Lektion 6

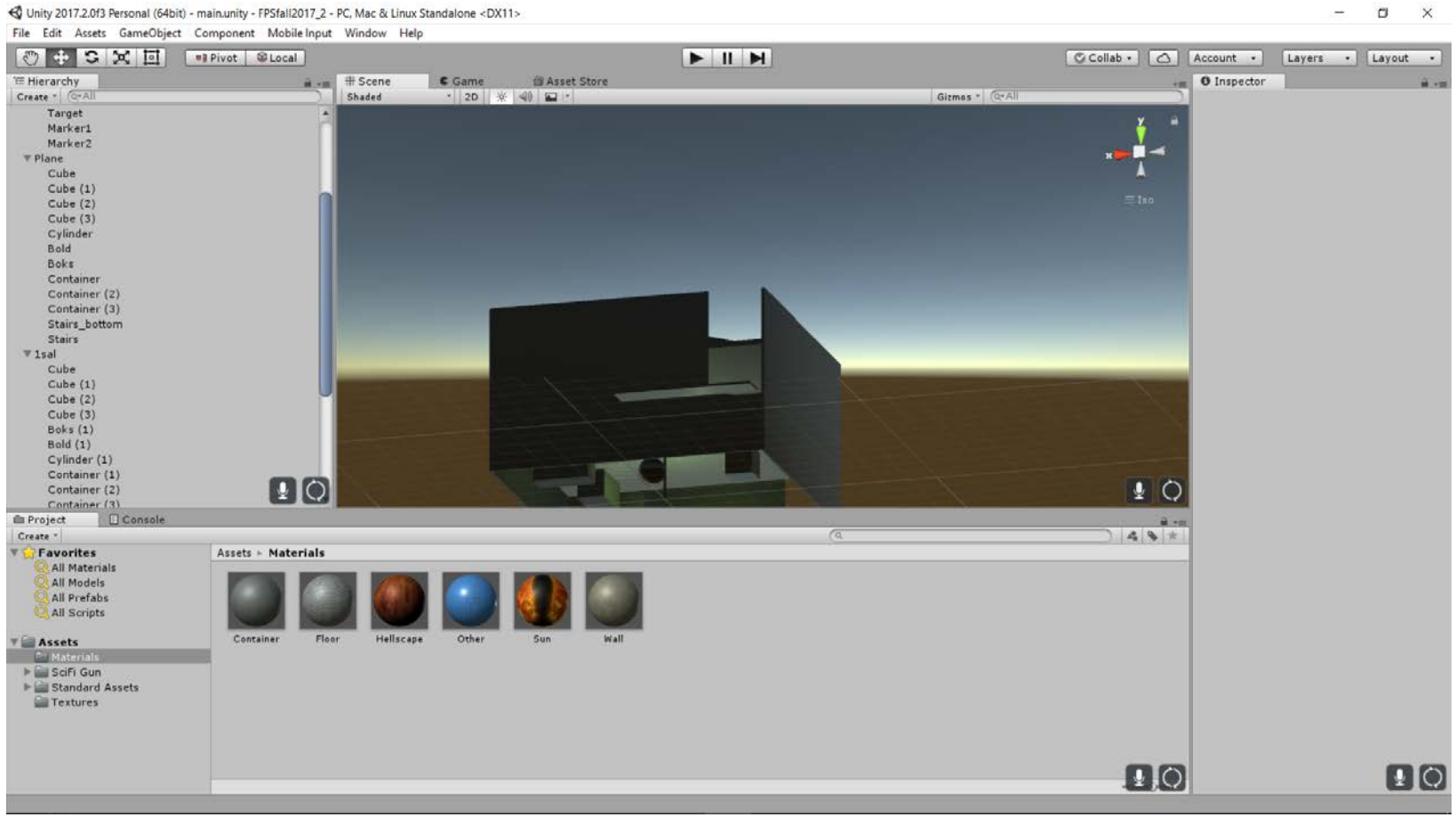
Grundlæggende programmering i VR

Plan for i dag

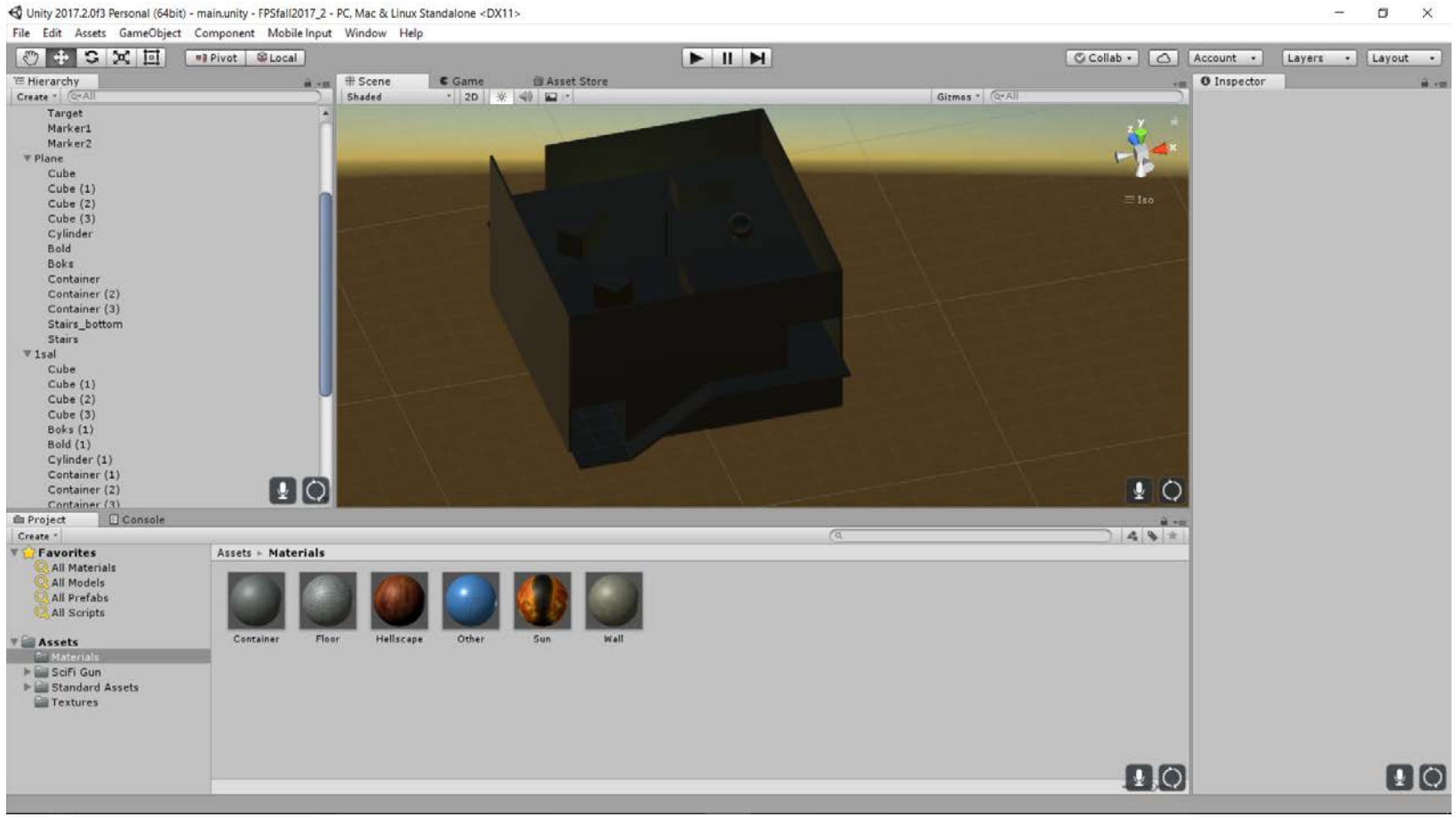
- Simpelt FPS
  - Triggerzones og animationer
- C# og objekt orienteret programmering
  - Interfaces
  - Generics
- Google Cardboard
  - Video om VRs fremtid
  - App til Google Cardboard
- Introduktion til eksamensopgave

Simpelt FPS

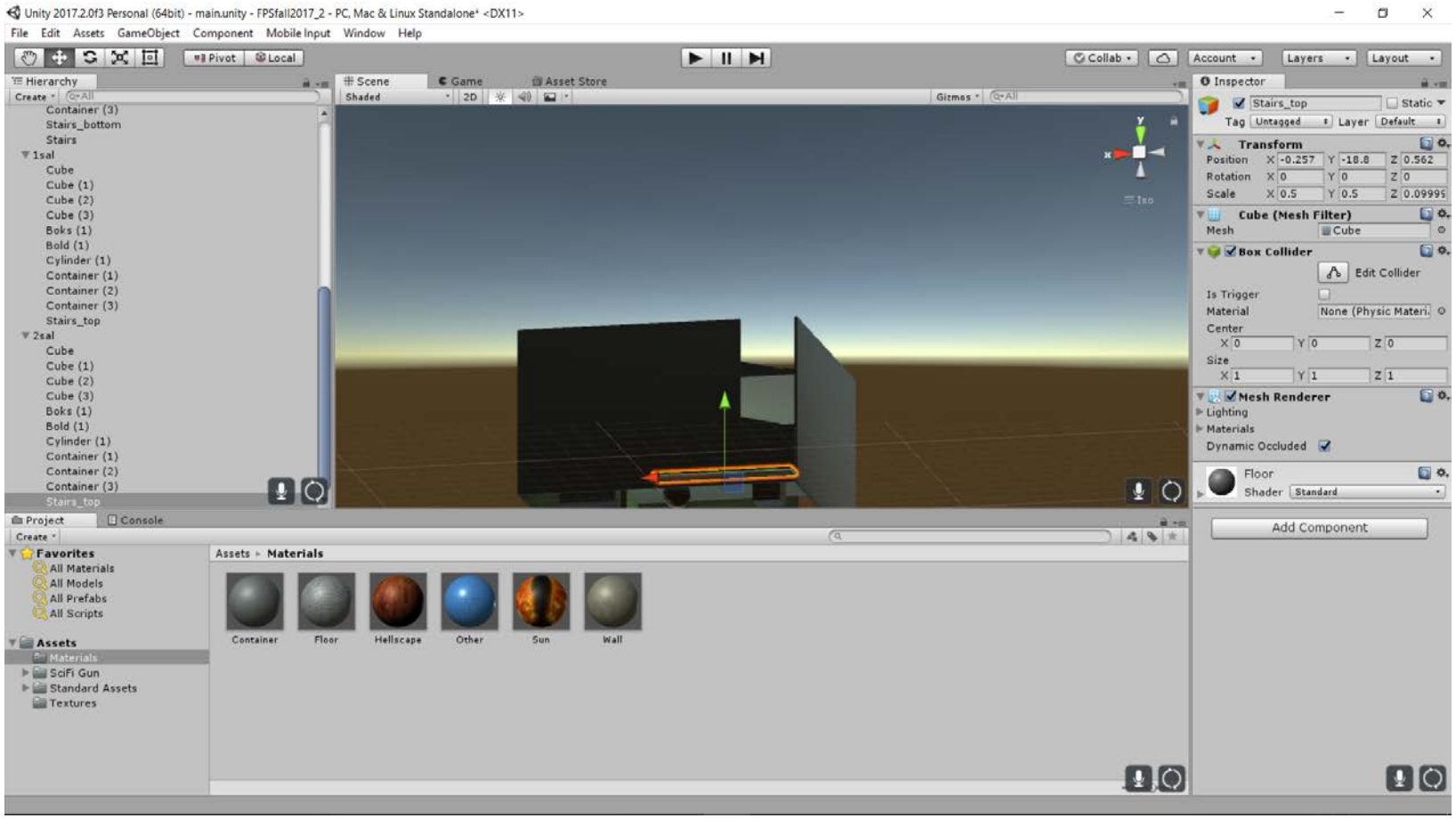
Føj til så jeres bygning har tre etager



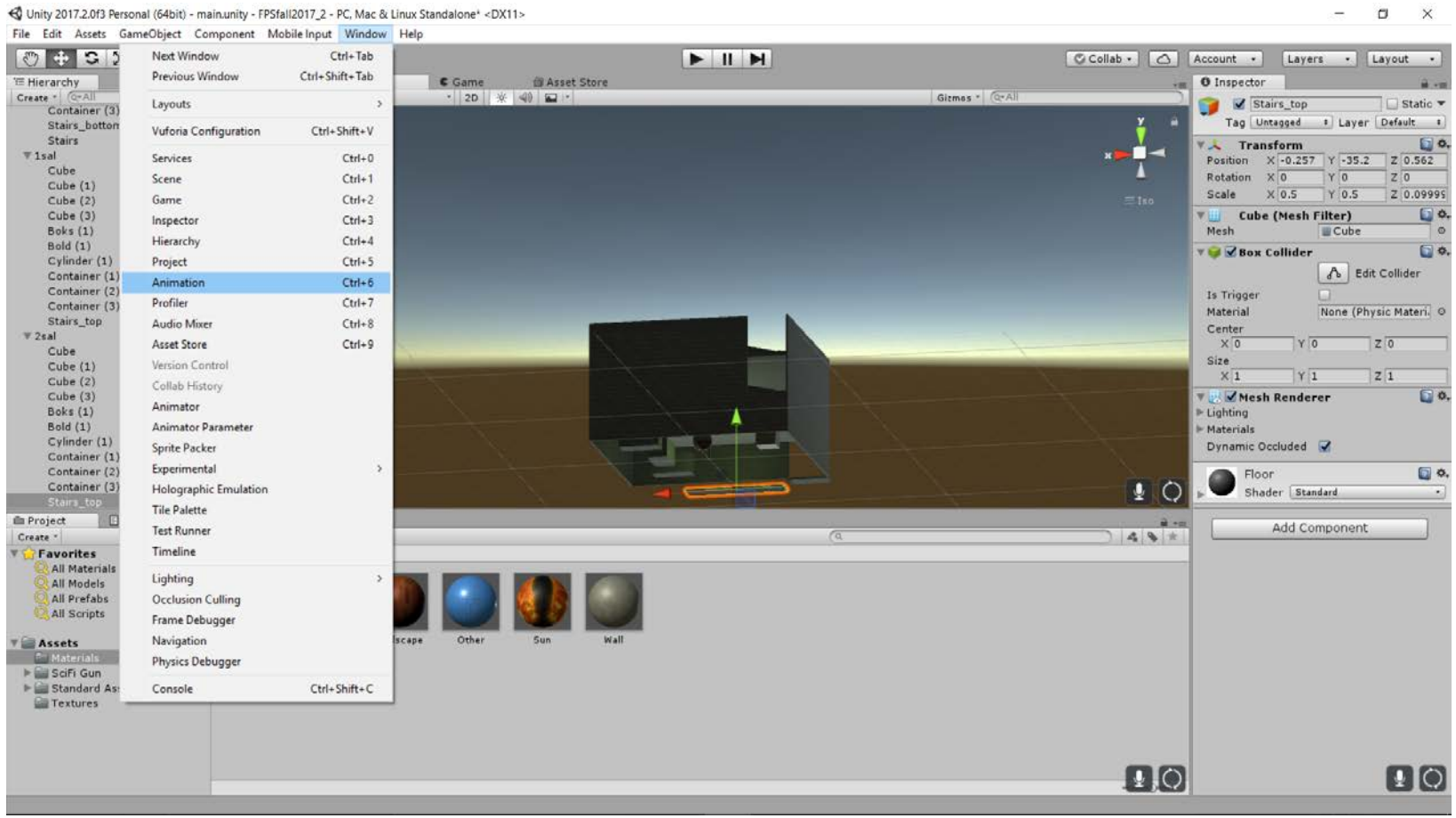
Lav f.eks. en rampe (drejet kube) så man kan komme op på 1ste salen



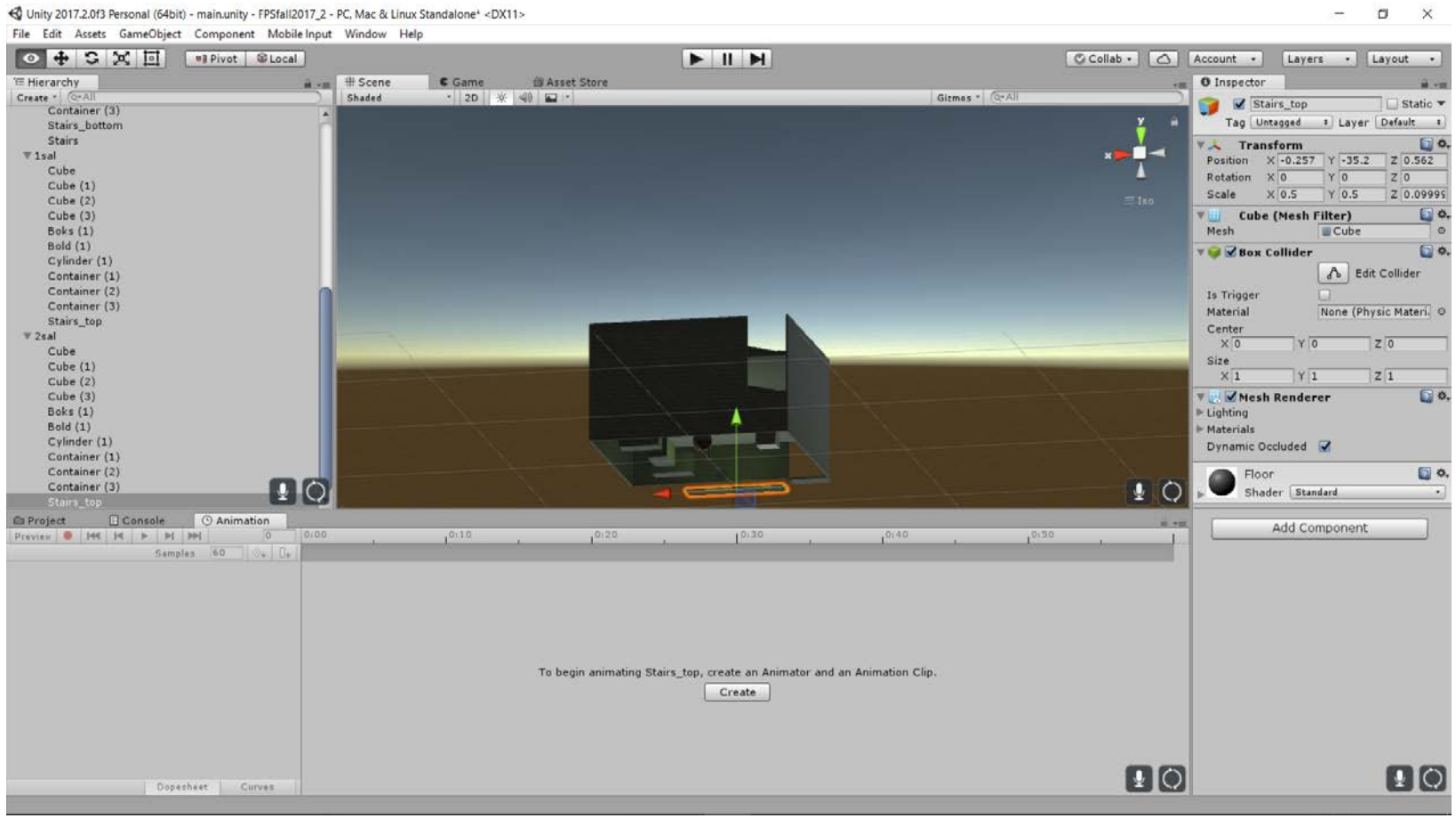
Der skal *ikke* være nogen direkte måde at komme fra 1 til 2 sal på!



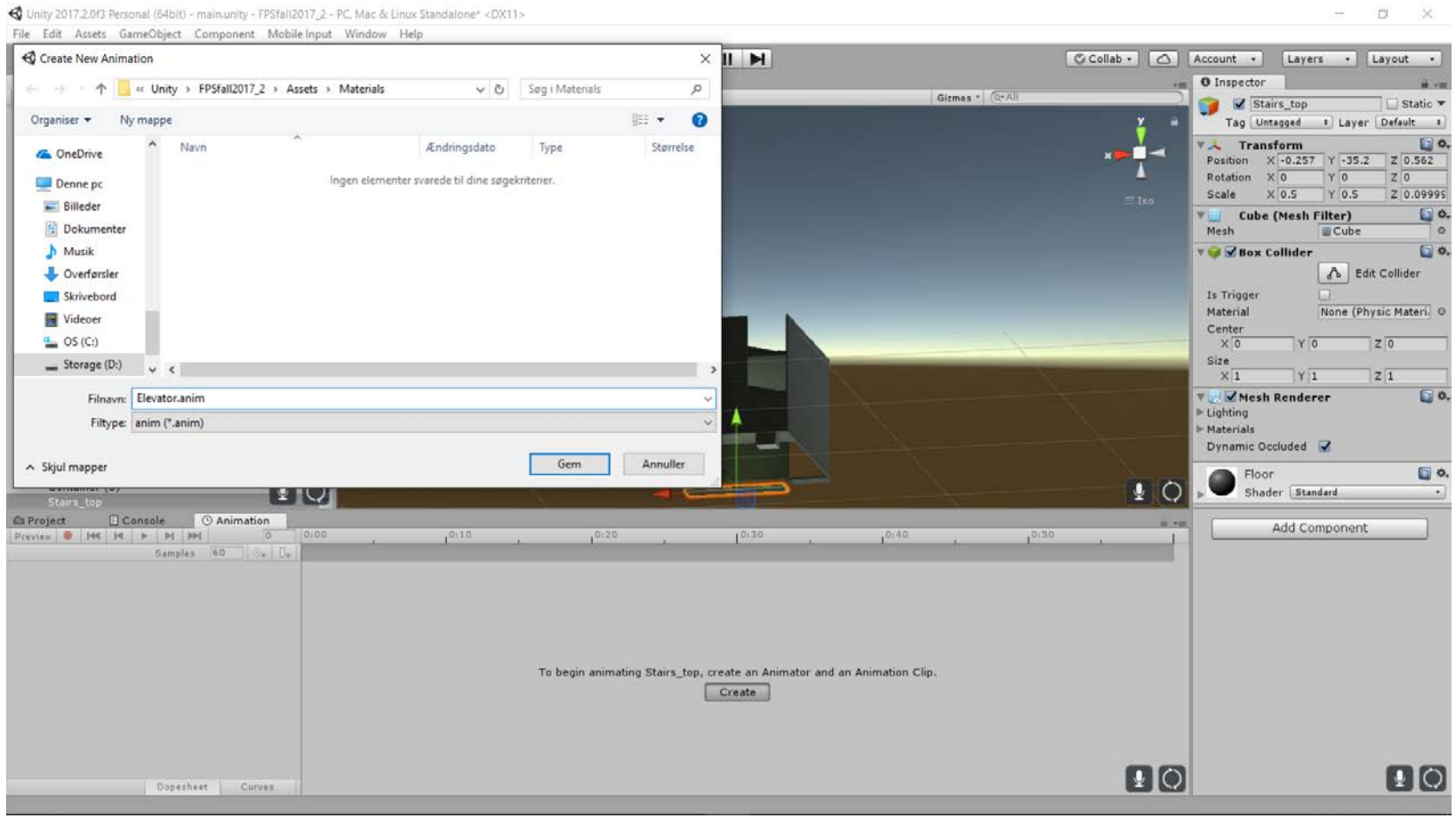
Åbn Animation vinduet



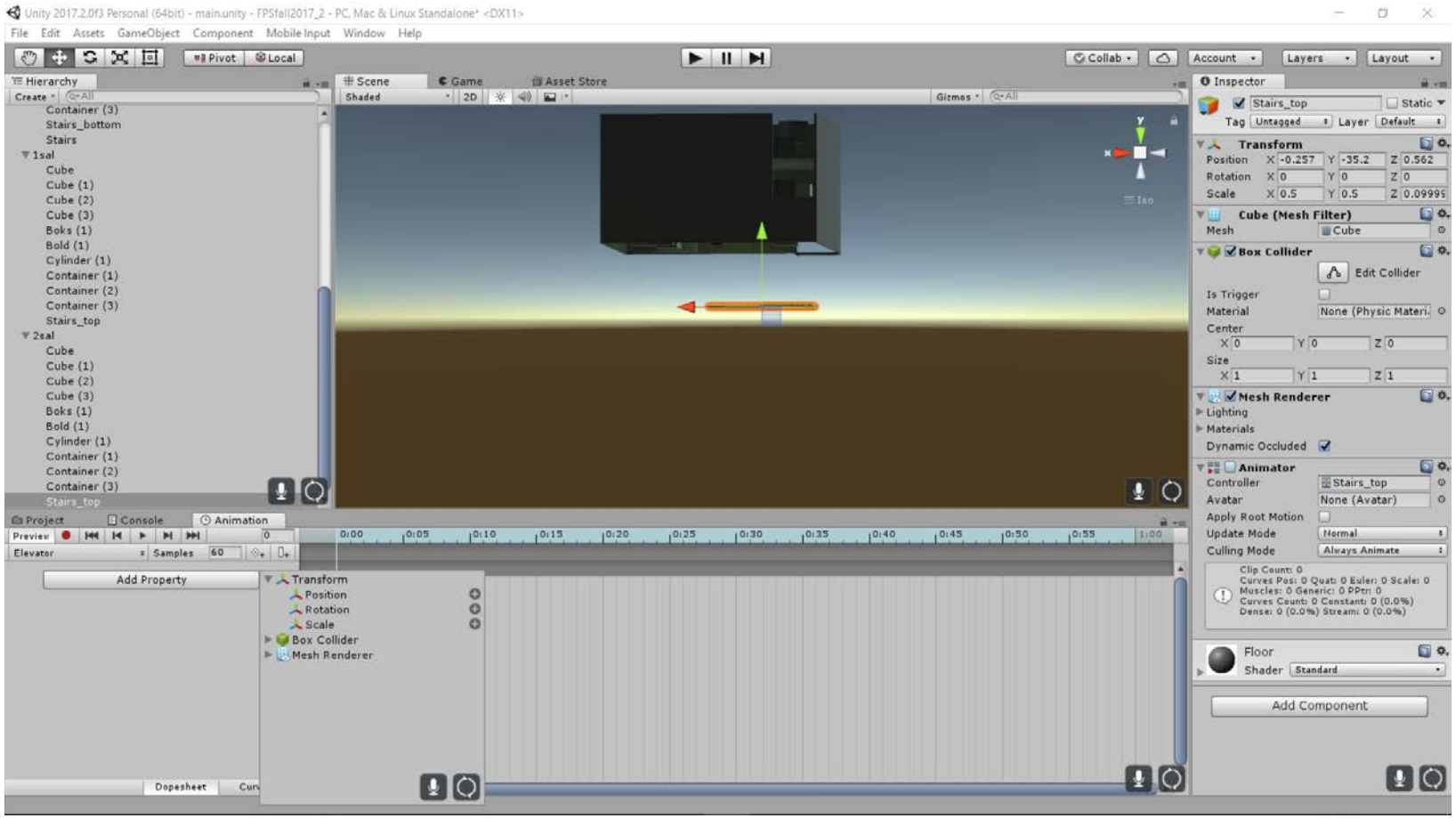
Dok det f.eks. i bunden



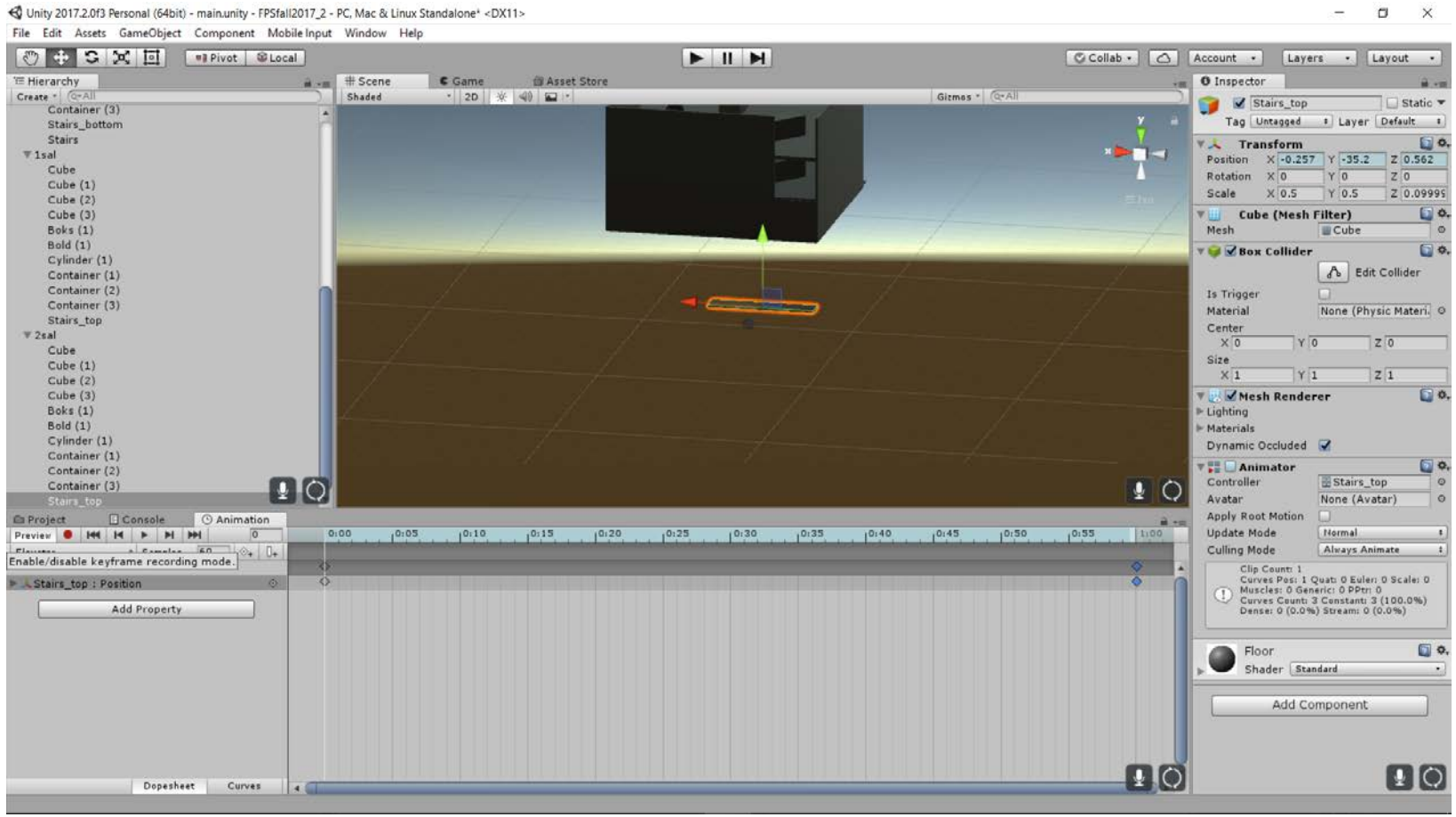
Lav en ny animation



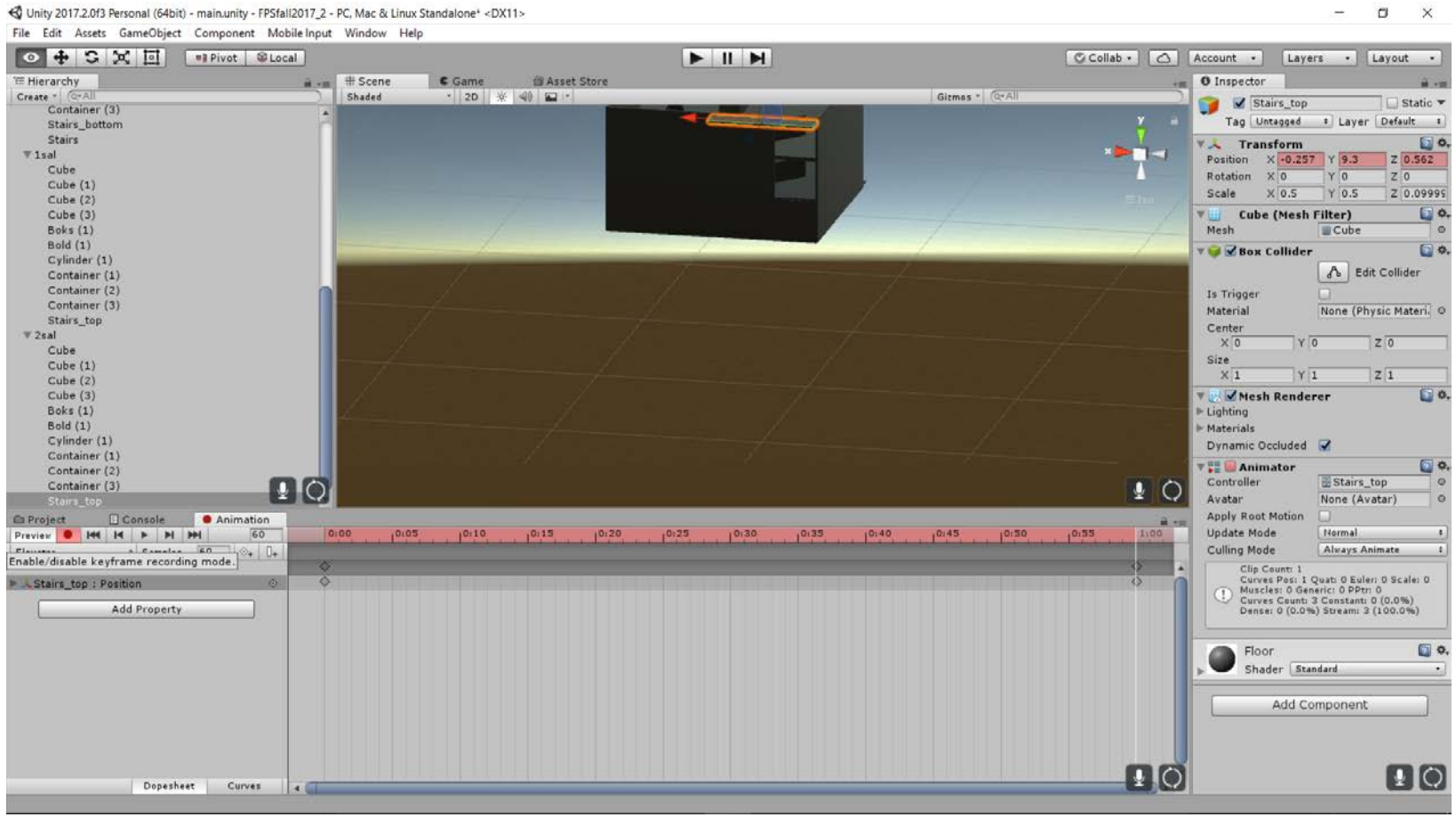
Sørg for at vores elevator platform er valgt og vælg Add Property > Transform da vi vil bevæge den



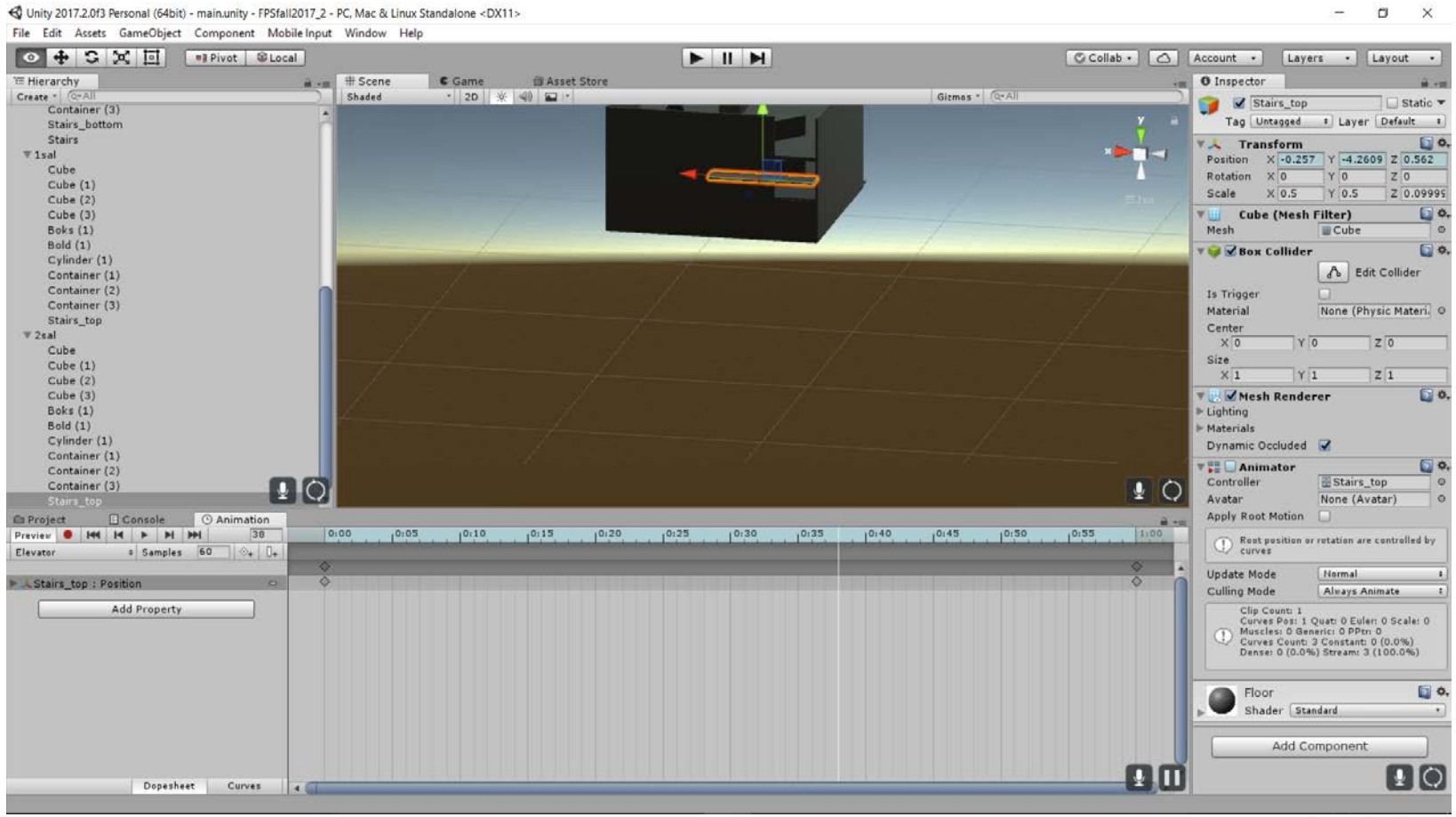
Sørg for at vi er på starten af tidslinjen og tryk optag knappen…



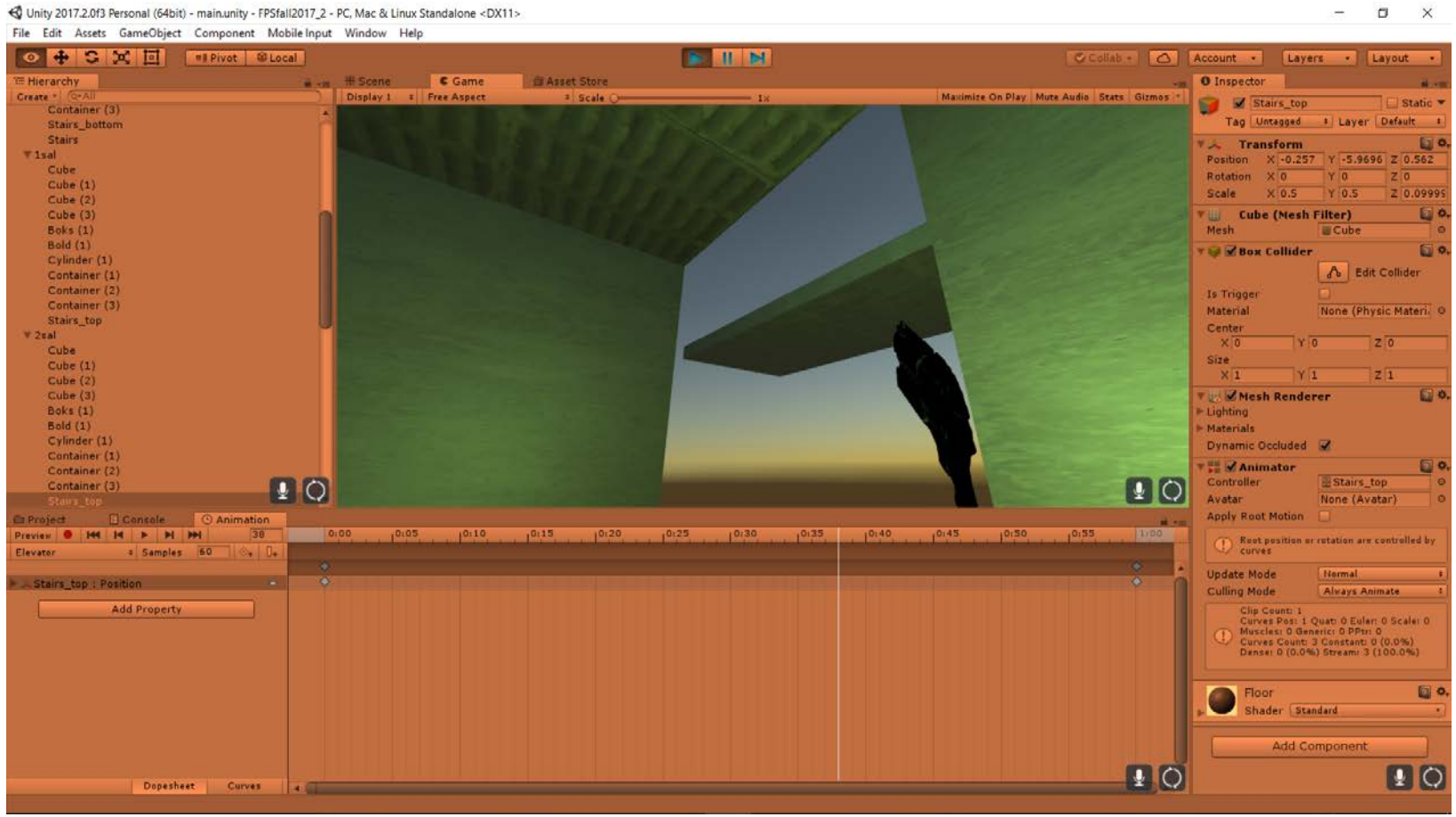
… ryk tidslinjen til 1:00 og elevatoren til hvor den skal ende og stop optagelse



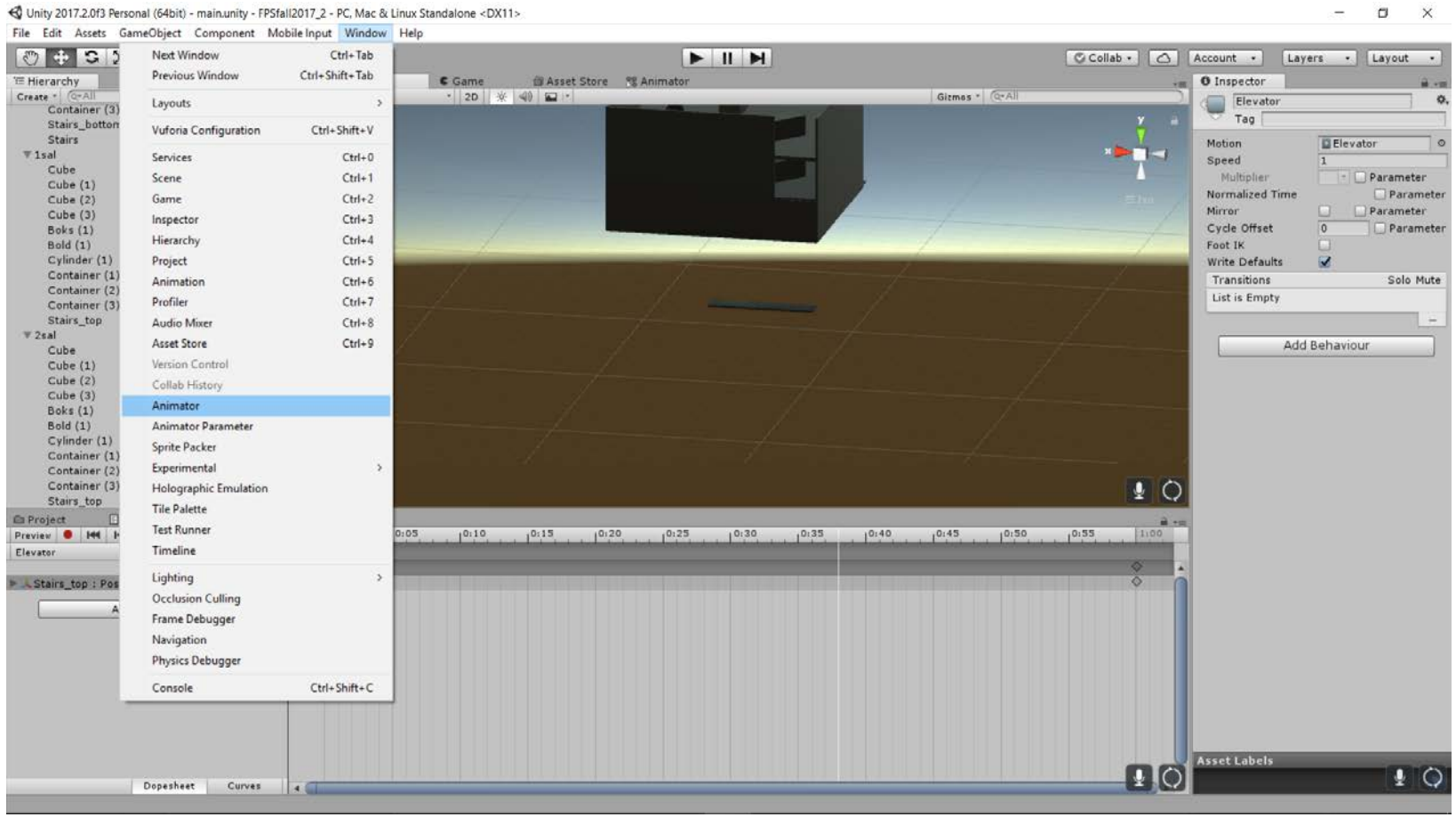
Alle skridtene imellem laver Unity for dig



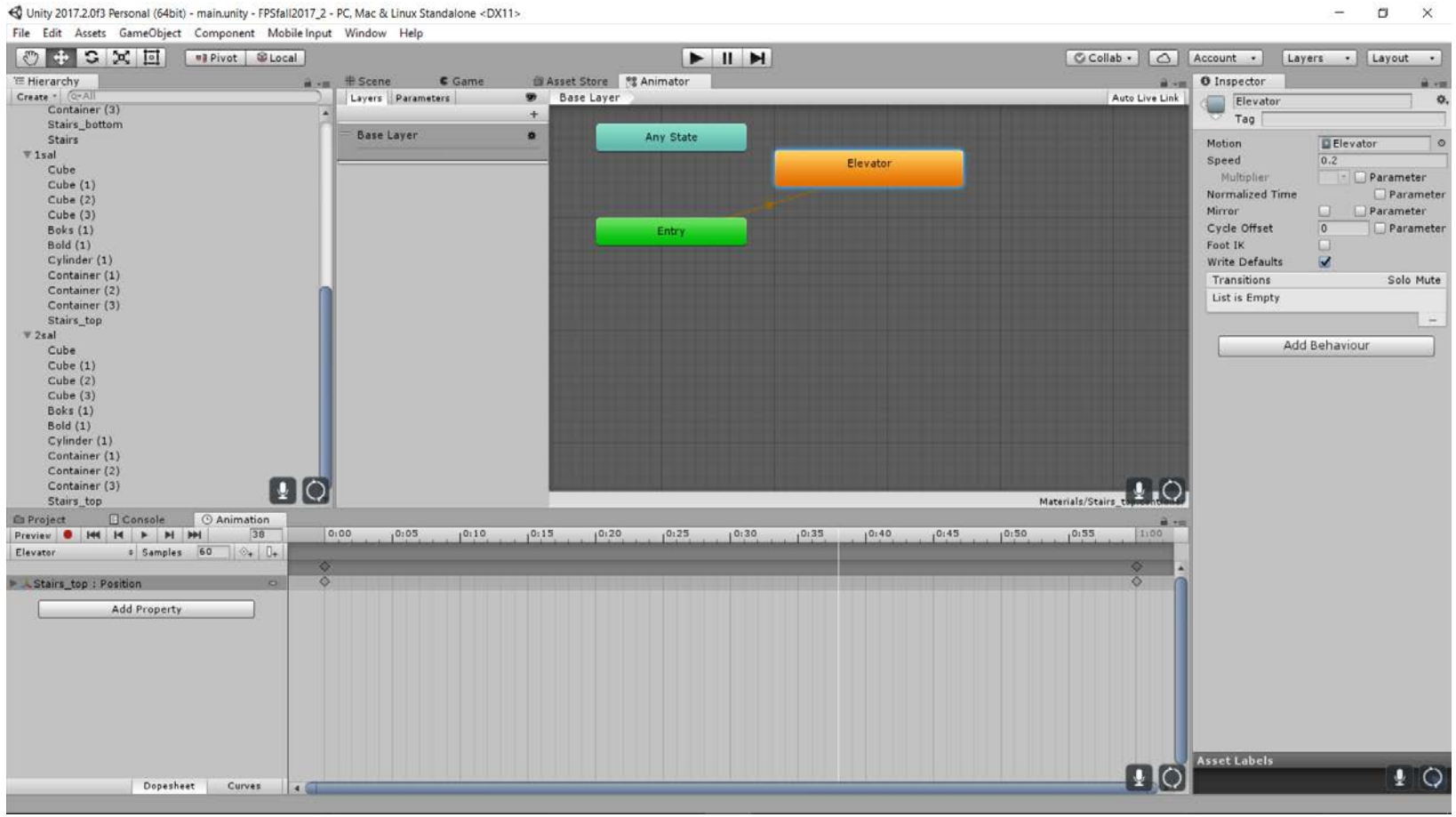
Den bevæger sig lidt hurtigt måske…



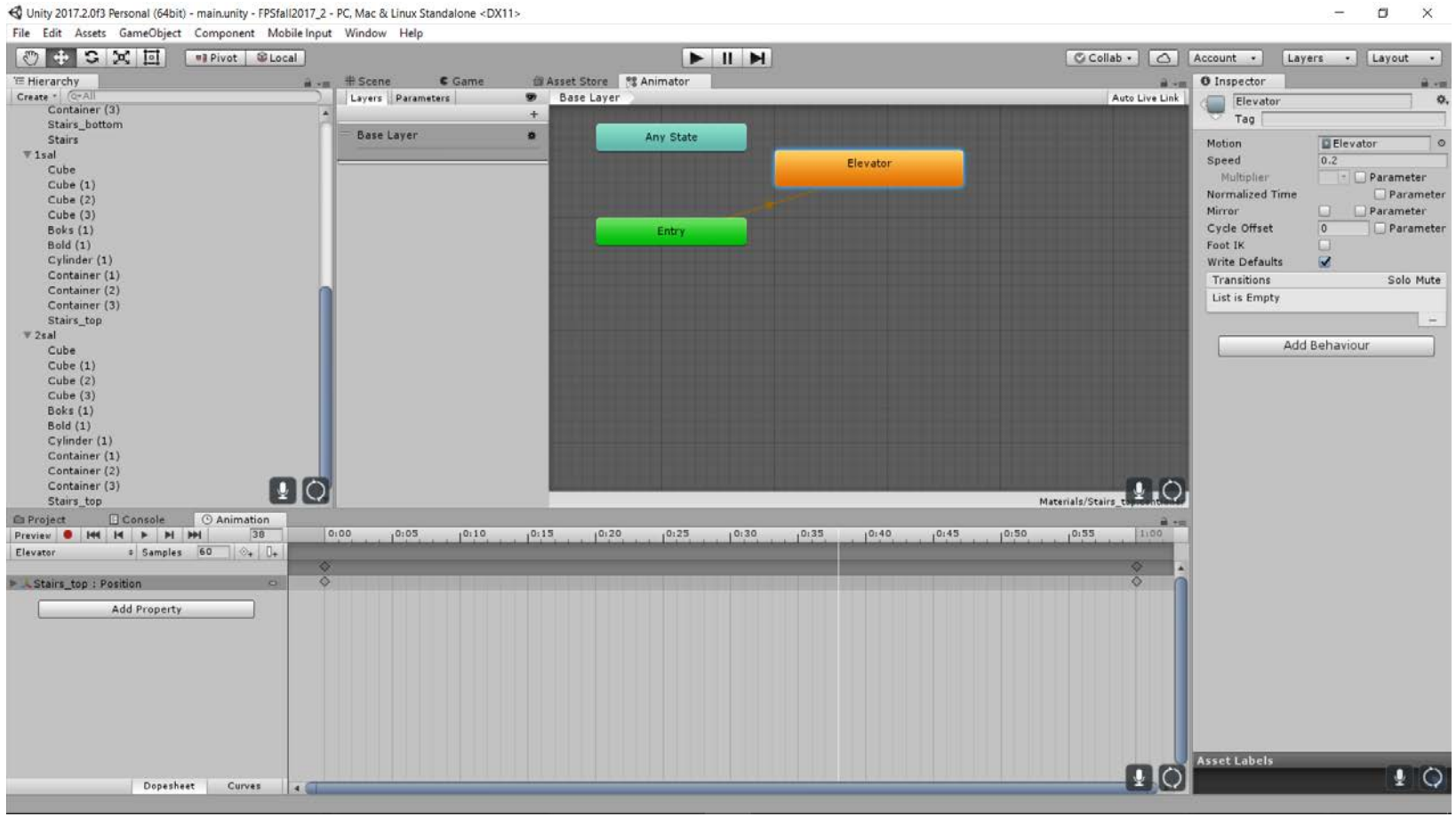
Gå ind i Animator



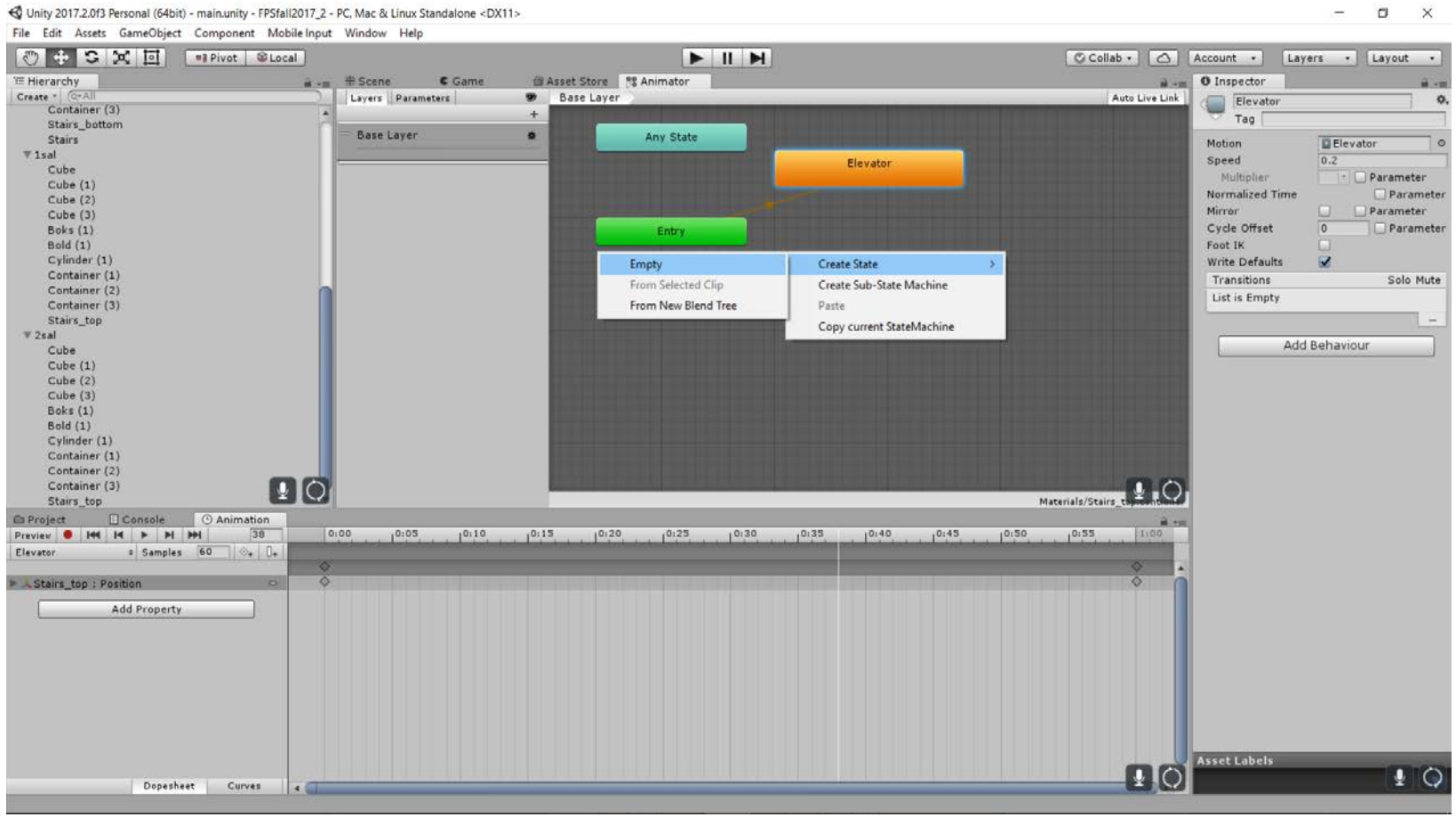
Her kan vi bl.a. justere hastigheden. Jeg har sat den ned fra 1 til 0.2



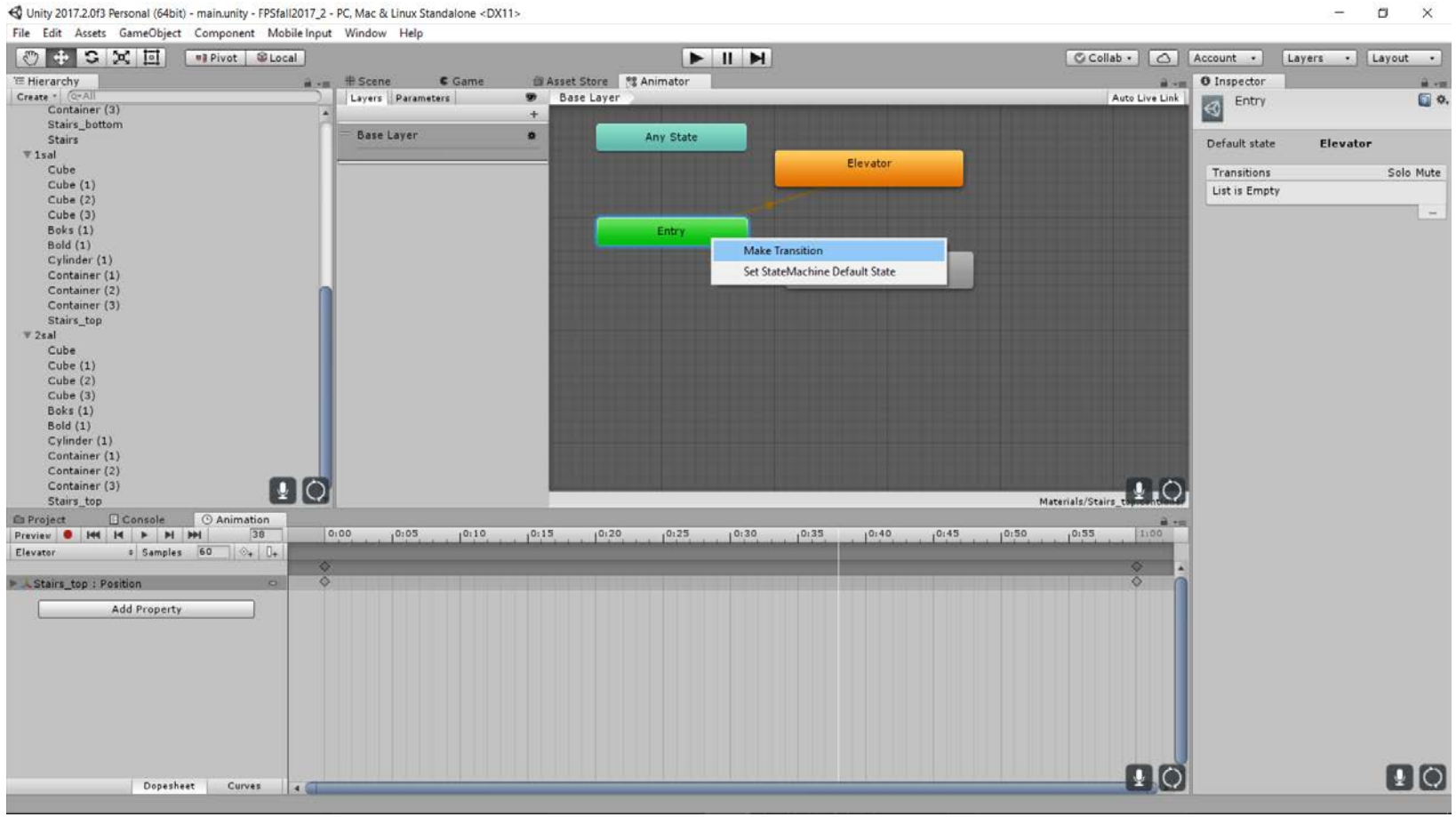
Som det er nu begynder animationen ved Entry, så snart programmet begynder



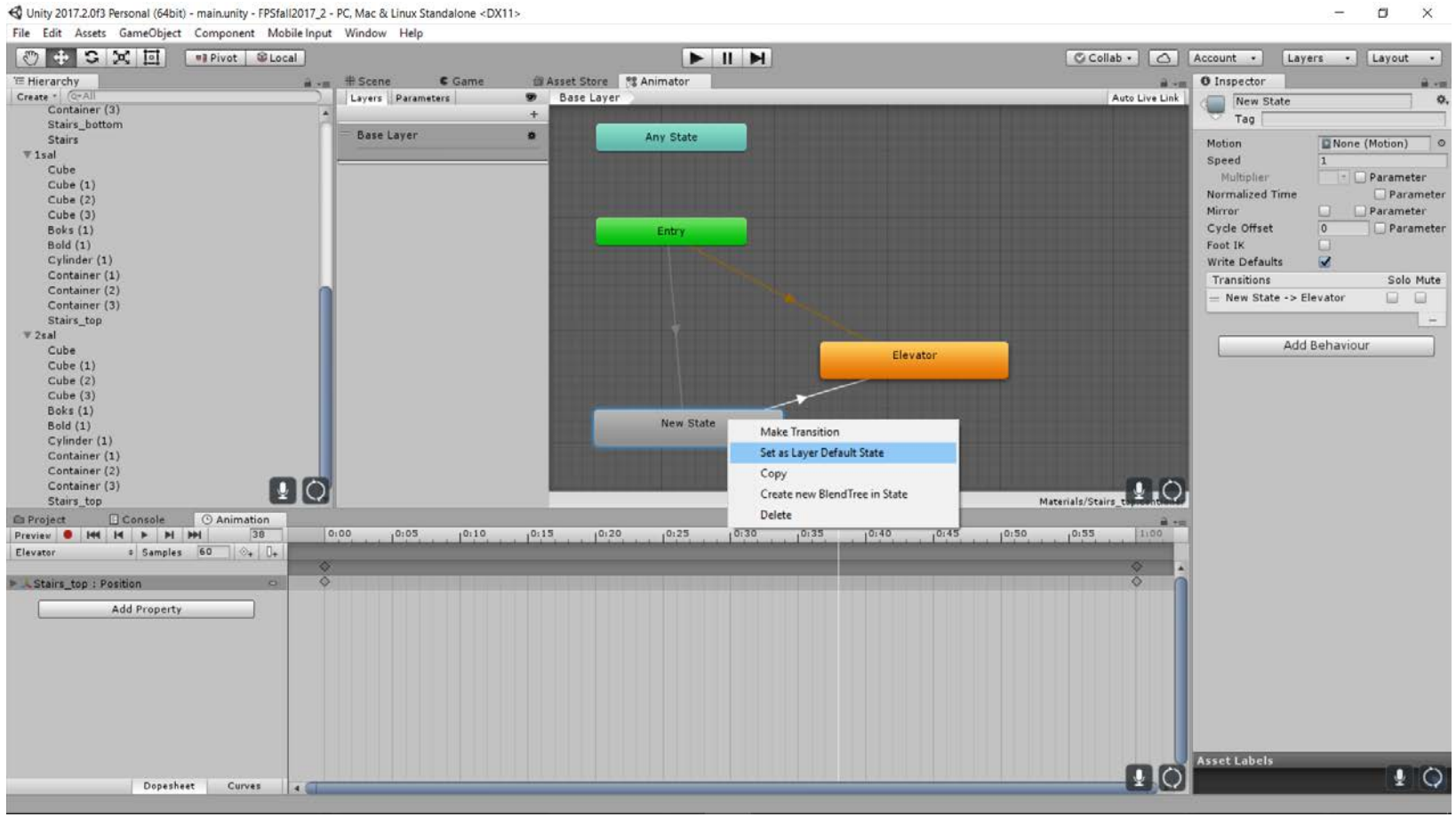
Lav et nyt Empty state



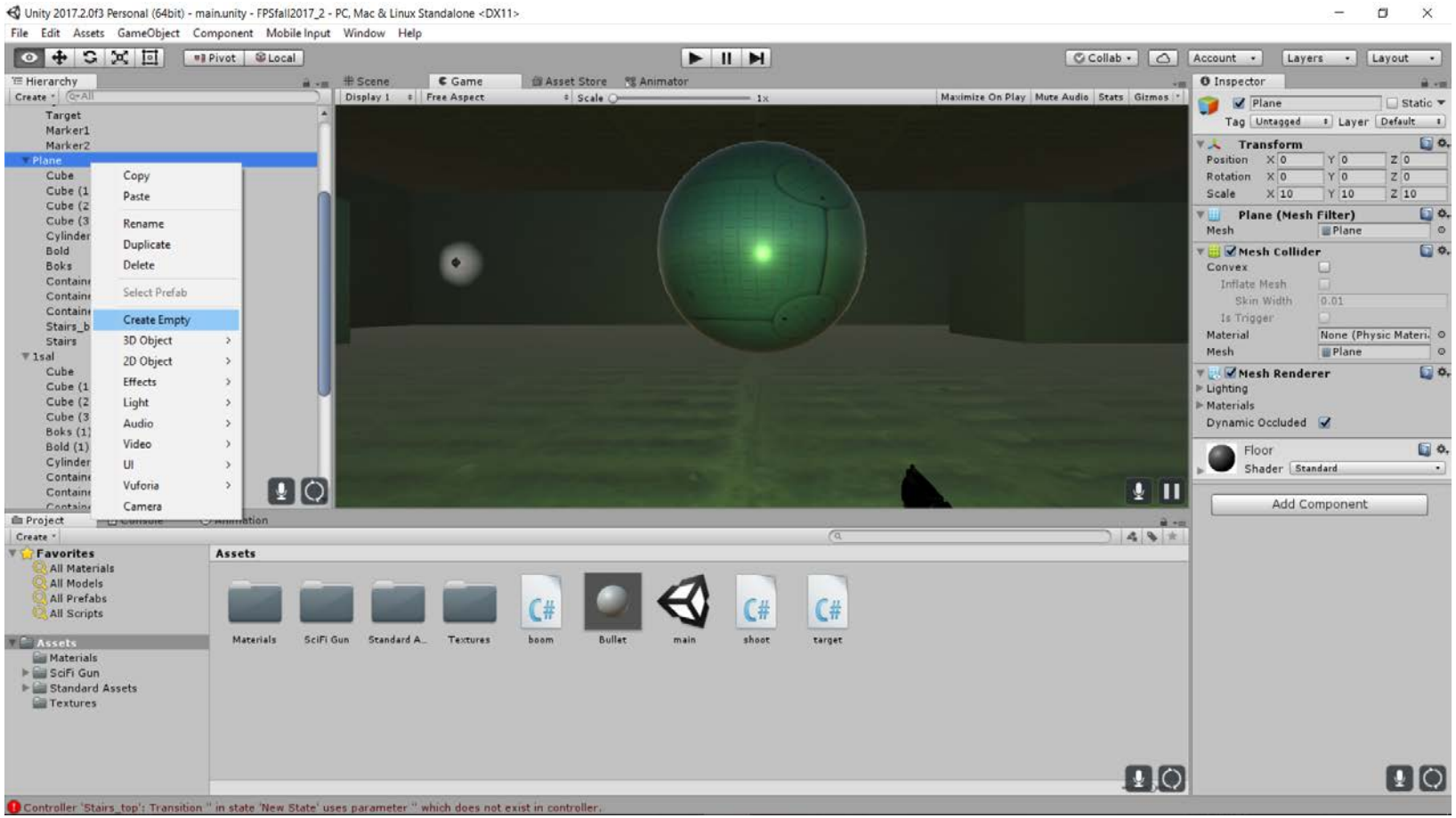
Forbind Entry med det ny state



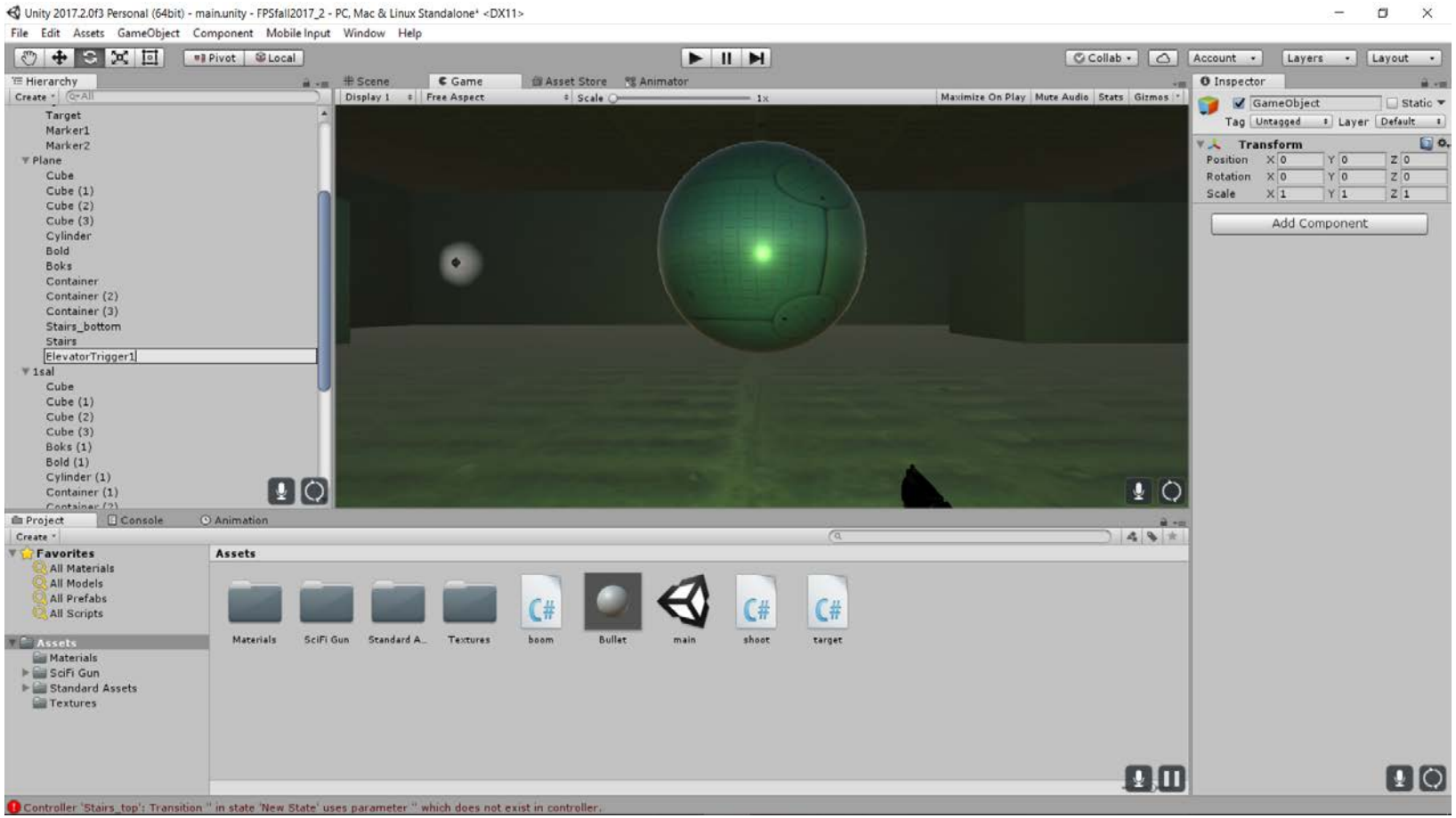
Sørg for at de ny State er default



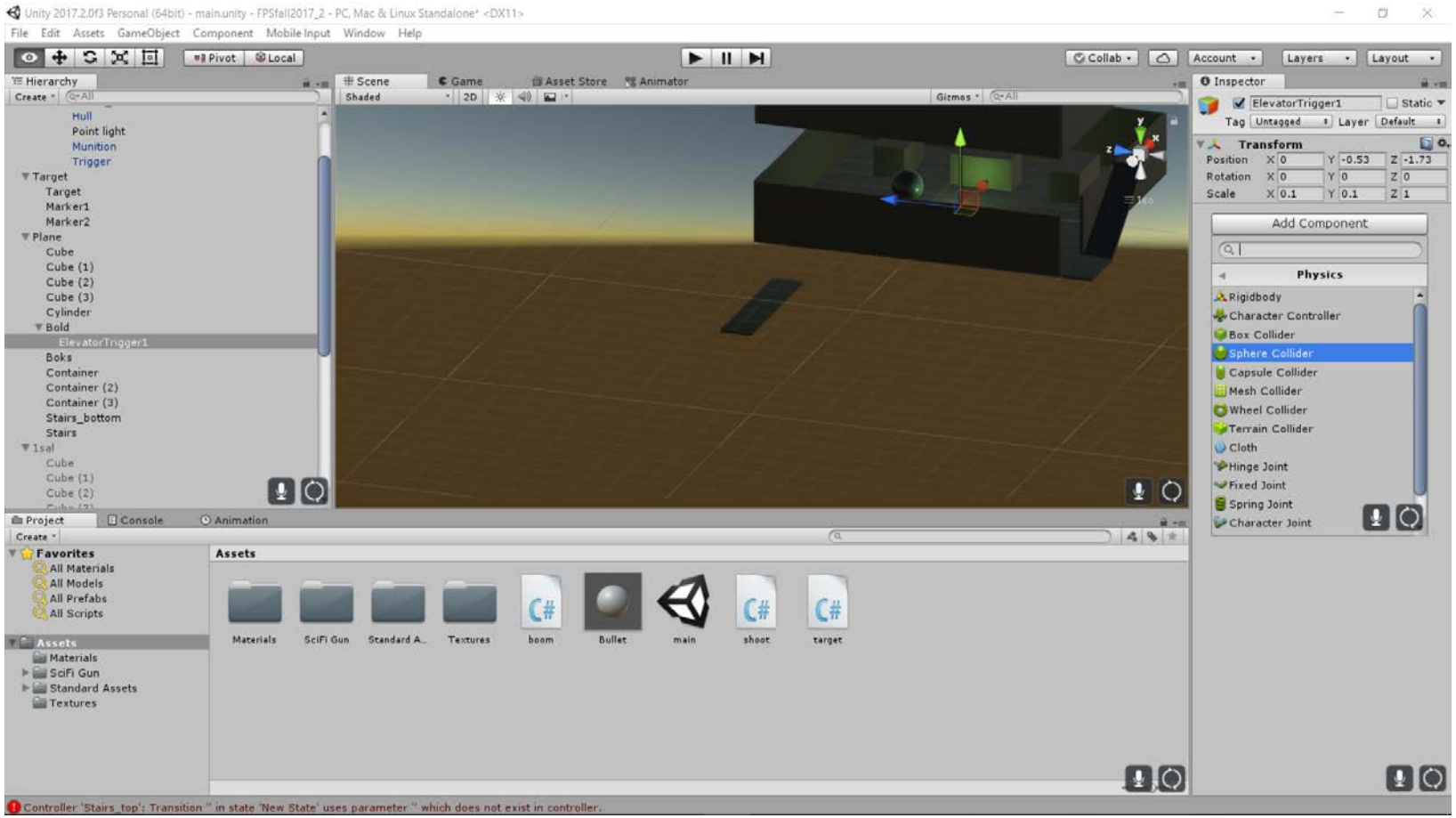
Vi tager et afbræk fra Animator og går tilbage til arbejdsområdet og lavet et Empty GameObject på vores nederste etage



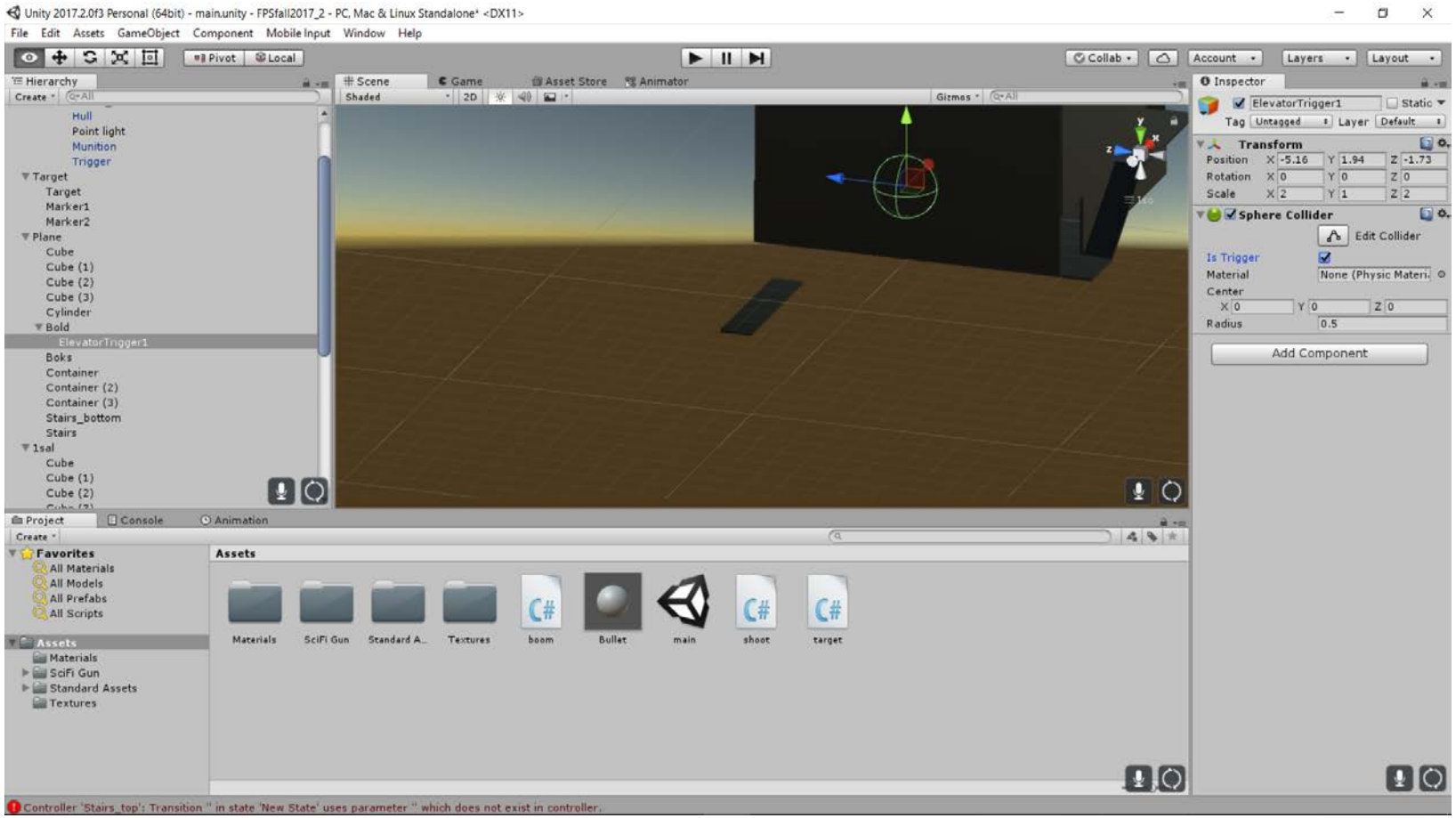
Vi kalder den ElevatorTrigger1



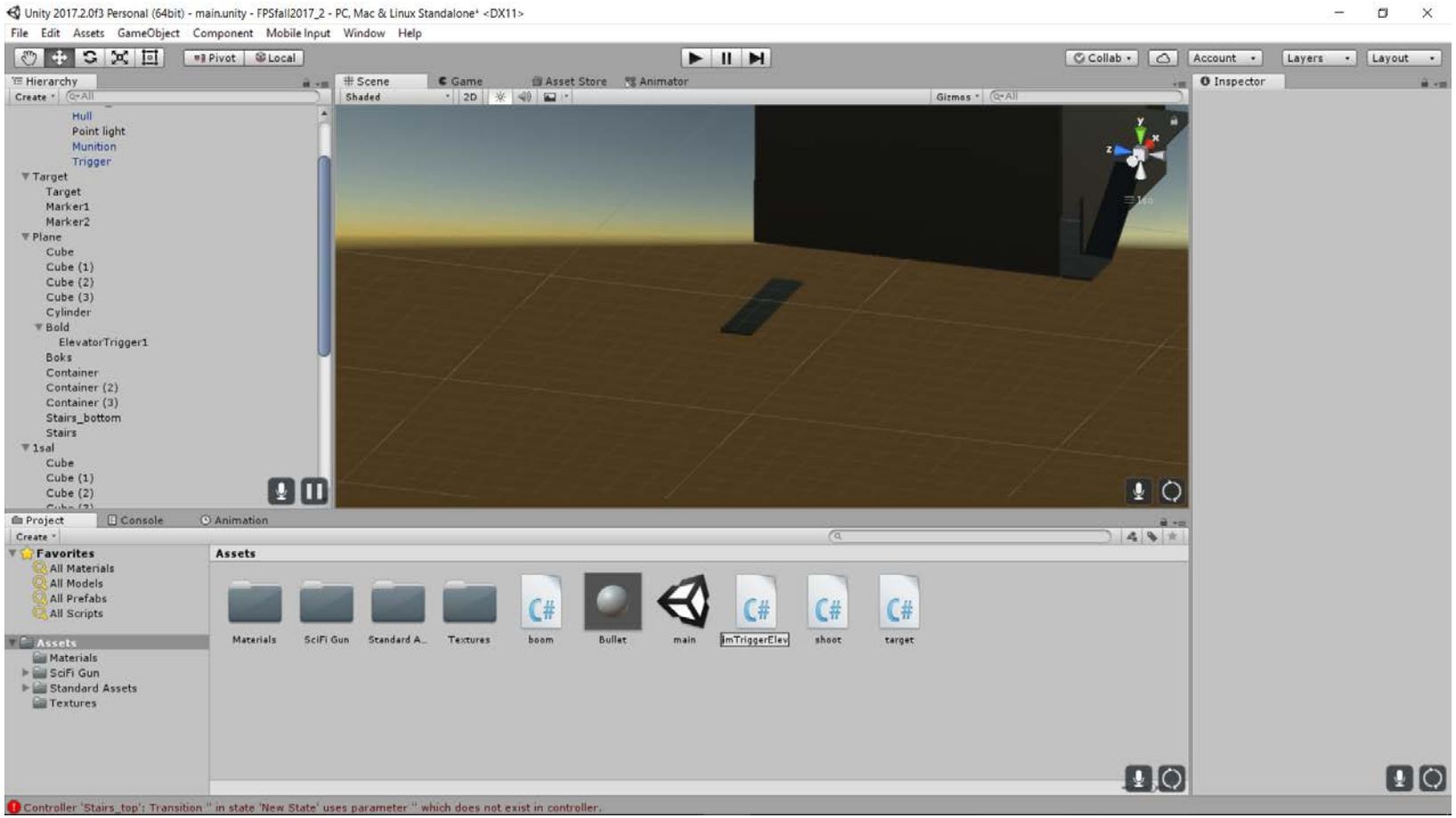
Giv vores trigger en passende Collider



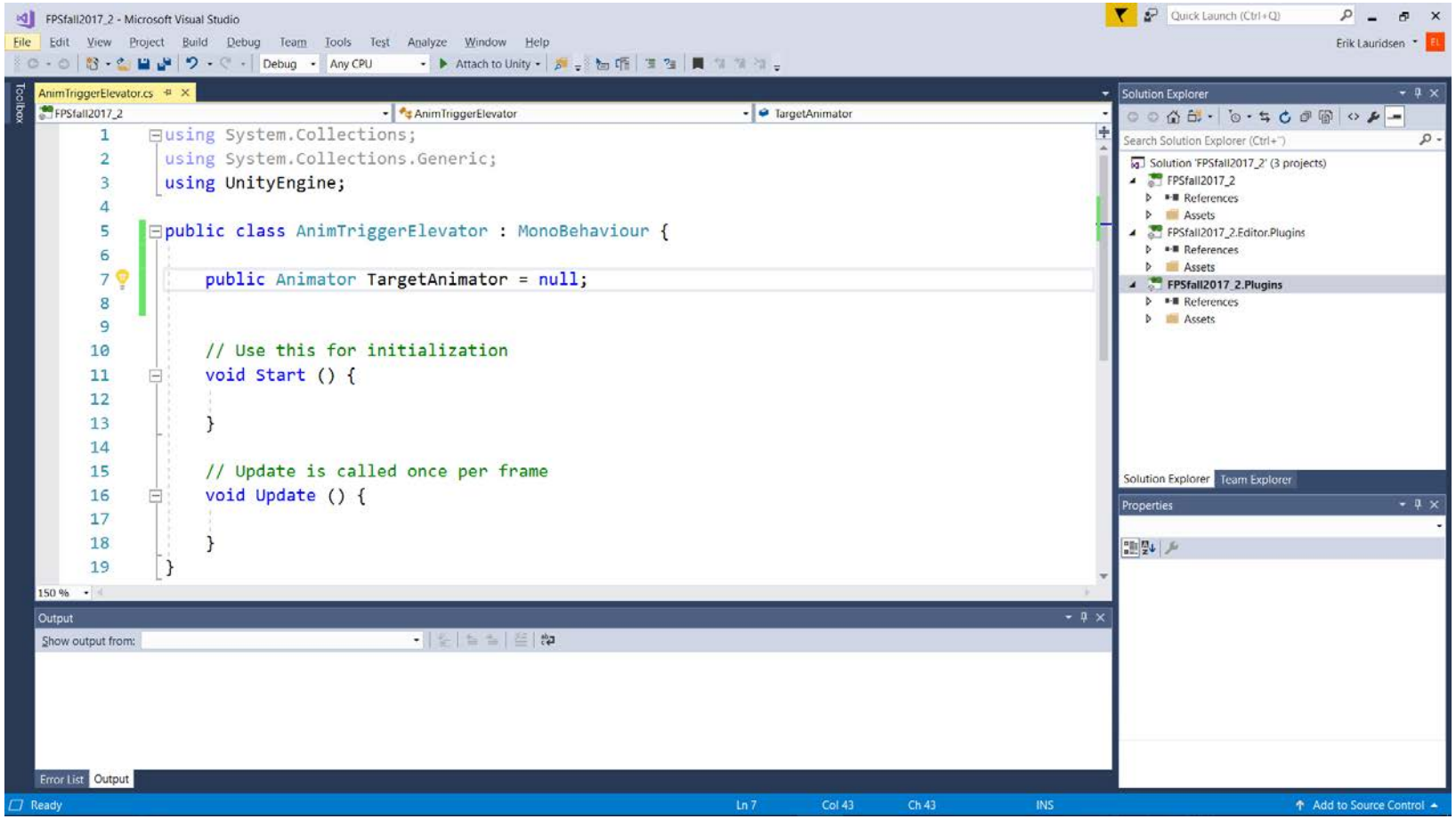
Sørg for at den er markeret som Is Trigger



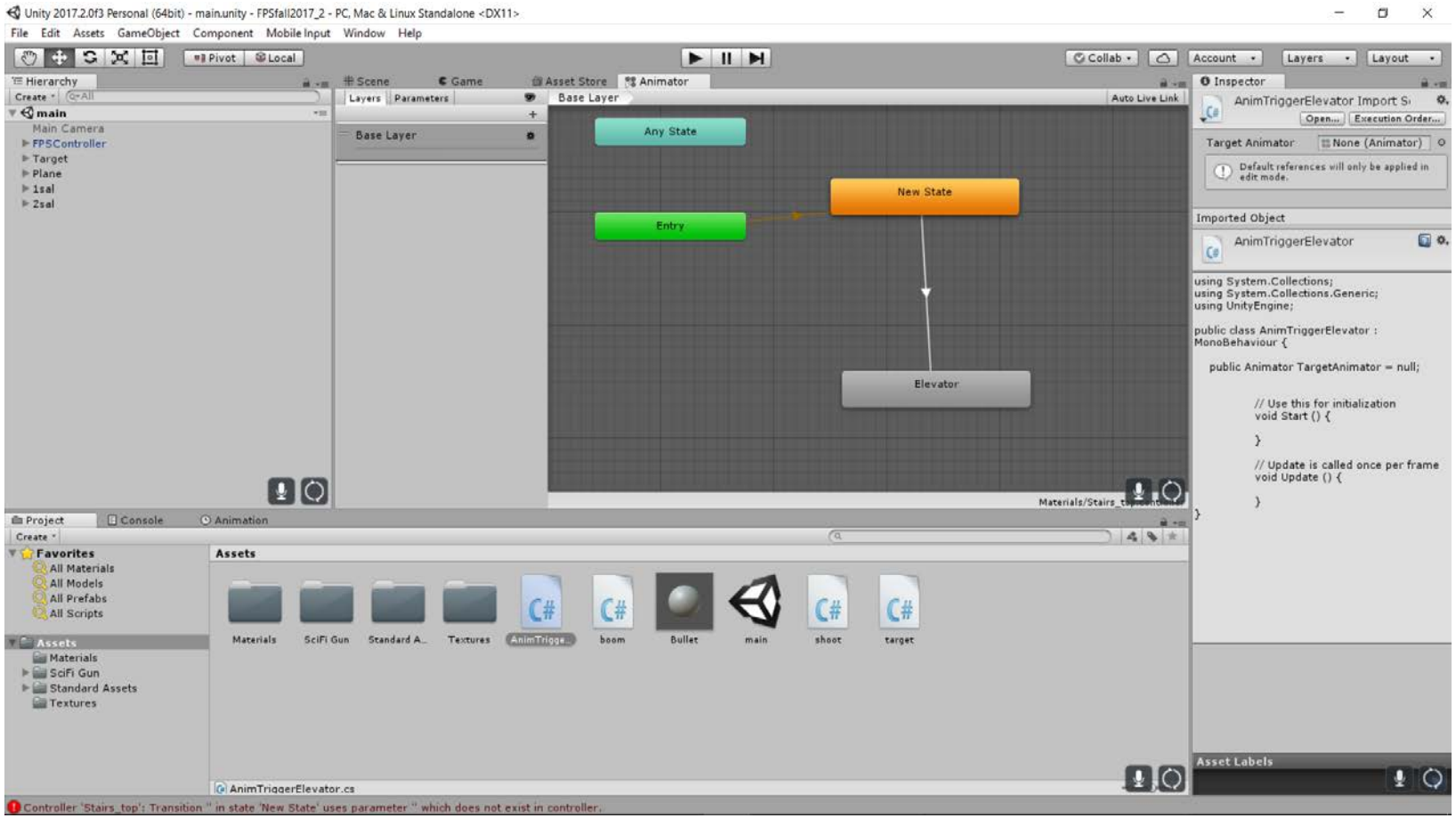
Lav et script der skal føjes til triggeren



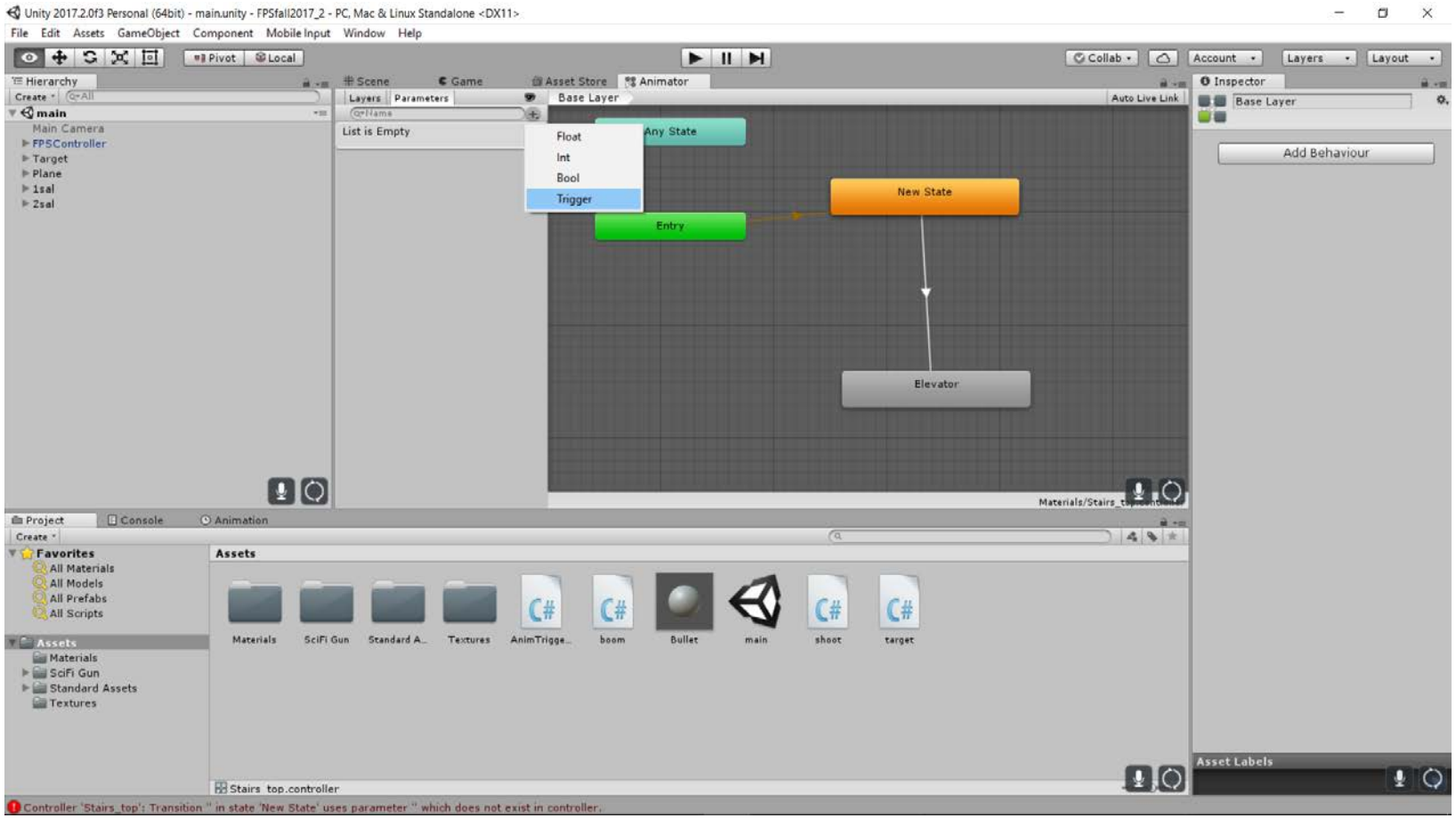
Vi laver en public Animator med target null



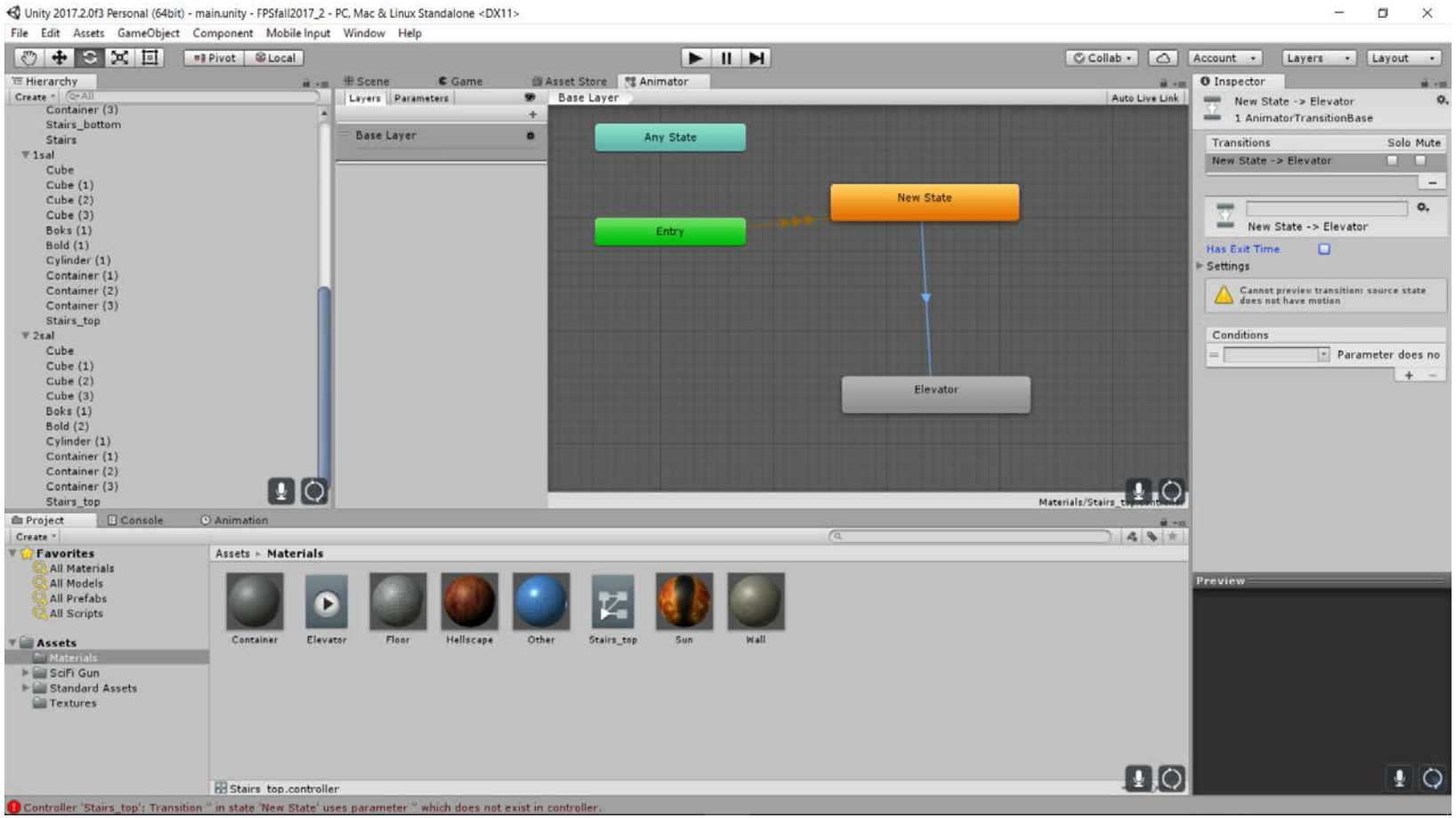
Vi skal have navnet på en trigger så vi går tilbage til Unitys Animator



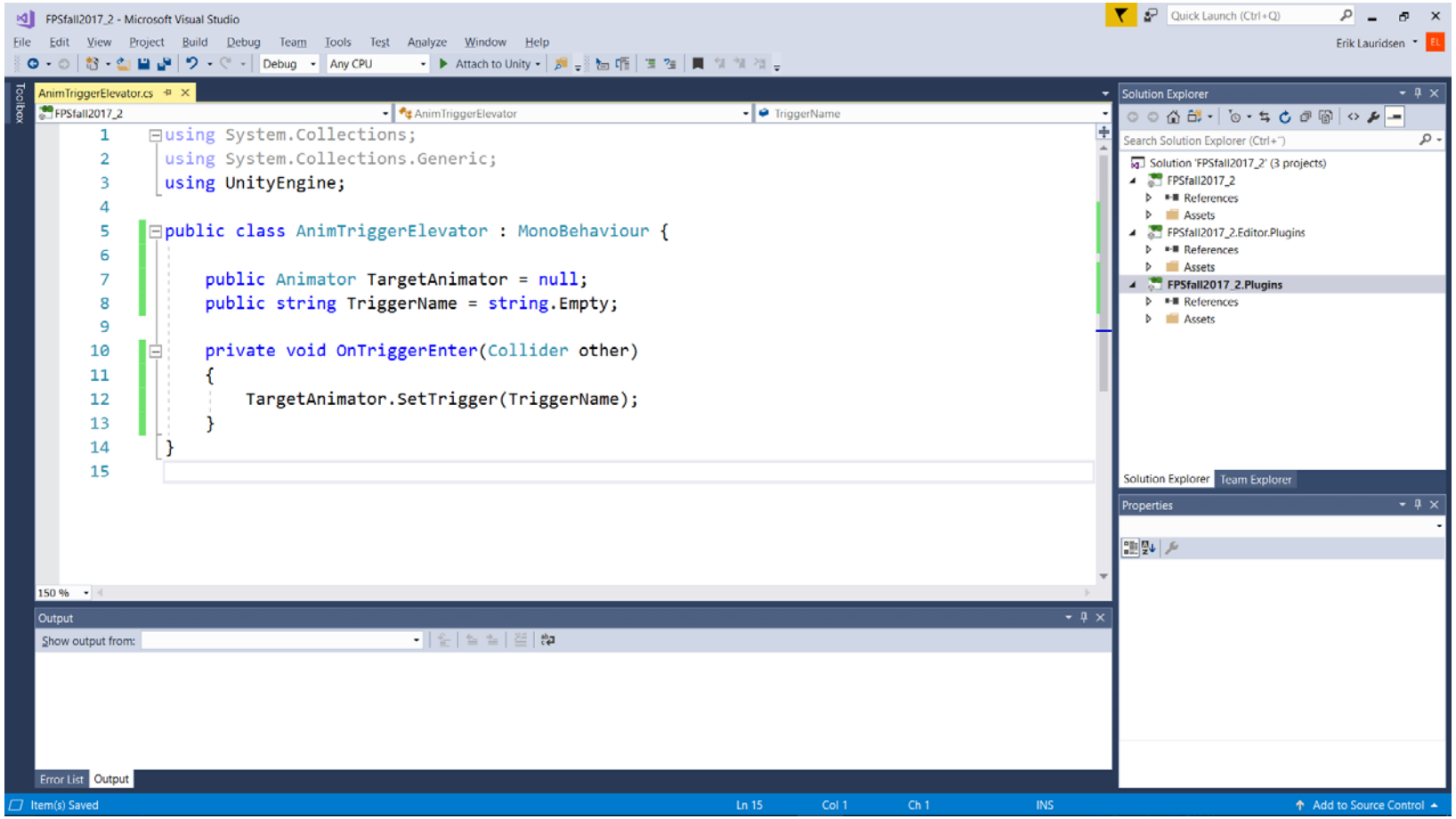
Vi går over i Parameters og tilføjer en Trigger



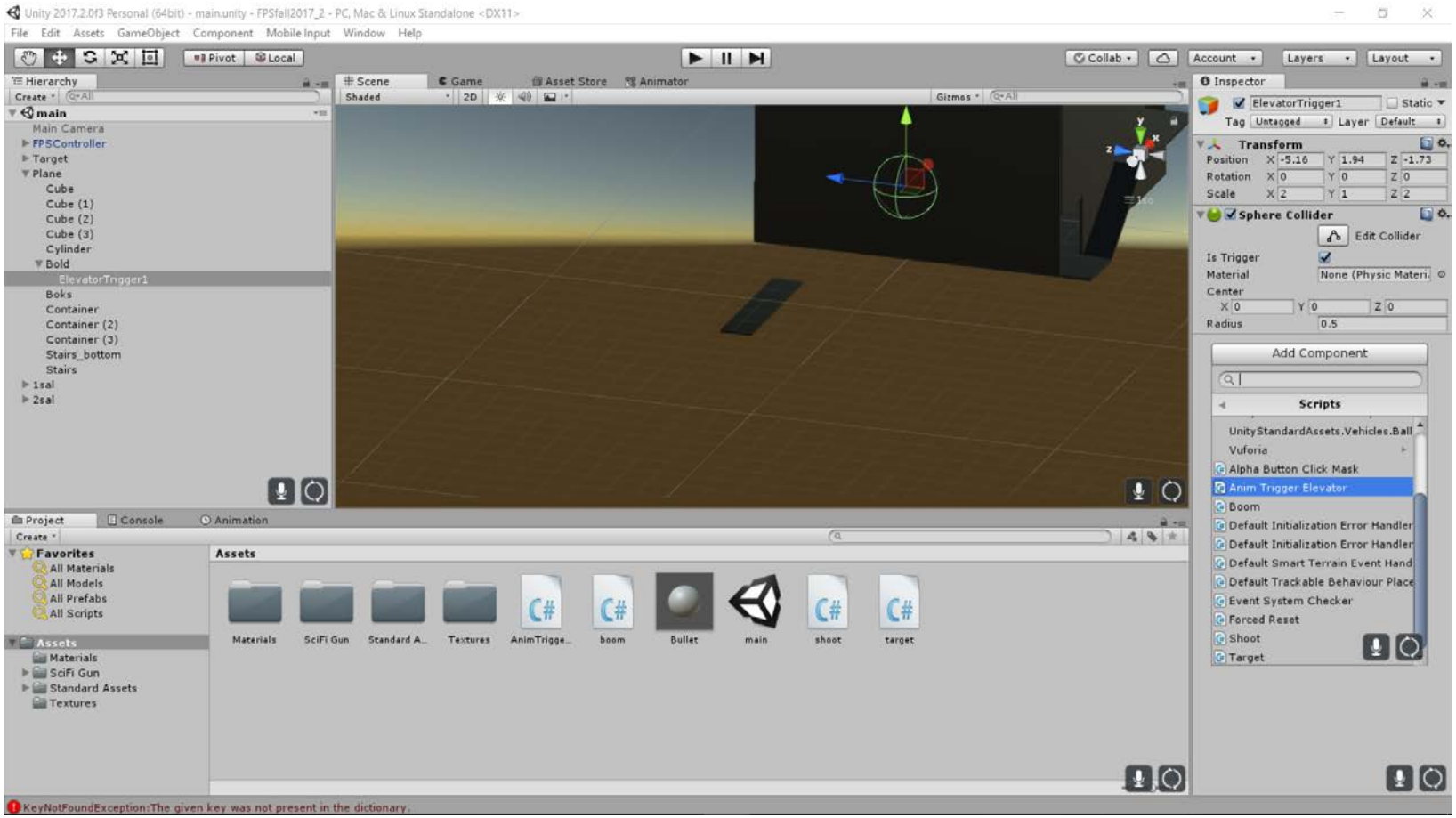
Fjern Has Exit Time fra forbindelsen mellem de to states



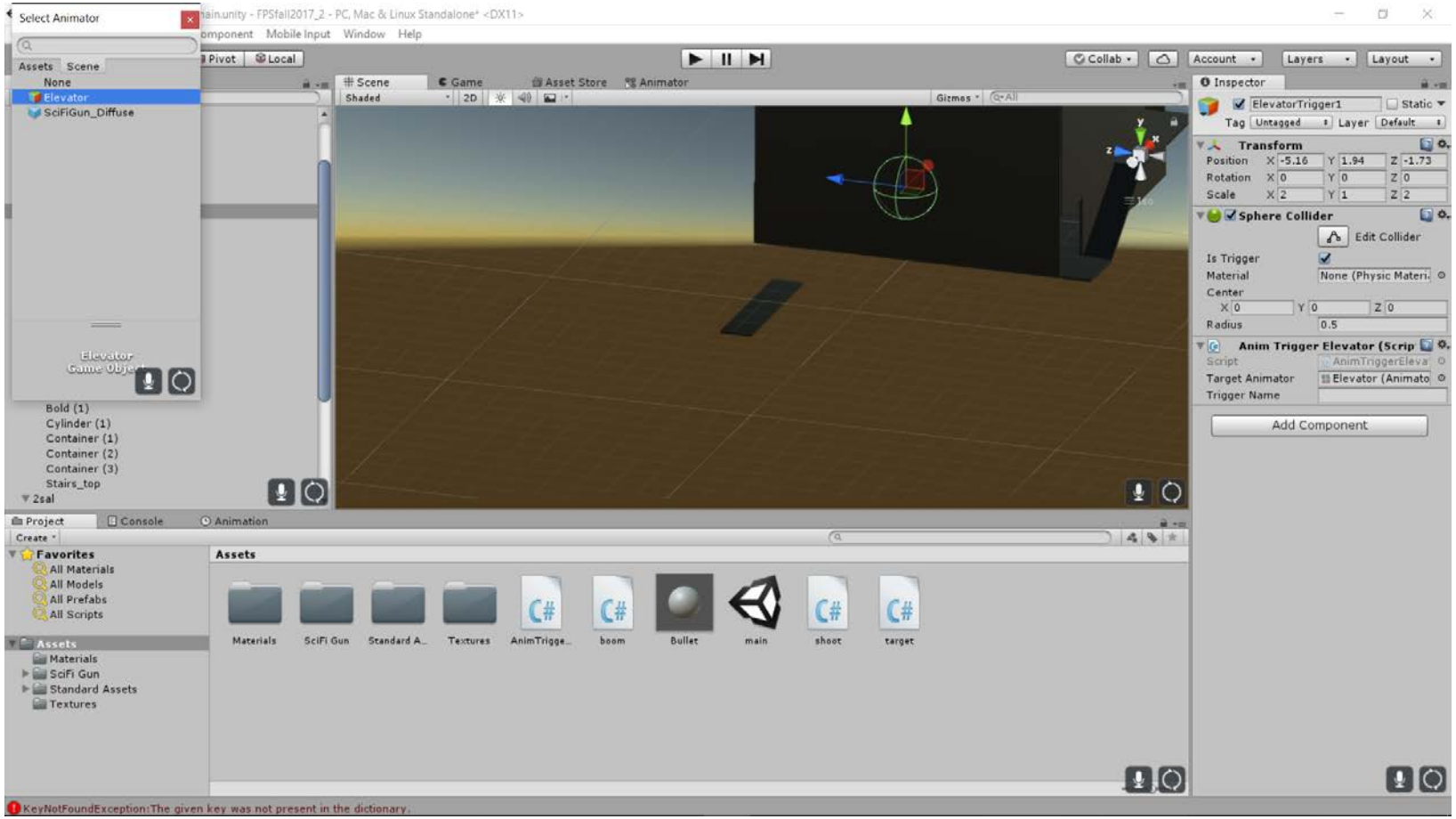
Vi går tilbage til koden og lader en angive et triggernavn og laver en OnTriggerEnter som blot aktiverer den angivne trigger



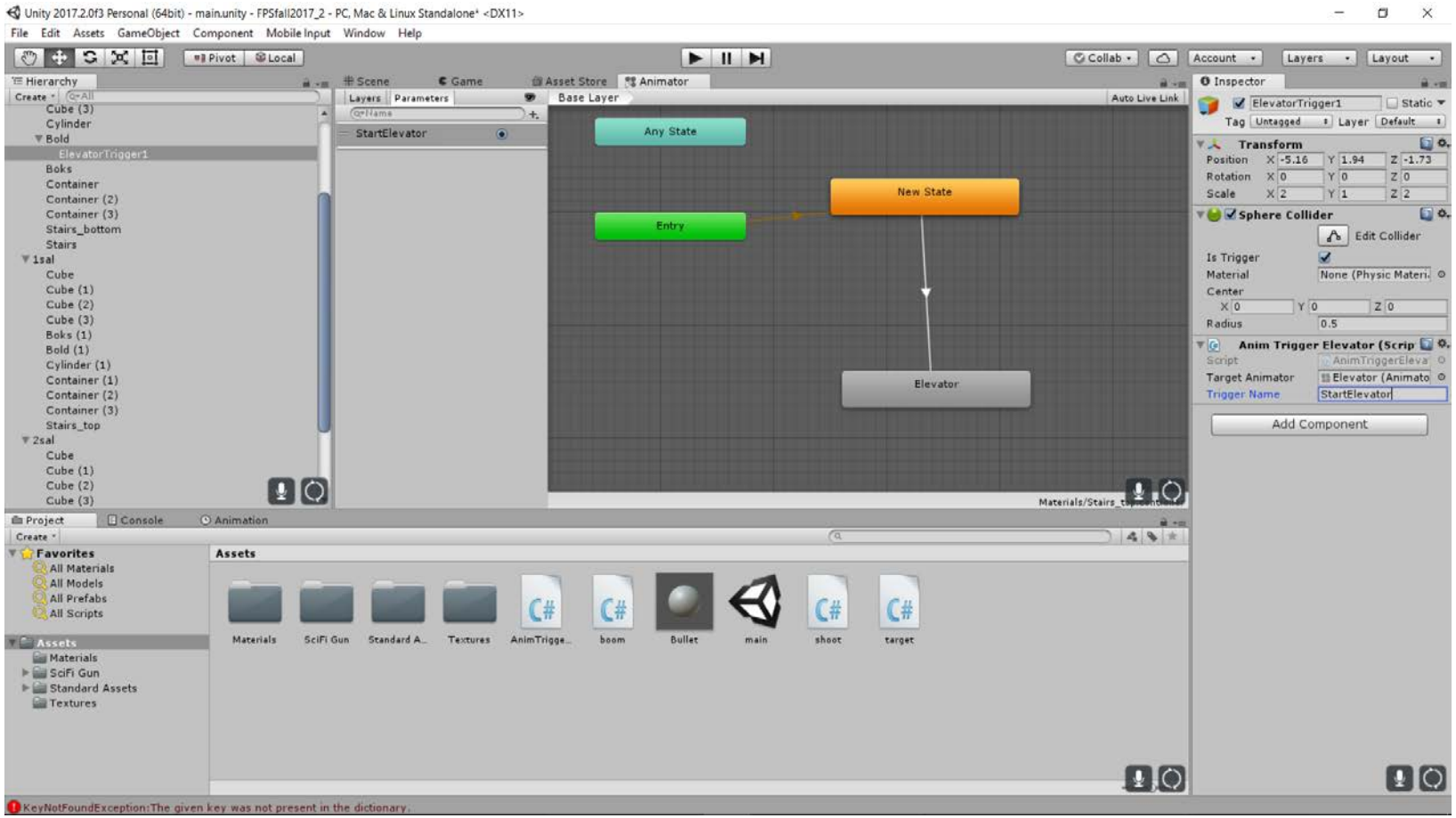
Vi føjer vores ny script til trigger elementet



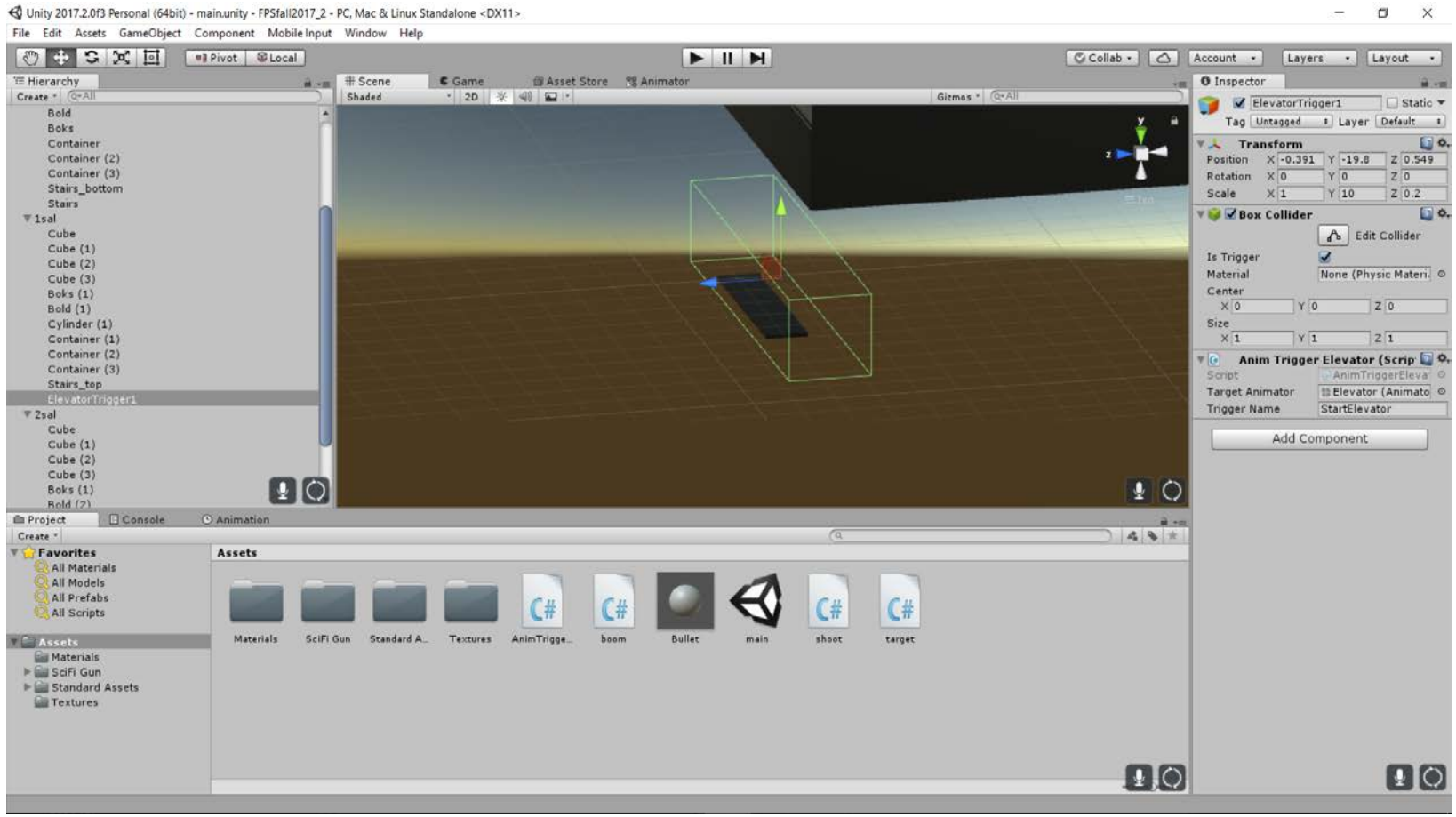
Vi sætter TargetAnimator til at være vores Elevator



Og vi sætter Trigger Name til vores StartElevator parameter



Flyt eventuelt jeres trigger objekt til et sted hvor I er sikker på at I let kan interagere med det



C# og objekt orienteret programmering

Interfaces og generics

Interfaces

- Hvor klasser definerer *hvordan* noget skal udføres sørger interfaces for *hvad* der skal udføres.
- Interfaces definerer properties, methods og events, der er medlemmer af interfacet.
- Interfaces indeholder kun deklarationen af medlemmerne. Det er den afledte klasses ansvar at definere medlemmerne.
- Interfacet hjælper ofte med at give en standard struktur, som de afledte klasser vil følge.
- Abstrakte klasser har til en vis grad samme formål, men de bliver primært brugt når der kun skal erklæres få metoder af base-klassen og de underliggende klasser implementerer funktionerne.

Interfaces

- Interfaces erklæres med interface nøgleordet.
  - Det svarer til en klasse erklæring.
- Interface-udsagn er offentlige som standard.
- Følgende er et eksempel på et interface erklæring:

```
public interface ITransactions
{
    // interface members
    void showTransaction();
    double getAmount();
}
```

In

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System;

namespace InterfaceApplication
{
    1 reference
    public interface ITransactions
    {
        // interface members
        3 references
        void showTransaction();
        2 references
        double getAmount();
    }

    6 references
    public class Transaction : ITransactions
    {
        private string tCode;
        private string date;
        private double amount;
        0 references
        public Transaction()
        {
            tCode = " ";
            date = " ";
            amount = 0.0;
        }

        2 references
        public Transaction(string c, string d, double a)
        {
            tCode = c;
            date = d;
            amount = a;
        }

        2 references
        public double getAmount()
        {
            return amount;
        }

        3 references
        public void showTransaction()
        {
            Console.WriteLine("Transaction: {0}", tCode);
            Console.WriteLine("Date: {0}", date);
            Console.WriteLine("Amount: {0}", getAmount());
        }
    }
    0 references
    class Tester
    {
        0 references
        static void Main(string[] args)
        {
            Transaction t1 = new Transaction("001", "8/10/2012", 78900.00);
            Transaction t2 = new Transaction("002", "9/10/2012", 451900.00);
            t1.showTransaction();
            t2.showTransaction();
            Console.ReadKey();
        }
    }
}
```


C#
#14
INTERFACES

Generics

- Generics gør det muligt at forsinke specifikation af datatypen for program elementer i en klasse eller en metode, indtil de faktisk bliver brugt i programmet.
  - Med generics kan man altså skrive en klasse eller metode, der kan arbejde med alle datatyper.
- Man skriver specifikationerne for klassen eller den metode, med alternative parametre for datatyper.
  - Når compileren møder en constructor for klassen eller et funktionskald for metoden, genererer den kode til at håndtere den specifikke datatype.

```
1  using System;
2  using System.Collections.Generic;
3
4  namespace GenericApplication
5  {
       5 references
6      public class MyGenericArray<T>
7      {
8          private T[] array;
           2 references
9          public MyGenericArray(int size)
10         {
11             array = new T[size + 1];
12         }
13
           2 references
14         public T getItem(int index)
15         {
16             return array[index];
17         }
18
           2 references
19         public void setItem(int index, T value)
20         {
21             array[index] = value;
22         }
23     }
24
       0 references
25     class Tester
26     {
           0 references
27         static void Main(string[] args)
28         {
29
30             //declaring an int array
31             MyGenericArray<int> intArray = new MyGenericArray<int>(5);
32
33             //setting values
34             for (int c = 0; c < 5; c++)
35             {
36                 intArray.setItem(c, c * 5);
37             }
38
39             //retrieving the values
40             for (int c = 0; c < 5; c++)
41             {
42                 Console.Write(intArray.getItem(c) + " ");
43             }
44
```

Fortsættes på næste slide med linje 45-66

Generics

```
45                  Console.WriteLine();
46
47                  //declaring a character array
48                  MyGenericArray<char> charArray = new MyGenericArray<char>(5);
49
50                  //setting values
51                  for (int c = 0; c < 5; c++)
52                  {
53                      charArray.setItem(c, (char)(c + 97));
54                  }
55
56                  //retrieving the values
57                  for (int c = 0; c < 5; c++)
58                  {
59                      Console.Write(charArray.getItem(c) + " ");
60                  }
61                  Console.WriteLine();
62
63                  Console.ReadKey();
64              }
65          }
66      }
```

Forsat fra linje 1-44 på sidste slide

Generics
Egenskaber

Generics kan forbedre ens programmer på følgende måder:

- De hjælper med at maksimere genbrug af kode, type sikkerhed og ydeevne.
- Du kan oprette generic collection klasser.
  - .NET Framework klasse bibliotek indeholder en række nye generic collection classes i System.Collections.Generic namespace. Man kan bruge disse generic collection classes i stedet for collection classes i System.Collections namespace.
- Man kan lave egne generic interfaces, classes, methods, events og delegates.
- Man kan lave generic classes tvunget til at give adgang til metoder for bestemte datatyper.
- Man kan få information om de typer, der anvendes i en generic datatype ved run-time ved hjælp af reflection.

Generics
Metoder

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

namespace GenericMethodAppl
{
    0 references
    class Program
    {
        2 references
        static void Swap<T>(ref T lhs, ref T rhs)
        {
            T temp;
            temp = lhs;
            lhs = rhs;
            rhs = temp;
        }
        0 references
        static void Main(string[] args)
        {
            int a, b;
            char c, d;
            a = 10;
            b = 20;
            c = 'I';
            d = 'V';

            //display values before swap:
            Console.WriteLine("Int values before calling swap:");
            Console.WriteLine("a = {0}, b = {1}", a, b);
            Console.WriteLine("Char values before calling swap:");
            Console.WriteLine("c = {0}, d = {1}", c, d);

            //call swap
            Swap<int>(ref a, ref b);
            Swap<char>(ref c, ref d);

            //display values after swap:
            Console.WriteLine("Int values after calling swap:");
            Console.WriteLine("a = {0}, b = {1}", a, b);
            Console.WriteLine("Char values after calling swap:");
            Console.WriteLine("c = {0}, d = {1}", c, d);

            Console.ReadKey();
        }
    }
}
```


C#
#15
GENERICS

TEDx talk om muligheder for VR

Google Cardboard nævnes


X
ED

Google Cardboard app fra Unity

En code-along video

Introduktion til eksamensopgave

Kilder

C# interfaces

- https://www.tutorialspoint.com/csharp/csharp_interfaces.htm
- https://youtu.be/IQpss9YAc4g

C# generics

- https://www.tutorialspoint.com/csharp/csharp_generics.htm
- https://youtu.be/ZrjCG0Fu5Ew

Kilder

TEDx talk

- https://youtu.be/DQMA5NNhN58

Google Carboard fra Unity

- https://youtu.be/DHBgundyLMU
- https://developers.google.com/vr/unity/