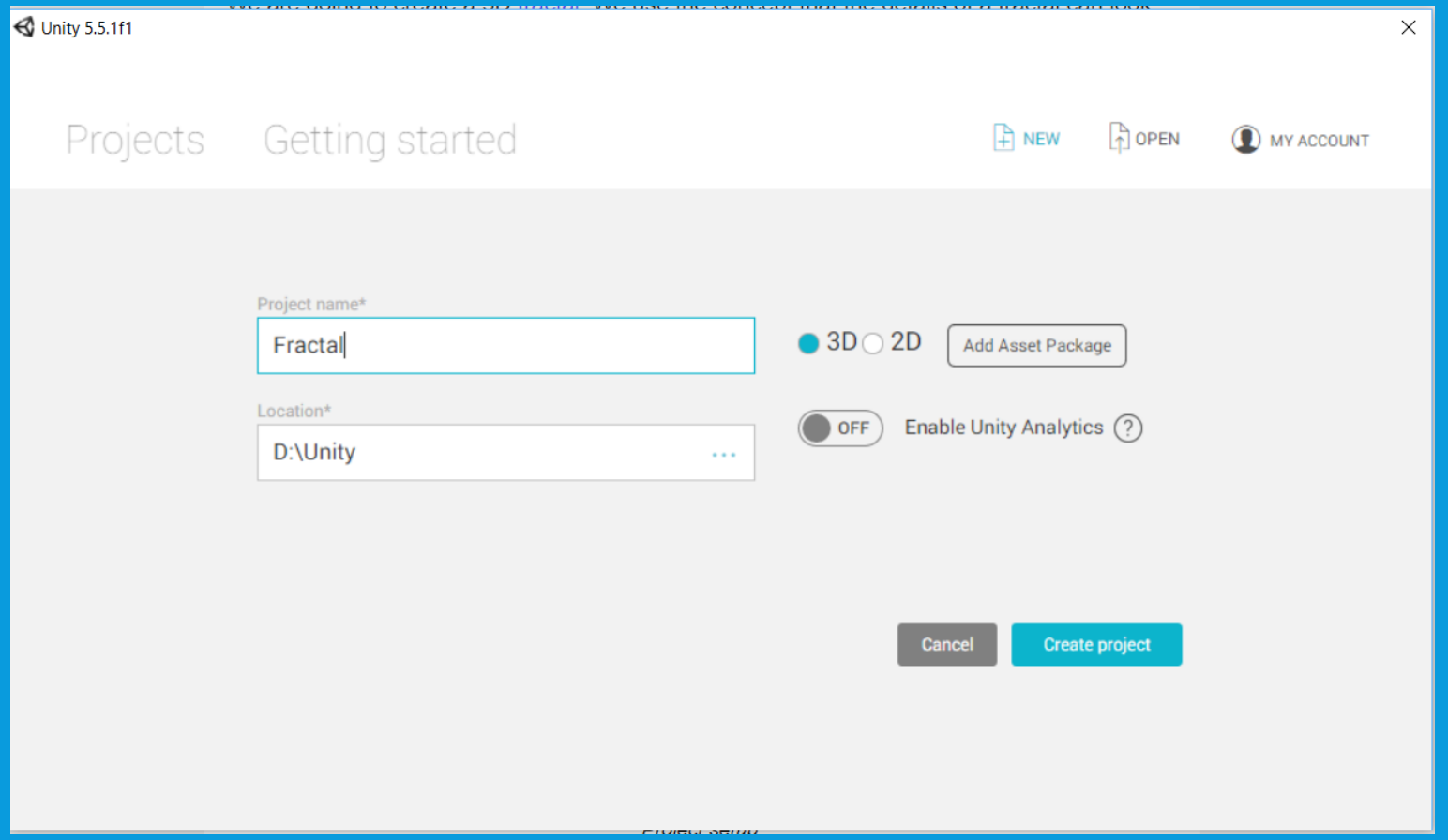
UNITY OG KODE

Manipuler objekter via C# kode

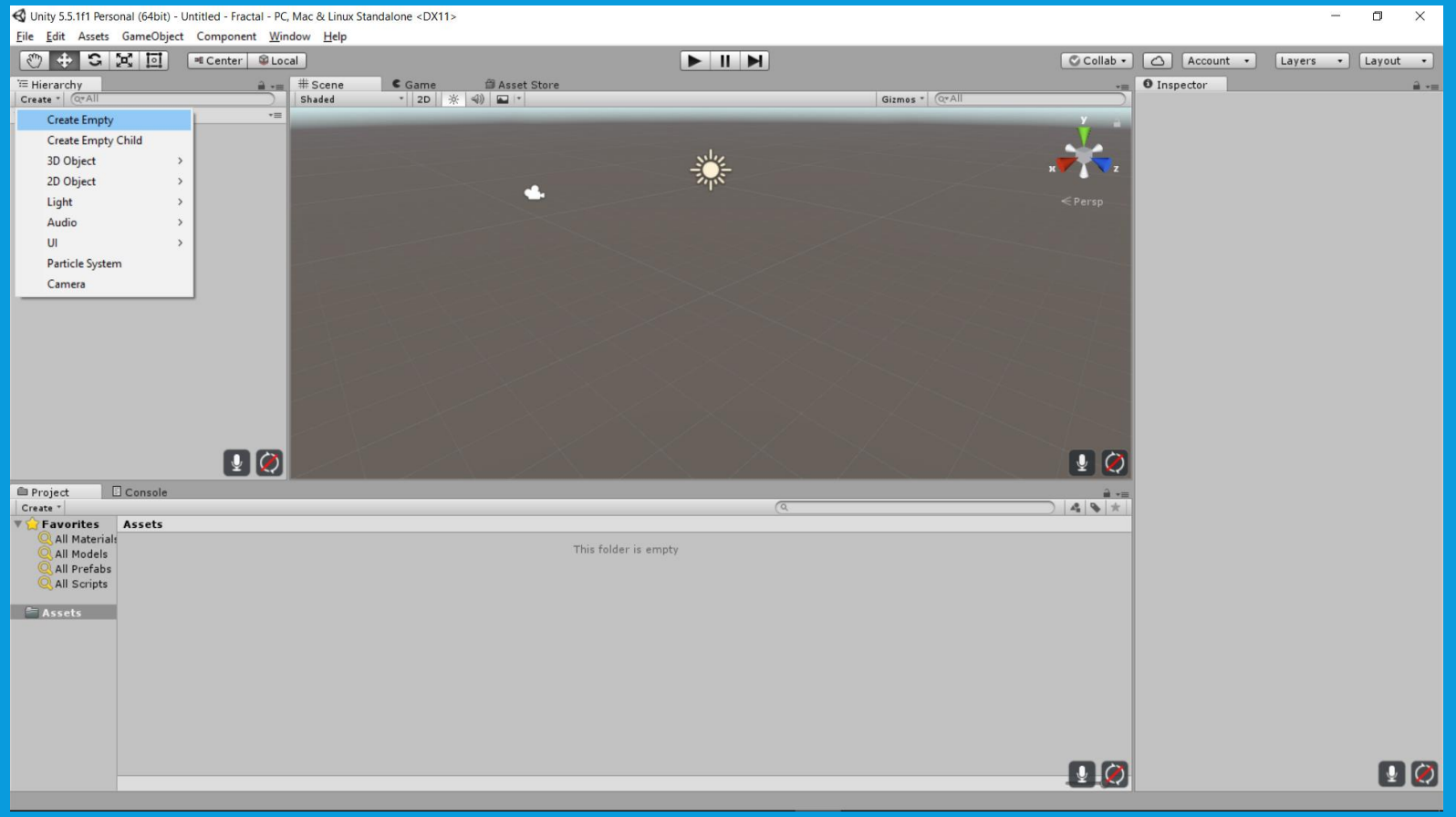
FRAKTAL

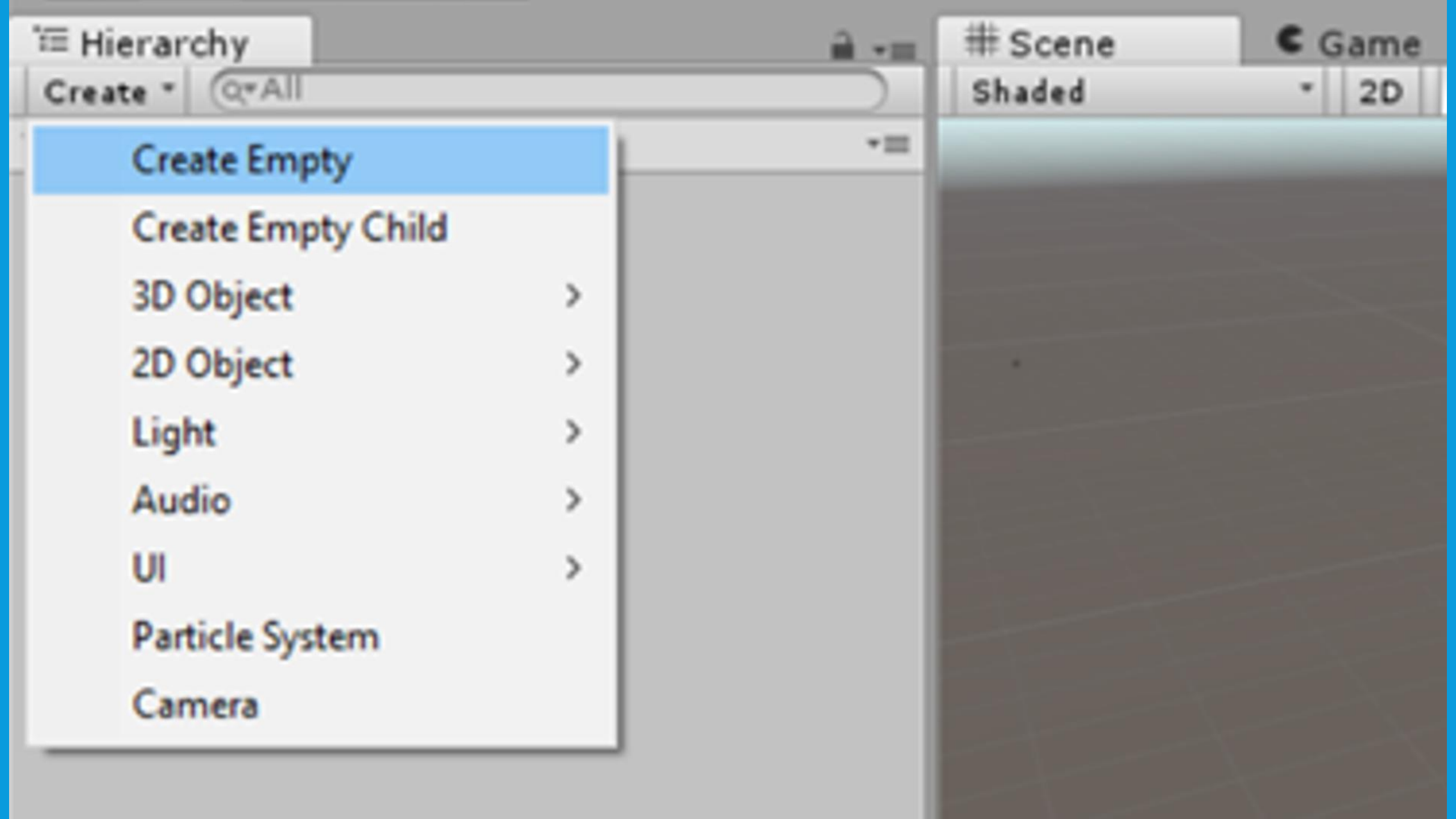
Et projekt der danner 3D fraktaler automatisk

LAV ET NYT 3D PROJEKT

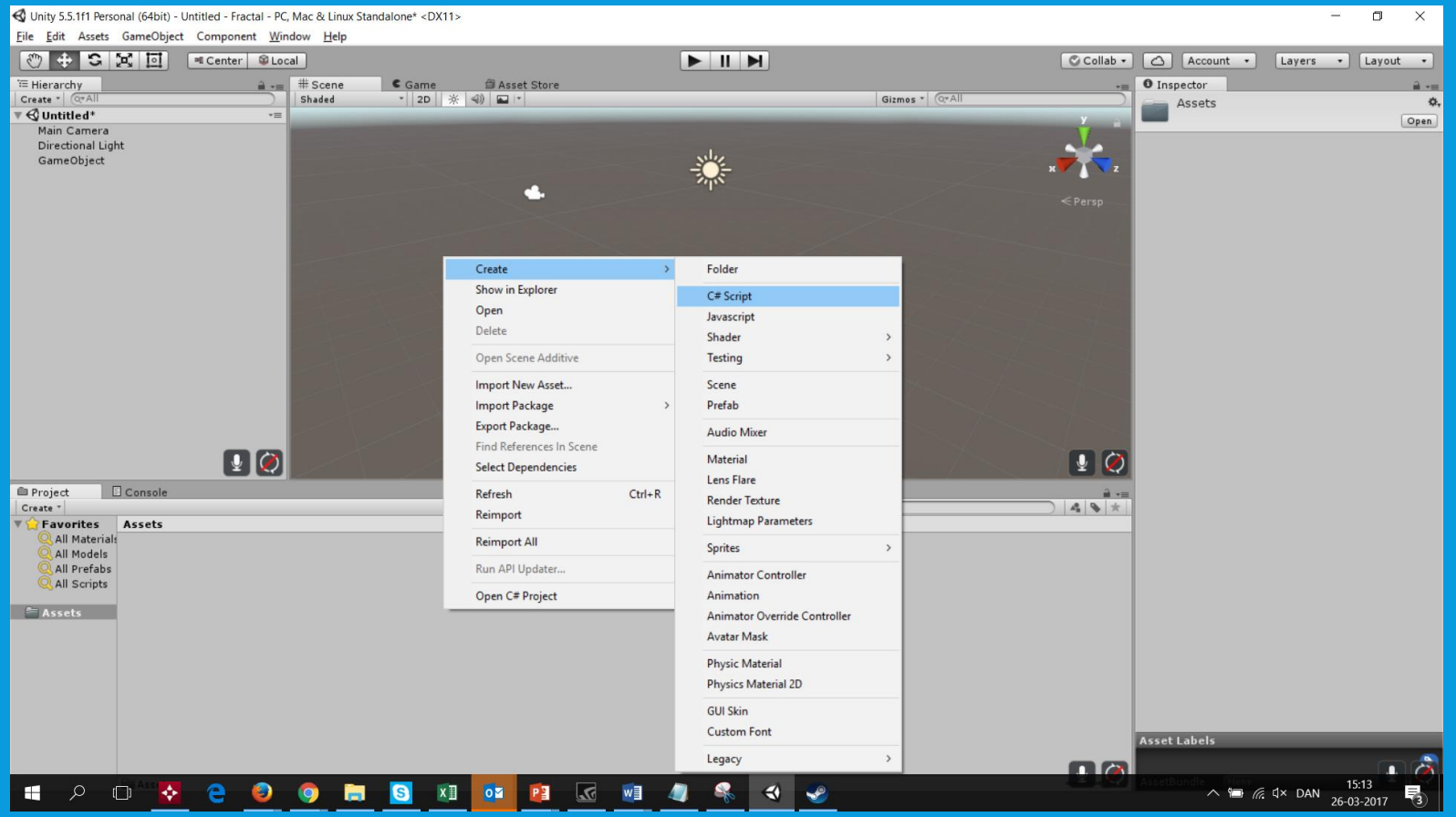


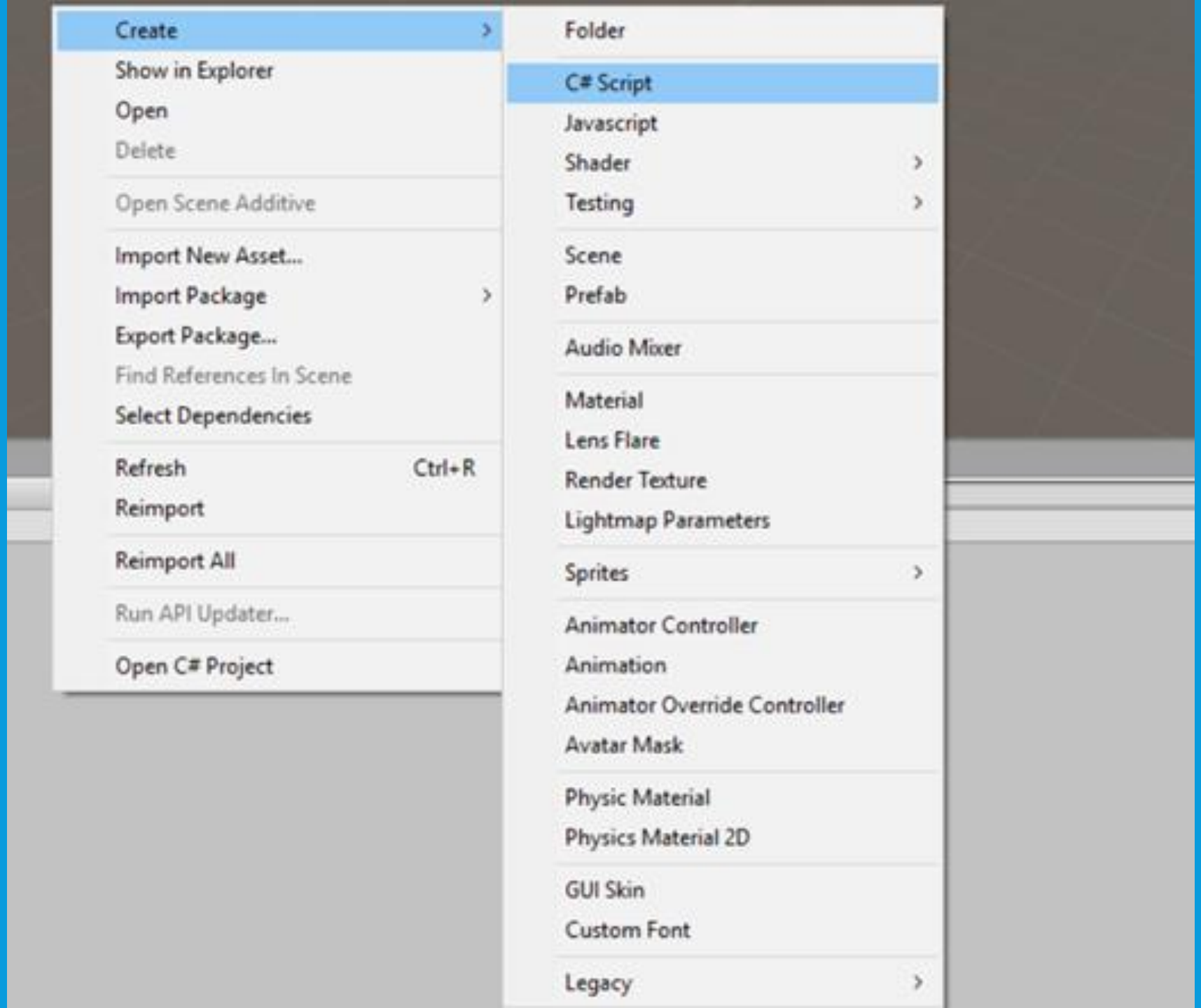
LAV ET TOMT GAMEOBJECT



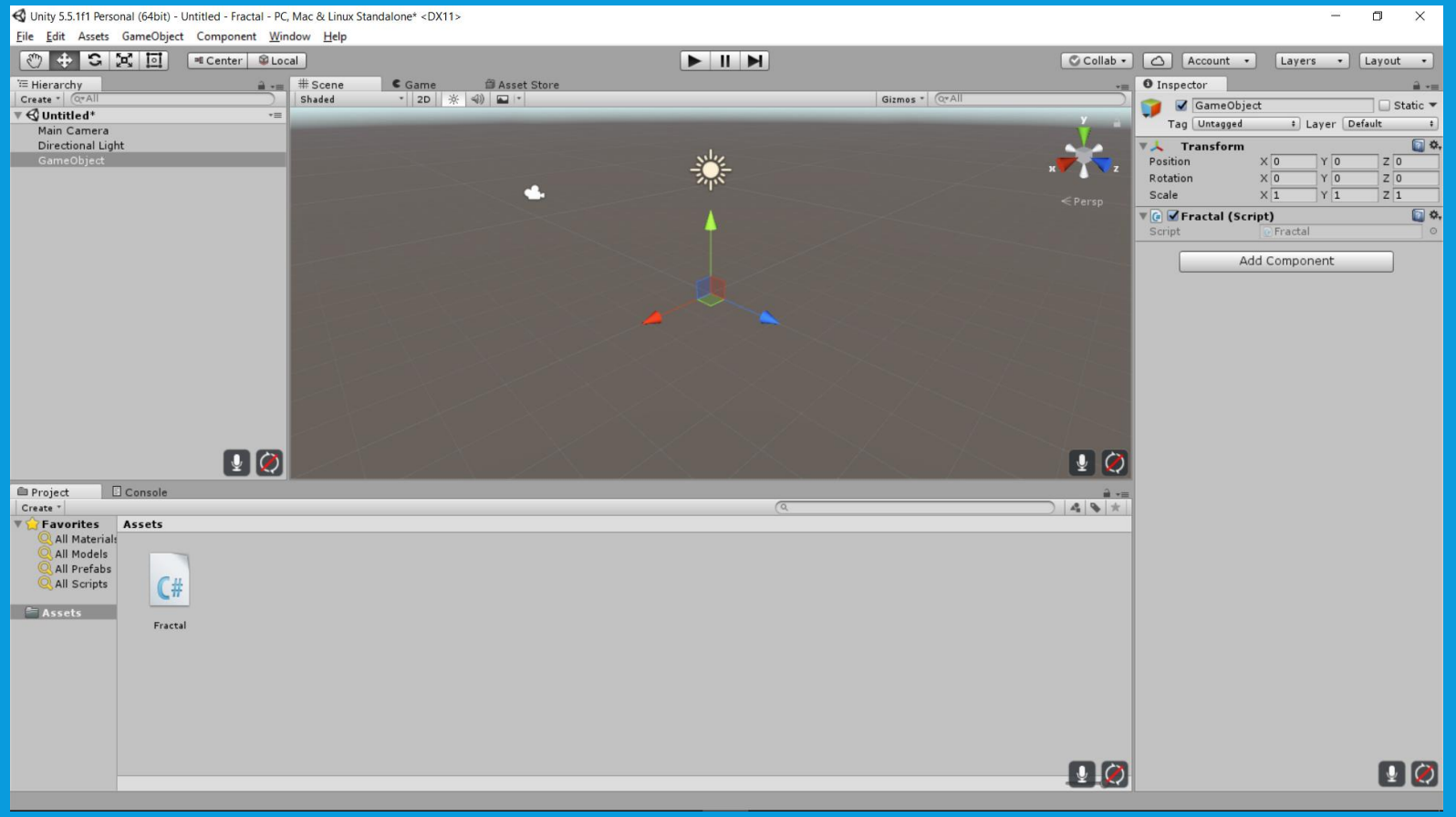

Hierarchy
Create
All
Create Empty
Create Empty Child
3D Object
2D Object
Light
Audio
UI
Particle System
Camera
Scene
Game
Shaded
2D

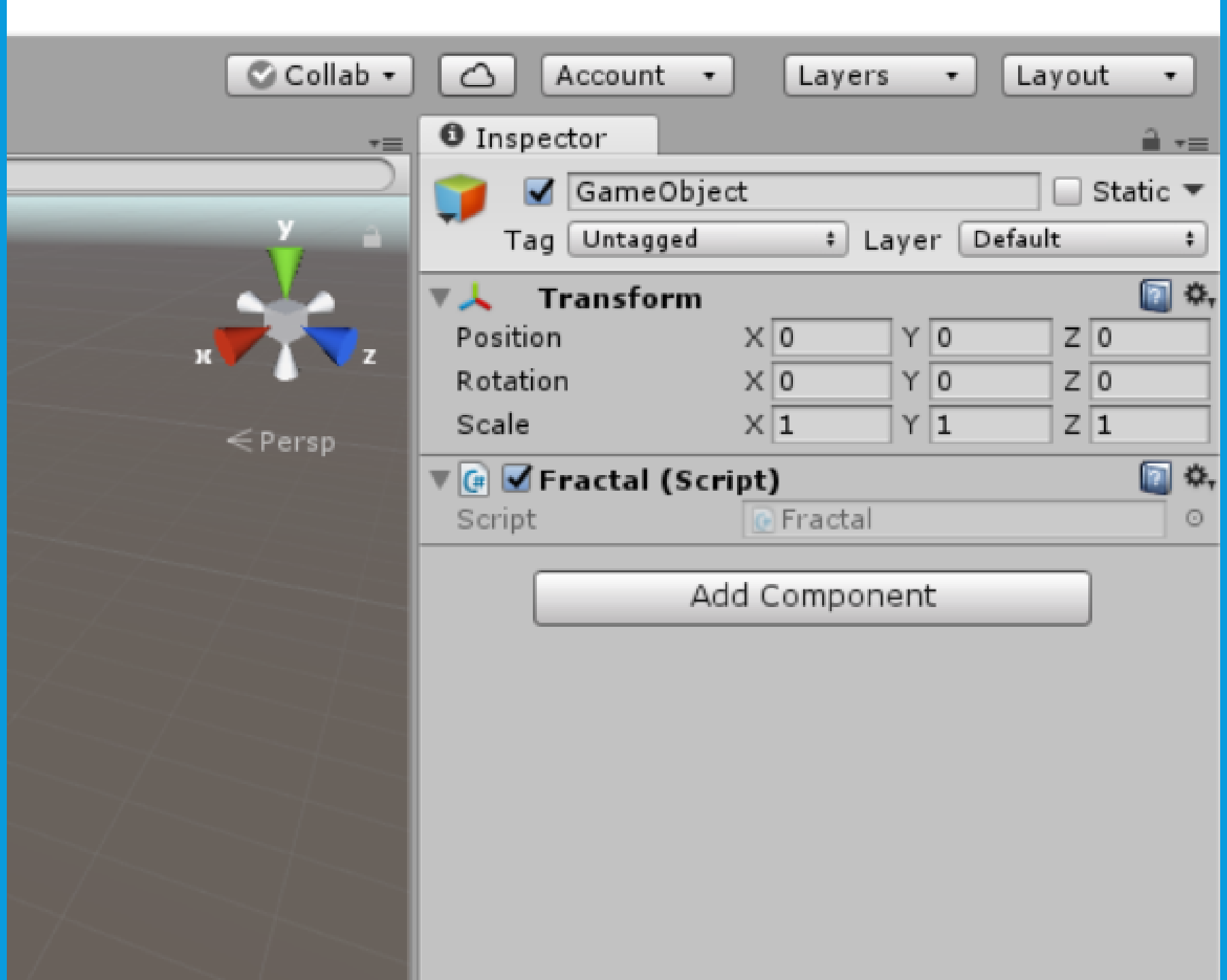
LAV ET NYT C# SCRIPT



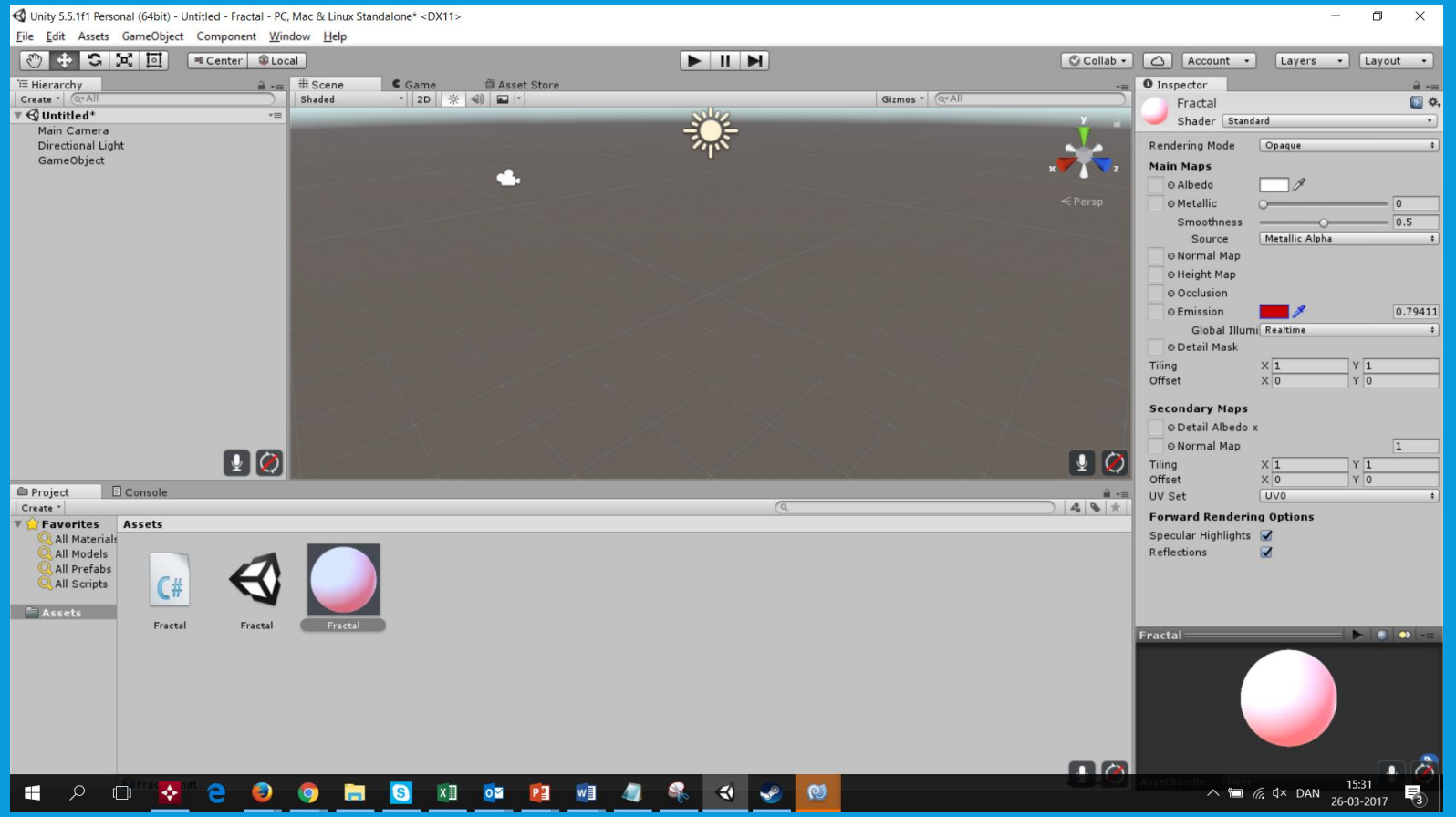

Create
Show in Explorer
Open
Delete
Open Scene Additive
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Run API Updater...
Open C# Project
Folder
C# Script
Javascript
Shader
Testing
Scene
Prefab
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Sprites
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Physic Material
Physics Material 2D
GUI Skin
Custom Font
Legacy

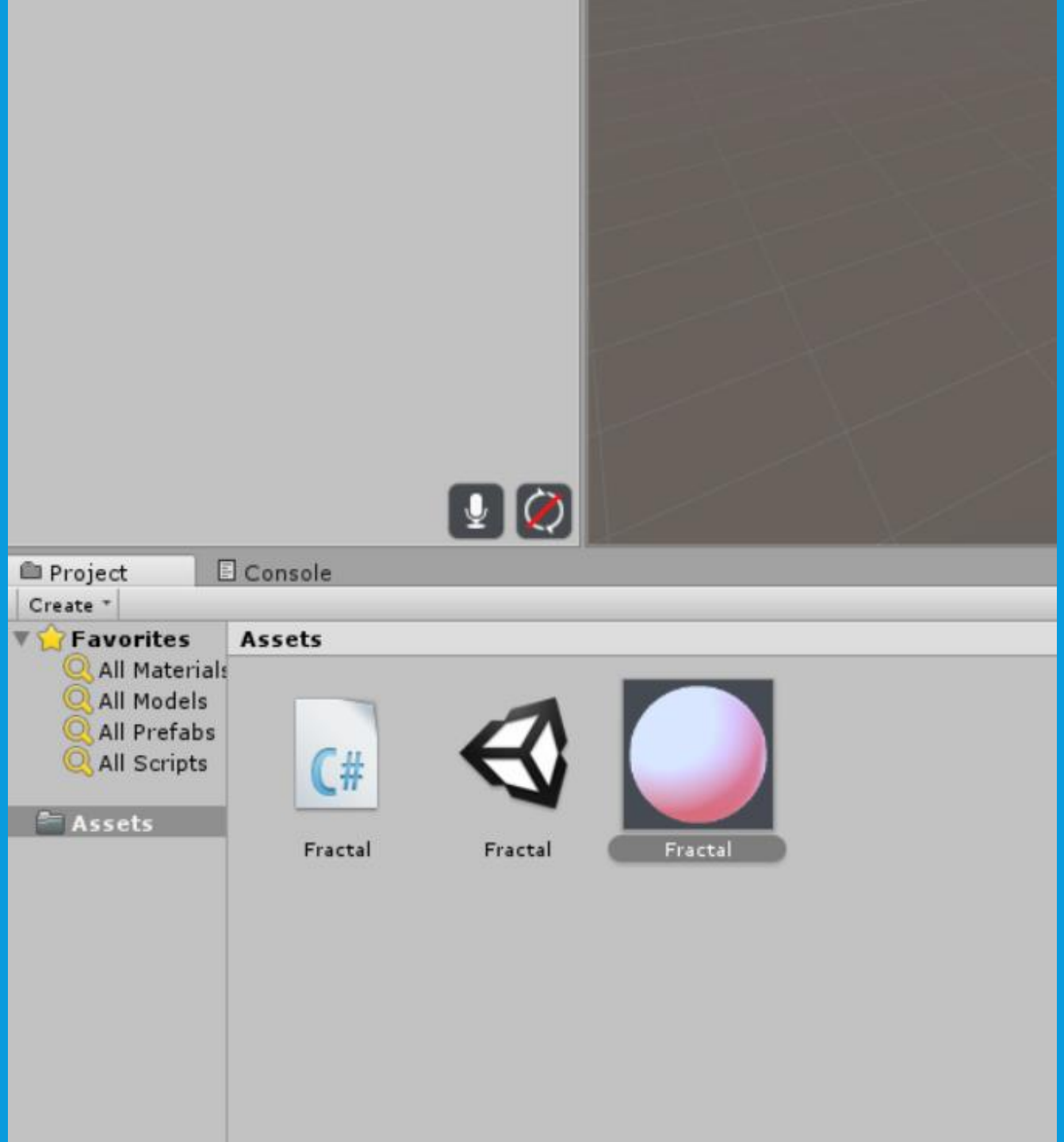
FØJ SCRIPTET TIL OBJEKTET



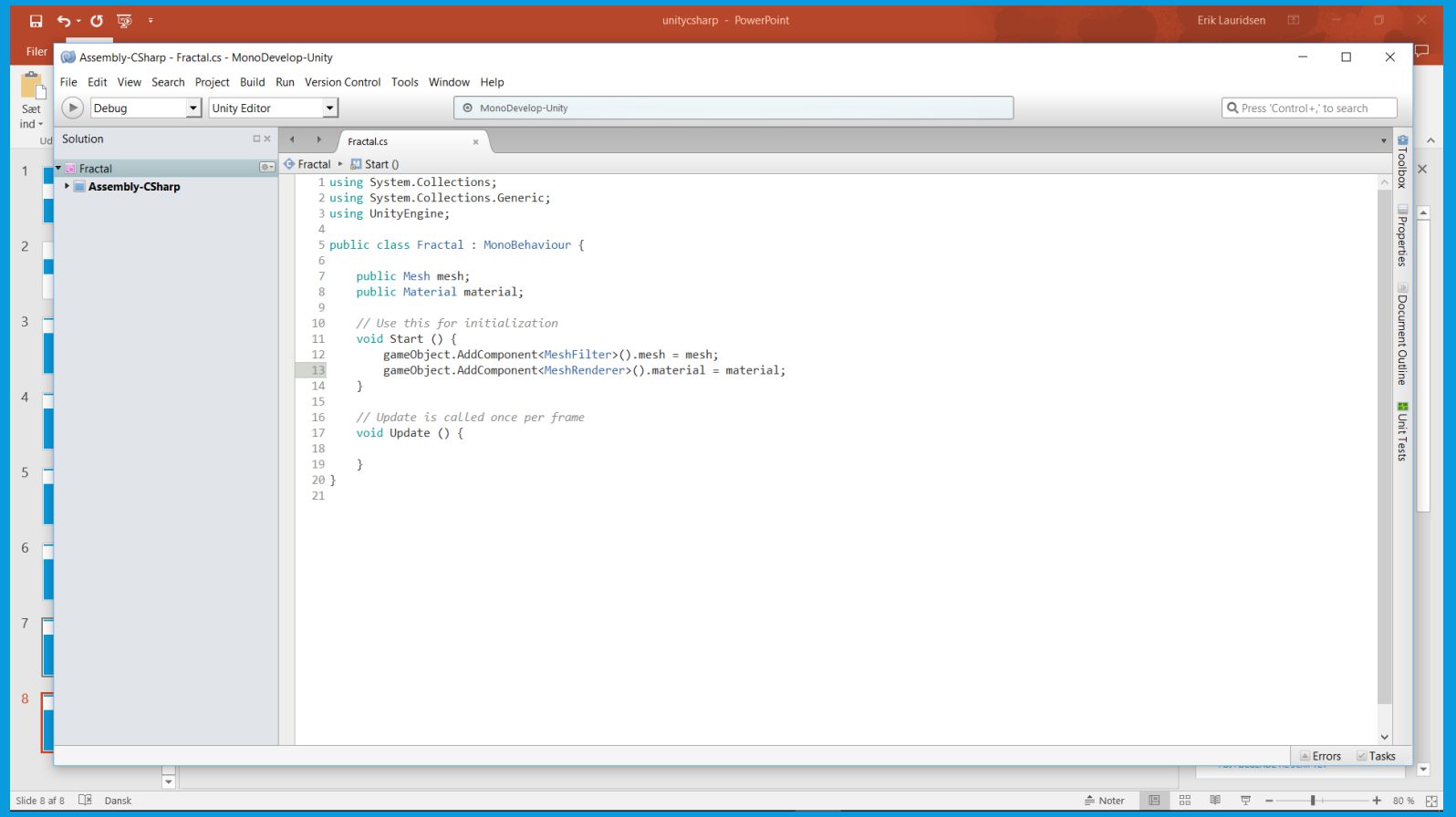

Collab
Account
Layers
Layout
Inspector
GameObject
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Fractal (Script)
Script
Fractal
Add Component
y
x
z
Persp

LAV EN SCENE OG ET MATERIALE




Project
Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Fractal

FØJ FØLGENDE TIL SCRIPTET



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Fractal : MonoBehaviour {
6
7     public Mesh mesh;
8     public Material material;
9
10    // Use this for initialization
11    void Start () {
12        gameObject.AddComponent<MeshFilter>().mesh = mesh;
13        gameObject.AddComponent<MeshRenderer>().material = material;
14    }
15
16    // Update is called once per frame
17    void Update () {
18
19    }
20 }
```

GENNEMGANG AF SCRIPTET INDTIL VIDERE

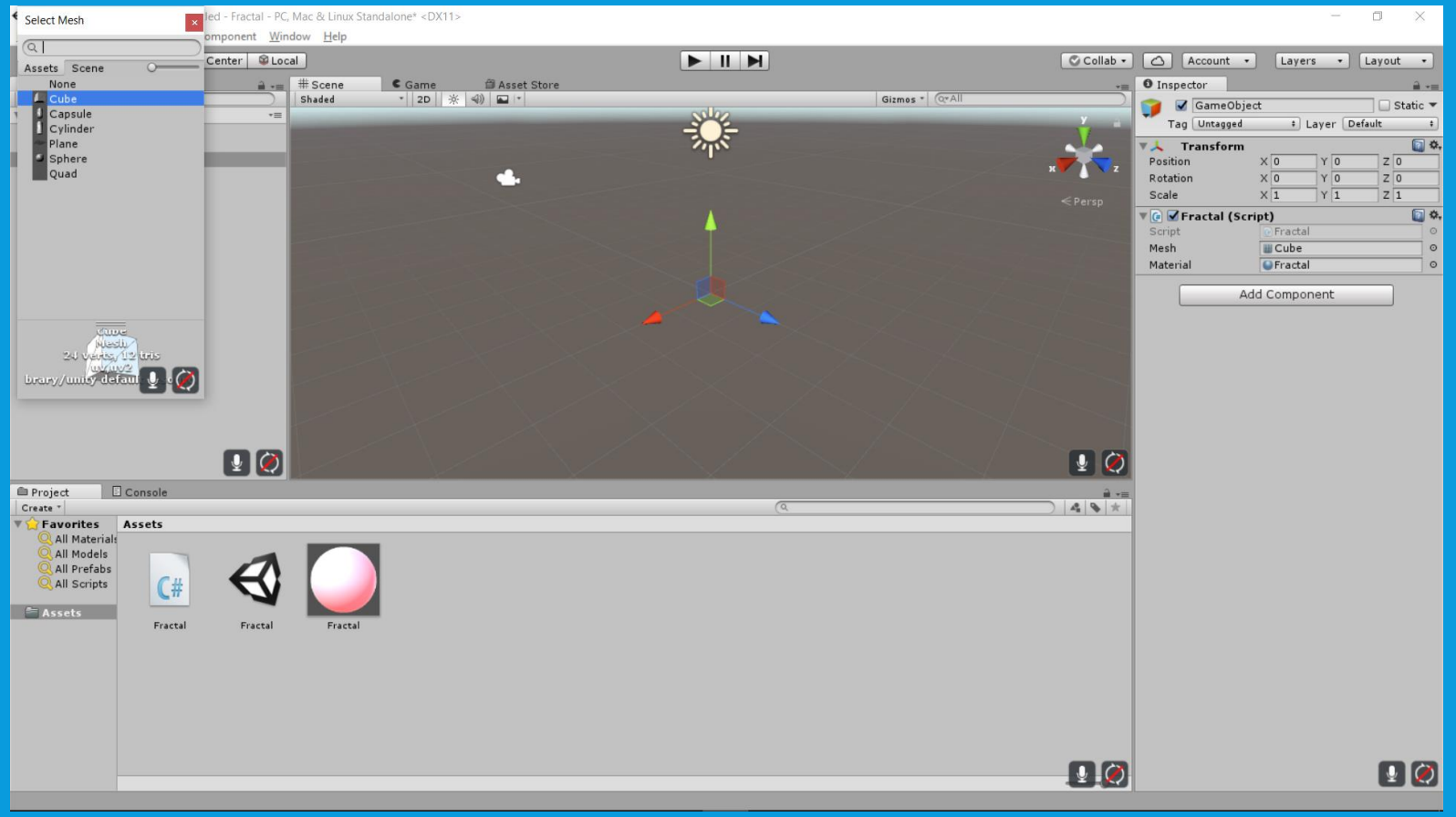
- **Hvad er et mesh?**
  - Et *mesh* er et construct, der anvendes af den grafiske hardware til at tegne komplekse ting. Det er et 3D-objekt, der er enten er importeret til Unity, en af Unitys standard figurer eller genereret af kode.
  - Et mesh indeholder mindst en samling af punkter i 3D-rum plus et sæt trekanter - de mest basale 2D figurer - defineret af disse punkter. Trekanterne udgør overfladen af uanset hvad mesh'et forestiller. Ofte vil du ikke tænke over at man kigger på en gruppe trekanter i stedet for et reelt objekt

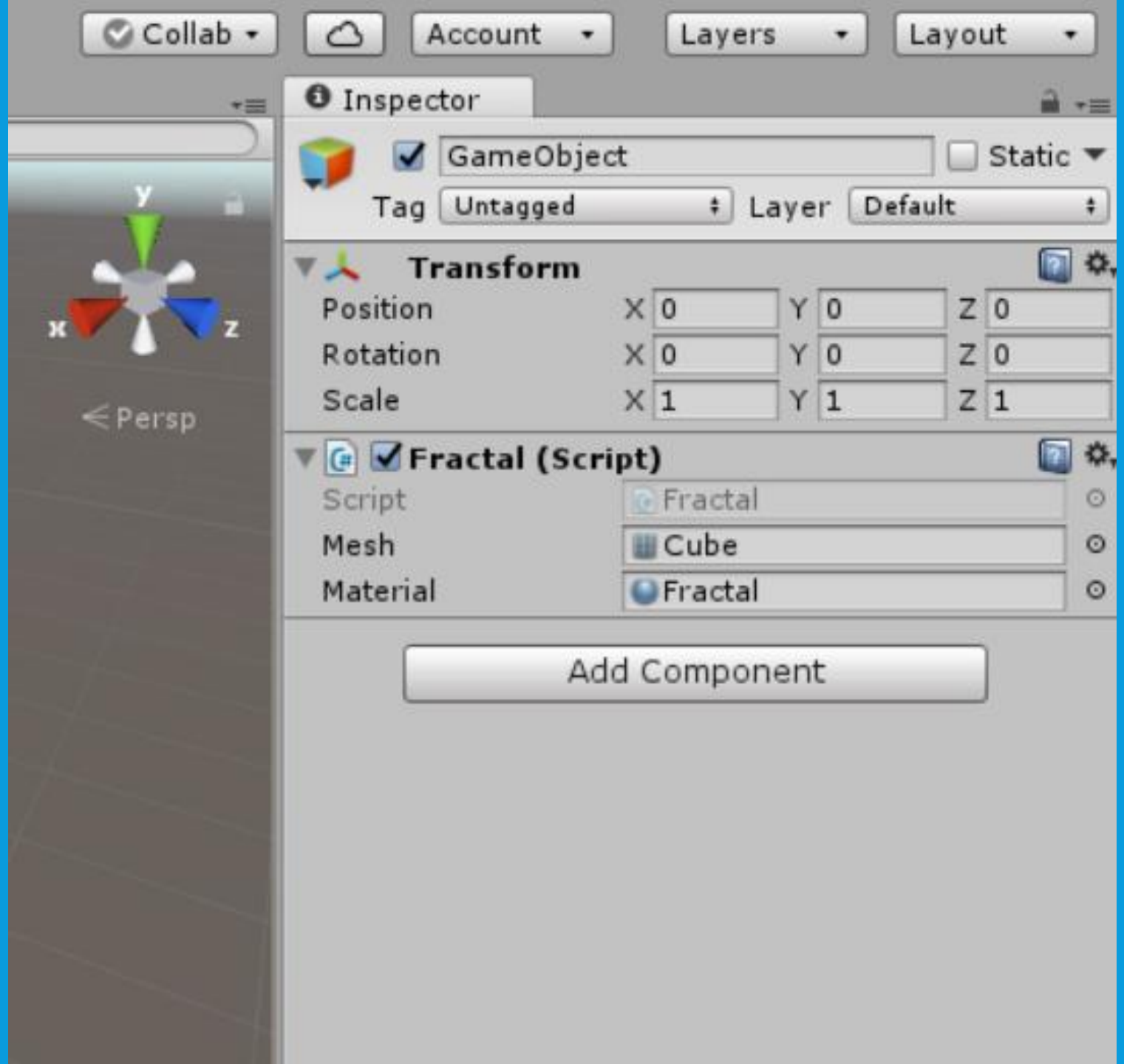
- **Hvad er et material?**
  - Materialer anvendes til at definere de visuelle egenskaber for objekter. De kan variere fra meget enkle, ligesom en konstant farve, til meget komplekse.
  - Materialer består af en shader og den data som shaderen måtte have behov for. Shaders er grundlæggende scripts der fortæller grafikkortet hvordan objektets polygoner bør udarbejdes.
  - Standard diffuse shader bruger en enkelt farve og en tekstur sammen lyskilderne i den scene til at bestemme udseendet af polygoner. Den lidt mere komplicerede specular shader simulerer også highlights.

GENNEMGANG AF SCRIPTET INDTIL VIDERE

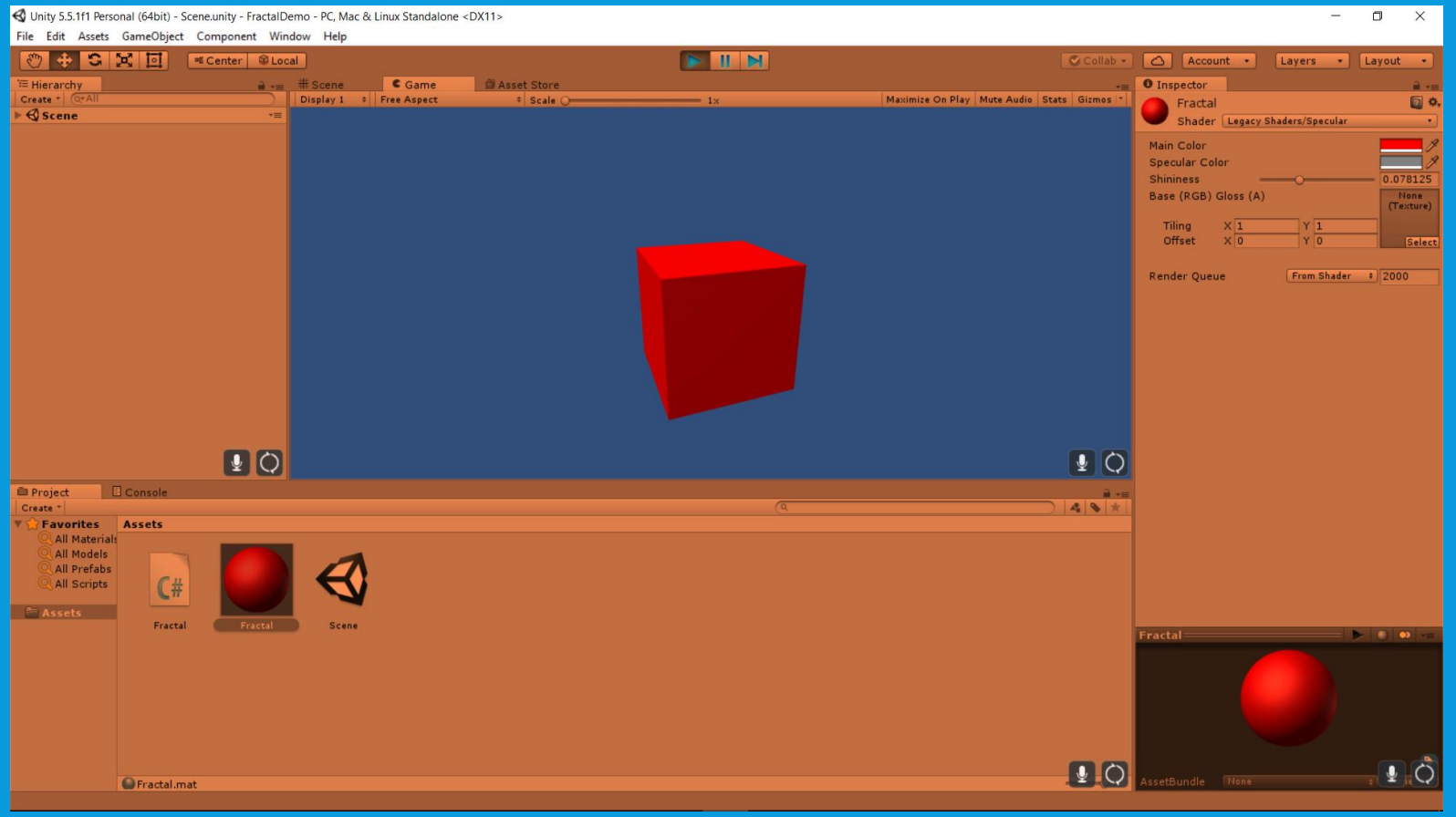
- **Hvornår bliver start kaldt (invoked)**
  - Start metoden kaldes af Unity efter komponenten er oprettet, når den er aktiv, lige før den første gang dens Update metode ville blive kaldt, hvis den har en. Den kaldes kun én gang.
- **Hvordan fungerer AddComponent?**
  - *AddComponent* metoden laver en ny komponent af en bestemt type, fastgjort til spil objektet, og returnerer en reference til det.
  - Derved kan vi straks få adgang til komponentens værdier. Man kan også benytte en mellemliggende variabel.

FØJ VORES MATERIALE TIL FRACTAL KOMPONENTEN OG GIV DEN ET STANDARD CUBE MESH



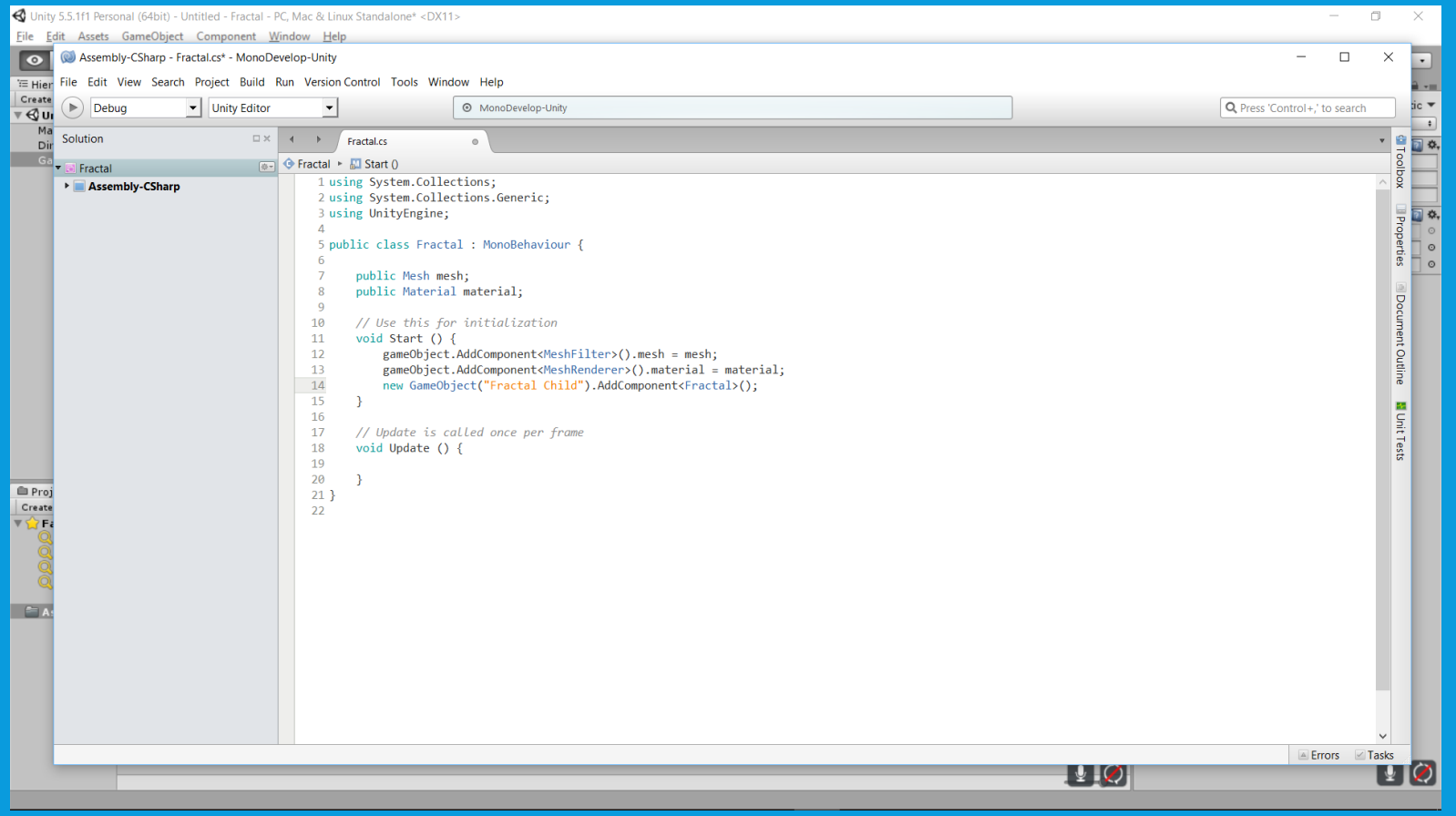

Collab
Account
Layers
Layout
Inspector
GameObject
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Fractal (Script)
Script
Fractal
Mesh
Cube
Material
Fractal
Add Component
y
x
z
Persp

TJEK DET VI HAR LAVET I PLAYMODE



Jeg har sat playmode for mig til at have en overlay farve så jeg kan se når jeg er i playmode.
Dette gøres under Edit > Preferences > Colors > Playmode tint

LAV BØRN AF FRACTAL MED DEN NY KODE PÅ LINJE 14



```
4
5  public class Fractal : MonoBehaviour {
6
7      public Mesh mesh;
8      public Material material;
9
10     // Use this for initialization
11     void Start () {
12         gameObject.AddComponent<MeshFilter>().mesh = mesh;
13         gameObject.AddComponent<MeshRenderer>().material = materia
14         new GameObject("Fractal Child").AddComponent<Fractal>();
15     }
16
17     // Update is called once per frame
18     void Update () {
19
20     }
21 }
```

GENNEMGANG AF SCRIPTET INDTIL VIDERE

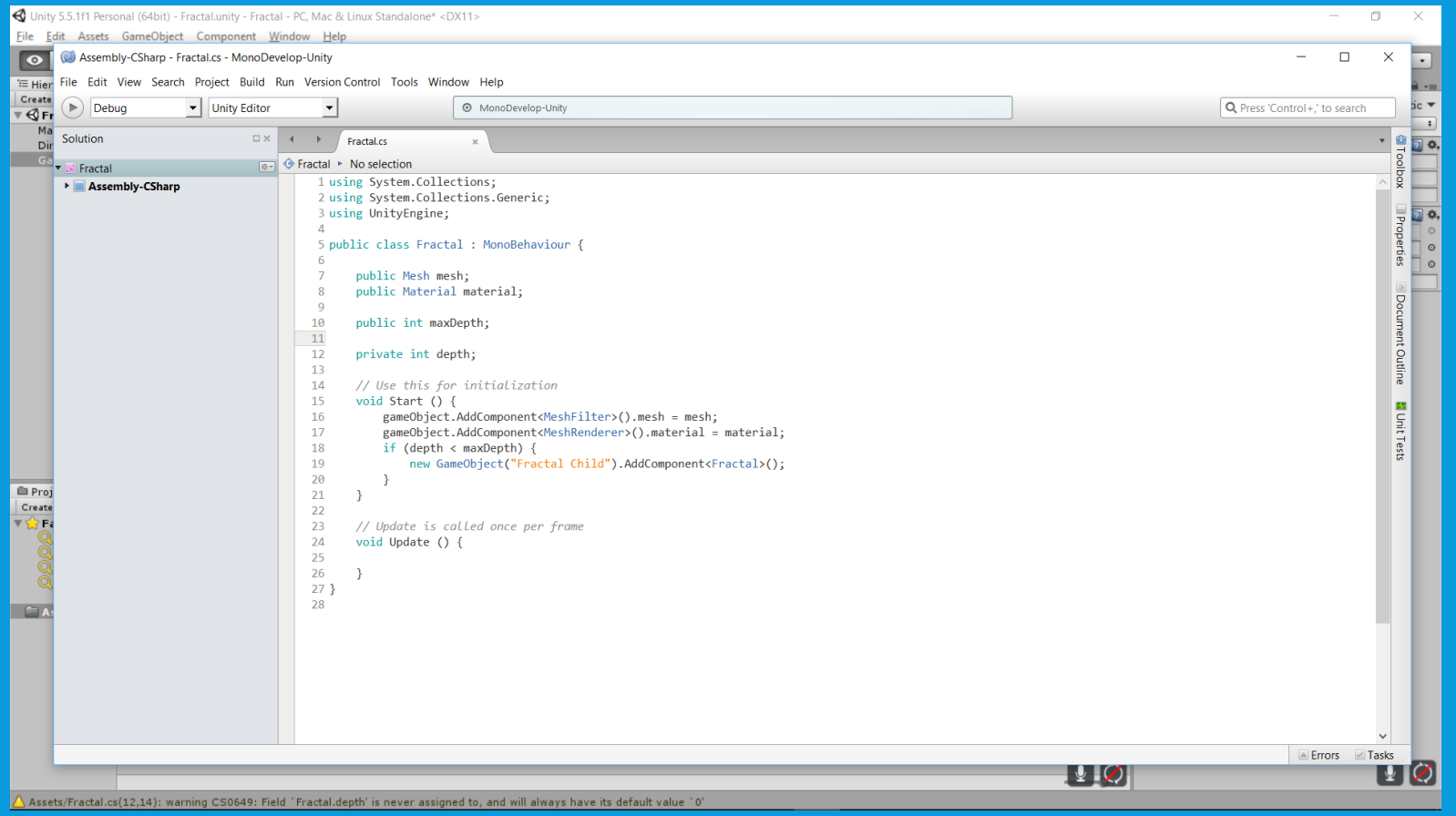
- **Hvad gør new?**
  - *new* nøgleordet bruges til at konstruere en ny forekomst af et objekt eller en struct. Det efterfølges af at kalde en speciel constructor metode, som har samme navn som den klasse eller struct det tilhører.

- **Vi har dog et problem!**
- Hver ny fraktal instans ønsker at skabe endnu en yderligere. Dette sker hver frame, uden ende. Lad det køre i et stykke tid, og din computer vil komme i problemer, da den løber tør for hukommelse.
- Typisk vil rekursive algoritmer der ikke stopper forbruge maskinens ressourcer næsten øjeblikkeligt og resultere i enten en stak overflow undtagelser eller et nedbrud.
- I dette tilfælde er det en temmelig godartet eksplosion, fordi det sker langsomt.

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Vi har dog et problem!** ***fortsat***
  - For at forhindre dette i at ske introducerer vi begrebet maksimal dybde.
  - Vores indledende fraktal instans vil have en dybde på nul.
  - Dets barn vil have en dybde på én.
  - Barnet af dette barn vil have en dybde på 2.
  - Og så videre, indtil den maksimale dybde er nået.
- For at gøre dette tilføjer vi en offentlig maxDepth heltalsvariabel og sætter den til 4 i inspektøren.
- Derefter tilføjer vi et privat heltal depth
- Derved kan der kun oprette et nyt barn, hvis vi er under den maksimale dybde.

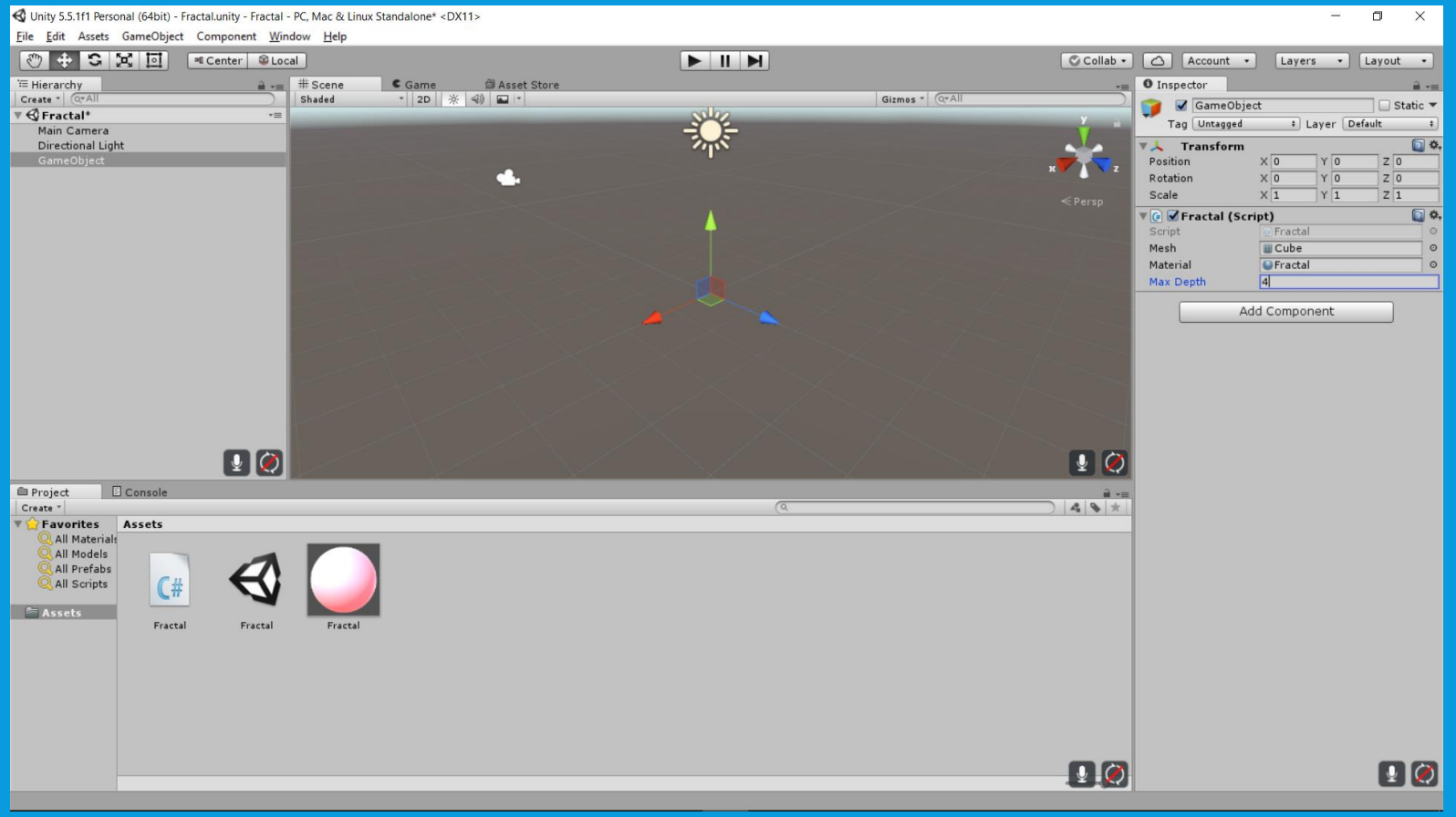
MAX DEPTH FØJES TIL SCRIPTET

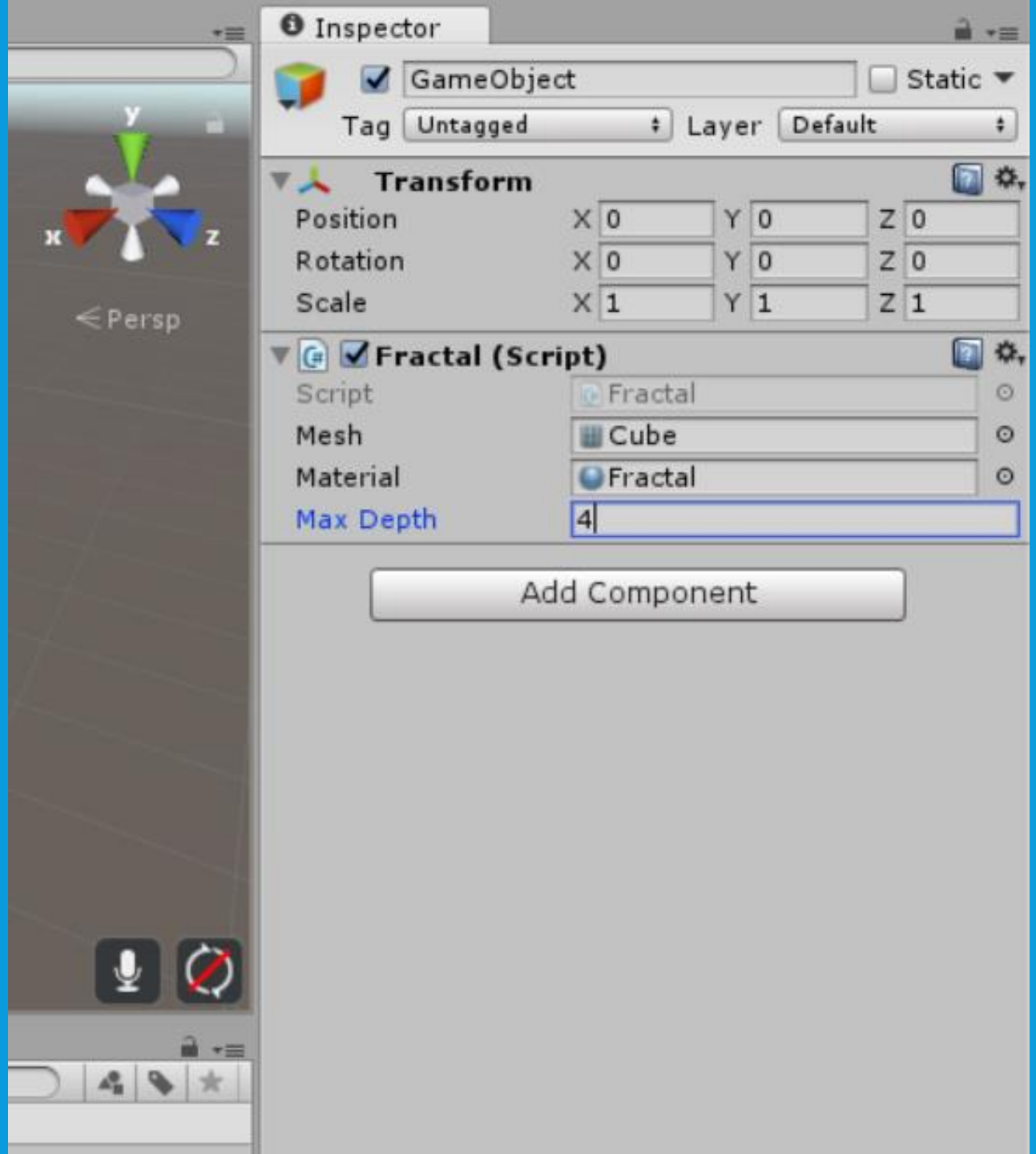


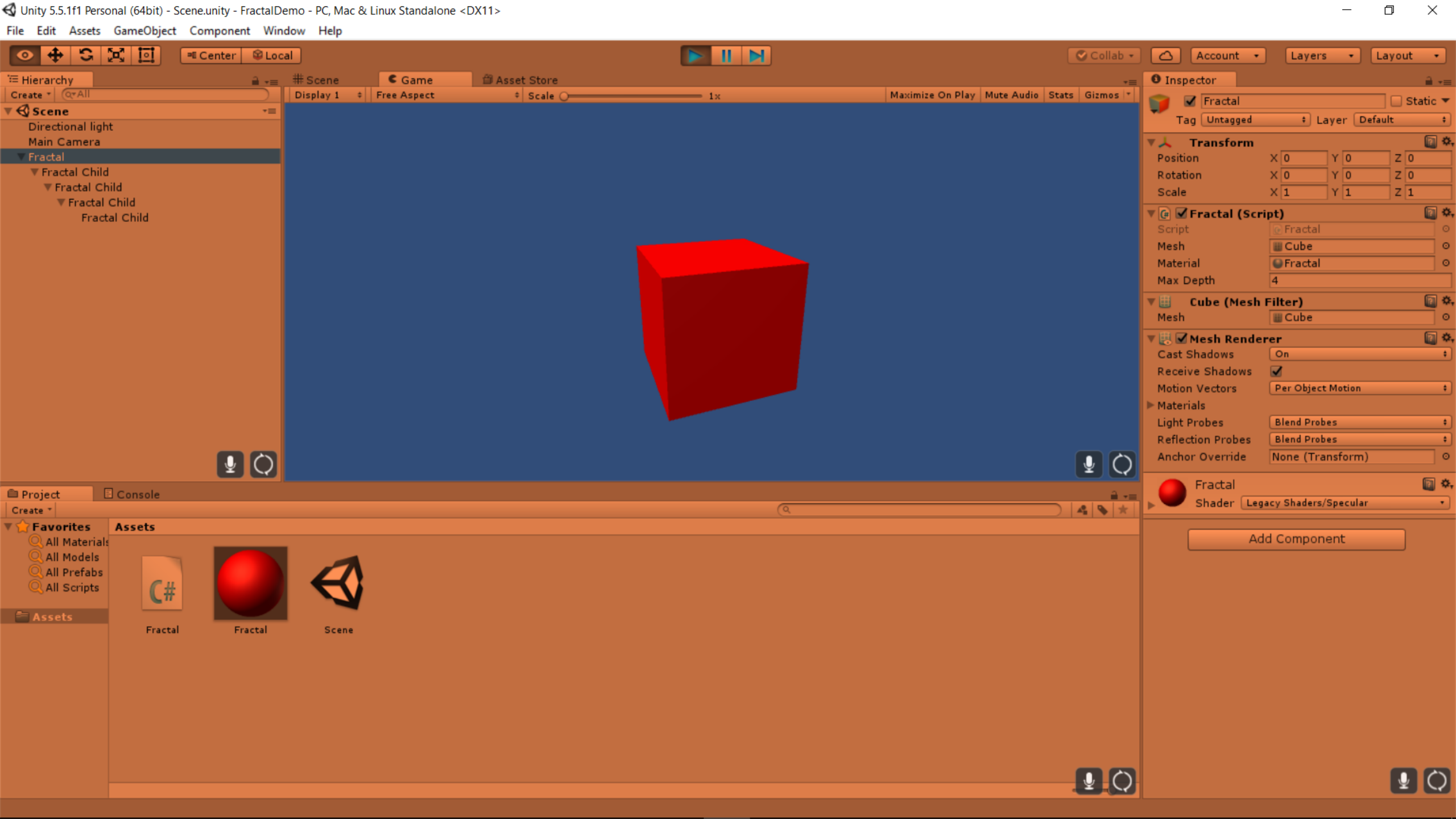
Føj linje 10-12 til og erstat den gamle linje 14 (nu 18) med de ny linjer 18-20

```
4
5 public class Fractal : MonoBehaviour {
6
7     public Mesh mesh;
8     public Material material;
9
10    public int maxDepth;
11
12    private int depth;
13
14    // Use this for initialization
15    void Start () {
16        gameObject.AddComponent<MeshFilter>().mesh = mesh;
17        gameObject.AddComponent<MeshRenderer>().material = material;
18        if (depth < maxDepth) {
19            new GameObject("Fractal Child").AddComponent<Fractal>();
20        }
21    }
22
```

SÆT MAX DEPTH TIL 4



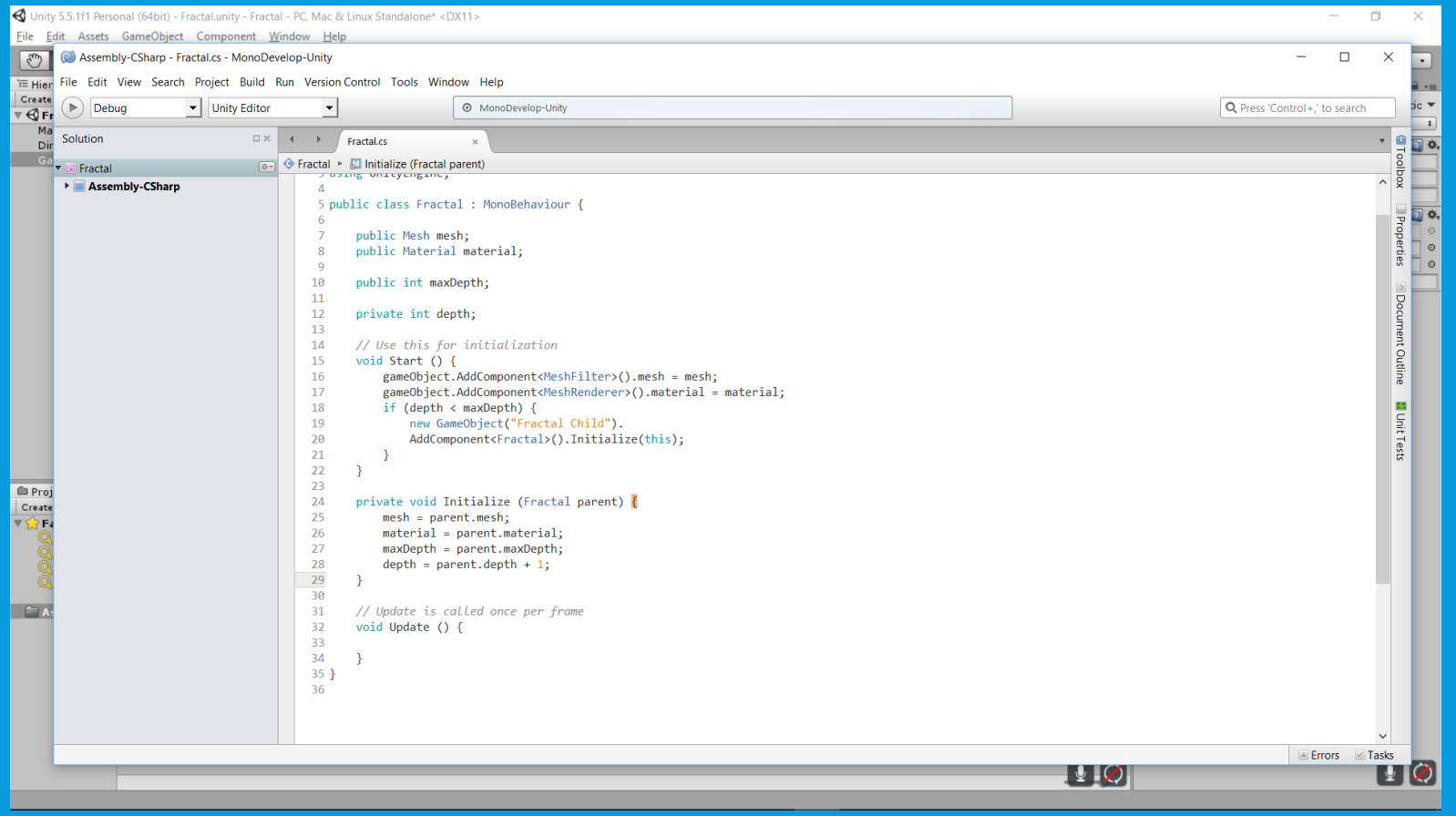

Inspector
GameObject
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Fractal (Script)
Script
Fractal
Mesh
Cube
Material
Fractal
Max Depth
4
Add Component
y
x
z
Persp


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Fractal Child
Fractal Child
Fractal Child
Fractal Child
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Mesh Cube
Material Fractal
Max Depth 4
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Add Component
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene

SKER DER NOGET?

- Præcis ét barn blev skabt da vi aldrig gav en dybde værdi. Derfor er den altid nul. Fordi nul er mindre end 4 har vores rod fraktal objekt skabt et barn. Barnets dybde værdi er derfor nul.
- Vi har ligeledes aldrig sat barnets maxDepth, så den er derfor nul. Derfor har barnet ligeledes ikke dannet yderligere.
- Vi kan heller ikke se barnet, da det mangler bade material og mesh.
- Vi er nødt til at kopiere disse referencer fra sit forældreren.
- Derfor tilføjer vi en ny metode der tager sig af alle de nødvendige initieringer.
-

VI FØJER DET TIL SCRIPTET



Føj linje 24-29 til og erstat de gamle linjer 19-20 med de ny linjer

```
15      void Start () {
16          gameObject.AddComponent<MeshFilter>().mesh = mesh;
17          gameObject.AddComponent<MeshRenderer>().material = material;
18          if (depth < maxDepth) {
19              new GameObject("Fractal Child").
20              AddComponent<Fractal>().Initialize(this);
21          }
22      }
23
24      private void Initialize (Fractal parent) {
25          mesh = parent.mesh;
26          material = parent.material;
27          maxDepth = parent.maxDepth;
28          depth = parent.depth + 1;
29      }
30
31      // Update is called once per frame
32      void Update () {
```

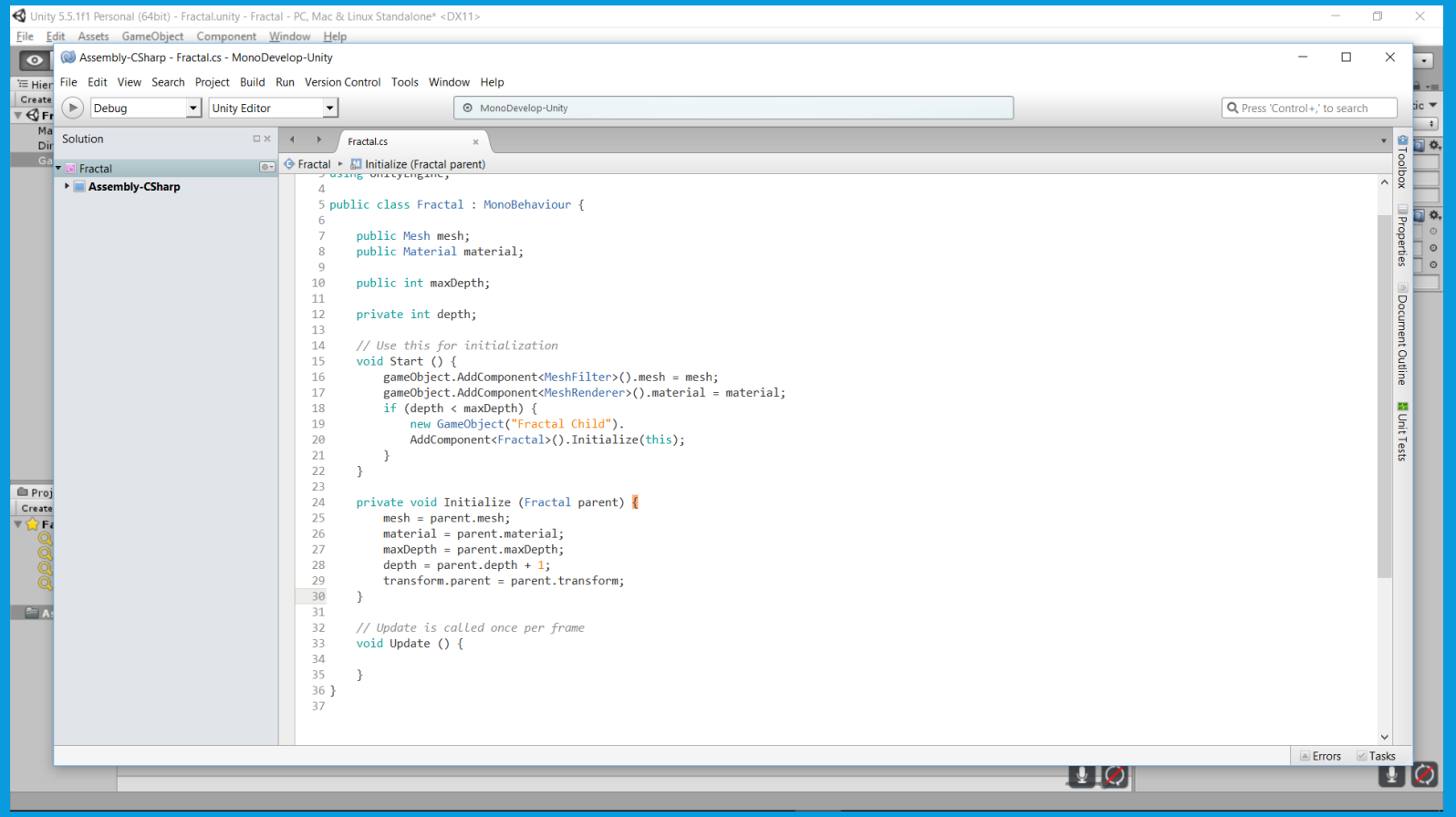
GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvad gør this?**
  - *this* nøgleordet henviser til det aktuelle objekt eller struct hvis metode bliver kaldt.
  - Det bliver brugt implicit hele tiden når der henvises til ting fra den samme klasse. For eksempel, når vi har adgang depth kan vi gøre det via this.depth.
  - Man bruger det typisk når man skal sende en reference til objektet selv, ligesom vi gør med *initialize*.
  - Vi kalder nemlig *initialize* metoden i det nye efterkommer objekt, ikke fra forælder objektet.
- **initialize kaldes før start**
  - Først opretter vi det nye spil objekt.
  - Så vil en ny Fractal komponent bliver oprettet og føjet til det.
  - På dette tidspunkt ville dens Awake og OnEnable metoder påberåbes, hvis de havde eksisteret.
  - Så gøres AddComponent metoden færdig.
  - Umiddelbart efter påberåber vi initialize.
  - Kaldet til at starte begynder ikke før næste frame.

SKER DER NOGET?

- Når vi nu går ind i playmode vil fire børn blive oprettet, som forventet.
- De er dog ikke rigtig børn, da de alle vises i hierarkiet rod. Forældre-barn forholdet mellem spil objekter er defineret ved deres transformations hierarki. Så et barn er nødt til at gøre således at forælderen af den transformerede komponent svarer til den fraktale forælders transform.

NY KODE DER ORDNER HIERAKIET

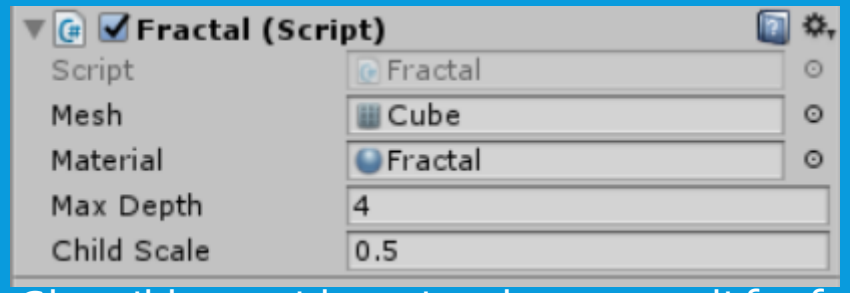


Føj linje 29 til

```
14      // Use this for initialization
15      void Start () {
16          gameObject.AddComponent<MeshFilter>().mesh = mesh;
17          gameObject.AddComponent<MeshRenderer>().material = material;
18          if (depth < maxDepth) {
19              new GameObject("Fractal Child").
20              AddComponent<Fractal>().Initialize(this);
21          }
22      }
23
24      private void Initialize (Fractal parent) {
25          mesh = parent.mesh;
26          material = parent.material;
27          maxDepth = parent.maxDepth;
28          depth = parent.depth + 1;
29          transform.parent = parent.transform;
30      }
```

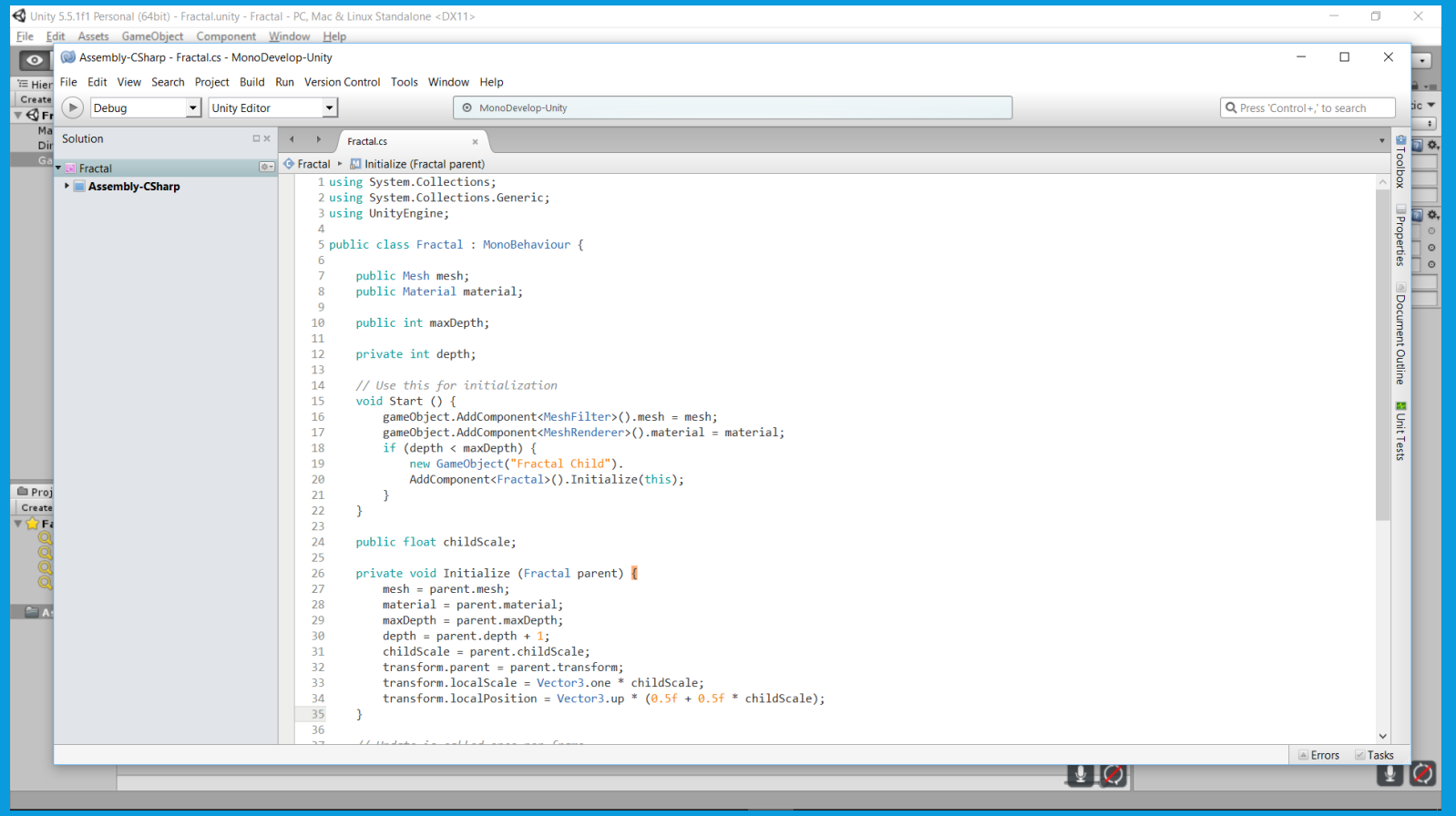
MEN HVOR ER BØRNENE?

- Indtil videre er børnene superimposed oven på deres forældre, hvilket betyder, at vi stadig kun se en enkelt kasse.
- Vi er nødt til at flytte dem ind i deres lokale rum, så de bliver synlige.
- Fordi de skal være mindre end deres forældre er vi nødt til at skalere dem også.
- Vi gør skaleringen konfigurerbar med en ny variabel med navnet childScale og tildeler den en værdi på 0,5 i inspektøren efter vi har føjet koden til.



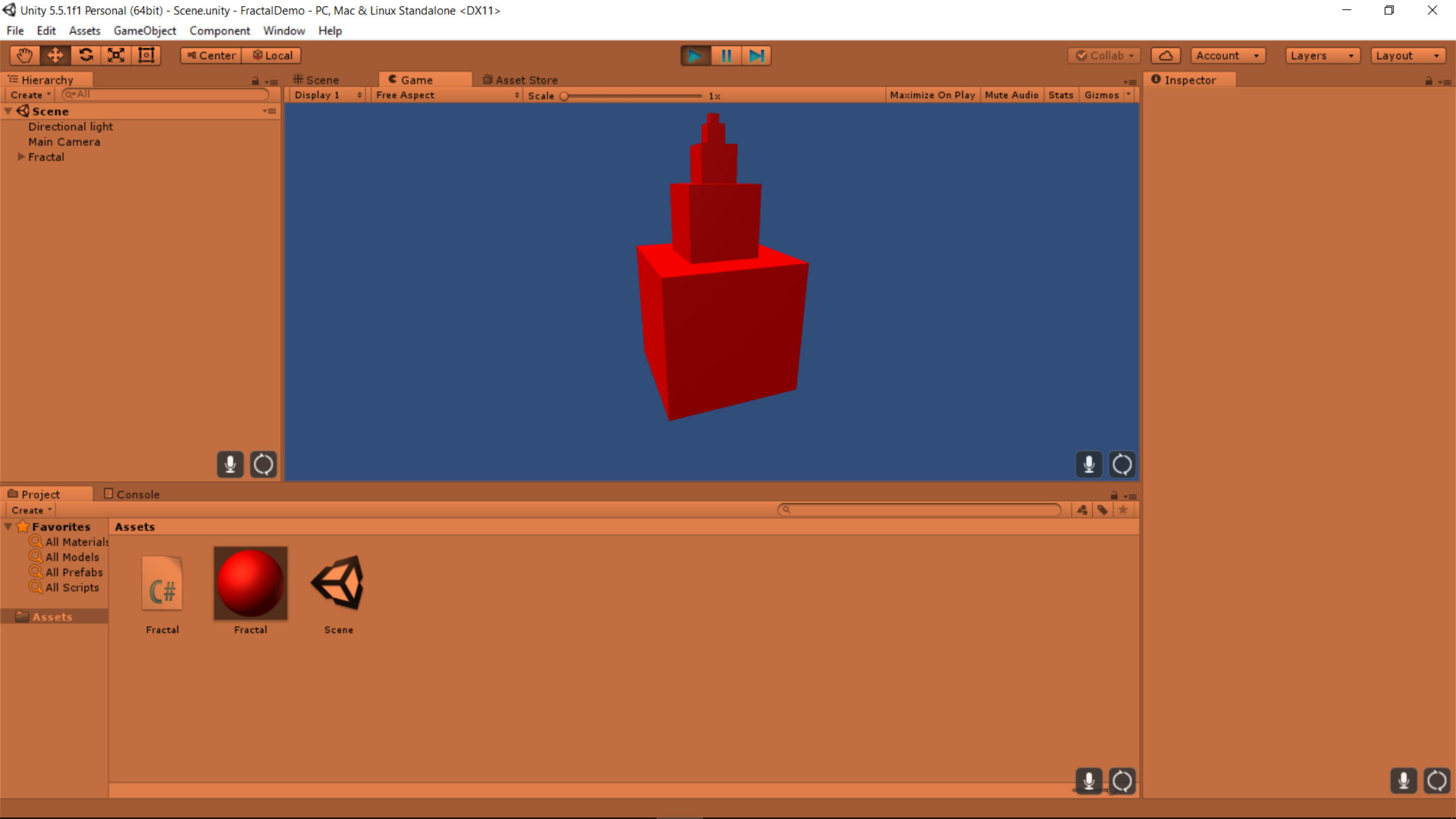
- Glem ikke at videregive denne værdi fra forælder til barn også. Så brug den til at sætte barnets lokale skala.
- Børnene skal som sagt flyttes så de kan ses, så lad os lægge dem lige op, så de rører deres forældre. Vi antager, at forældrene har en størrelse på én i alle retninger, hvilket er tilfældet for den kube, som vi bruger. Flytning op af en halv sætter os på det punkt, hvor forældre og barn skal røre.
  Så vi er nødt til at flytte et yderligere værdi svarende til halvdelen af størrelsen af barnet.

NY KODE DER FLYTTER OG SKALERER



Erstat linje 24-30 med linje 24-35

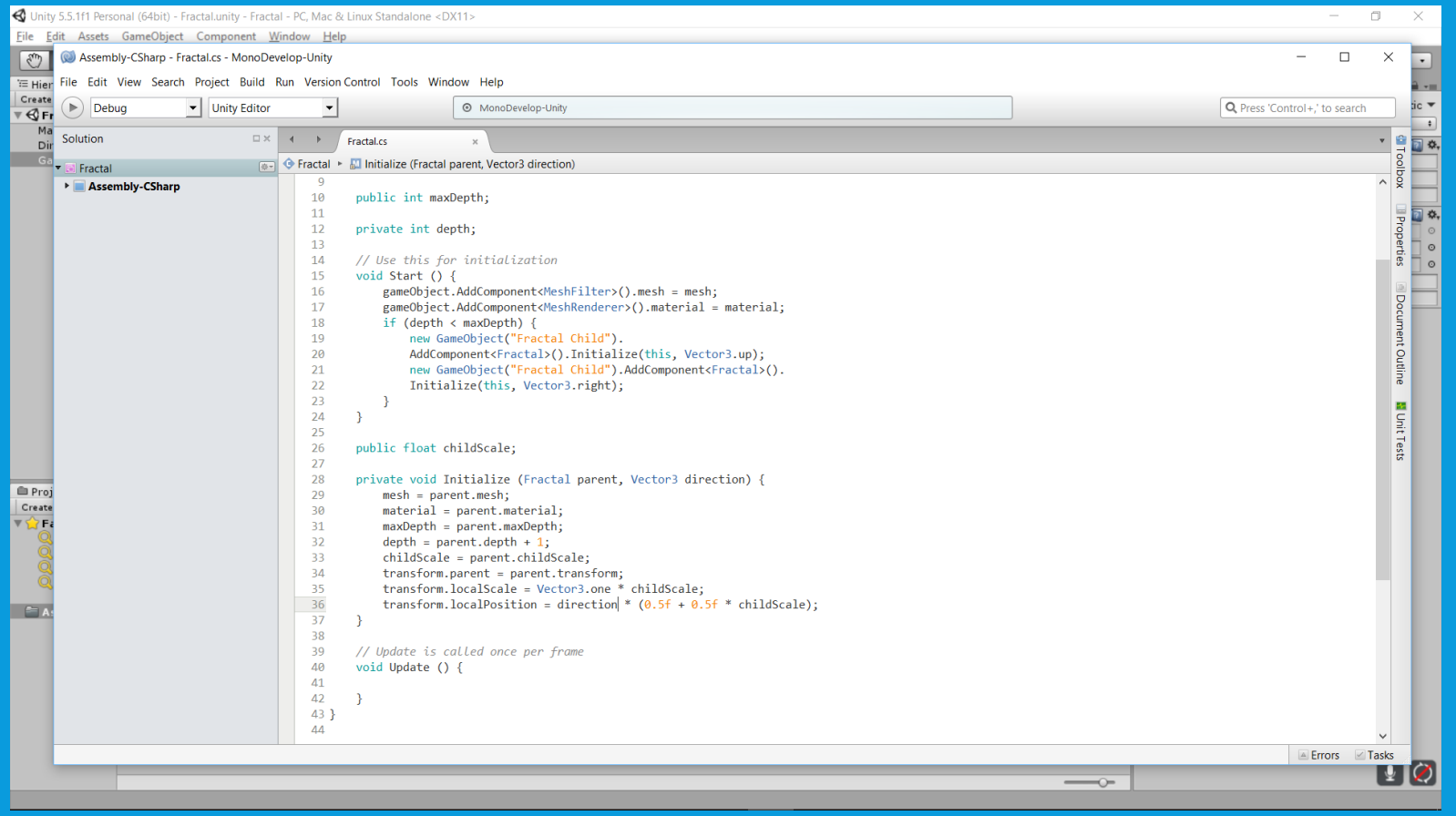
```
16          gameObject.AddComponent<MeshFilter>().mesh = mesh;
17          gameObject.AddComponent<MeshRenderer>().material = material;
18          if (depth < maxDepth) {
19              new GameObject("Fractal Child").
20              AddComponent<Fractal>().Initialize(this);
21          }
22      }
23
24      public float childScale;
25
26      private void Initialize (Fractal parent) {
27          mesh = parent.mesh;
28          material = parent.material;
29          maxDepth = parent.maxDepth;
30          depth = parent.depth + 1;
31          childScale = parent.childScale;
32          transform.parent = parent.transform;
33          transform.localScale = Vector3.one * childScale;
34          transform.localPosition = Vector3.up * (0.5f + 0.5f * childScale);
35      }
```


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene

FLERE BØRN I ANDRE RETNINGER

- Hvad vi laver nu ligner et tårn, men ikke rigtig en fraktal. Vi skal have det til at forgrene sig.
- Det gør vi ved at oprette flere børn per forælder.
- Det er nemt at bare oprette en anden genstand, men den skal også vokse i en anden retning.
- Derfor tilføjer vi en retnings parameter til vores Initialize metode og benytter den til at placere det andet barn til højre i stedet for op.

FLERE BØRN I ANDRE RETNINGER



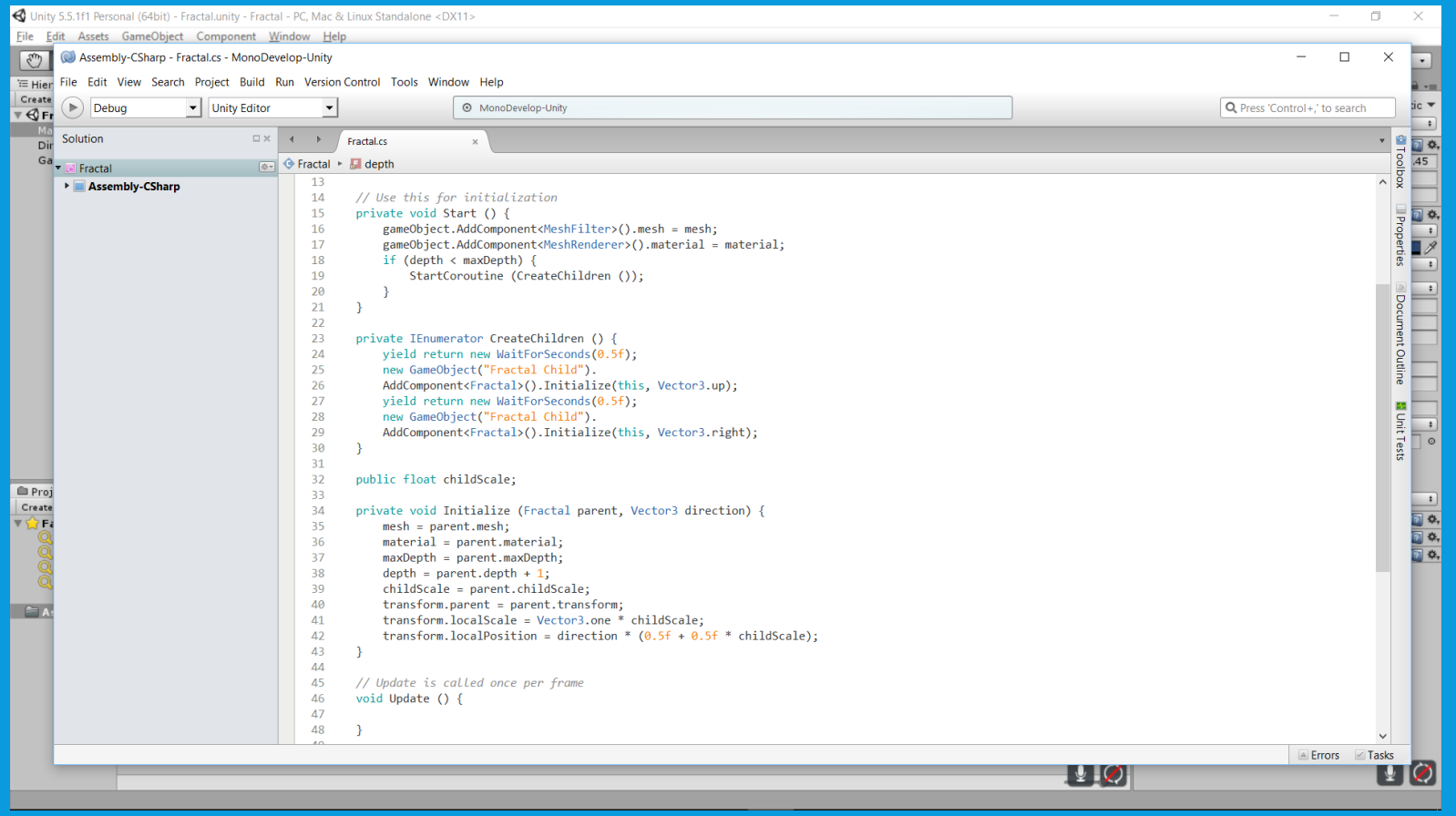
Erstat linje 20 med linje 20-22 og føj det ny til linje 26 og 34 (28 og 36 hvis du har sat de første linjer ind)

```
18          if (depth < maxDepth) {
19              new GameObject("Fractal Child").
20              AddComponent<Fractal>().Initialize(this, Vector3.up);
21              new GameObject("Fractal Child").AddComponent<Fractal>().
22              Initialize(this, Vector3.right);
23          }
24      }
25
26      public float childScale;
27
28      private void Initialize (Fractal parent, Vector3 direction) {
29          mesh = parent.mesh;
30          material = parent.material;
31          maxDepth = parent.maxDepth;
32          depth = parent.depth + 1;
33          childScale = parent.childScale;
34          transform.parent = parent.transform;
35          transform.localScale = Vector3.one * childScale;
36          transform.localPosition = direction * (0.5f + 0.5f * childScale);
37      }
```

LAD OS SÆNKE TEMPOET SÅ VI KAN SE FRAKTALEN GRO

- Da alle figurerne er skabt inden for et par frames går det for hurtigt til at vi kan se det vokse.
- Lad os bremse denne proces ned, så vi kan se det ske. Vi kan gøre dette ved at lade en coroutine skabe børnene.
- Tænk på coroutines som metoder, hvor du kan indsætte pause udsagn.
  - Mens metodeaktiveringen er sat på pause fortsætter resten af programmet.
  - Selvom dette synspunkt meget forenklet, er det hvad vi har brug for at gøre brug af lige nu.
- Vi flytter derfor de to linjer, der skaber børn, ind i en ny metode, CreateChildren.
- Denne metode er nødt til at have IEnumerator som en return type, hvilket stammer fra System.Collections namespace.
  - Det er derfor Unity inkluderer det I deres standard script template og vi har bibeholdt det.
- I stedet for bare at kalde denne metode i Start, er vi nødt til at kalde det som et argument for Unitys StartCoroutine metode.
- Derefter føjes et pause direktiv inden vi opretter hvert barn.
  - Vi gør dette ved at skabe et nyt WaitForSeconds objekt i et halvt sekund eller deromkring, hvorefter det gives tilbage til Unity.

LAD OS SÆNKE TEMPOET SÅ VI KAN SE FRAKTALEN GRO



Erstat linje 18 og frem med den ny kode

```
18          if (depth < maxDepth) {
19              StartCoroutine (CreateChildren ());
20          }
21      }
22
23      private IEnumerator CreateChildren () {
24          yield return new WaitForSeconds(0.5f);
25          new GameObject("Fractal Child").
26          AddComponent<Fractal>().Initialize(this, Vector3.up);
27          yield return new WaitForSeconds(0.5f);
28          new GameObject("Fractal Child").
29          AddComponent<Fractal>().Initialize(this, Vector3.right);
30      }
31
32      public float childScale;
33
34      private void Initialize (Fractal parent, Vector3 direction) {
35          mesh = parent.mesh;
36          material = parent.material;
37          maxDepth = parent.maxDepth;
```

```
30      }
31
32      public float childScale;
33
34      private void Initialize (Fractal parent, Vector3 direction) {
35          mesh = parent.mesh;
36          material = parent.material;
37          maxDepth = parent.maxDepth;
38          depth = parent.depth + 1;
39          childScale = parent.childScale;
40          transform.parent = parent.transform;
41          transform.localScale = Vector3.one * childScale;
42          transform.localPosition = direction * (0.5f + 0.5f * childScale);
43      }
44
45      // Update is called once per frame
46      void Update () {
47
48      }
```

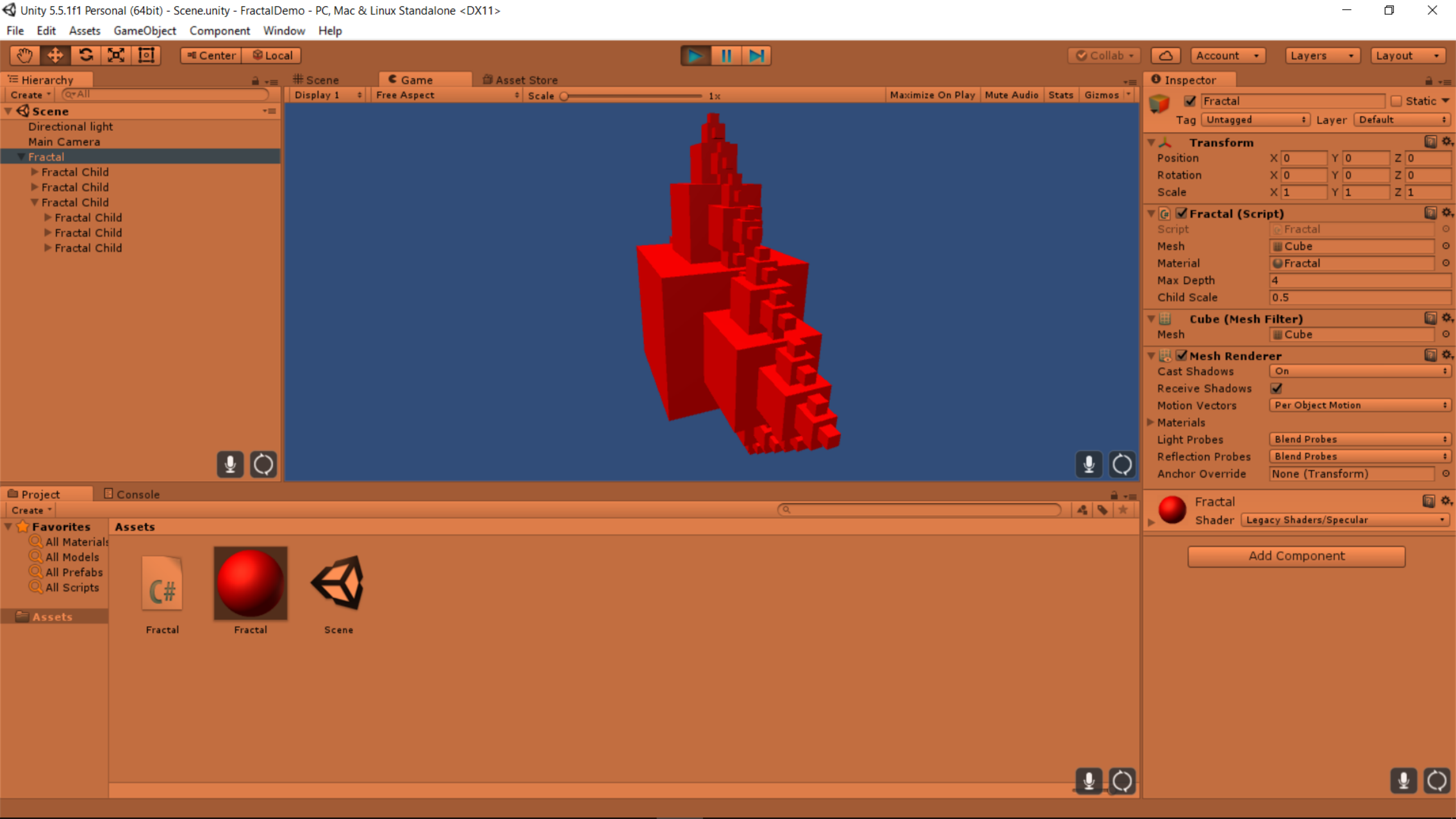
GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvad er en enumerator?**
- Optælling (*enumeration*) er begrebet når man går gennem en samling (collection) ét element ad gangen, ligesom looping igennem alle elementer i et array.
- En optæller (enumerator) - eller iteratoren - er et objekt, der giver en grænseflade til denne funktion.
  System.Collections.IEnumerator beskriver en sådan grænseflade.
- Coroutines benytte dette, hvilket også er grunden til Unity har System.Collections i deres standard script skabelon, og derfor vi har den med.

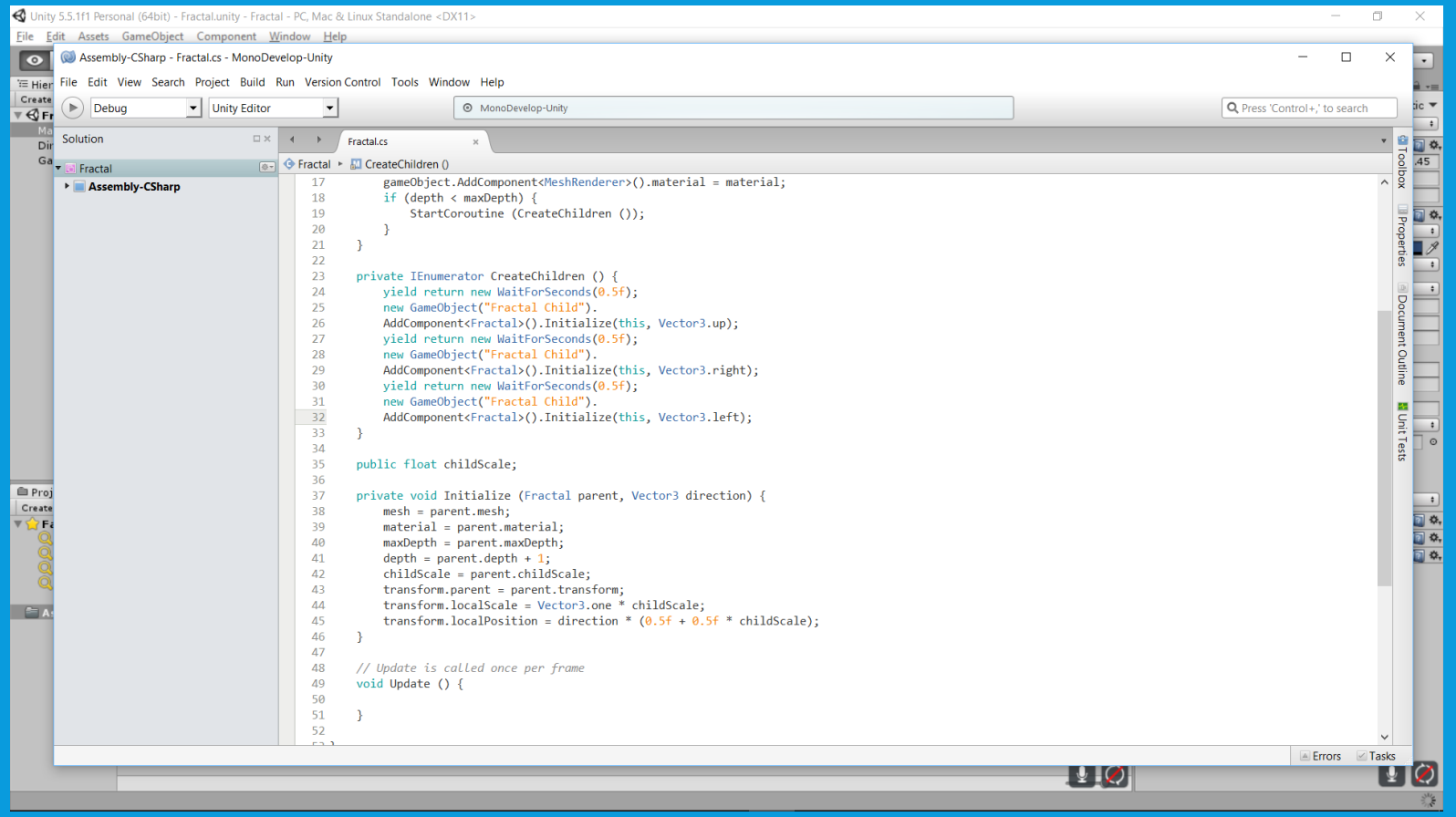
- **Hvad gør return?**
  - Du bruger *return* nøgleordet til at indikere, at en metode er færdig og hvad dens resultat er. Hvad du sender tilbage, skal passe til metodens type. Hvis det er en void metode skal man blot returnere noget.
  - Det er ikke nødvendigt at have en return-sætning i slutningen af en void eller en special constructor metode, for alle andre metoder er det påkrævet.
  - Det er muligt at have flere return-sætninger inde i en metode. I så fald er der flere mulige udgangspunkter.
    Man bruger typisk if statements til at afgøre, hvilken return der bliver brugt.

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvad gør yield?**
  - Et *yield* statement bruges af iterators.
  - For at gøre enumeration mulig er man nødt til at holde styr på ens fremskridt.
    Dette indebærer noget standardtekst (boilerplate) kode, der stort set altid er den samme.
    Det vi gerne vil er reelt bare at kunne skrive return firstItem; return secondItem; indtil vi er færdige. Dette tillader yield statementet.
  - Når man benytter yield bliver et enumerator objekt skabt bag kulissen. Derfor har vores CreateChildren metode IEnumerator som dens return type.
  - Man kan også yield'e en anden iterator. I så fald vil denne anden iterator blive behandlet fuldstændigt, så du kan kombinere dem på kreative måder.
- **Hvordan virker coroutines?**
  - Når man opretter en coroutine i Unity laver man reelt en iterator.
  - Når man sender den til StartCoroutine metoden bliver den opbevaret og bliver bedt om sin næste punkt hver frame, indtil den er færdig.
  - Yield statements producerer elementerne. Statements i mellem - de ting, man ønsker skal ske - er bivirkninger af iteratoren der gør sit arbejde.
  - Du kan give yield særlige ting som WaitForSeconds for at have mere kontrol over, hvornår din egen kode fortsætter, men den overordnede tilgang er blot en iterator.


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Fractal Child
Fractal Child
Fractal Child
Fractal Child
Fractal Child
Fractal Child
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Mesh Cube
Material Fractal
Max Depth 4
Child Scale 0.5
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Add Component
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene

LAD OS FØJE ET BARN TIL DEN ANDEN SIDE OGSÅ



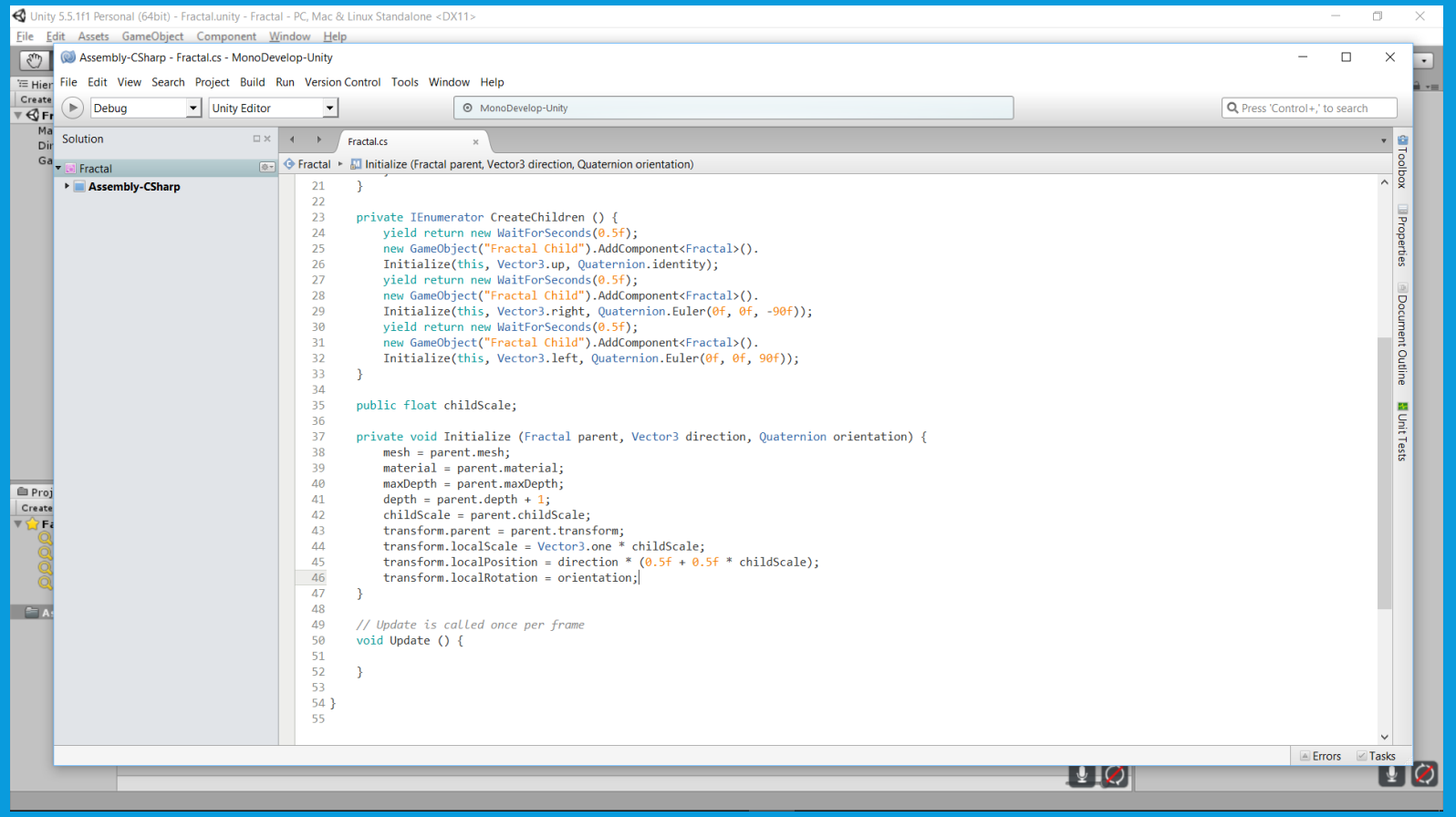
Føj linje 30-33 til efter linje 29

```
23      private IEnumerator CreateChildren () {
24          yield return new WaitForSeconds(0.5f);
25          new GameObject("Fractal Child").
26          AddComponent<Fractal>().Initialize(this, Vector3.up);
27          yield return new WaitForSeconds(0.5f);
28          new GameObject("Fractal Child").
29          AddComponent<Fractal>().Initialize(this, Vector3.right);
30          yield return new WaitForSeconds(0.5f);
31          new GameObject("Fractal Child").
32          AddComponent<Fractal>().Initialize(this, Vector3.left);
33      }
34
35      public float childScale;
36
37      private void Initialize (Fractal parent, Vector3 direction) {
38          mesh = parent.mesh;
39          material = parent.material;
40          maxDepth = parent.maxDepth;
41          depth = parent.depth + 1;
42          childScale = parent.childScale;
```

BØRNENES RETNING RETTES TIL

- Det er et problem at børnene har samme retning som deres forældre.
- Det betyder, at et venstre barn, hvis forælder selv er et højre barn, vil befinde sig inde i sin bedsteforælder og vice versa.
- For at løse dette, vi nødt til at rotere børnene, så deres opadgående retning vil pege væk fra deres forældre.
- Dette løses ved at tilføje en orientering parameter til Initialize.
- Det er en quaternion der bruges til at indstille den lokale rotation af det nye barn.
- Det opadgående barn behøver ingen rotation, det højre barn har brug for at rotere 90 grader med uret, og venstre barn har brug for at rotere i den modsatte retning.

BØRNENES RETNING RETTES TIL



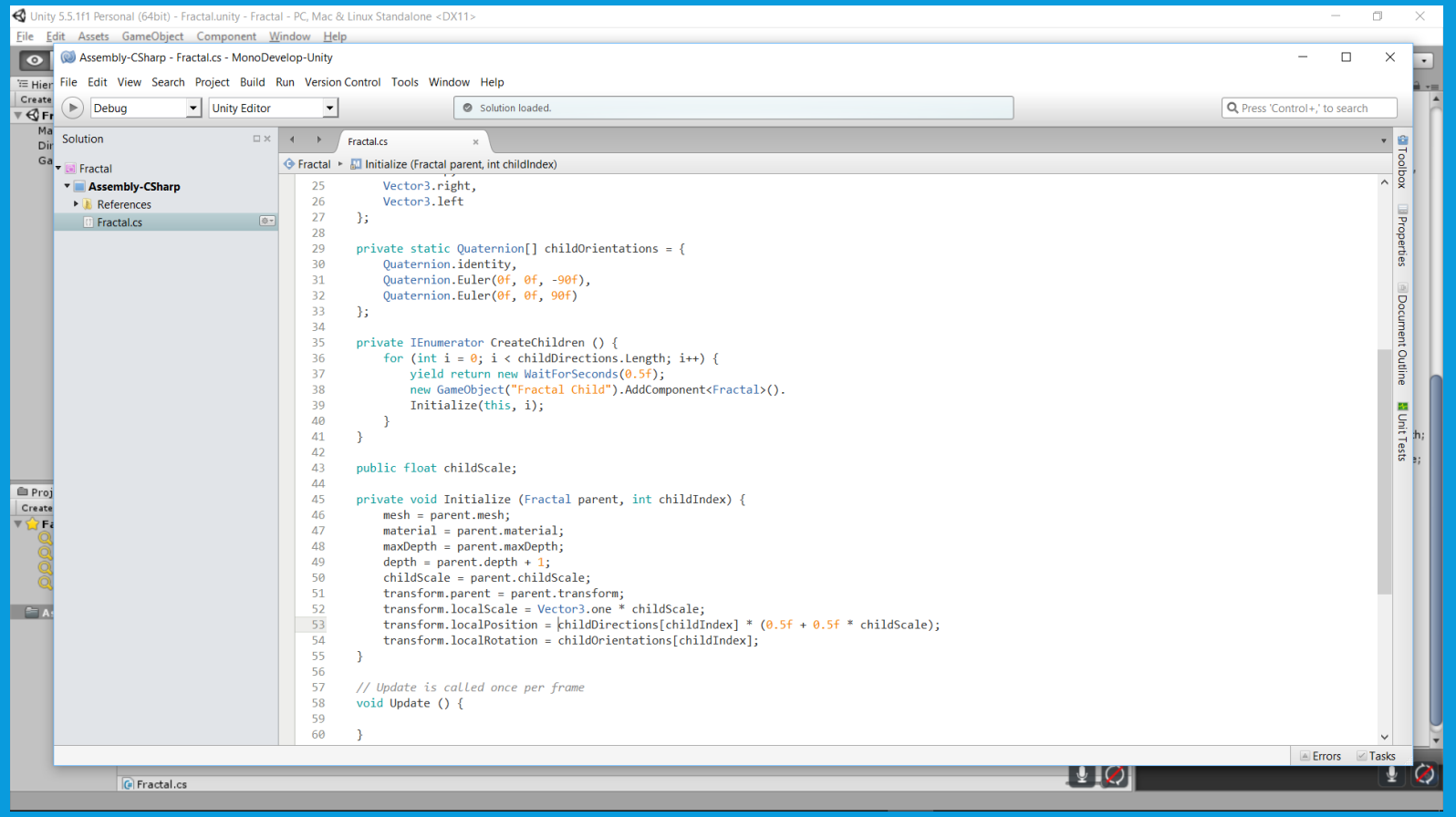
Føj rettelserne til linje 23-33 og linje 37 og føj linje 46 til

```
23      private IEnumerator CreateChildren () {
24          yield return new WaitForSeconds(0.5f);
25          new GameObject("Fractal Child").AddComponent<Fractal>().
26          Initialize(this, Vector3.up, Quaternion.identity);
27          yield return new WaitForSeconds(0.5f);
28          new GameObject("Fractal Child").AddComponent<Fractal>().
29          Initialize(this, Vector3.right, Quaternion.Euler(0f, 0f, -90f));
30          yield return new WaitForSeconds(0.5f);
31          new GameObject("Fractal Child").AddComponent<Fractal>().
32          Initialize(this, Vector3.left, Quaternion.Euler(0f, 0f, 90f));
33      }
34
35      public float childScale;
36
37      private void Initialize (Fractal parent, Vector3 direction, Quaternion orientation) {
38          mesh = parent.mesh;
39          material = parent.material;
40          maxDepth = parent.maxDepth;
41          depth = parent.depth + 1;
42          childScale = parent.childScale;
43          transform.parent = parent.transform;
44          transform.localScale = Vector3.one * childScale;
45          transform.localPosition = direction * (0.5f + 0.5f * childScale);
46          transform.localRotation = orientation;
47      }
```

OPTIMERING AF KODEN GENNEM ARRAYS

- Koden er blevet lidt uhåndterlig.
- Lad os generalisere ved at flytte data retning og orientering til statiske arrays.
- Derved kan vi reducere CreateChildren til et kort loop og bruge barnet indeks som parameter for Initialize.

OPTIMERING AF KODEN GENNEM ARRAYS



Erstat linje 23-33 med linjerne 29-41, linje 37 (nu 45 efter ovenstående føjet til) rettes til og linje 45-56 (nu 53-54) ændres

```
29      private static Quaternion[] childOrientations = {
30          Quaternion.identity,
31          Quaternion.Euler(0f, 0f, -90f),
32          Quaternion.Euler(0f, 0f, 90f)
33      };
34
35      private IEnumerator CreateChildren () {
36          for (int i = 0; i < childDirections.Length; i++) {
37              yield return new WaitForSeconds(0.5f);
38              new GameObject("Fractal Child").AddComponent<Fractal>().
39              Initialize(this, i);
40          }
41      }
42
43      public float childScale;
44
45      private void Initialize (Fractal parent, int childIndex) {
46          mesh = parent.mesh;
47          material = parent.material;
48          maxDepth = parent.maxDepth;
49          depth = parent.depth + 1;
50          childScale = parent.childScale;
51          transform.parent = parent.transform;
52          transform.localScale = Vector3.one * childScale;
53          transform.localPosition = childDirections[childIndex] * (0.5f + 0.5f * childScale);
54          transform.localRotation = childOrientations[childIndex];
55      }
```

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvordan virker arrays?**
  - Arrays er objekter på en fast længde, som indeholder en lineær sekvens af variabler.
  - Når man erklærer en variabel, angiver man ved at sætte kantede parenteser bag dens type at man ønsker et array af denne type.
    - Så *int myVariable;* giver et heltal medens *int [] myVariable;* giver et array af heltal.
  - Adgang til en af posterne inde i en array fås ved at angive dens array indeks - ikke dets position - i firkantet parentes bag variablen.
    - Ved *myVariable [0]* får du den første post i arrayet, ved *myVariable [1]* får du den anden, og så videre.
  - Det at lave et array og tildele det til variablen gøres med
    myVariable = new int [10];
    Her laves der et nyt array med plads til ti poster.
  - Alternativt kan du oprette det implicit ved angive dets oprindelige værdier mellem krøllede parenteser, hvilket *myVariable = {1, 2, 3};* gør.

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvordan virker et for loop?**
  - En for-løkke er en kompakt måde at skrive en løkke, der gentager noget.

    for(int i = 0; i < 10; i++) { DoStuff(i); }

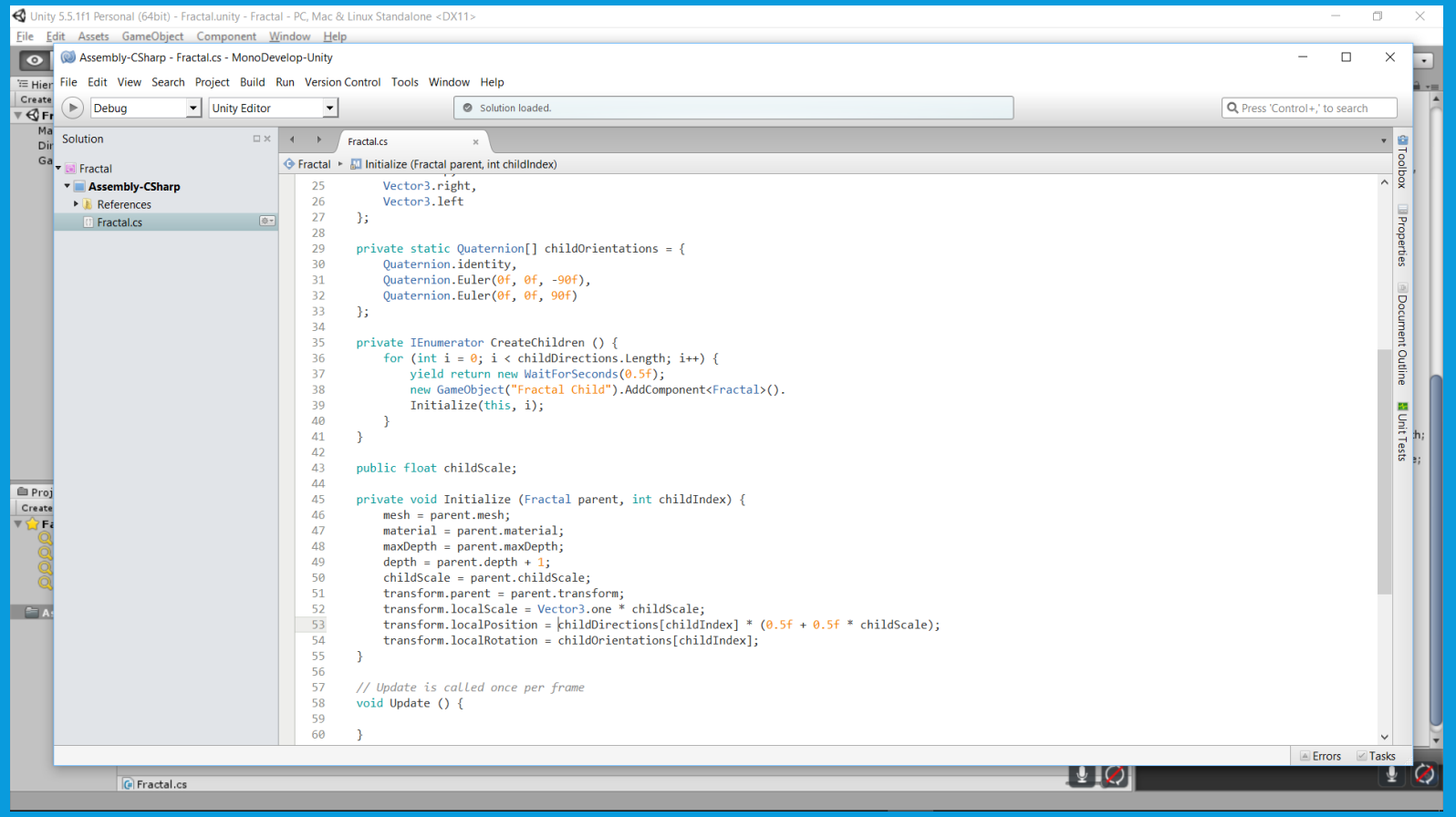
    - I dette tilfælde bruger vi et heltal ved navn i som iteratoren. Den første del erklærer iterator heltal, den anden del kontrollerer løkkens tilstand, og den tredje del inkrementerer iteratoren.
  - Du kan bruge en while-løkke til at få det nøjagtigt samme resultat, men det grupperer ikke iterator koden så pænt.
    - Således er koden herefter det same som linjen ovenfor.

int i = 0; while(i < 10) { DoStuff(i); i++; }

- *i*++ er en forkortelse for *i*+=*1*, som er en forkortelse for *i* = *i* + *1*.

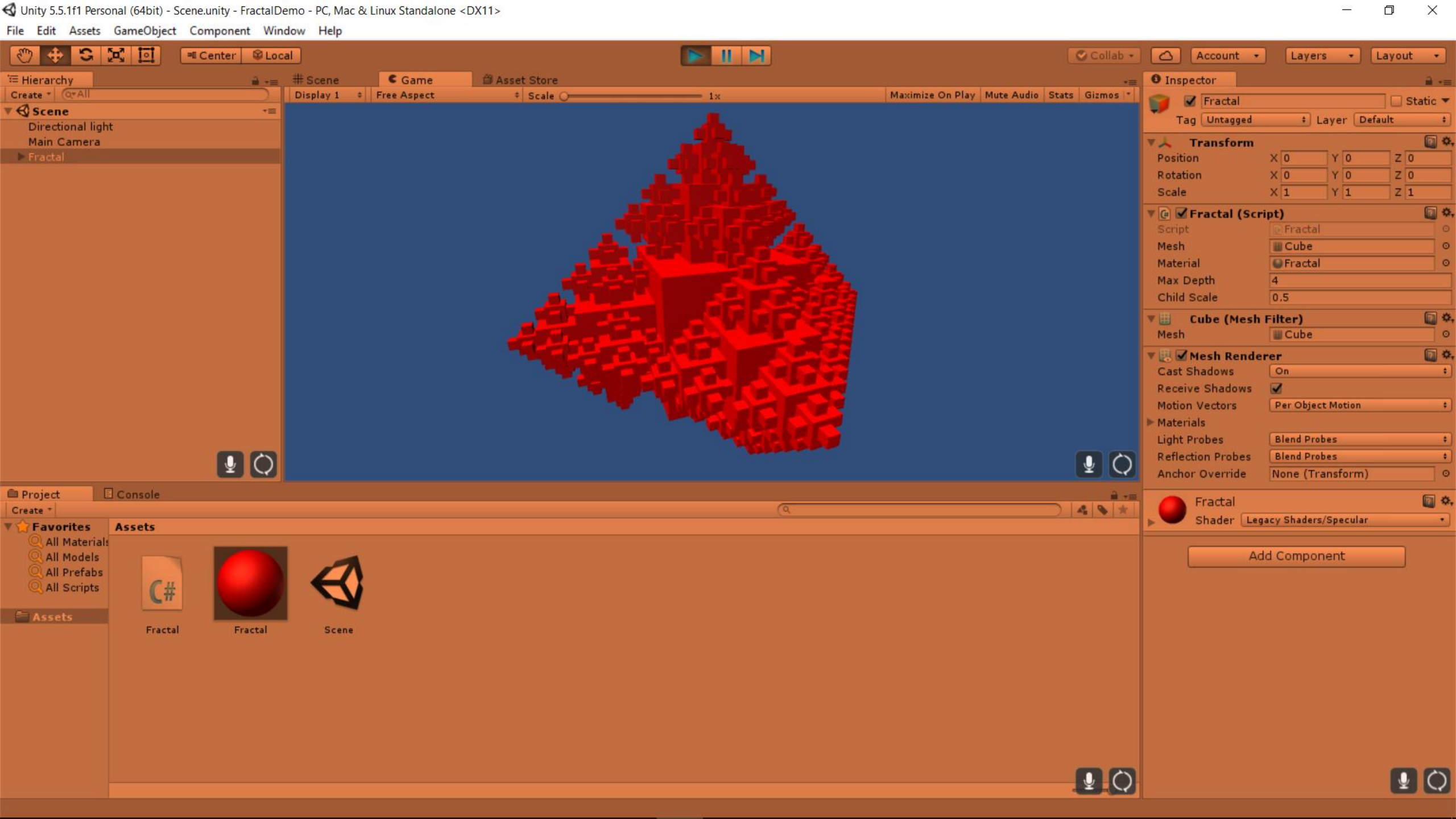
- Lad os nu indføre to børn med ved simpelt at tilføje deres data til arrays.
  - En fremadrettet, den anden går baglæns.

TO BØRN MERE GENNEM ARRAYS



Sæt linje 27-28 ind efter linje 26 og linje 35-36 efter linje 34 (32 uden de andre ny linjer)

```
        Vector3.right,
        Vector3.left
    };

    private static Quaternion[] childOrientations = {
        Quaternion.identity,
        Quaternion.Euler(0f, 0f, -90f),
        Quaternion.Euler(0f, 0f, 90f)
    };

    private IEnumerator CreateChildren () {
        for (int i = 0; i < childDirections.Length; i++) {
            yield return new WaitForSeconds(0.5f);
            new GameObject("Fractal Child").AddComponent<Fractal>().
            Initialize(this, i);
        }
    }

    public float childScale;

    private void Initialize (Fractal parent, int childIndex) {
        mesh = parent.mesh;
        material = parent.material;
        maxDepth = parent.maxDepth;
```

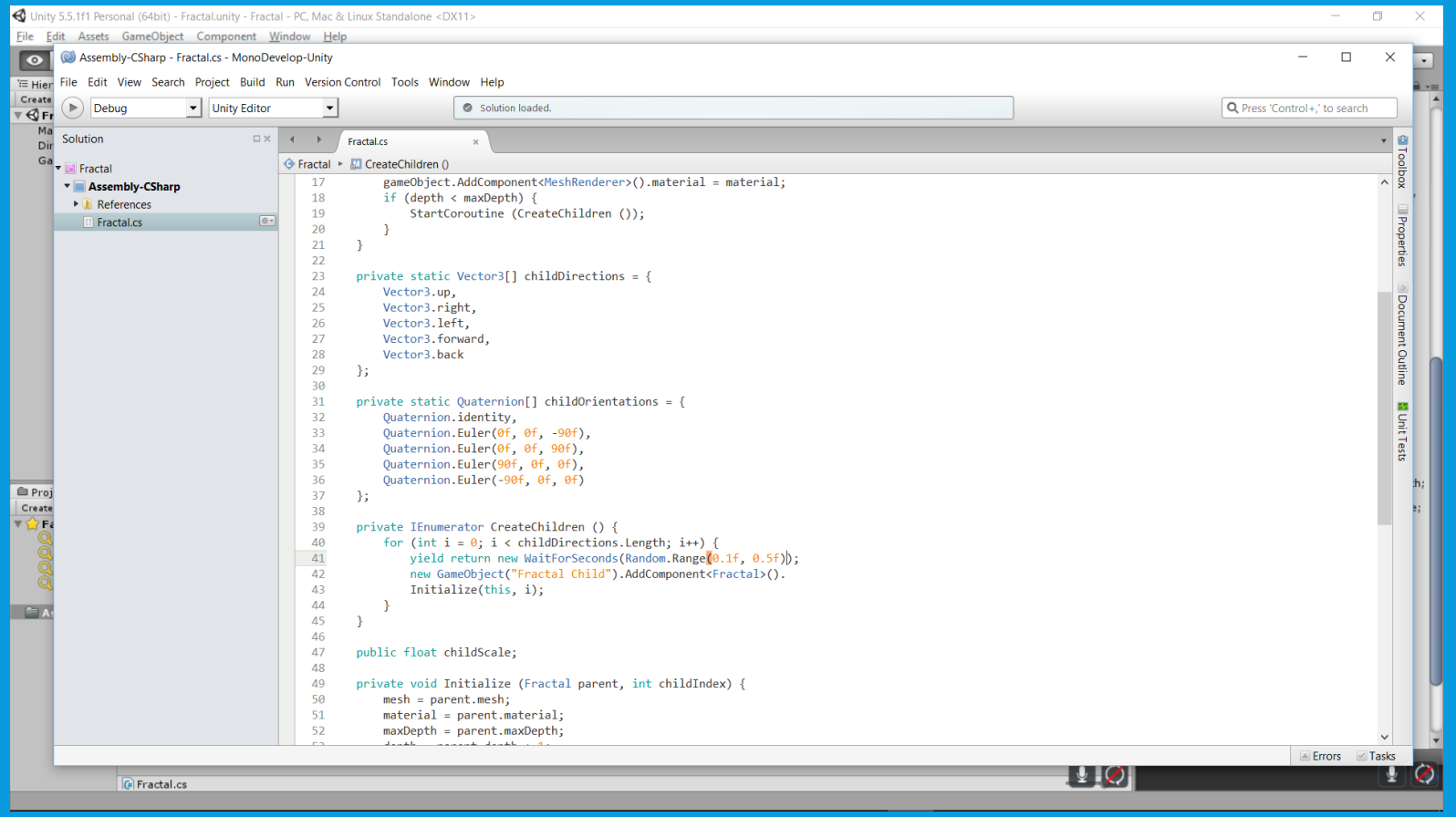

Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Mesh Cube
Material Fractal
Max Depth 4
Child Scale 0.5
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Add Component
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene

EKSPLOSIV VÆKST

- Hvor mange kuber laver vi reelt?
  - Da vi opretter fem børn per forælder vil det samlede antal kuber afhænge af den maksimale dybde.
  - En maksimal dybde på nul frembringer kun en terning, den oprindelige rod.
  - En maksimal dybde på en frembringer fem yderligere børn, for i alt seks kuber.
  - Da det er en fraktal vil dette mønster gentages, og vi kan skrive det som funktionen f (0) = 1, f (n) = 5 × f (n - 1) + 1.
  - Ovenstående funktion frembringer sekvensen 1, 6, 31, 156, 781, 3906, 19531, 97.656, og så videre.
  - Du vil se disse tal vise sig som mængden af draw opkald i game view statistics i Unity.

    Hvis du har dynamic batching aktiveret, vil det være summen af Draw opkald og Saved by batching.
  - Pas på en en for høj maksimal dybe, da en for høj vil resultere i en elendig framerate.
- Udover mængde er varighed er også et problem.
  - Lige nu holder vi pause et halvt sekund, før der oprettes et nyt barn. Dette frembringer synkroniserede byger af vækst i et par sekunder.
  - Vi kan distribuere væksten mere jævnt ved randomisering af forsinkelserne. Dette resulterer også i et mere uforudsigeligt og organisk mønster.

ERSTAT DEN FASTE FORSINKELSE MED ET TILFÆLDIGT INTERVAL MELLEM 0,1 OG 0,5 OG ØG DEN MAKSIMALE DYBDE TIL 5

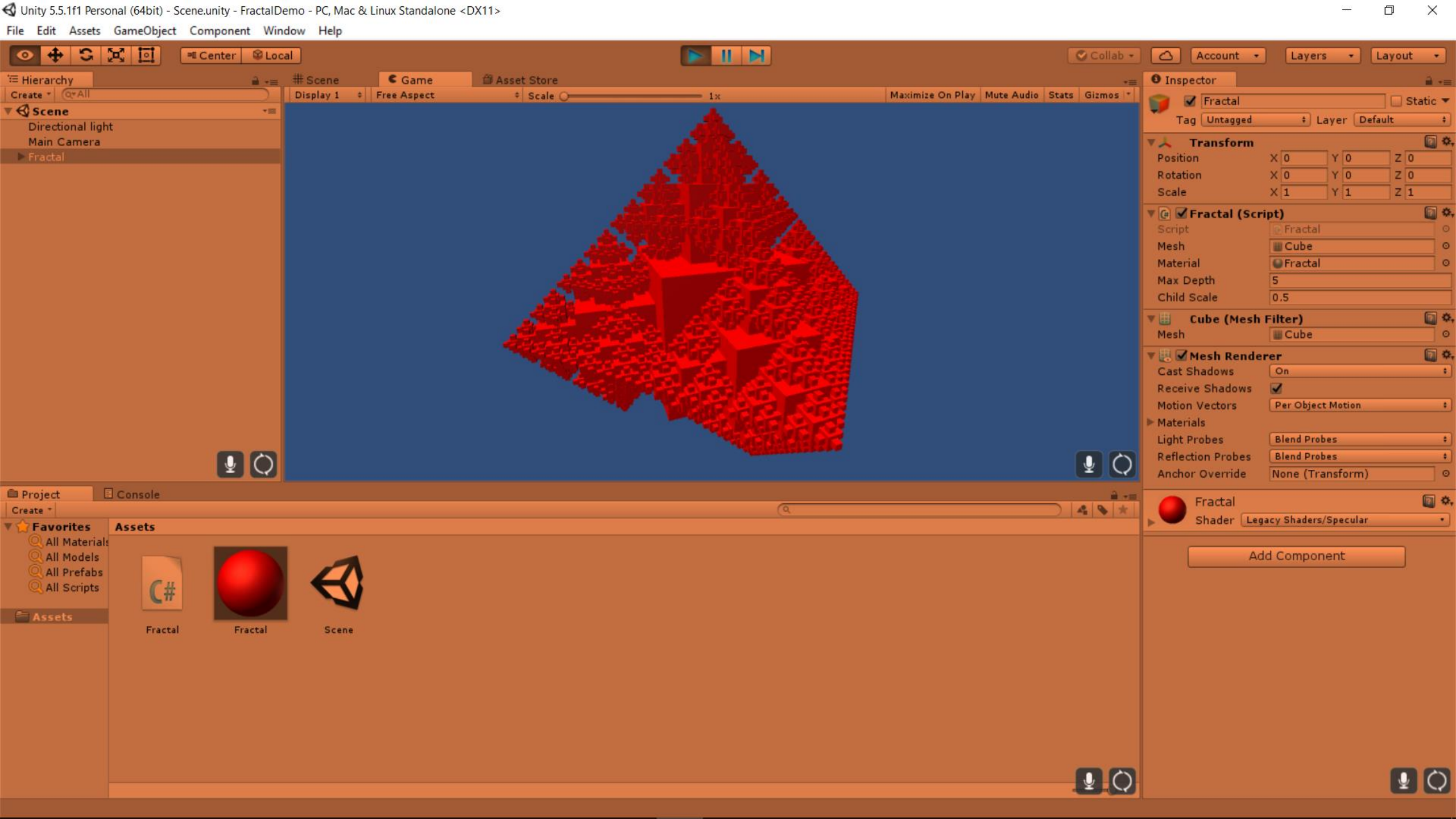


Erstat indholdet i parentesen på linje 41

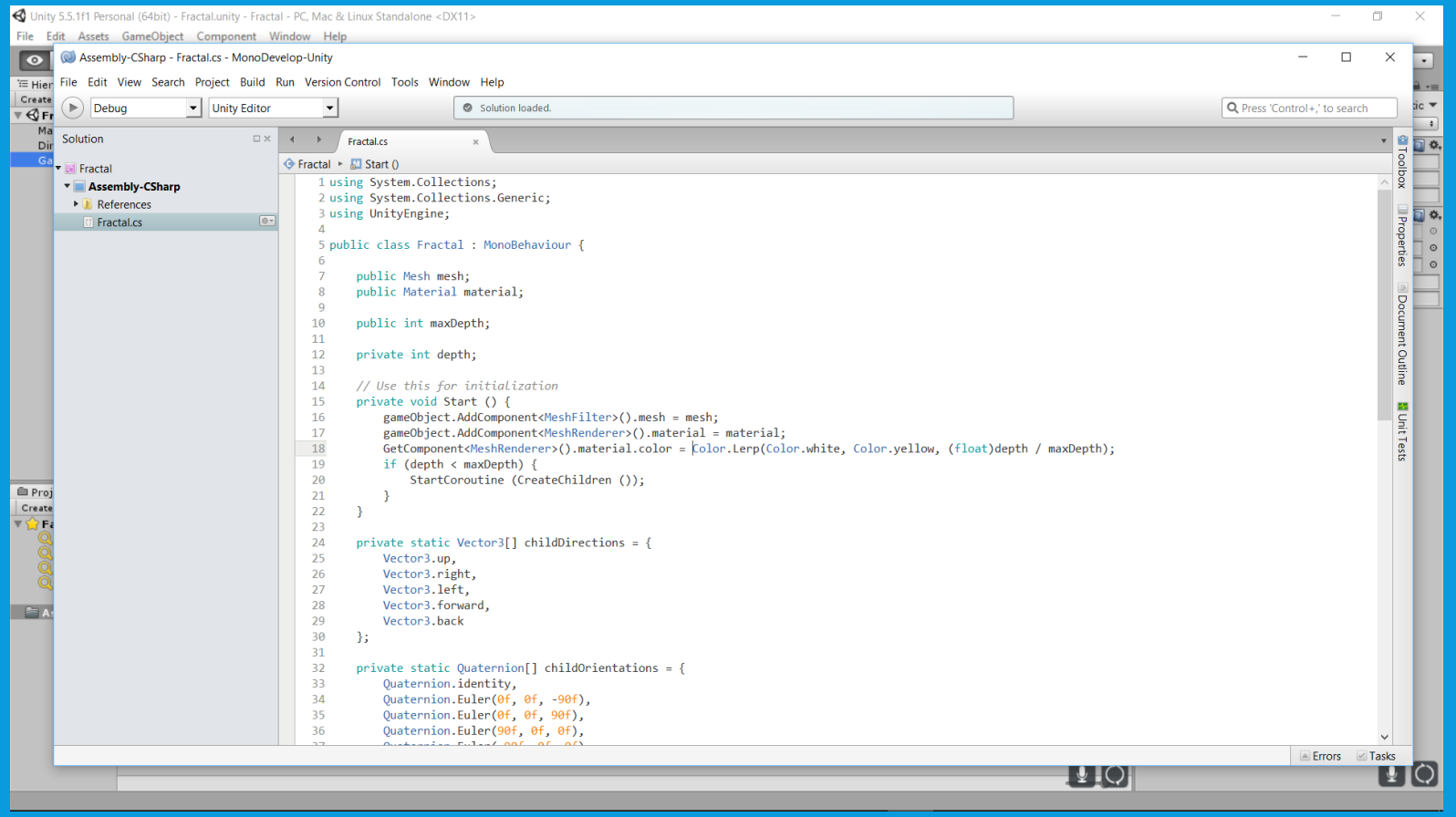
```
27          Vector3.forward,
28          Vector3.back
29      };
30
31      private static Quaternion[] childOrientations = {
32          Quaternion.identity,
33          Quaternion.Euler(0f, 0f, -90f),
34          Quaternion.Euler(0f, 0f, 90f),
35          Quaternion.Euler(90f, 0f, 0f),
36          Quaternion.Euler(-90f, 0f, 0f)
37      };
38
39      private IEnumerator CreateChildren () {
40          for (int i = 0; i < childDirections.Length; i++) {
41              yield return new WaitForSeconds(Random.Range(0.1f, 0.5f));
42              new GameObject("Fractal Child").AddComponent<Fractal>().
43              Initialize(this, i);
44          }
45      }
46
47      public float childScale;
48
49      private void Initialize (Fractal parent, int childIndex) {
```

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvordan virker random.range?**
  - *Random* er en hjælpeprogram klasse, som indeholder ting til at skabe tilfældige værdier.
    - Dens Range metode kan bruges til at generere en tilfældig værdi inden for en rækkevidde.
  - Der er to versioner af Range metoden.
    - Man kan kalde den med to floats, i hvilket tilfælde den returnerer en float mellem minimum og maksimum værdi, begge inklusive.
    - Alternativt kan man kalde range med to heltal, i hvilket tilfælde det returnerer et heltal mellem den mindste, inklusive, og maksimum, eksklusiv.
      Det typiske eksempel på brug af denne version er at vælge et indeks tilfældigt, *someArray[Random.Range(o, someArray.Length)].*


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Mesh Cube
Material Fractal
Max Depth 5
Child Scale 0.5
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Add Component
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene

LAD OS FØJE FARVE TIL



Føj koden på linje 18 til

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

public class Fractal : MonoBehaviour {

	public Mesh mesh;
	public Material material;

	public int maxDepth;

	private int depth;

	// Use this for initialization
	private void Start () {
		gameObject.AddComponent<MeshFilter>().mesh = mesh;
		gameObject.AddComponent<MeshRenderer>().material = material;
		GetComponent<MeshRenderer>().material.color = Color.Lerp(Color.white, Color.yellow, (float)depth / maxDepth);
		if (depth < maxDepth) {
			StartCoroutine (CreateChildren ());
		}
	}

	private static Vector3[] childDirections = {
		Vector3.up,
		Vector3.right,
		Vector3.left,
		Vector3.forward,
		Vector3.back
	};

```

LAD OS FØJE FARVE TIL

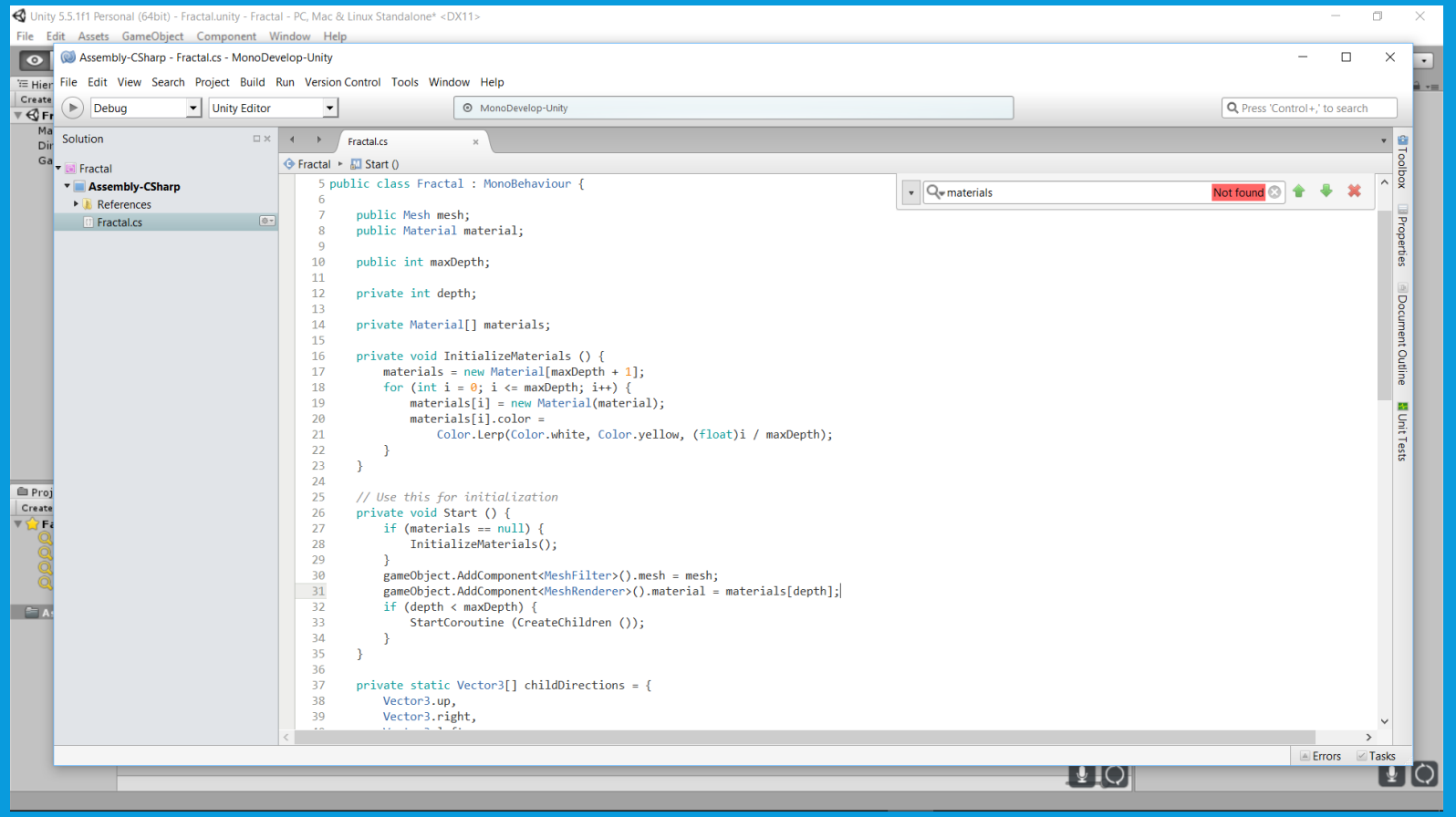
- Vi føjer farve til ved at interpolere fra hvid ved roden til gul på de mindste børn.
- Den statiske Color.Lerp metode er en praktisk måde at gøre dette på.
- Interpolatoren går fra nul til én, som vi gør ved at dividere vores aktuelle dybde med den maksimale dybde.
  - Da vi ikke ønsker en heltal division konverterer vi dybde til en float først.

- **Hvad gør lerp?**
  - Lerp er en forkortelse for lineær interpolation.
  - Dens typiske signatur er *Lerp (a, b, t)* og den beregner *a + (b - a) * t*, med t fastspændt til 0-1 intervallet.
  - Flere versioner findes med forskellige former, herunder float, vektorer og farver.

LAD OS OPTIMERE LIDT

- Lad os nøjes med at oprette ét materiale kopi per dybde i stedet for per terning.
- Vi tilføjer et nyt aray felt til at holde materialerne.
- Derefter kontrollerer vi om en matrix findes i Start, og hvis ikke kaldes en ny InitializeMaterials metode.
- I denne metode vil vi eksplicit kopiere vores materiale og ændre dens farve per dybde.
- I stedet for at videresende materiale henvisning fra forældre til barn, videresendes materiale arrray henvisningen i stedet.
- Hvis vi ikke gør det vil hvert barn blive tvunget til at skabe sit eget materialer array og vi ville ikke have løst problemet.

LAD OS OPTIMERE LIDT



Føj koden linje 14-23 og 27-29 til, ret slutningen af linje 31 og fjern linje 32.

```
14      private Material[] materials;
15
16      private void InitializeMaterials () {
17          materials = new Material[maxDepth + 1];
18          for (int i = 0; i <= maxDepth; i++) {
19              materials[i] = new Material(material);
20              materials[i].color =
21                  Color.Lerp(Color.white, Color.yellow, (float)i / maxDepth);
22          }
23      }
24
25      // Use this for initialization
26      private void Start () {
27          if (materials == null) {
28              InitializeMaterials();
29          }
30          gameObject.AddComponent<MeshFilter>().mesh = mesh;
31          gameObject.AddComponent<MeshRenderer>().material = materials[depth];
32          if (depth < maxDepth) {
33              StartCoroutine (CreateChildren ());
34          }
35      }
```

GENNEMGANG AF SCRIPTET INDTIL VIDERE

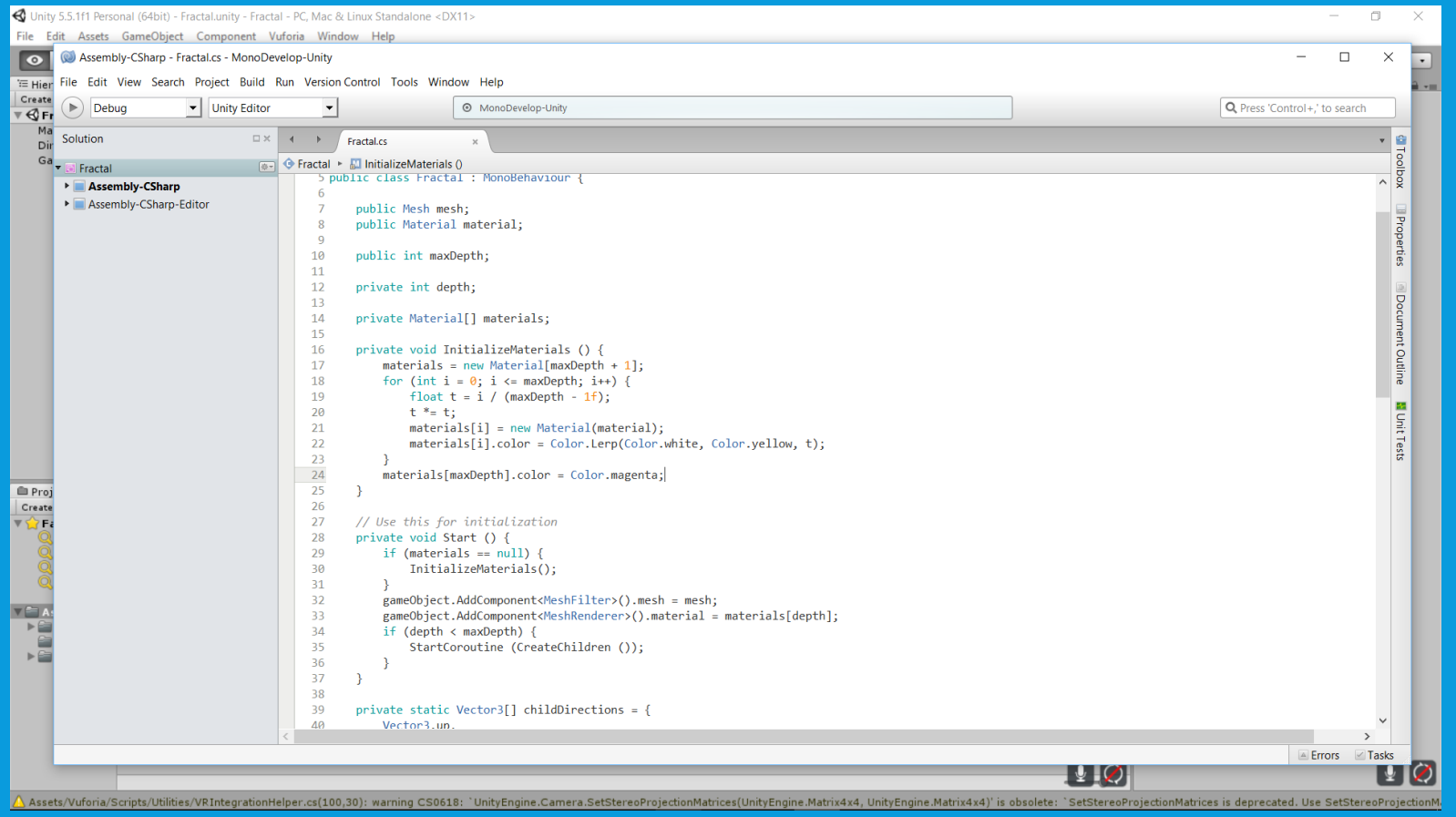
- **Hvad er null?**
  - Standardværdien af en variabel, der er ikke en simpel værdi, er null. Det betyder, at den variable ikke refererer noget. Forsøger at påberåbe sig eller få adgang til alt fra en variabel, der er null resulterer i en fejl.
  - Du kan teste for denne værdi for at sikre, at det ikke sker. Du kan også indstille en sådan variabel til null selv, hvis du ikke længere har brug for, hvad det var refererer.
- Bemærk, at objekter ikke automatisk ophøre med at eksistere, når du indstiller en henvisning til dem til null.
- Kun når der er ingen tilbage med en henvisning til dem, vil de blive kandidater til fjernelse af affaldsindsamling.

GENNEMGANG AF SCRIPTET INDTIL VIDERE

- **Hvorfor er materials ikke static?**
  - Vi gør ikke materiale array'et statisk, fordi det afhænger af den maksimale dybde, som kan være forskellig fra fraktal til fraktal.
  - På den måde sikrer vi at der kan være flere fraktaler på samme tid og selvfølgelig kan de have forskellige maksimale dybder.
  -

- Lad os ændre den sidste farve til magenta bagefterVi retter også interpolatoren, så vi stadig kan se den fulde overgang til gul. Vi sørger også lige for at det resulterer i en lidt pænere transition.

VI ÆNDRER FARVEN TIL MAGENTA



Føj koden linje 19-20, ret slutningen af linje 22 (sæt 22 og 23 sammen og ret) og føj linje 24 til

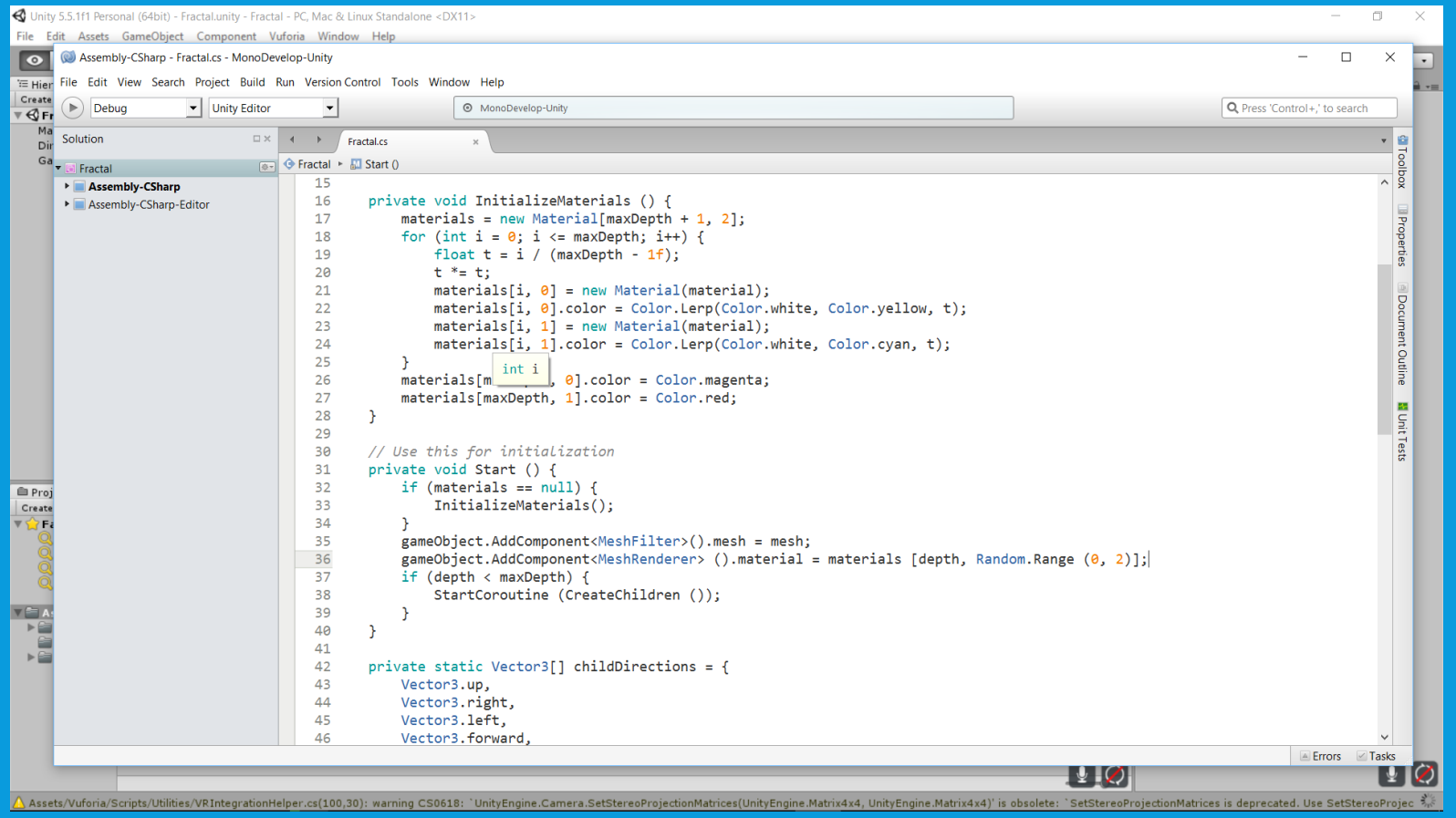
```
public class Fractal : MonoBehaviour {

	public Mesh mesh;
	public Material material;

	public int maxDepth;

	private int depth;

	private Material[] materials;

	private void InitializeMaterials () {
		materials = new Material[maxDepth + 1];
		for (int i = 0; i <= maxDepth; i++) {
			float t = i / (maxDepth - 1f);
			t *= t;
			materials[i] = new Material(material);
			materials[i].color = Color.Lerp(Color.white, Color.yellow, t);
		}
		materials[maxDepth].color = Color.magenta;
	}

```

FLERE FARVER

- Vi tilføjer en anden farve progression, fra hvid til cyan med røde spidser.
- Vi bruger et enkelt todimensional array til at holde dem begge, og vælger derefter et tilfældigt, når vi har brug for et materiale
- På den måde vil vores fraktal se anderledes ud hver gang går vi afspilningstilstand.

- **Hvordan virker to-dimensionelle arrays?**
  - Man kan tilføje en anden dimension til et array ved at indsætte et komma inde i dets parentes.
  - Man er derefter også nødt til at give to indekser, når man vil have adgang til et af array-elementerne.
  - Denne fremgangsmåde omfatter også højere dimensioner.

FLERE FARVER



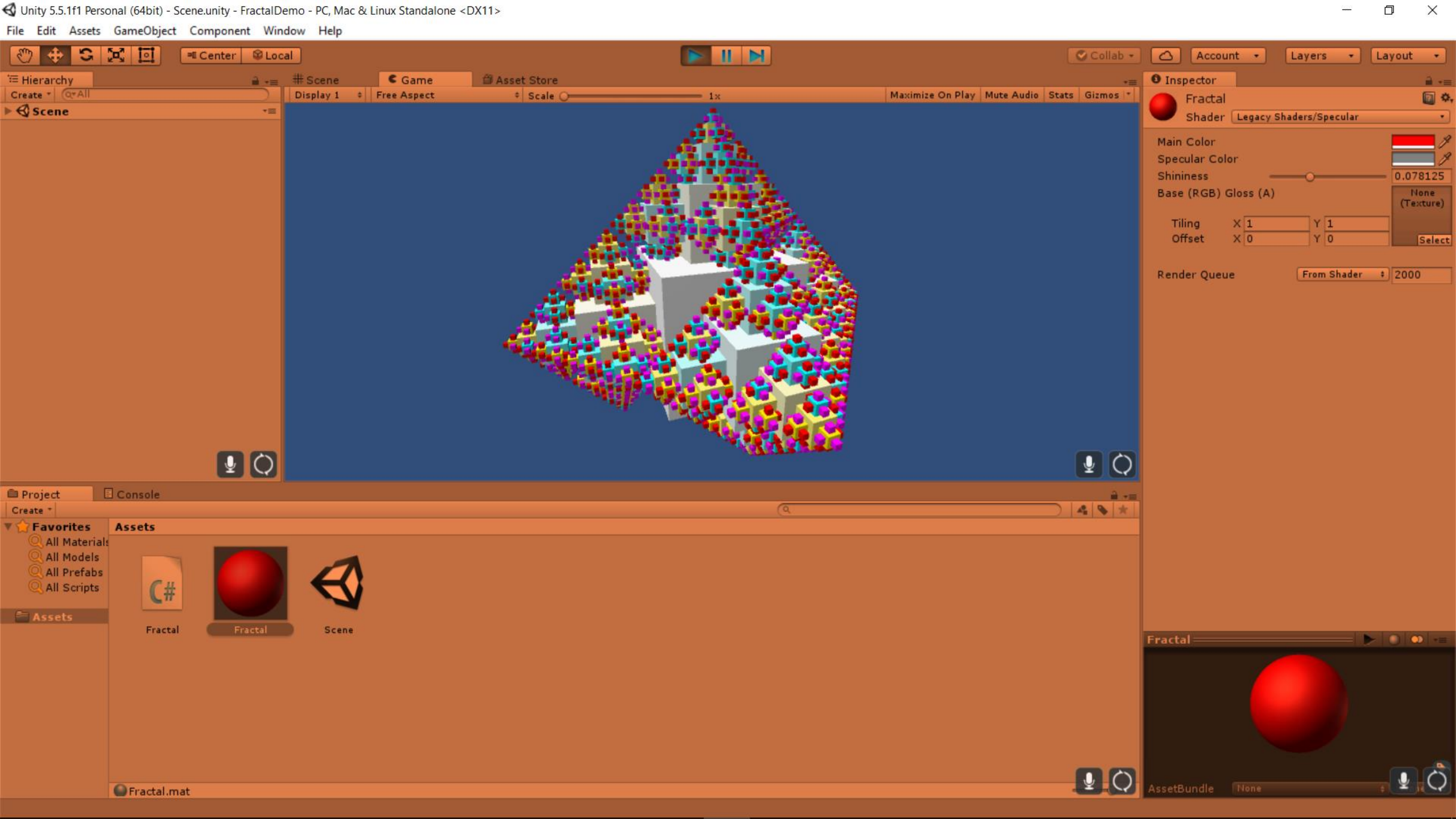
Føj , til [] i linje 14, føj , 2 til linje 17, ret linje 21-27 og føj ændringerne til slutningen af linje 36

Fractal.cs

Fractal ▸ Start ()

```
15
16      private void InitializeMaterials () {
17          materials = new Material[maxDepth + 1, 2];
18          for (int i = 0; i <= maxDepth; i++) {
19              float t = i / (maxDepth - 1f);
20              t *= t;
21              materials[i, 0] = new Material(material);
22              materials[i, 0].color = Color.Lerp(Color.white, Color.yellow, t);
23              materials[i, 1] = new Material(material);
24              materials[i, 1].color = Color.Lerp(Color.white, Color.cyan, t);
25          }
26          materials[m                , 0].color = Color.magenta;
27          materials[maxDepth, 1].color = Color.red;
28      }
29
30      // Use this for initialization
31      private void Start () {
32          if (materials == null) {
33              InitializeMaterials();
34          }
35          gameObject.AddComponent<MeshFilter>().mesh = mesh;
36          gameObject.AddComponent<MeshRenderer> ().material = materials [depth, Random.Range (0, 2)];
37          if (depth < maxDepth) {
38              StartCoroutine (CreateChildren ());
39          }
```

int i


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal
Shader Legacy Shaders/Specular
Main Color
Specular Color
Shininess 0.078125
Base (RGB) Gloss (A)
None (Texture)
Tiling X 1 Y 1
Offset X 0 Y 0
Select
Render Queue From Shader 2000
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal.mat
Fractal
AssetBundle None

FRAKTAL

Koden indtil videre – ryddet en smule op

Assembly-CSharp - Fractal.cs - MonoDevelop-Unity

File Edit View Search Project Build Run Version Control Tools Window Help

Debug | Unity Editor | MonoDevelop-Unity | Press 'Control+,' to search

Solution

- FractalDemo
  - **Assembly-CSharp**
    - References
    - Fractal.cs

Fractal.cs

Fractal ▸ Initialize (Fractal parent, int childIndex)

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Fractal : MonoBehaviour {
5
6      private static Vector3[] childDirections = {
7          Vector3.up,
8          Vector3.right,
9          Vector3.left,
10         Vector3.forward,
11         Vector3.back
12     };
13
14     private static Quaternion[] childOrientations = {
15         Quaternion.identity,
16         Quaternion.Euler(0f, 0f, -90f),
17         Quaternion.Euler(0f, 0f, 90f),
18         Quaternion.Euler(90f, 0f, 0f),
19         Quaternion.Euler(-90f, 0f, 0f)
20     };
21
22     public Mesh mesh;
23     public Material material;
24     public int maxDepth;
25     public float childScale;
26
27     private int depth;
28     private Material[,] materials;
29
30     private void Start () {
31         if (materials == null) {
32             InitializeMaterials();
33         }
34         gameObject.AddComponent<MeshFilter>().mesh = mesh;
35         gameObject.AddComponent<MeshRenderer>().material =
36             materials[depth, Random.Range(0, 2)];
37         if (depth < maxDepth) {
38             StartCoroutine(CreateChildren());
39         }
40     }
41
```

Toolbox Properties Document Outline Unit Tests

Errors Tasks

Assembly-CSharp - Fractal.cs - MonoDevelop-Unity

File Edit View Search Project Build Run Version Control Tools Window Help

Debug | Unity Editor | Solution loaded. | Press 'Control+,' to search

Solution

FractalDemo

Assembly-CSharp

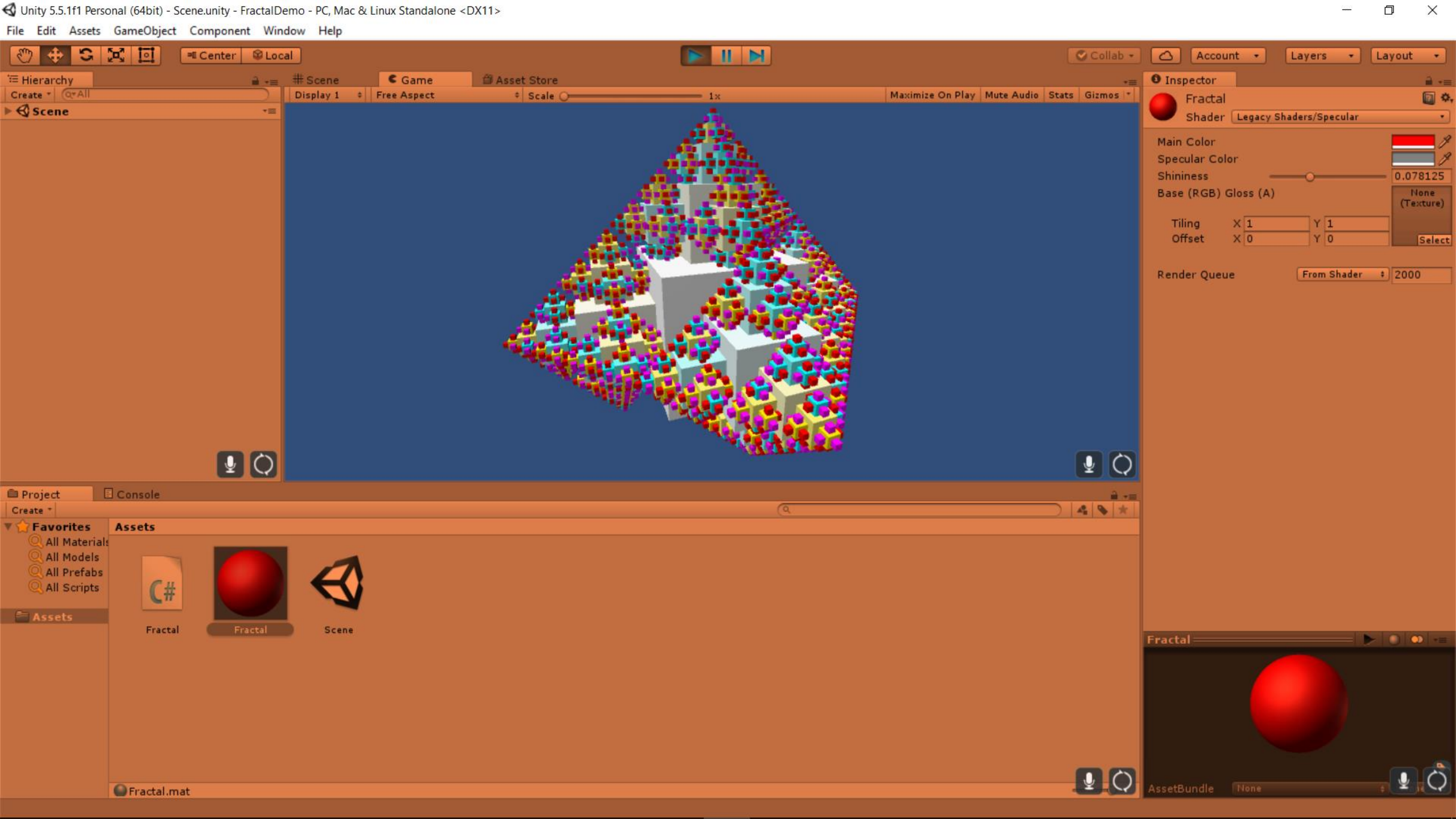
Fractal.cs

Fractal ▸ Initialize (Fractal parent, int childIndex)

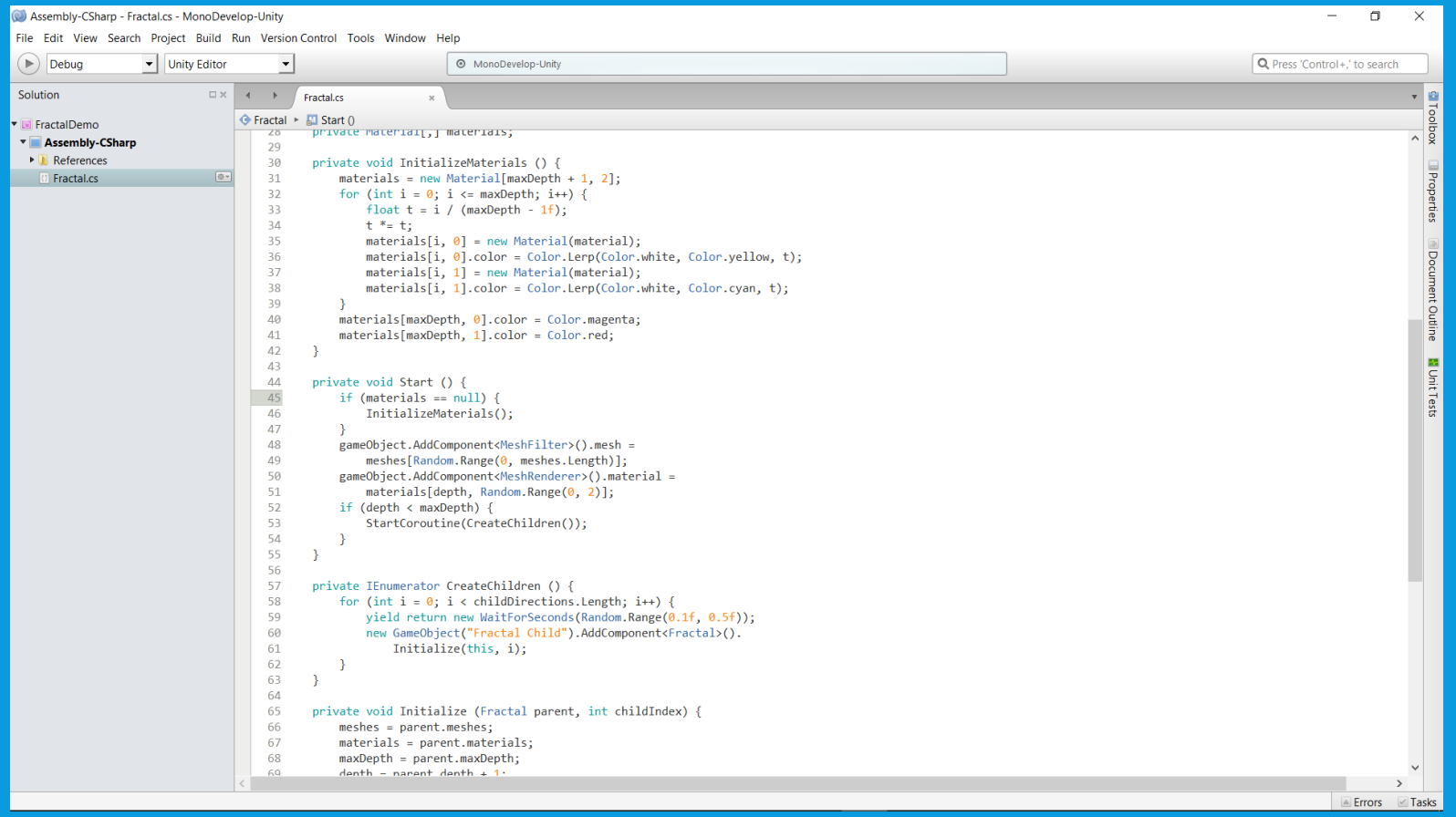
```
40      }
41
42      private void InitializeMaterials () {
43          materials = new Material[maxDepth + 1, 2];
44          for (int i = 0; i <= maxDepth; i++) {
45              float t = i / (maxDepth - 1f);
46              t *= t;
47              materials[i, 0] = new Material(material);
48              materials[i, 0].color = Color.Lerp(Color.white, Color.yellow, t);
49              materials[i, 1] = new Material(material);
50              materials[i, 1].color = Color.Lerp(Color.white, Color.cyan, t);
51          }
52          materials[maxDepth, 0].color = Color.magenta;
53          materials[maxDepth, 1].color = Color.red;
54      }
55
56      private IEnumerator CreateChildren () {
57          for (int i = 0; i < childDirections.Length; i++) {
58              yield return new WaitForSeconds(Random.Range(0.1f, 0.5f));
59              new GameObject("Fractal Child").AddComponent<Fractal>().
60                  Initialize(this, i);
61          }
62      }
63
64      private void Initialize (Fractal parent, int childIndex) {
65          mesh = parent.mesh;
66          materials = parent.materials;
67          maxDepth = parent.maxDepth;
68          depth = parent.depth + 1;
69          childScale = parent.childScale;
70          transform.parent = parent.transform;
71          transform.localScale = Vector3.one * childScale;
72          transform.localPosition =
73              childDirections[childIndex] * (0.5f + 0.5f * childScale);
74          transform.localRotation = childOrientations[childIndex];
75      }
76  }
```

Toolbox Properties Document Outline Unit Tests

Errors Tasks


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal
Shader Legacy Shaders/Specular
Main Color
Specular Color
Shininess 0.078125
Base (RGB) Gloss (A)
None (Texture)
Tiling X 1 Y 1
Offset X 0 Y 0
Select
Render Queue From Shader 2000
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal.mat
Fractal
AssetBundle None

FORSKELLIGE MESHES FOR MERE VARIATION




```
28      private Material[,] materials;
29
30      private void InitializeMaterials () {
31          materials = new Material[maxDepth + 1, 2];
32          for (int i = 0; i <= maxDepth; i++) {
33              float t = i / (maxDepth - 1f);
34              t *= t;
35              materials[i, 0] = new Material(material);
36              materials[i, 0].color = Color.Lerp(Color.white, Color.yellow, t);
37              materials[i, 1] = new Material(material);
38              materials[i, 1].color = Color.Lerp(Color.white, Color.cyan, t);
39          }
40          materials[maxDepth, 0].color = Color.magenta;
41          materials[maxDepth, 1].color = Color.red;
42      }
43
44      private void Start () {
45          if (materials == null) {
46              InitializeMaterials();
47          }
48          gameObject.AddComponent<MeshFilter>().mesh =
49              meshes[Random.Range(0, meshes.Length)];
50          gameObject.AddComponent<MeshRenderer>().material =
51              materials[depth, Random.Range(0, 2)];
52          if (depth < maxDepth) {
53              StartCoroutine(CreateChildren());
54          }
55      }
56
57      private IEnumerator CreateChildren () {
58          for (int i = 0; i < childDirections.Length; i++) {
59              yield return new WaitForSeconds(Random.Range(0.1f, 0.5f));
60              new GameObject("Fractal Child").AddComponent<Fractal>().
61                  Initialize(this, i);
62          }
63      }
64
65      private void Initialize (Fractal parent, int childIndex) {
66          meshes = parent.meshes;
67          materials = parent.materials;
68          maxDepth = parent.maxDepth;
69          depth = parent.depth + 1;
```

Føj ændringerne på linje 22, linje 49 og linje 66

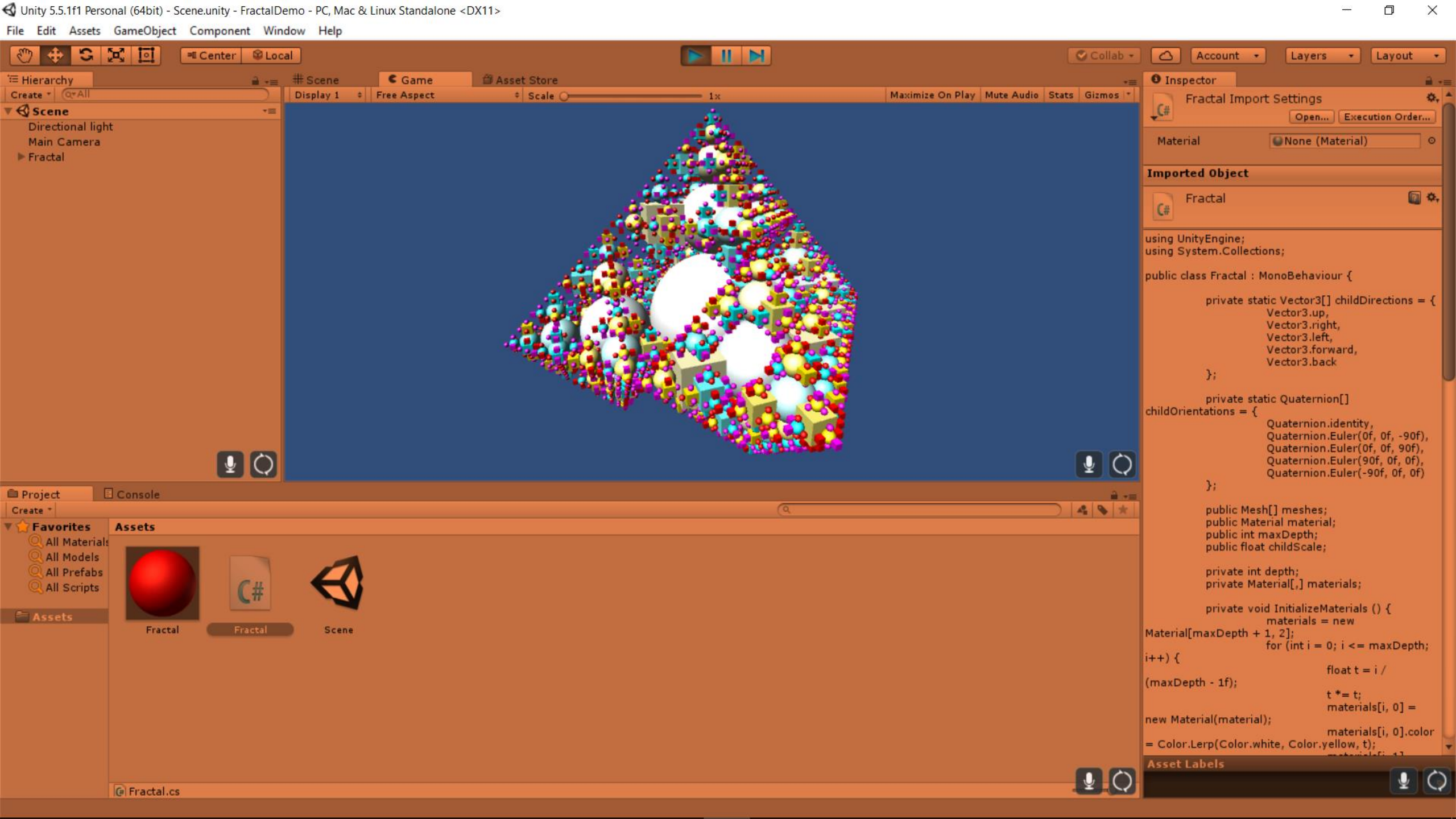
Fractal.cs

Fractal ▸ Start ()

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Fractal : MonoBehaviour {
5
6     private static Vector3[] childDirections = {
7         Vector3.up,
8         Vector3.right,
9         Vector3.left,
10        Vector3.forward,
11        Vector3.back
12    };
13
14    private static Quaternion[] childOrientations = {
15        Quaternion.identity,
16        Quaternion.Euler(0f, 0f, -90f),
17        Quaternion.Euler(0f, 0f, 90f),
18        Quaternion.Euler(90f, 0f, 0f),
19        Quaternion.Euler(-90f, 0f, 0f)
20    };
21
22    public Mesh[] meshes;
23    public Material material;
24    public int maxDepth;
25    public float childScale;
```

```
28      private Material[,] materials;
29
30      private void InitializeMaterials () {
31          materials = new Material[maxDepth + 1, 2];
32          for (int i = 0; i <= maxDepth; i++) {
33              float t = i / (maxDepth - 1f);
34              t *= t;
35              materials[i, 0] = new Material(material);
36              materials[i, 0].color = Color.Lerp(Color.white, Color.yellow, t);
37              materials[i, 1] = new Material(material);
38              materials[i, 1].color = Color.Lerp(Color.white, Color.cyan, t);
39          }
40          materials[maxDepth, 0].color = Color.magenta;
41          materials[maxDepth, 1].color = Color.red;
42      }
43
44      private void Start () {
45          if (materials == null) {
46              InitializeMaterials();
47          }
48          gameObject.AddComponent<MeshFilter>().mesh =
49              meshes[Random.Range(0, meshes.Length)];
50          gameObject.AddComponent<MeshRenderer>().material =
51              materials[depth, Random.Range(0, 2)];
52          if (depth < maxDepth) {
53              StartCoroutine(CreateChildren());
54          }
```

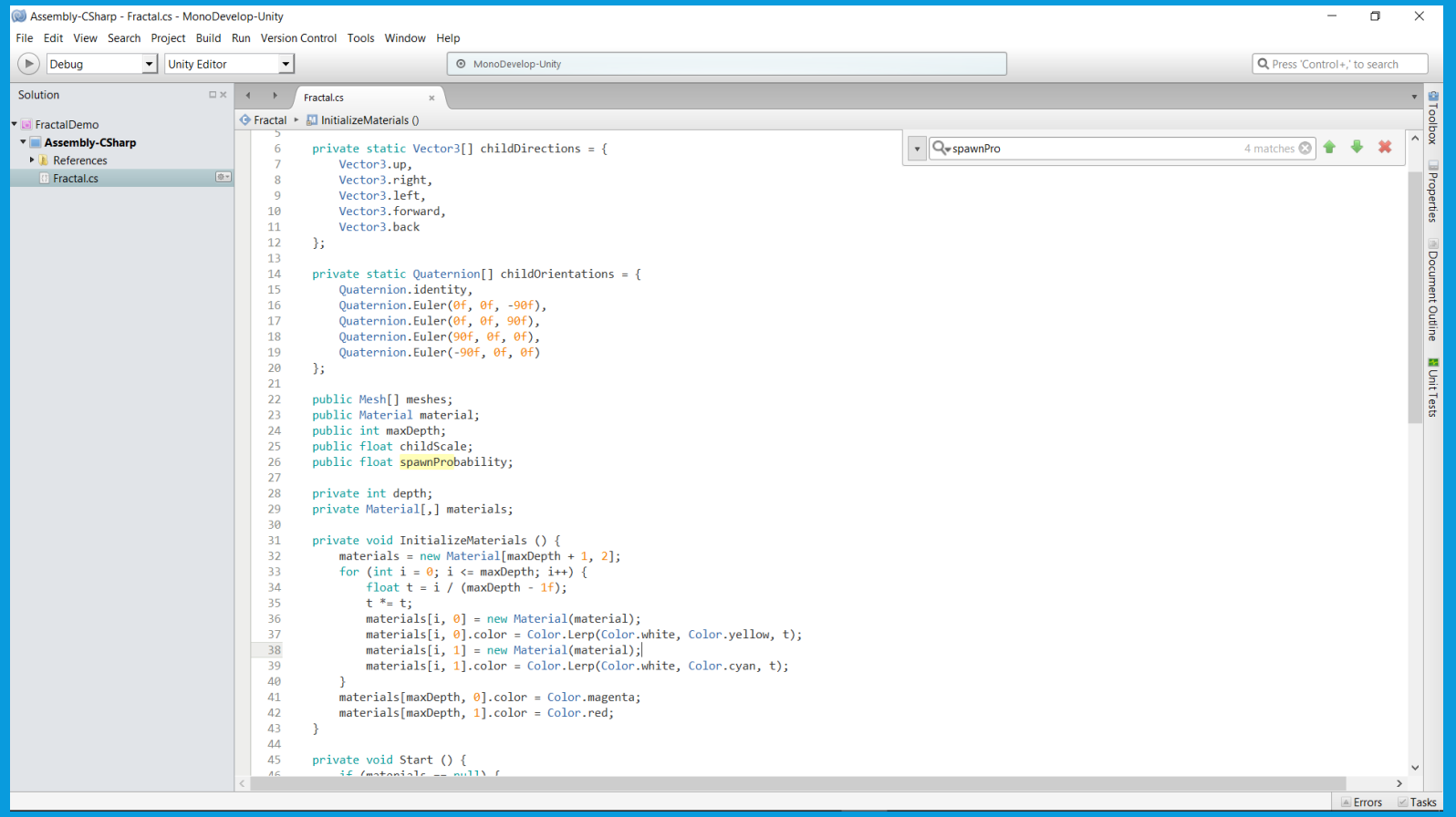
```
56
57      private IEnumerator CreateChildren () {
58          for (int i = 0; i < childDirections.Length; i++) {
59              yield return new WaitForSeconds(Random.Range(0.1f, 0.5f));
60              new GameObject("Fractal Child").AddComponent<Fractal>().
61                  Initialize(this, i);
62          }
63      }
64
65      private void Initialize (Fractal parent, int childIndex) {
66          meshes = parent.meshes;
67          materials = parent.materials;
68          maxDepth = parent.maxDepth;
69          depth = parent.depth + 1;
70          childScale = parent.childScale;
71          transform.parent = parent.transform;
72          transform.localScale = Vector3.one * childScale;
73          transform.localPosition =
74              childDirections[childIndex] * (0.5f + 0.5f * childScale);
75          transform.localRotation = childOrientations[childIndex];
76      }
77  }
```


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Import Settings
Open... Execution Order...
Material None (Material)
Imported Object
Fractal
using UnityEngine;
using System.Collections;
public class Fractal : MonoBehaviour {
private static Vector3[] childDirections = {
Vector3.up,
Vector3.right,
Vector3.left,
Vector3.forward,
Vector3.back
};
private static Quaternion[]
childOrientations = {
Quaternion.identity,
Quaternion.Euler(0f, 0f, -90f),
Quaternion.Euler(0f, 0f, 90f),
Quaternion.Euler(90f, 0f, 0f),
Quaternion.Euler(-90f, 0f, 0f)
};
public Mesh[] meshes;
public Material material;
public int maxDepth;
public float childScale;
private int depth;
private Material[,] materials;
private void InitializeMaterials () {
materials = new
Material[maxDepth + 1, 2];
for (int i = 0; i <= maxDepth;
i++) {
float t = i /
(maxDepth - 1f);
t *= t;
materials[i, 0] =
new Material(material);
materials[i, 0].color
= Color.Lerp(Color.white, Color.yellow, t);
Asset Labels
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal.cs

MERE IRREGULÆR OG ORGANISK

- Selvom vores fraktal virker nu ser den ikke specielt organisk ud, da alle arme er lige lange. Lad os gøre noget ved det!
- Til det formål introducerer vi en ny public spawnProbability variabel.
  - Vi sender den med rundt og lader den tilfældigt bestemme om der skal dannes et barn eller om det skal hoppes over.
  - En sandsynlighed på 0 betyder at der ikke vil dannes nogle børn overhovedet medens en på 1 betyder at alle børn vil dannes.
- Den statiske Random.value skaber en værdi mellem nul og et. Den sammenlignes med spawnProbability lader os bestemme om der skal dannes et nyt barn eller ej.

MERE IRREGULÆR OG ORGANISK



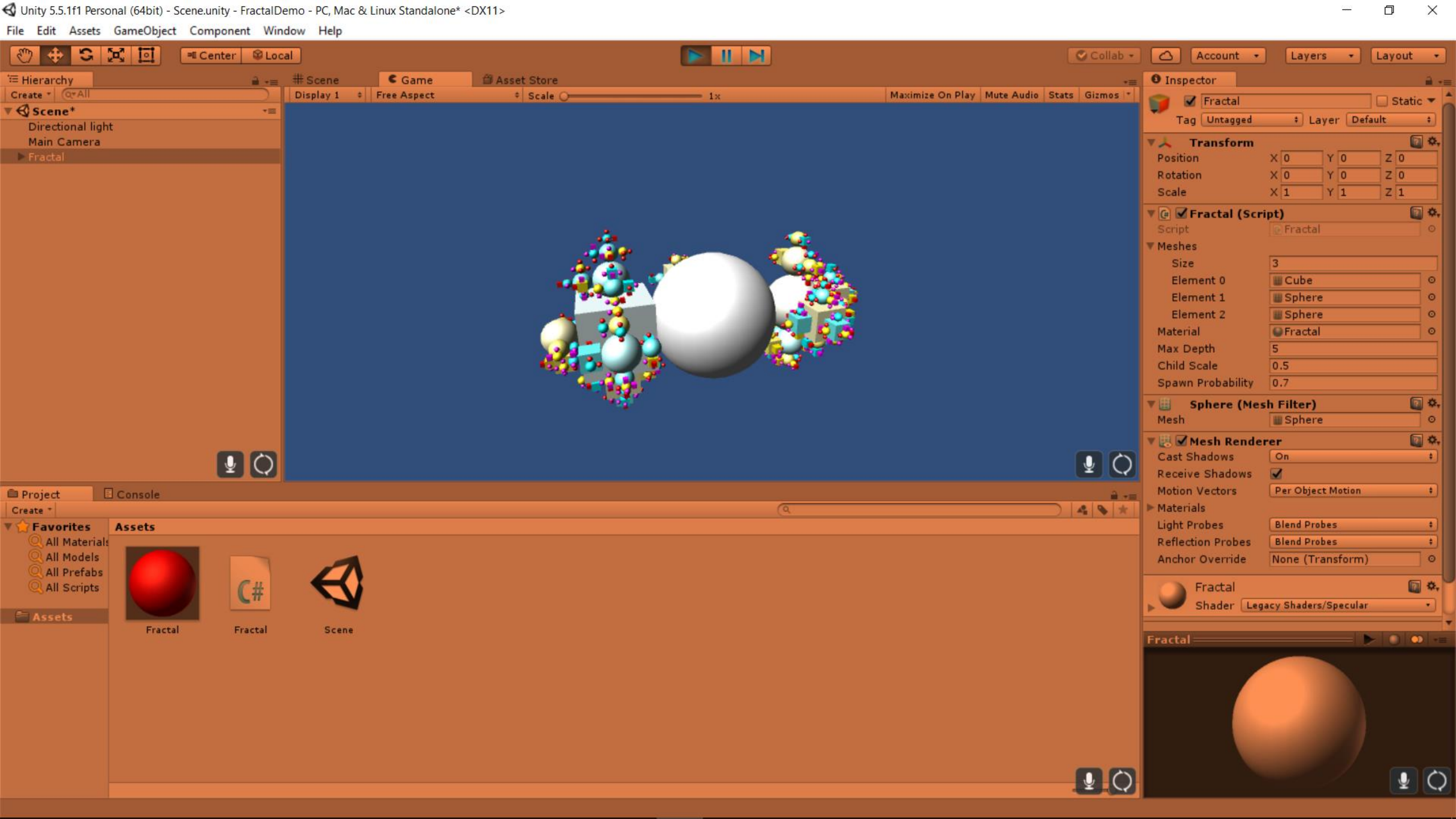

```
6   private static Vector3[] childDirections = {
7       Vector3.up,
8       Vector3.right,
9       Vector3.left,
10      Vector3.forward,
11      Vector3.back
12  };
13
14  private static Quaternion[] childOrientations = {
15      Quaternion.identity,
16      Quaternion.Euler(0f, 0f, -90f),
17      Quaternion.Euler(0f, 0f, 90f),
18      Quaternion.Euler(90f, 0f, 0f),
19      Quaternion.Euler(-90f, 0f, 0f)
20  };
21
22  public Mesh[] meshes;
23  public Material material;
24  public int maxDepth;
25  public float childScale;
26  public float spawnProbability;
27
28  private int depth;
29  private Material[,] materials;
30
31  private void InitializeMaterials () {
32      materials = new Material[maxDepth + 1, 2];
33      for (int i = 0; i <= maxDepth; i++) {
34          float t = i / (maxDepth - 1f);
35          t *= t;
36          materials[i, 0] = new Material(material);
37          materials[i, 0].color = Color.Lerp(Color.white, Color.yellow, t);
38          materials[i, 1] = new Material(material);
39          materials[i, 1].color = Color.Lerp(Color.white, Color.cyan, t);
40      }
41      materials[maxDepth, 0].color = Color.magenta;
42      materials[maxDepth, 1].color = Color.red;
43  }
44
45  private void Start () {
```

Føj ændringerne på linje 26, linje 60 og linje 72

```
22      public Mesh[] meshes;
23      public Material material;
24      public int maxDepth;
25      public float childScale;
26      public float spawnProbability;
27
```

```
58      private IEnumerator CreateChildren () {
59          for (int i = 0; i < childDirections.Length; i++) {
60              if (Random.value < spawnProbability) {
61                  yield return new WaitForSeconds(Random.Range(0.1f, 0.5f));
62                  new GameObject("Fractal Child").AddComponent<Fractal>().
63                      Initialize(this, i);
64              }
65          }
66      }
```

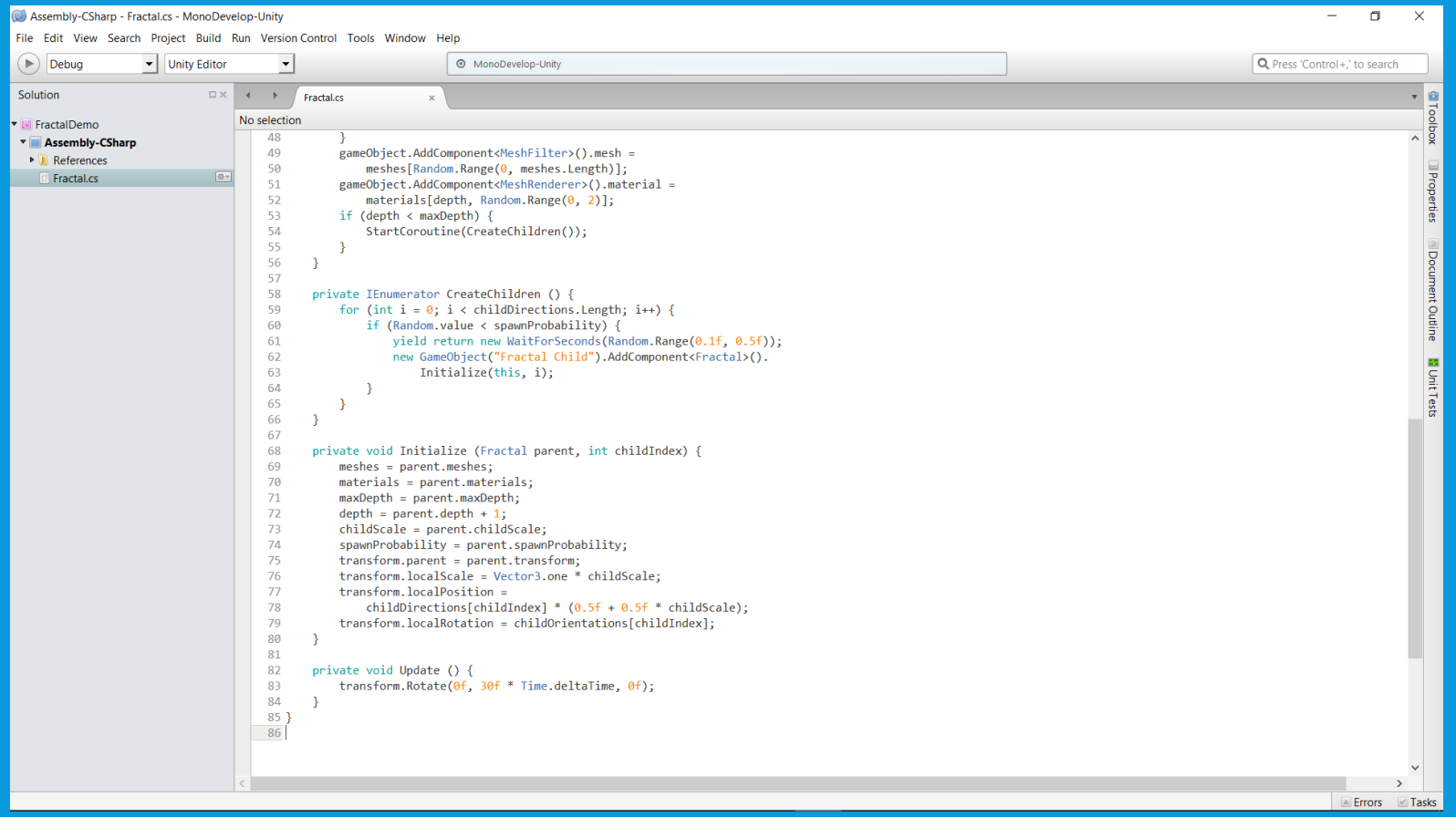
```
68      private void Initialize (Fractal parent, int childIndex) {
69          meshes = parent.meshes;
70          materials = parent.materials;
71          maxDepth = parent.maxDepth;
72          depth = parent.depth + 1;
73          childScale = parent.childScale;
74          spawnProbability = parent.spawnProbability;
75          transform.parent = parent.transform;
76          transform.localScale = Vector3.one * childScale;
77          transform.localPosition =
78              childDirections[childIndex] * (0.5f + 0.5f * childScale);
79          transform.localRotation = childOrientations[childIndex];
80      }
81  }
```


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene*
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Meshes
Size 3
Element 0 Cube
Element 1 Sphere
Element 2 Sphere
Material Fractal
Max Depth 5
Child Scale 0.5
Spawn Probability 0.7
Sphere (Mesh Filter)
Mesh Sphere
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal

ROTERING AF FRAKTALEN

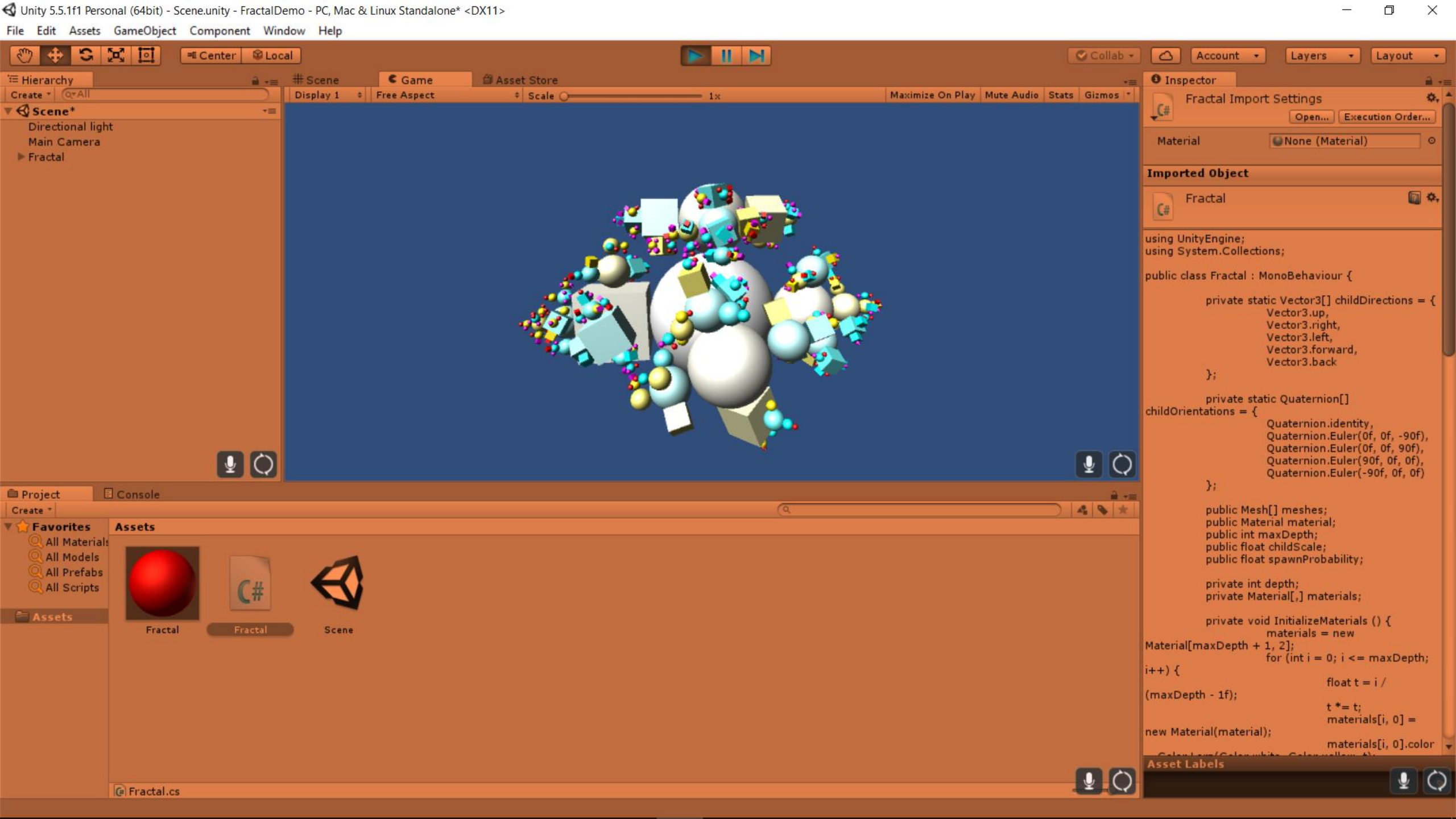
- Indtil videre er vores fraktal ubevægelig, lad os gøre den mere interessant ved at lade den bevæge sig.
- Vi gør dette ved at tilføje en simpel update metode der roterer y-aksen med en hastighed på 30 grader i sekundet.

ROTERING AF FRAKTALEN



Føj koden på linje 82-85 til

```
68      private void Initialize (Fractal parent, int childIndex) {
69          meshes = parent.meshes;
70          materials = parent.materials;
71          maxDepth = parent.maxDepth;
72          depth = parent.depth + 1;
73          childScale = parent.childScale;
74          spawnProbability = parent.spawnProbability;
75          transform.parent = parent.transform;
76          transform.localScale = Vector3.one * childScale;
77          transform.localPosition =
78              childDirections[childIndex] * (0.5f + 0.5f * childScale);
79          transform.localRotation = childOrientations[childIndex];
80      }
81
82      private void Update () {
83          transform.Rotate(0f, 30f * Time.deltaTime, 0f);
84      }
85  }
86
```

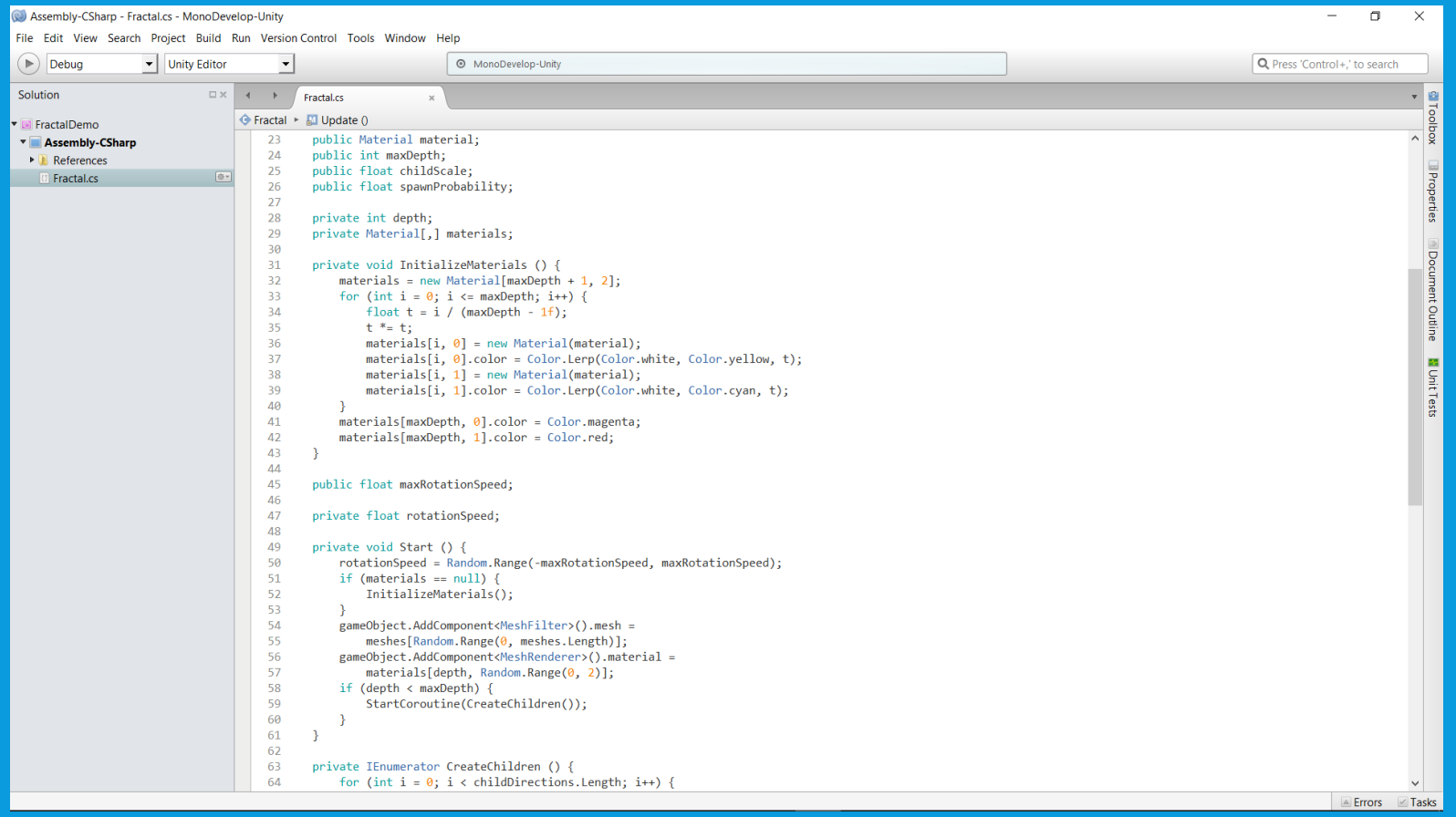

Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene*
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Import Settings
Open... Execution Order...
Material None (Material)
Imported Object
Fractal
using UnityEngine;
using System.Collections;
public class Fractal : MonoBehaviour {
private static Vector3[] childDirections = {
Vector3.up,
Vector3.right,
Vector3.left,
Vector3.forward,
Vector3.back
};
private static Quaternion[]
childOrientations = {
Quaternion.identity,
Quaternion.Euler(0f, 0f, -90f),
Quaternion.Euler(0f, 0f, 90f),
Quaternion.Euler(90f, 0f, 0f),
Quaternion.Euler(-90f, 0f, 0f)
};
public Mesh[] meshes;
public Material material;
public int maxDepth;
public float childScale;
public float spawnProbability;
private int depth;
private Material[,] materials;
private void InitializeMaterials () {
materials = new
Material[maxDepth + 1, 2];
for (int i = 0; i <= maxDepth;
i++) {
float t = i /
(maxDepth - 1f);
t *= t;
materials[i, 0] =
new Material(material);
materials[i, 0].color
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal.cs
Asset Labels

ROTERING AF FRAKTALEN

- Nu roterer fraktalen, men med et fast tempo hvor alt andet er tilfældigt genereret.
- Det kan vi ikke have, men vi skal også have en maksimum hastighed så man kan nå at se rotationerne.

- Bemærk, at vi er nødt til at initialisere vores rotationshastighed i Start - ikke i Initialize - fordi rod elementet også skal rotere.

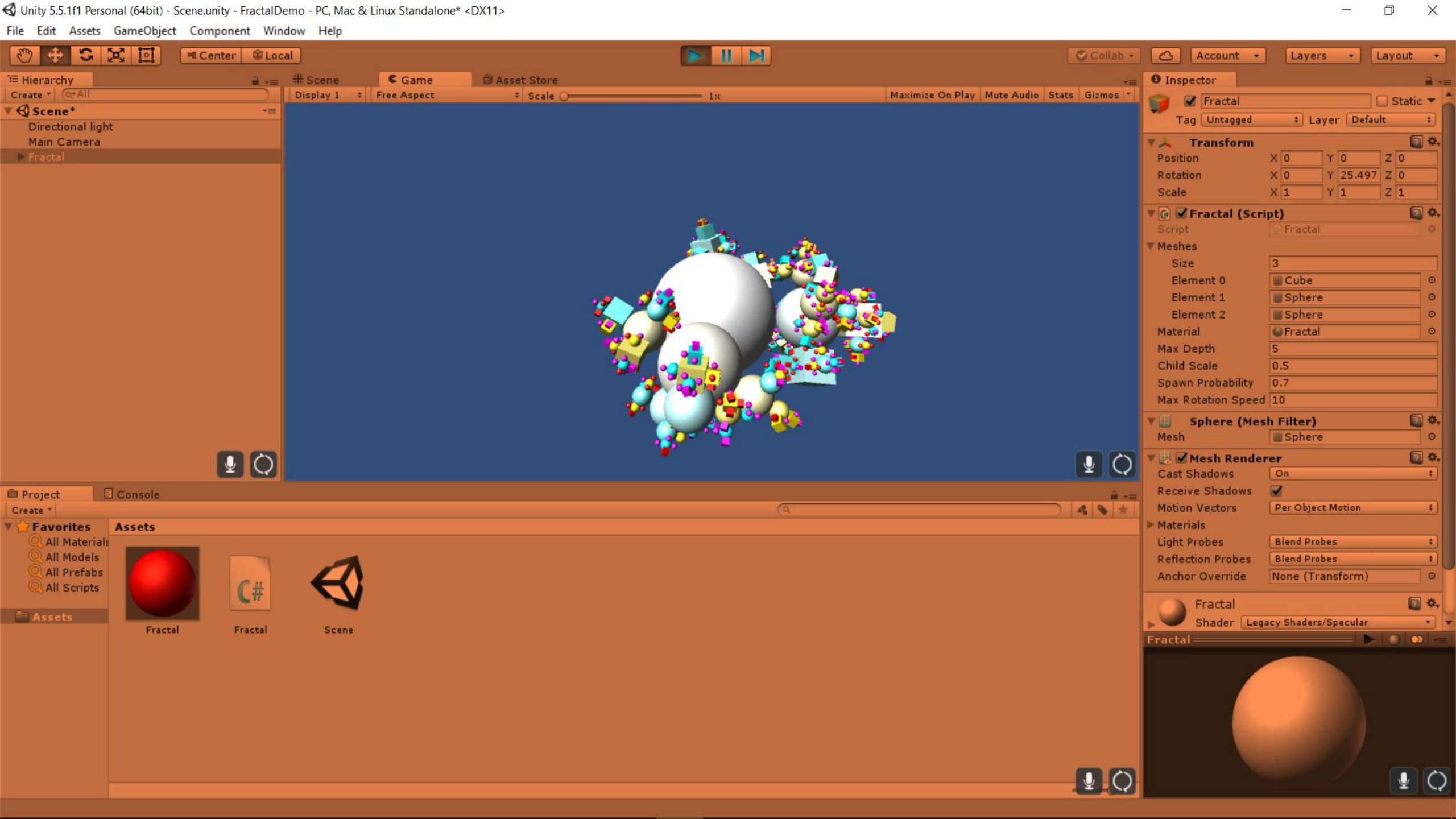
ROTERING AF FRAKTALEN



Føj koden linje på 45-47, linje 50 og linje 78 til og ret linje 89

```
41          materials[maxDepth, 0].color = Color.magenta;
42          materials[maxDepth, 1].color = Color.red;
43      }
44
45      public float maxRotationSpeed;
46
47      private float rotationSpeed;
48
49      private void Start () {
50          rotationSpeed = Random.Range(-maxRotationSpeed, maxRotationSpeed);
51          if (materials == null) {
52              InitializeMaterials();
53          }
54          gameObject.AddComponent<MeshFilter>().mesh =
55              meshes[Random.Range(0, meshes.Length)];
56          gameObject.AddComponent<MeshRenderer>().material =
57              materials[depth, Random.Range(0, 2)];
58          if (depth < maxDepth) {
59              StartCoroutine(CreateChildren());
60          }
61      }
```

```
73      private void Initialize (Fractal parent, int childIndex) {
74          meshes = parent.meshes;
75          materials = parent.materials;
76          maxDepth = parent.maxDepth;
77          depth = parent.depth + 1;
78          maxRotationSpeed = parent.maxRotationSpeed;
79          childScale = parent.childScale;
80          spawnProbability = parent.spawnProbability;
81          transform.parent = parent.transform;
82          transform.localScale = Vector3.one * childScale;
83          transform.localPosition =
84              childDirections[childIndex] * (0.5f + 0.5f * childScale);
85          transform.localRotation = childOrientations[childIndex];
86      }
87
88      private void Update () {
89          transform.Rotate(0f, rotationSpeed * Time.deltaTime, 0f);
90      }
91  }
```

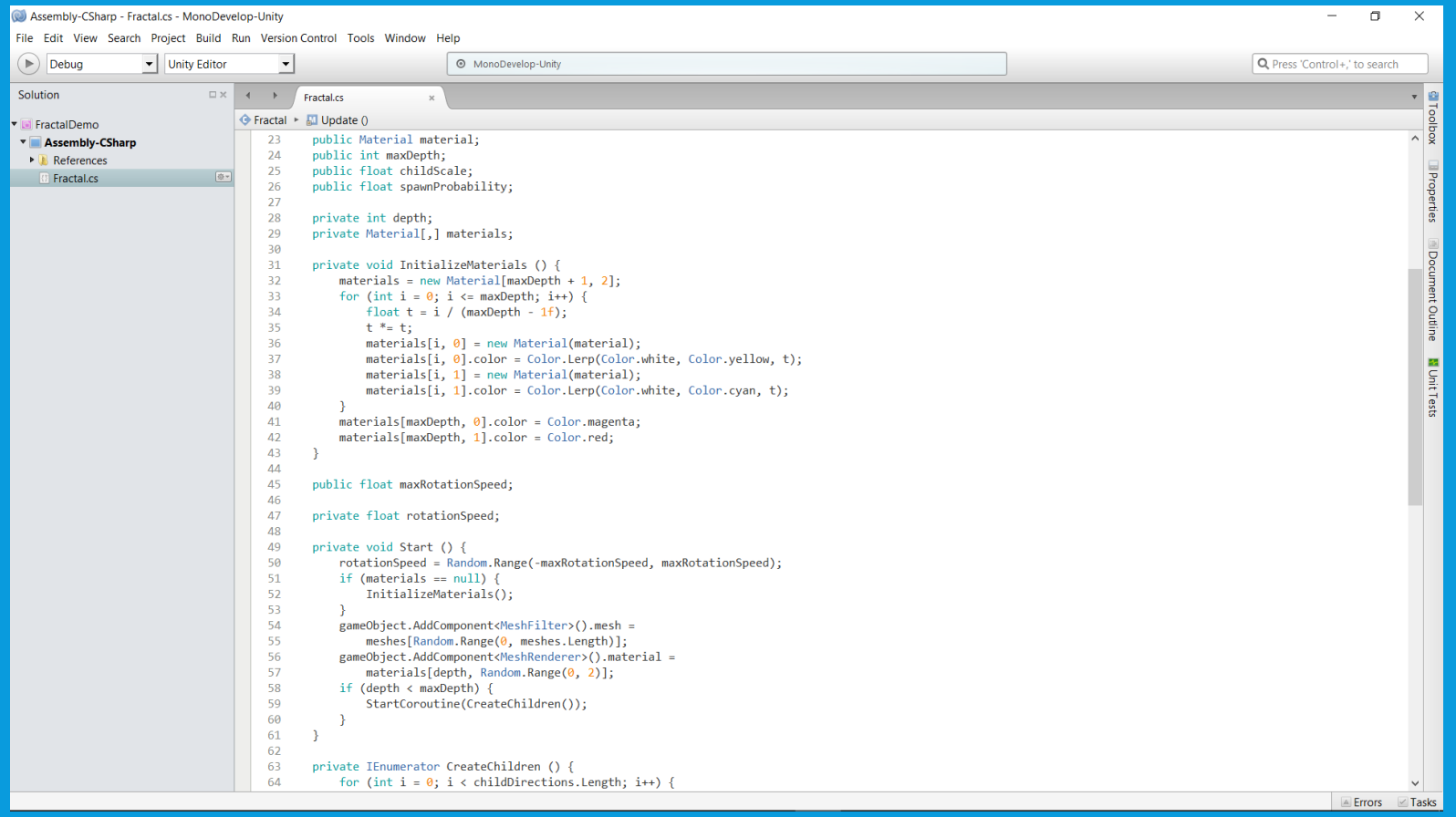

Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene*
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 25.497 Z 0
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Meshes
Size 3
Element 0 Cube
Element 1 Sphere
Element 2 Sphere
Material Fractal
Max Depth 5
Child Scale 0.5
Spawn Probability 0.7
Max Rotation Speed 10
Sphere (Mesh Filter)
Mesh Sphere
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Fractal
Shader Legacy Shaders/Specular
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal

LAD OS SLUTTE AF MED EN SMULE MERE KAOS

- Vi kan gøre fraktalen endnu mere skæv på mange måder, men lad os slutte af med at skubbe dens elementer en smule med en twist rotation.

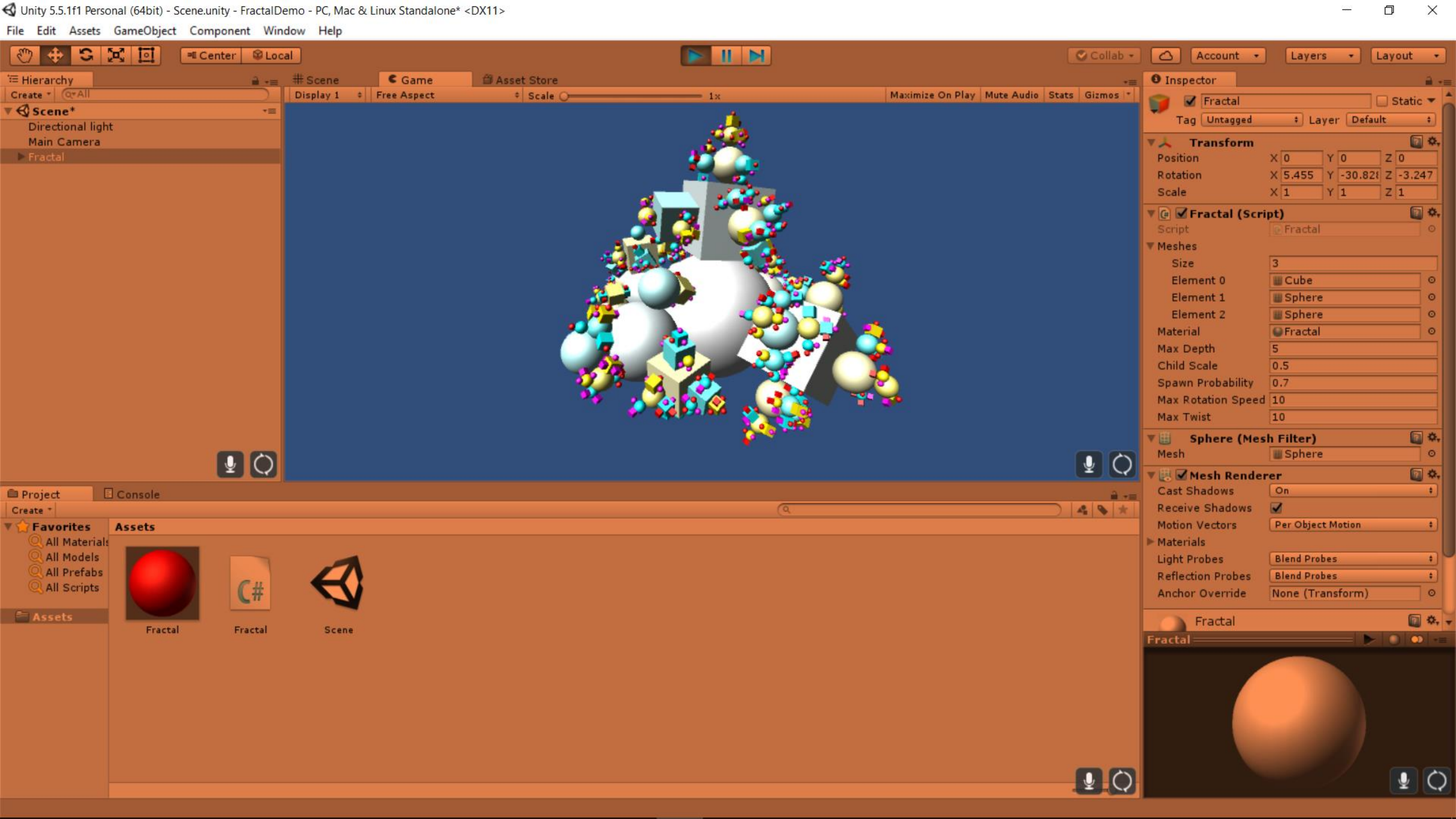
- I kan gøre meget, meget mere ved at pille ved de forskellige parametre! Prøv jer frem!

ROTERING AF FRAKTALEN



Føj koden linje på 48, linje 53 og linje 82 til

```
46      public float maxRotationSpeed;
47      private float rotationSpeed;
48      public float maxTwist;

51      private void Start () {
52          rotationSpeed = Random.Range(-maxRotationSpeed, maxRotationSpeed);
53          transform.Rotate(Random.Range(-maxTwist, maxTwist), 0f, 0f);
54          if (materials == null) {
55              InitializeMaterials();
56          }

76      private void Initialize (Fractal parent, int childIndex) {
77          meshes = parent.meshes;
78          materials = parent.materials;
79          maxDepth = parent.maxDepth;
80          depth = parent.depth + 1;
81          maxRotationSpeed = parent.maxRotationSpeed;
82          maxTwist = parent.maxTwist;
83          childScale = parent.childScale;
84          spawnProbability = parent.spawnProbability;
```


Unity 5.5.1f1 Personal (64bit) - Scene.unity - FractalDemo - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create All
Scene*
Directional light
Main Camera
Fractal
Scene Game Asset Store
Display 1 Free Aspect Scale 1x Maximize On Play Mute Audio Stats Gizmos
Inspector
Fractal
Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 5.455 Y -30.82 Z -3.247
Scale X 1 Y 1 Z 1
Fractal (Script)
Script Fractal
Meshes
Size 3
Element 0 Cube
Element 1 Sphere
Element 2 Sphere
Material Fractal
Max Depth 5
Child Scale 0.5
Spawn Probability 0.7
Max Rotation Speed 10
Max Twist 10
Sphere (Mesh Filter)
Mesh Sphere
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors Per Object Motion
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Assets
Fractal
Fractal
Scene
Fractal
Fractal

TAK FORDI I FULGTE MED!

Lad os lave noget andet sammen en anden gang