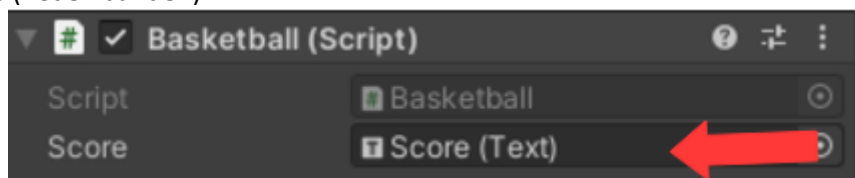
Og disse linjer i bunden af koden (kun det med **fed**):

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.name == "Reset")
    {
        Reset();
    }
}

private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.name == "KurvHit")
    {
        point++;
        score.text = "Score: " + point;
    }
}
```

Gem filen og i **Hierarchy**'et vælg *Basketball* objektet.

I **Hierarchy**'et tryk på *Score* objektet og træk det over i **Inspector**'en for *Basketball* objektet og ind i feltet som hedder **Score** (nede i bunden).



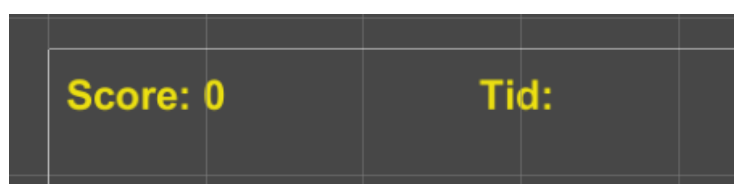
Prøv spillet og se om Score tæller op, når man scorer point.

1.9 Tidsbegrænsning

Vi skal nu lave et lille ur, så man kan se hvor mange mål man kan score på 1 minut.

Først skal vi have lavet et tekstfelt mere. Det letteste er at kopier den tekst vi allerede har og så rette lidt i den.

1. I **Hierarchy**'et vælg *Score* (tryk på den) og tryk derefter på Ctrl-D. Dette laver en kopi af *Score*.
2. I **Hierarchy**'et, ret navnet til *Tid*.
3. I **Inspector**'en ret **Text** til *Tid*.
4. Flyt teksten så den står til højre for *Score*:



Nu skal vi rette i koden så tiden bliver talt ned fra 60 sekunder. Tilføj følgende kode (kun det med **fed**):

```

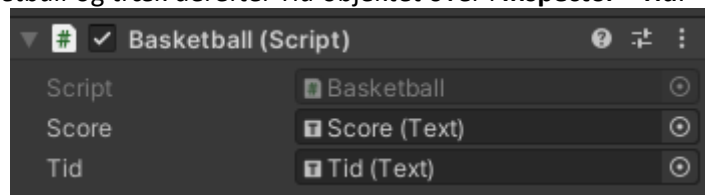
public class Basketball : MonoBehaviour
{
    [SerializeField] Text score;
    [SerializeField] Text tid;
    Vector3 startOfMovement;
    Vector3 originalPosition;
    int point;
    float timeLeft = 60;

    private void Start()
    {
        originalPosition = transform.position;
    }

    private void Update()
    {
        timeLeft -= Time.deltaTime;
        if (timeLeft < 0)
        {
            timeLeft = 0;
        }
        tid.text = "Tid: " + timeLeft.ToString("00");
    }
}

```

I **Hierarchy**'et vælg *basketball* og træk derefter *Tid* objektet over i **Inspector**->**Tid**.



Prøv spillet og se at tiden tæller ned. Når tiden når nul, så stopper tiden.

1.10 Lyd

Nu skal vi have spillet til at spille en lyd hver gang man scorer.

I **Hierarchy**'et vælg *basketball* og i **Inspector**'en tilføj en komponent som hedder: **Audio Source**.

Der skal tilføjes lidt kode flere steder (kun det med **fed**):

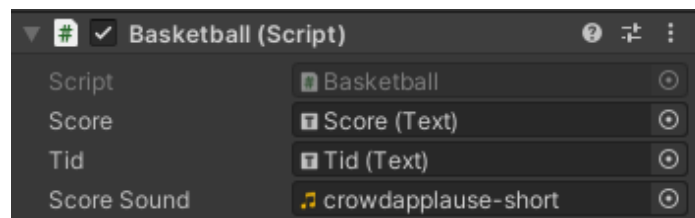
```
public class Basketball : MonoBehaviour
{
    [SerializeField] Text score;
    [SerializeField] Text tid;
    [SerializeField] AudioClip scoreSound;

    AudioSource audioPlayer;
    Vector3 startOfMovement;
    Vector3 originalPosition;
    int point;
    float timeLeft = 60;

    private void Start()
    {
        originalPosition = transform.position;
        audioPlayer = GetComponent();
    }
}
```

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.name == "KurvHit")
    {
        point++;
        score.text = "Score: " + point;
        audioPlayer.PlayOneShot(scoreSound, 1.0f);
    }
}
```

I *Assets/Sounds* folderen ligger der nogle lydfiler. Træk *crowdapplause-short* over i *basketball Inspector->Score Sound*.



Prøv kode og hør hvordan det lyder når man scorer.

2 Links

Tips videoer	TOP 10 UNITY TIPS - 2017 TOP 10 UNITY TIPS #2
Unity manual	https://docs.unity3d.com/2020.2/Documentation/ScriptReference/index.html
Unity assets store	https://assetstore.unity.com
Fonts	https://fonts.google.com/ (Fx 'Press Start 2D')
Dansk C#/Unity document	https://github.com/Grailas/CodingPiratesAalborg/blob/master/Guides/Hj%C3%A6lpeguide.pdf