# Lektion 3

Grundlæggende programmering i VR

# Plan for i dag

- UML
  - Usecase diagrammer
  - Aktivitets diagrammer
  - Klasse diagrammer
- Udforskning af forskelligt VR og andre måder at udvide virkelig-heden på
  - Cardboard
  - Vive
  - Hologram
  - Realfiction Dreamoc Pop
- Introduktion til objektorienteret programmering
- Simpelt FPS

# UML

Unified Model Language

# UML

Hvad er UML

- UML er et sprog
- Regler for, hvordan elementer er sat sammen
- Regler for organisationen
- UML vise  hvordan elementer forholder sig til hinanden
- Kan både anvendes i software værktøjer, på white boards og på papir

# UML

Hvilken software kan jeg bruge til at lave UML diagrammer

- Et stykke papir
- Gliffy - www.gliffy.com
- Astah - www.astah.net
- Der er dog **mange** andre muligheder, så find din egen foretrukne løsning

# UML

## OPGAVE

- Hent Astah community edition (http://www.astah.net/download)
- Installer det på din computer

# UML

Der er grundlæggende **to slags diagrammer**

- Adfærds diagrammer
  - Krav, drift, indre tilstande
- Struktur diagrammer
  - Fysisk organisation

# UML

Adfærds diagrammer

- Use case diagram
  - Funktionelle krav til et system
  - Hvad et system skal gøre
  - Gør det muligt for den der laver modellen at fokusere på brugerens behov snarere end detaljer i produktionen
- Aktivitet diagram
  - Vis strømmen fra en adfærd eller aktivitet, til den næste
  - Med udtryksfuld end en klassisk flowchart

Struktur diagrammer

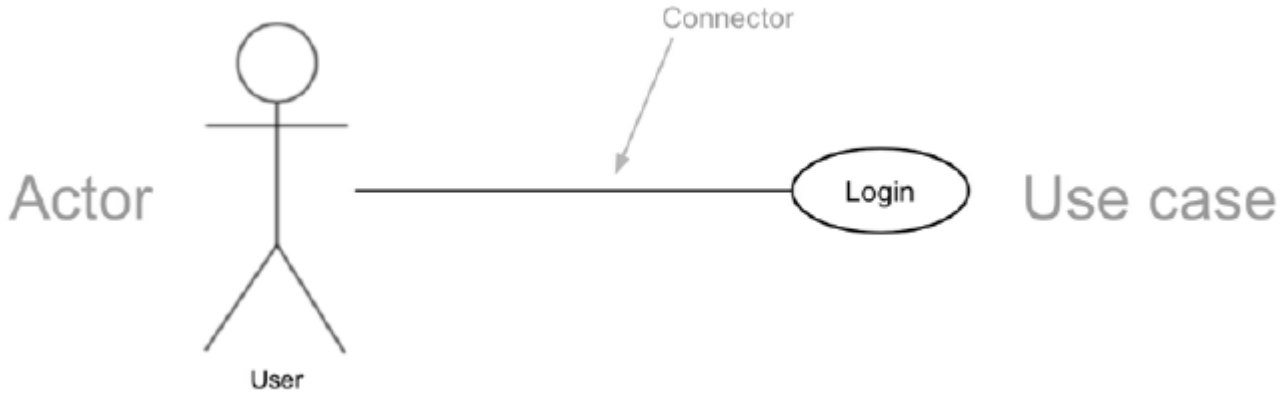
- Klasse diagrammer
  - Viser enheder i et system og forholdet mellem dem
  - Kan være detaljeret og generere kildekode eller simple skitser

# UML

## Adfærds diagram

Use case diagrammer

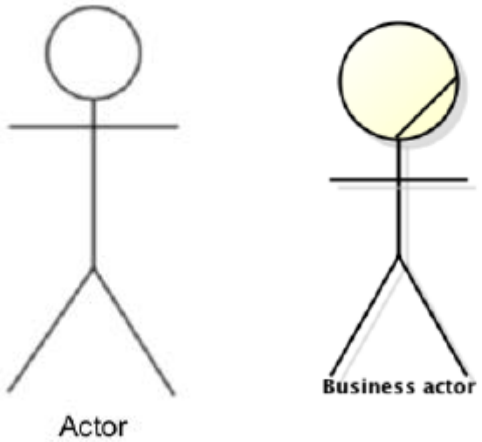
- Grafisk oversigt over en eller flere aktørers involvering i et system.



# UML
## Adfærds diagram

Use case diagrammer

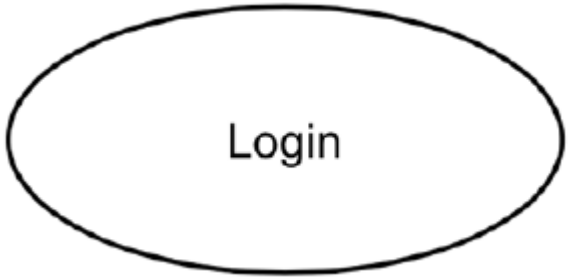
- Aktør
  - En enhed, der udfører en rolle i et system
    - Kan være
      - En person
      - Et eksternt system



# UML
## Adfærds diagram

Use case diagrammer

- Use case
  - Et use case er en funktion eller på handling inden for systemet
    - Det kan være
    - Log on
    - At ændre profil billede
    - At skrive på vens ”væg”

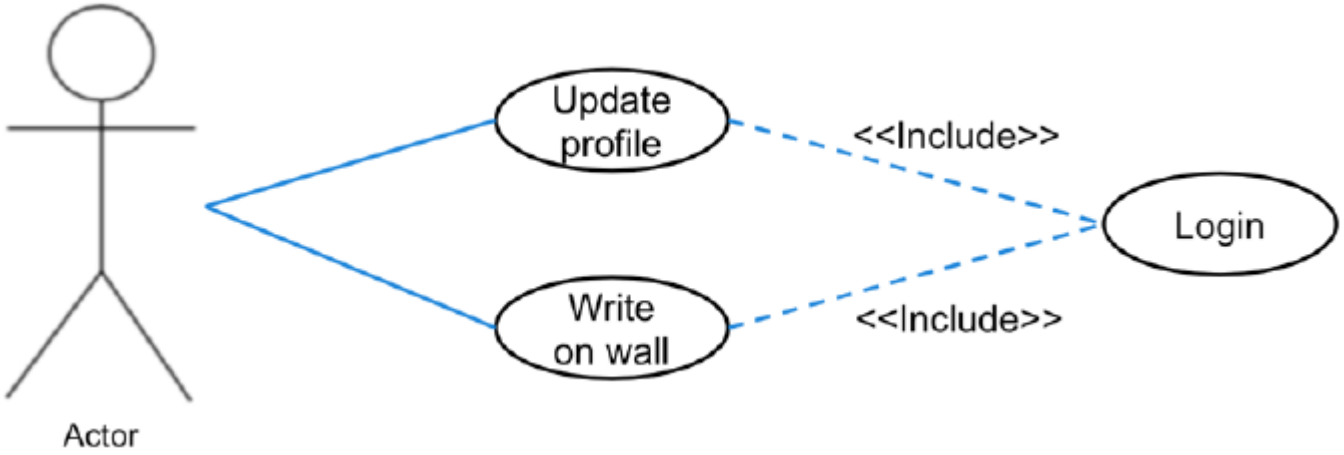


# UML

## Adfærds diagram

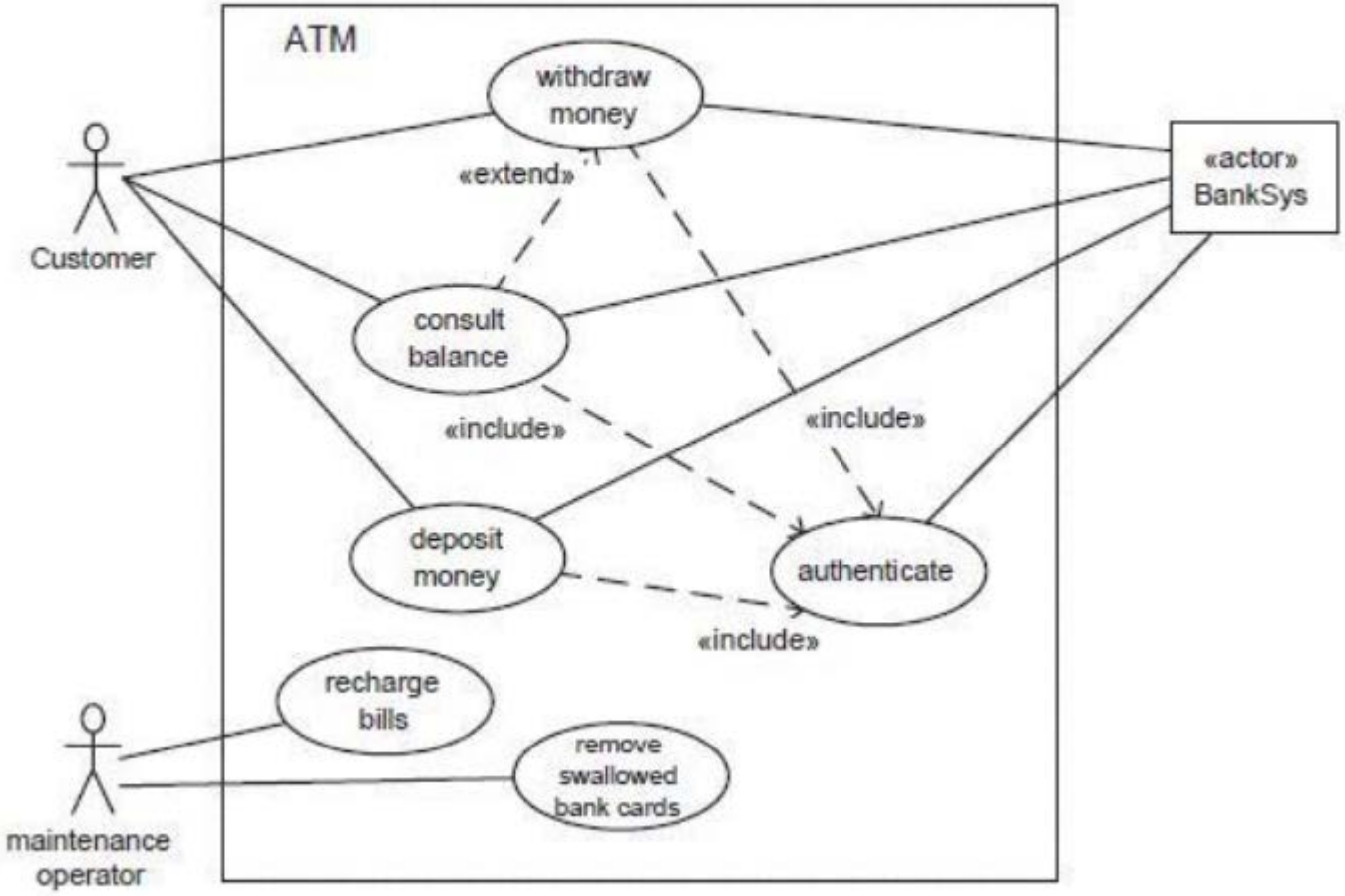
Use case diagrammer

- **Include** bruges til fælles funktioner som kan g
- Metoder vil blive anv include anvendes.



# UML
## Adfærds diagram

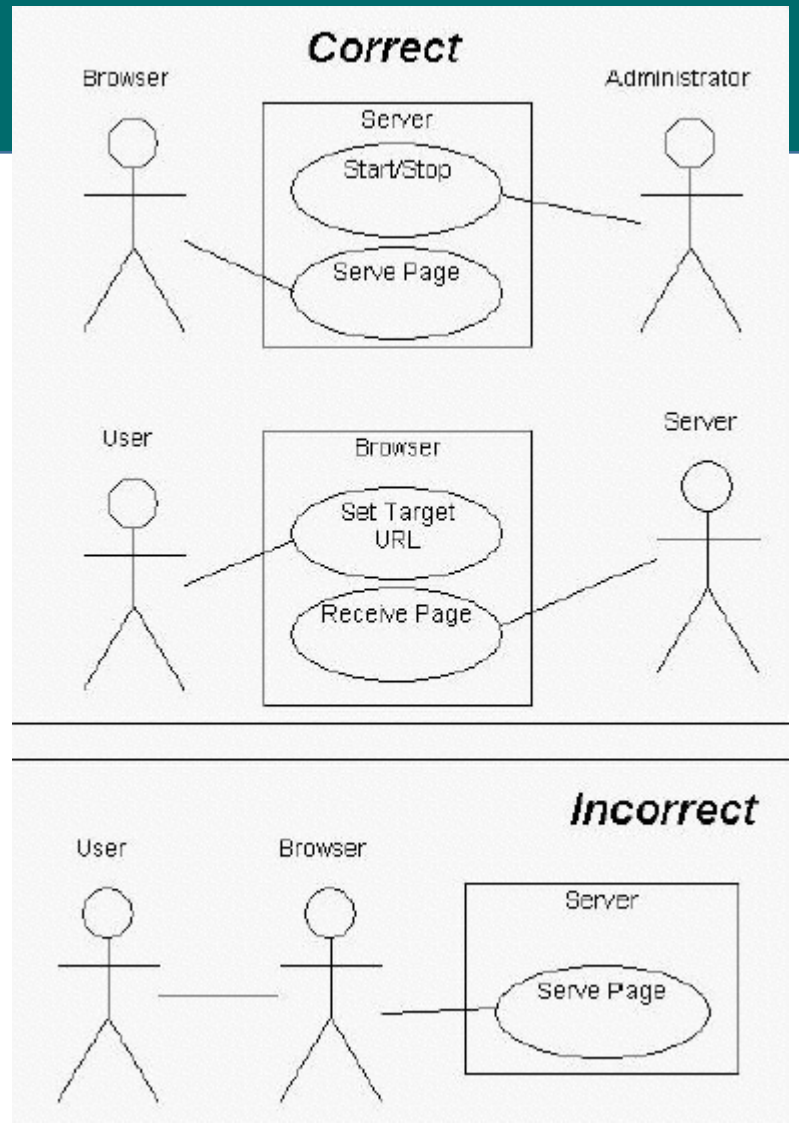
### Use Case Diagram



# UML

## Adfærds diagram

Use case diagrammer



# UML
## Adfærds diagram

### OPGAVE

- Start Astah hvis I har lukket det
- Lav et use case diagram
- Der er en bruger, der gerne vil skrive en email til hans chef hvor han fortæller at han er syg
- Vis hvilke ”brugs” trin det at han vil skrive en mail går igennem og hvordan kæden mellem ham og chefen hænger sammen
  - Vi kigger ikke på hvad der sker undervejs i detaljer, kun brugs-stadier

# UML

## Adfærds diagram

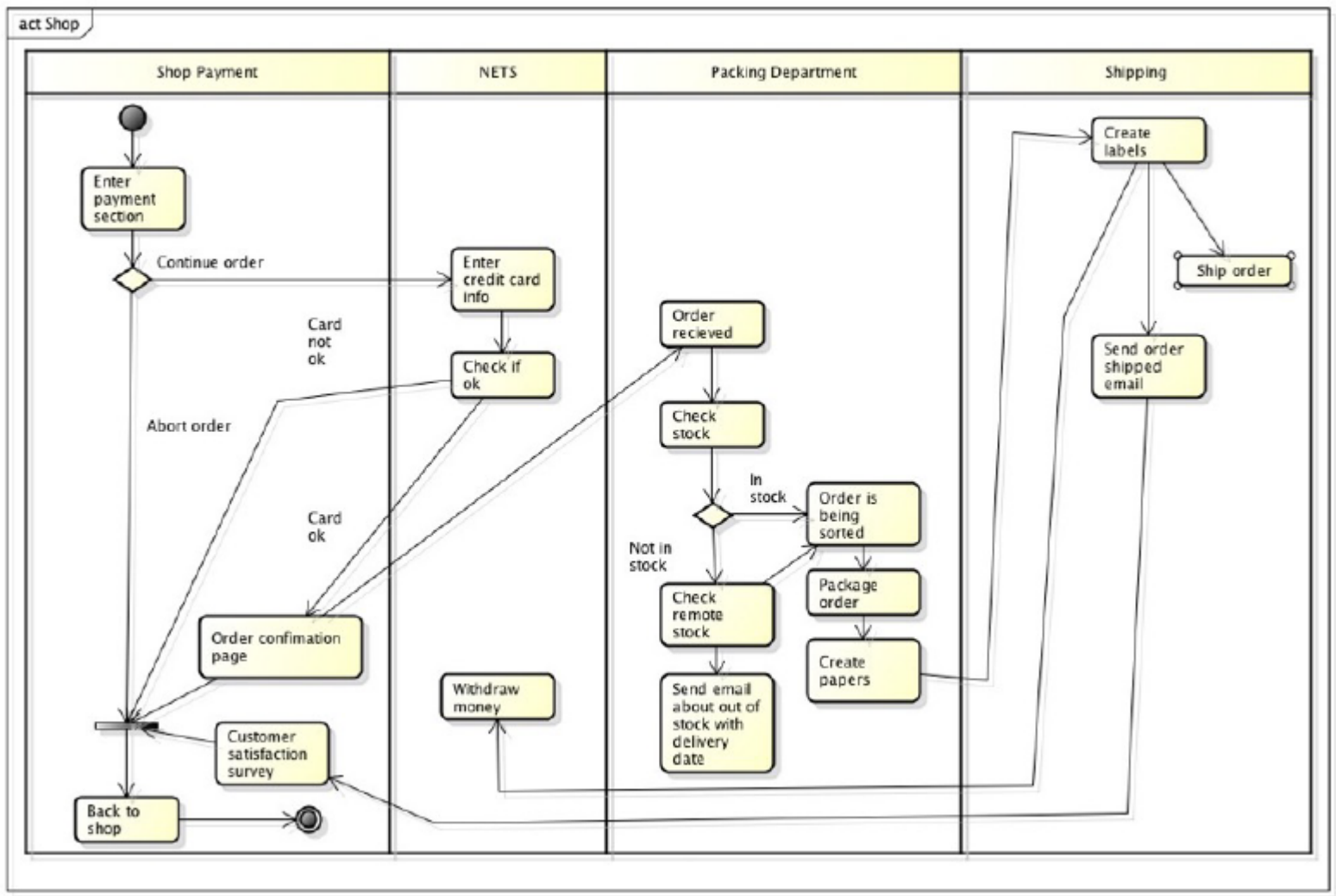
Aktivitets diagrammer

- Opdelt i opgaver af lodrette "kasser" ved siden af hinanden
- Afrundede rektangler = handlinger
- Diamanter = beslutninger
- Barer = splitter eller sammenføjede aktiviteter
- Sort cirkel = start workflow (oprindelige tilstand)
- Omkranset sort cirkel = ende af flow (endelige tilstand)

# UML

## Adfærds diagram

Aktivitets diagrammer



# UML

## Adfærds diagram

### OPGAVE

- Start Astah hvis I har lukket det
- Lav et aktivitets diagram (Activity Diagram)
- Der er en bruger, der starter et spil på sin egen PC
- Spillet tjekker om der er opdateringer
  - For at gøre dette kontakter det spil-producentens server
  - Den svarer enten ja eller nej til opdatering
  - Spillet opdaterer først eller starter
- Spillet startes
  - Brugeren spiller
- Brugeren lukker spillet ned og afslutter

# UML
## Struktur diagram

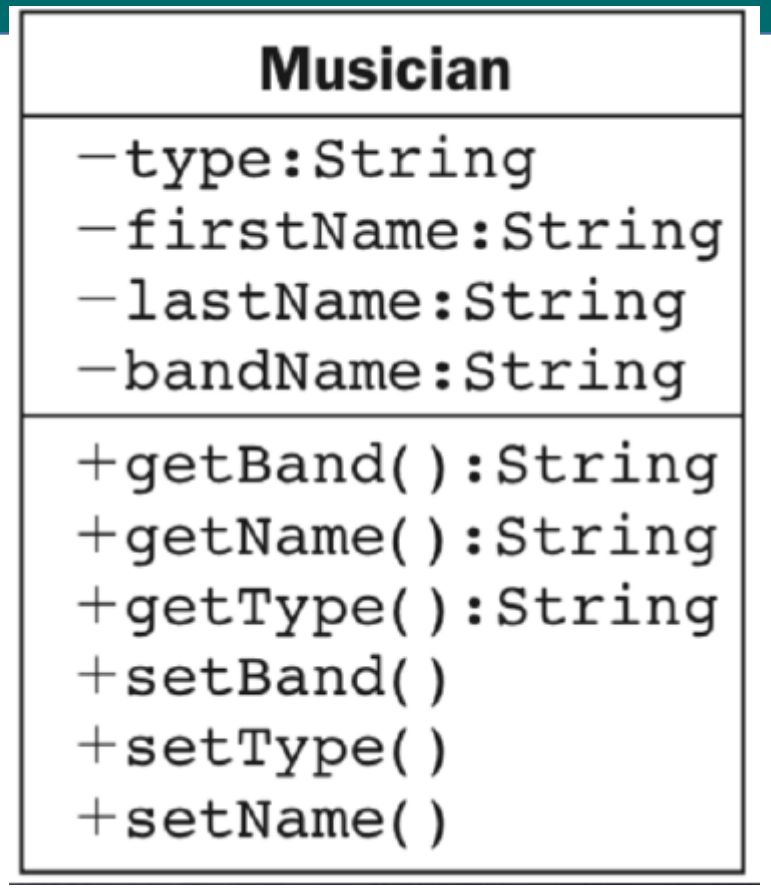
Klasse diagrammer

- En klasse repræsenterer en gruppe af ting, der har fælles tilstand og adfærd
- En klasse er repræsenteret af en rektangulær kasse opdelt i rum
  - Første rum indeholder navnet på klassen
  - Det andet er attributter
  - Det tredje er operationer
- Et klasse diagram bruges især til at udlægge variabler og

# UML
## Struktur diagram

Klasse diagrammer

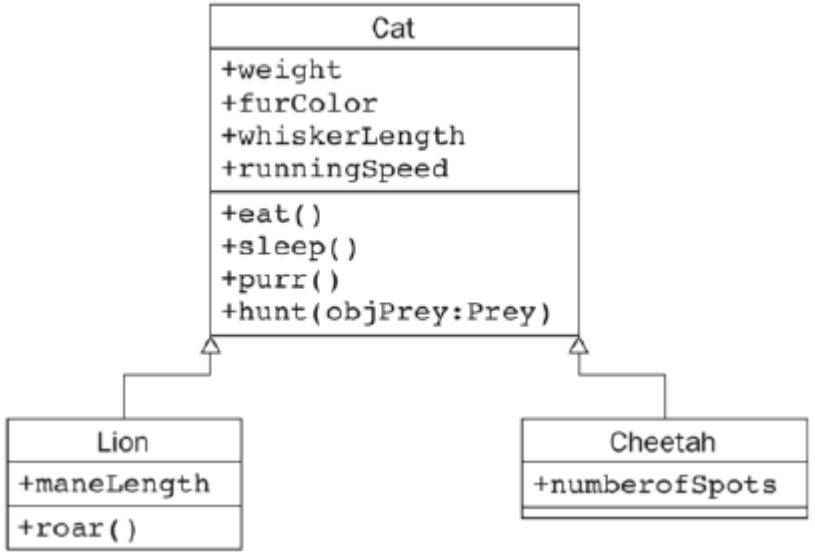
- Klasse navn
- Egenskaber
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :type
- Operationer ()
  - (+ / - / #)
    - offentlig / privat / beskyttet
  - :return



# UML
## Struktur diagram

Klasse diagrammer

- Nedarvning (inheritance)
  - En klasse arver fra sin forælder-klasse
  - Overskriver værdier I forælderen
  - Bibeholde forælderen



# UML
## Struktur diagram

Klasse diagrammer og programmering

- Klasse diagrammer er de mest populære UML diagrammer brugt i forbindelse med konstruktion af software-applikationer. Så det er meget vigtigt at lære proceduren for klassediagram tegning
- Klassediagrammer danner grundlag for overvejelser medens man tegner dem
- Et klassediagram er dybest set en grafisk repræsentation af det statiske billede af systemet og repræsenterer forskellige aspekter af anvendelsen. Så en samling af klassediagrammer repræsenterer hele systemet.
- Generelt er UML diagrammer er ikke direkte forbundet med eventuelle objektorienterede programmeringssprog men klassediagrammet er en undtagelse
  - Klasse diagram viser tydeligt forbindelsen til objektorienterede sprog som Java, C ++ osv.
  - Vi kommer senere i forløbet ind på hvordan man direkte kan bruge et klassediagram som grundlag for objekter i C#

# UML
## Struktur diagram

OPGAVE

- Start Astah (hvis du ikke har det kørende allerede)
- Lav et klasse diagram (Class Diagram)
- Lav klassen Bil
  - Den skal have egenskaberne
    - Producent (bogstav værdi)
    - Model (bogstav værdi)
    - Vægt (tal værdi)
    - Motor (bogstav værdi)
    - Hjul (tal værdi)
  - Den skal have operationerne
    - Køre (hastighed) (tal værdi)

# UML
## Struktur diagram

OPGAVE

- Lav klassen Personbil
  - Den skal være barn af Bil klassen
  - Den skal føje egenskaben ”Døre (tal værdi)” til
  - Den skal føje egenskaben ”Antal pladser (tal værdi” til
  - Den skal føje egenskaben ”Bagagerum (ja/nej (boolean) værdi)” til
  - Den skal føje egenskaben ”Trailertræk (ja/nej (boolean) værdi)” til
- Lav klassen Varebil
  - Den skal være barn af Bil klassen
  - Den skal føje egenskaben ”Skydedør i siden (ja/nej (boolean) værdi)” til
  - Den skal føje egenskaben ”Trailertræk (ja/nej (boolean) værdi)” til

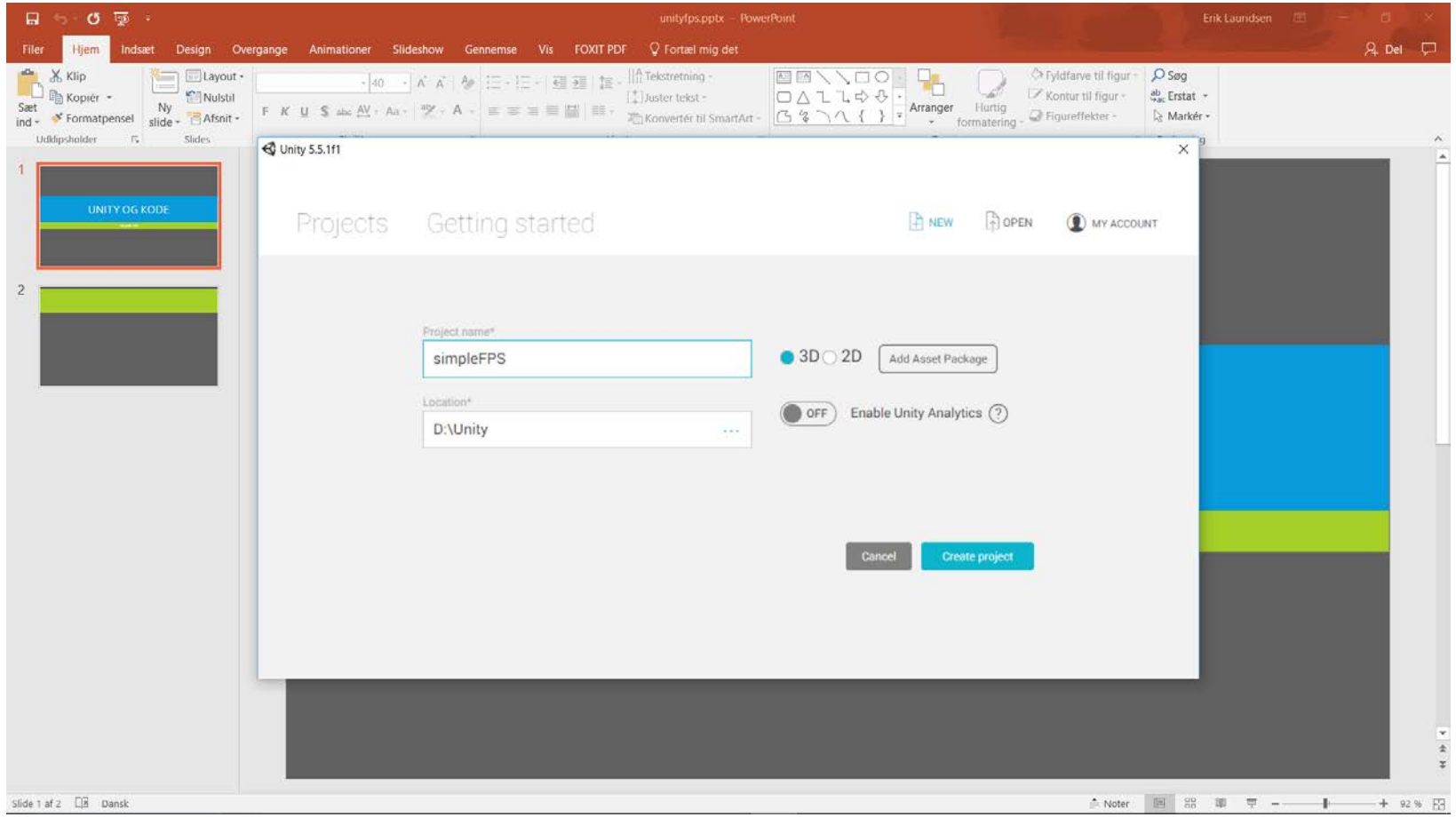
# Værktøjer til projektplanlægning

# Introduktion til Objekt Orienteret Programmering

Animal
A
B
C
Mammal
D
E
A
B
C
Cat
F
G
D
E
A
B
C

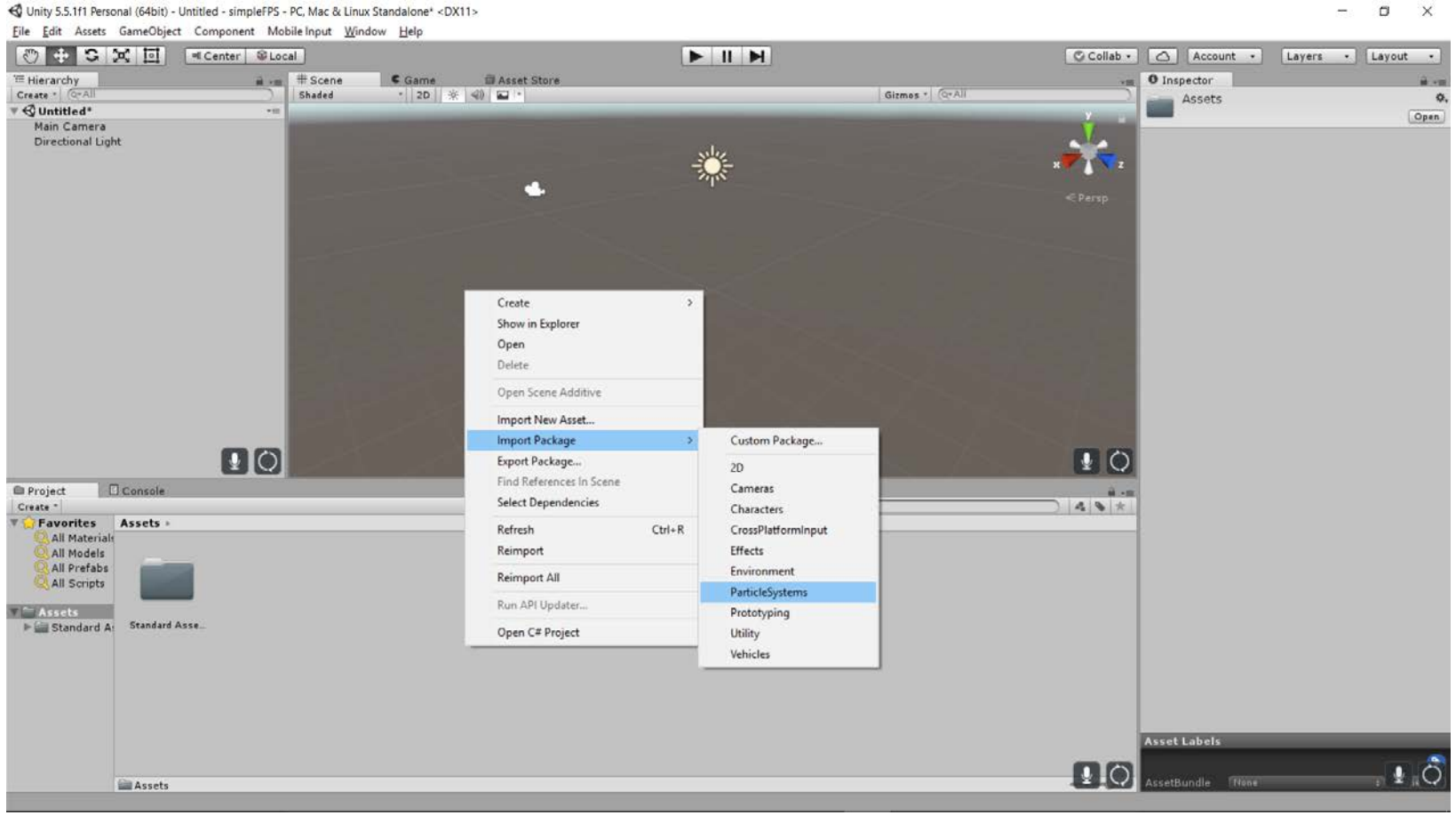
# Simpelt FPS

# Lav et nyt unity 3D projekt



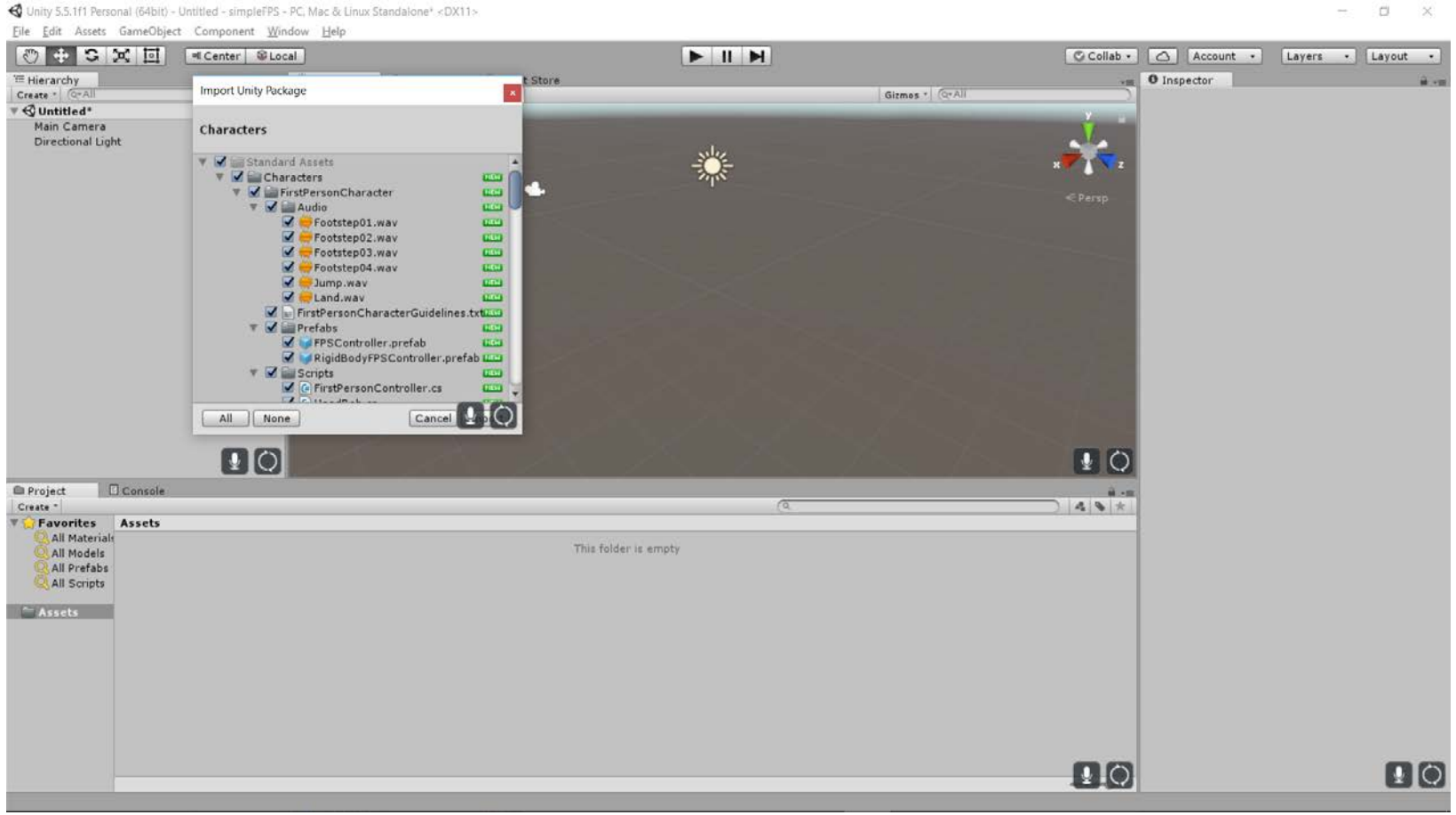
# Importer particlesystems pakken

standard indstillinger

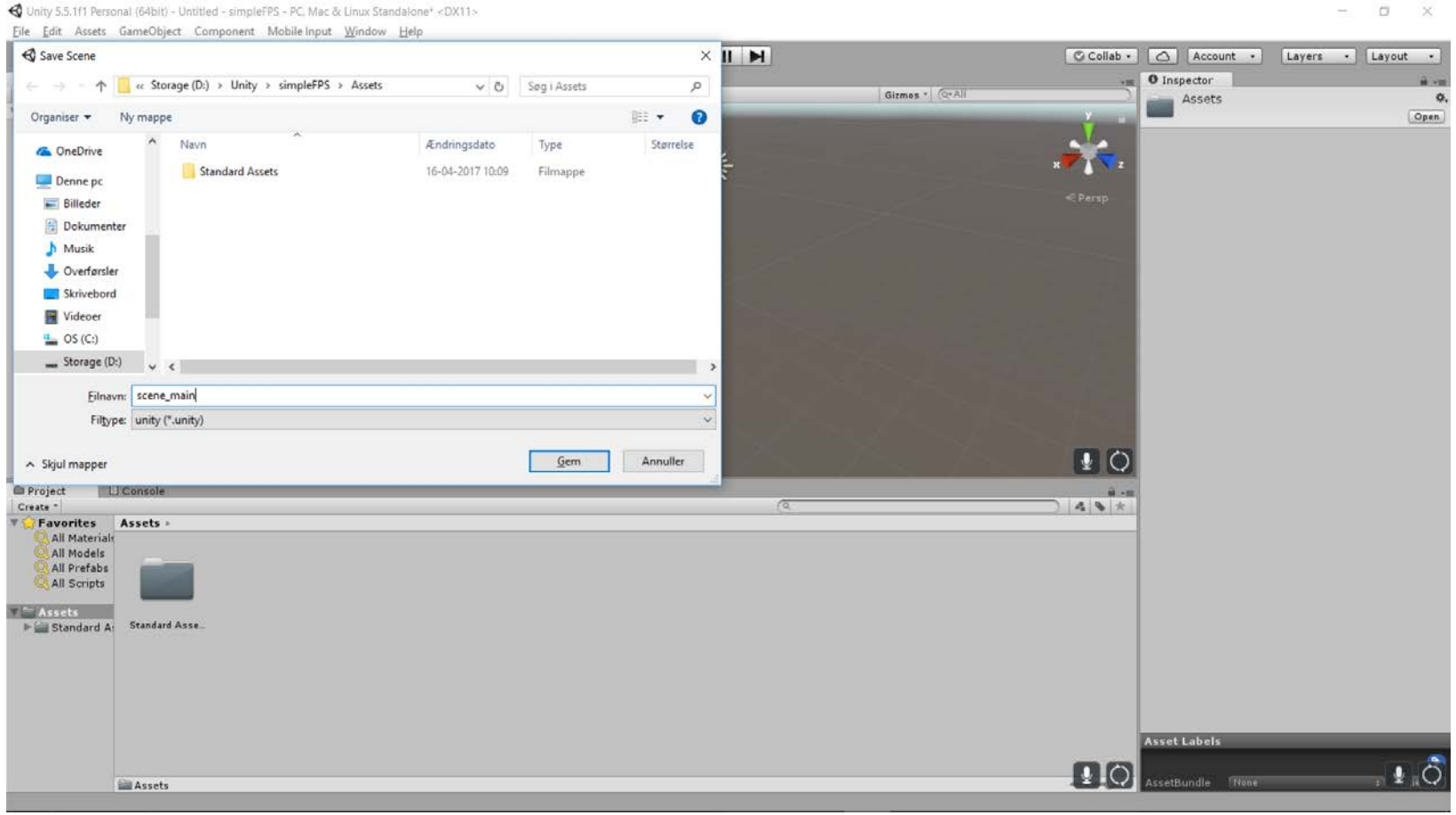


# Importer characters pakken

standard indstillinger

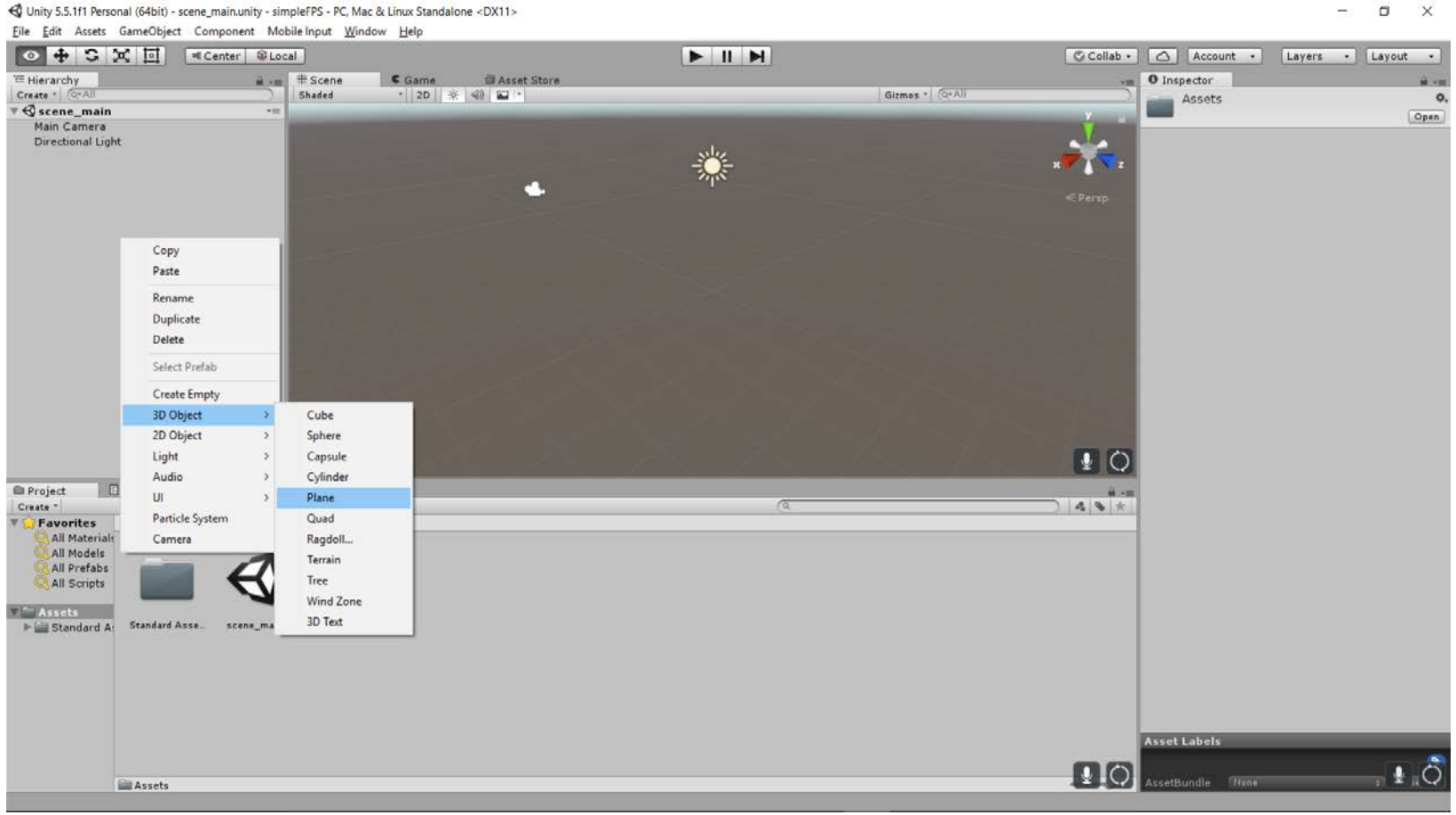


# Gem scenen som scene_main

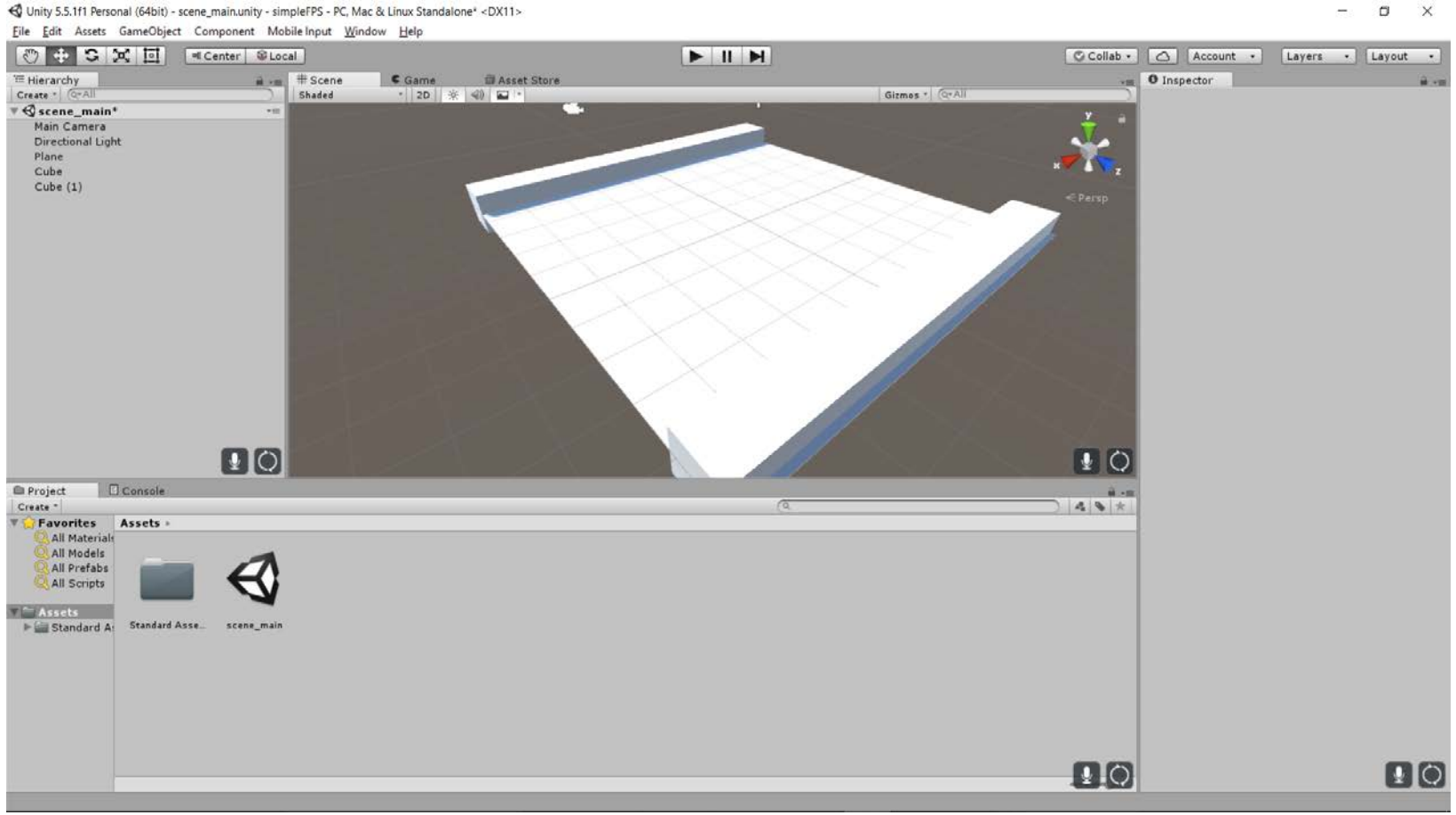


# Føj et plane til

spilleren skal have noget at stå på

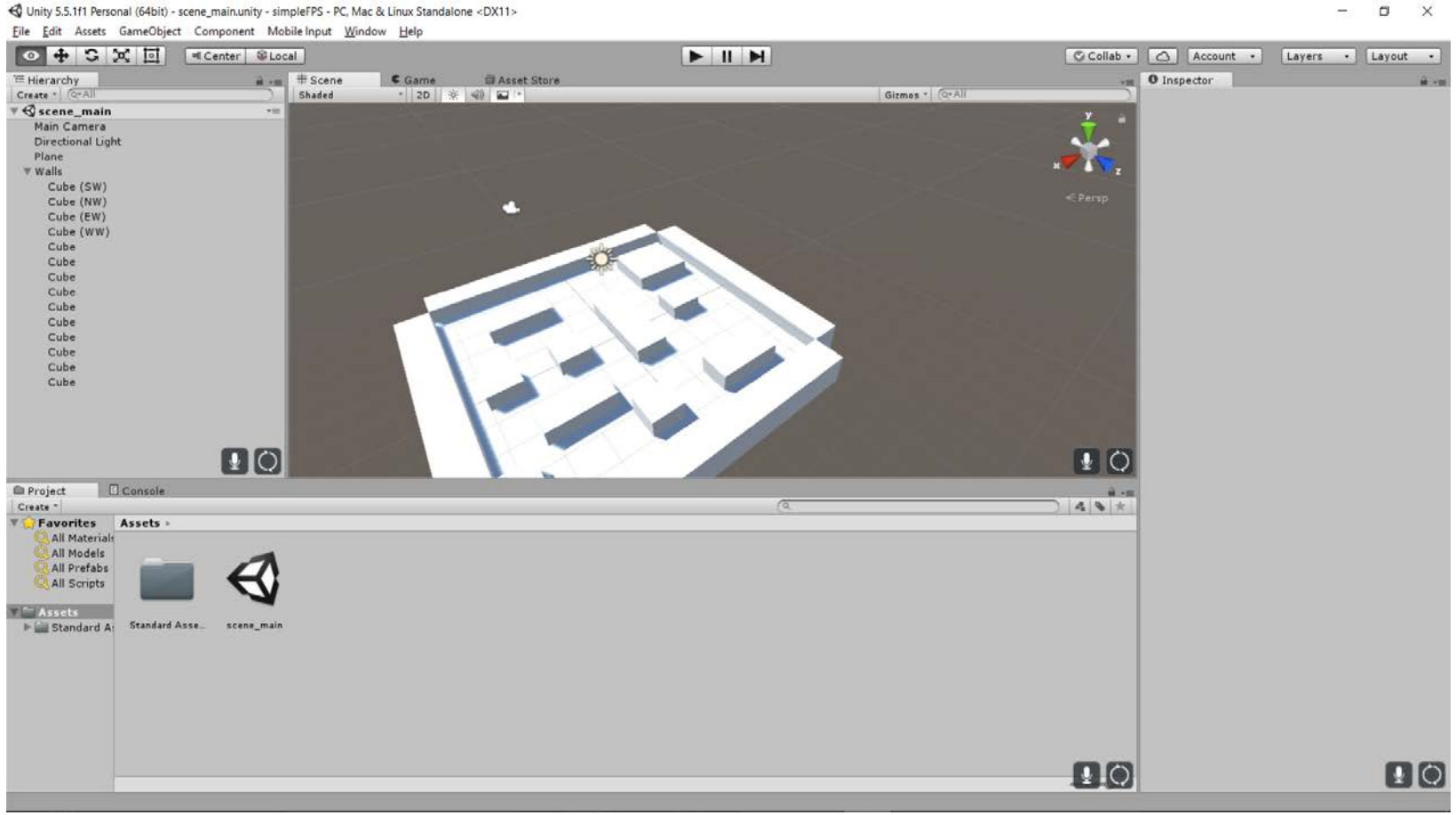


# Begynd at føj cubes til så vi får vægge

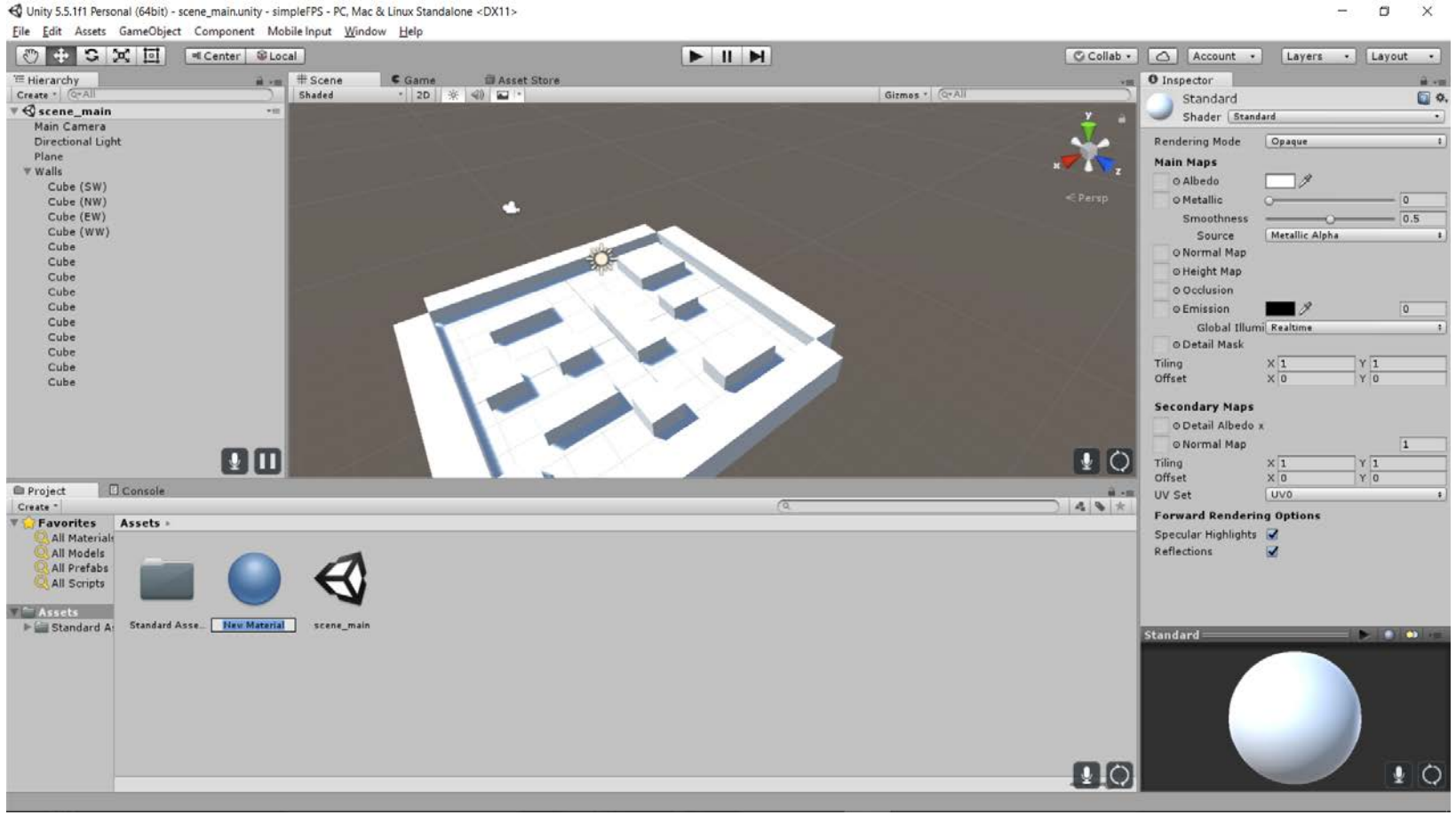


# … også inde på ”banen”

## JEG HAR OGSÅ ORGANISERET DEM EN SMULE I EN GRUPPE FOR OVERSKUELIGHED

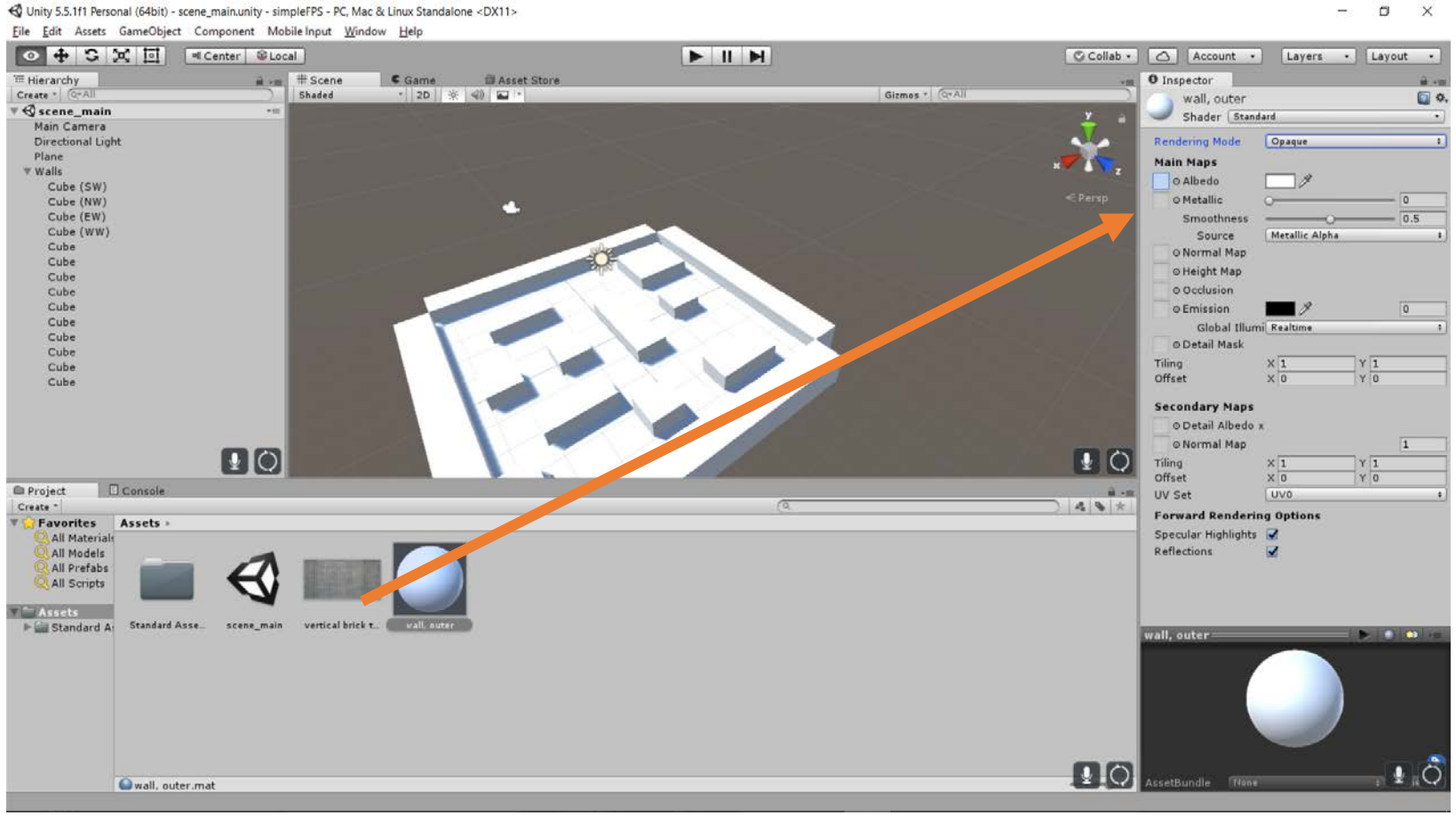


# Lav et materiale til ydervæggene

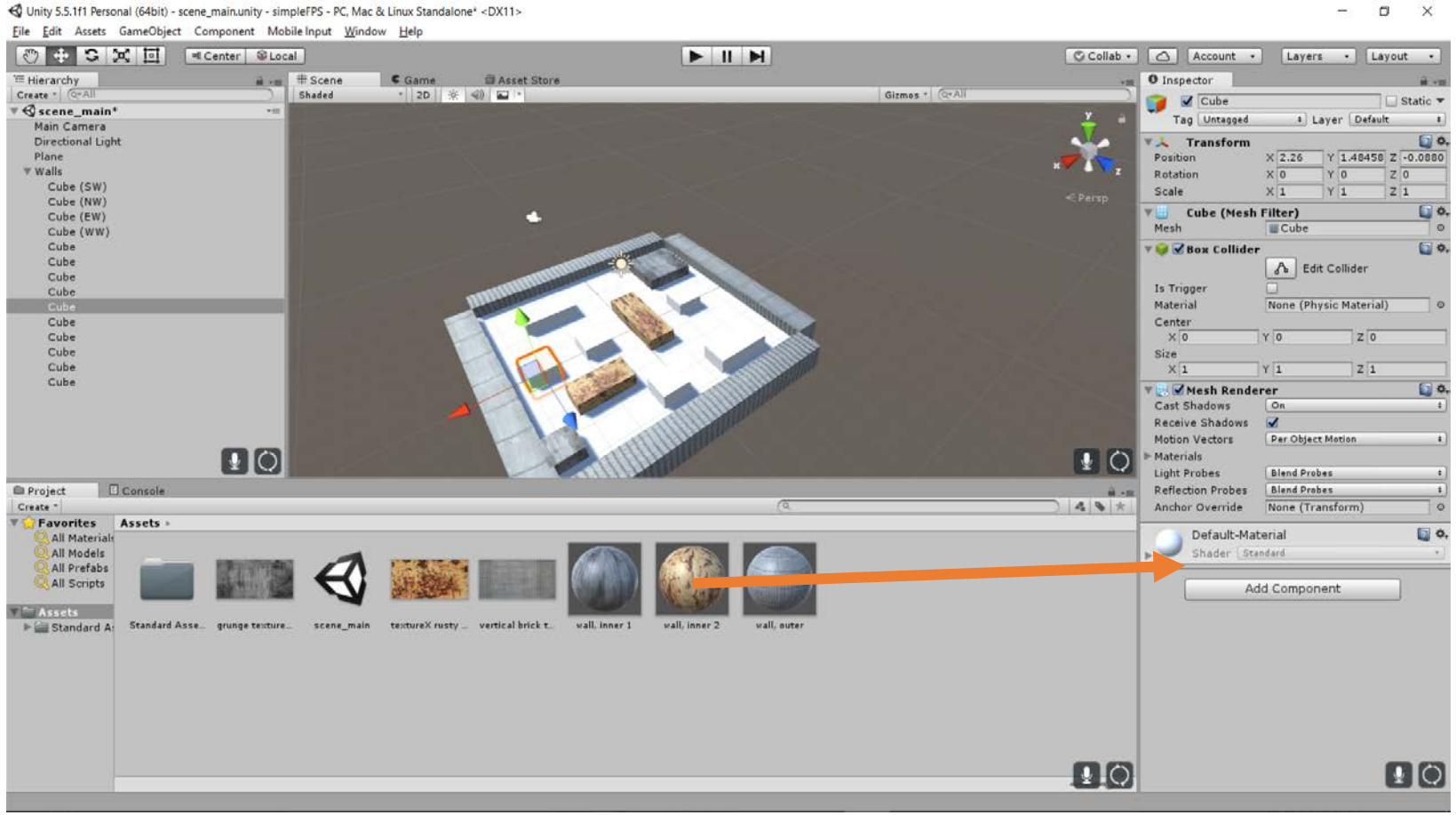


# Importer et billede og gør det til materialets tekstur

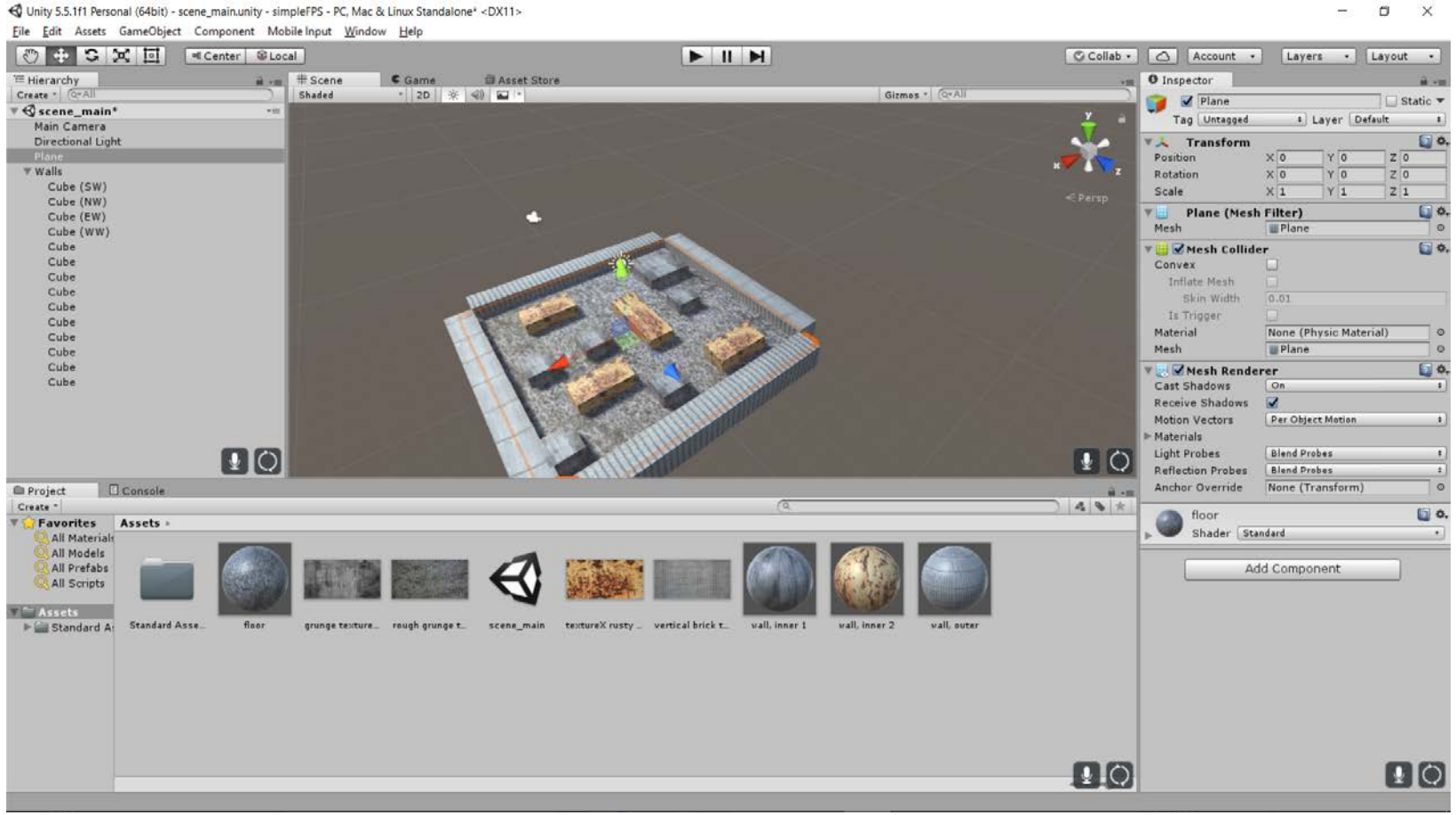
Kan bl.a. hentes gratis på http://www.texturex.com/



# Vi føjer materialer og teksturer til både indre og ydre vægge



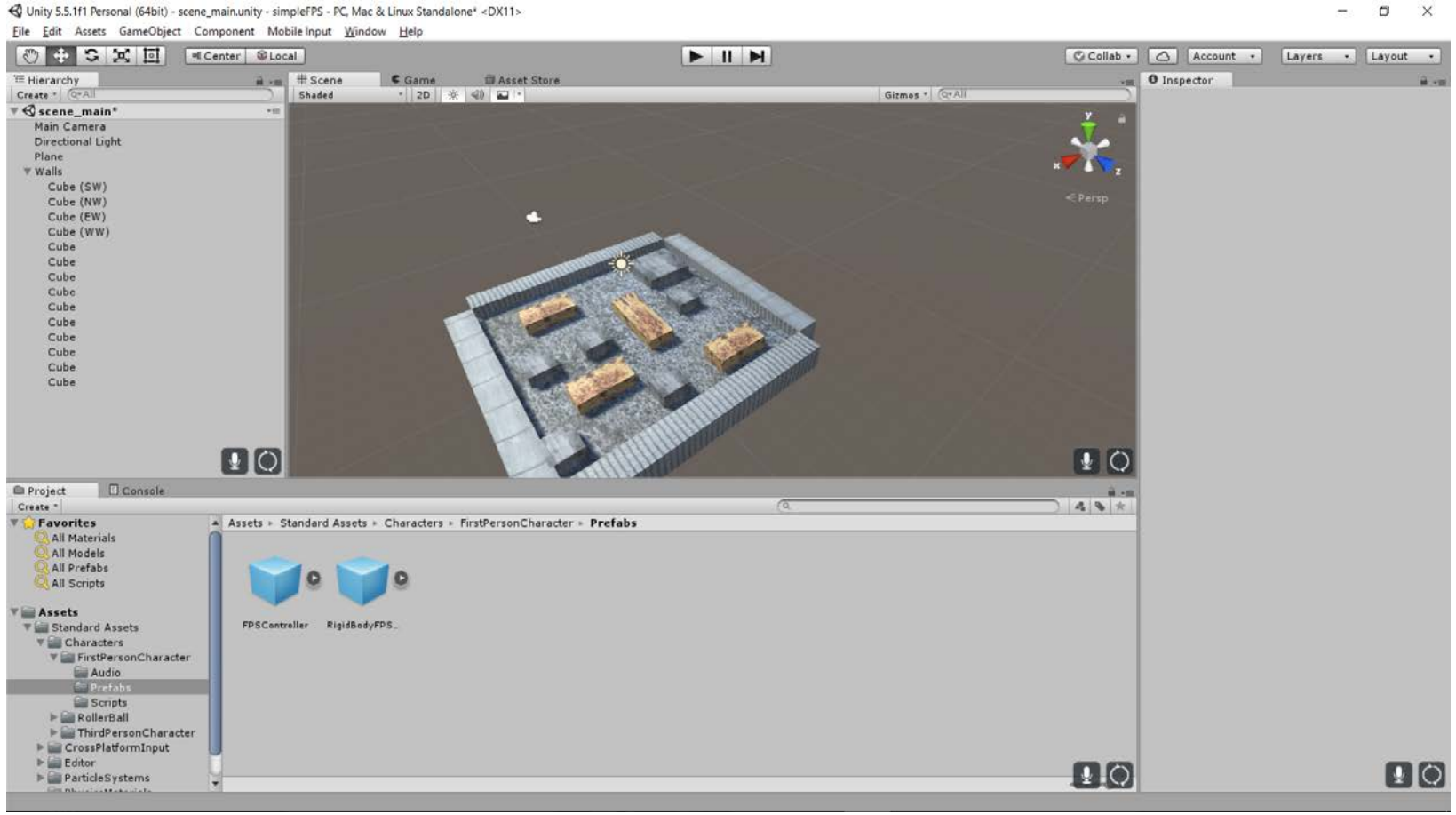
# Og til sidst til gulvet



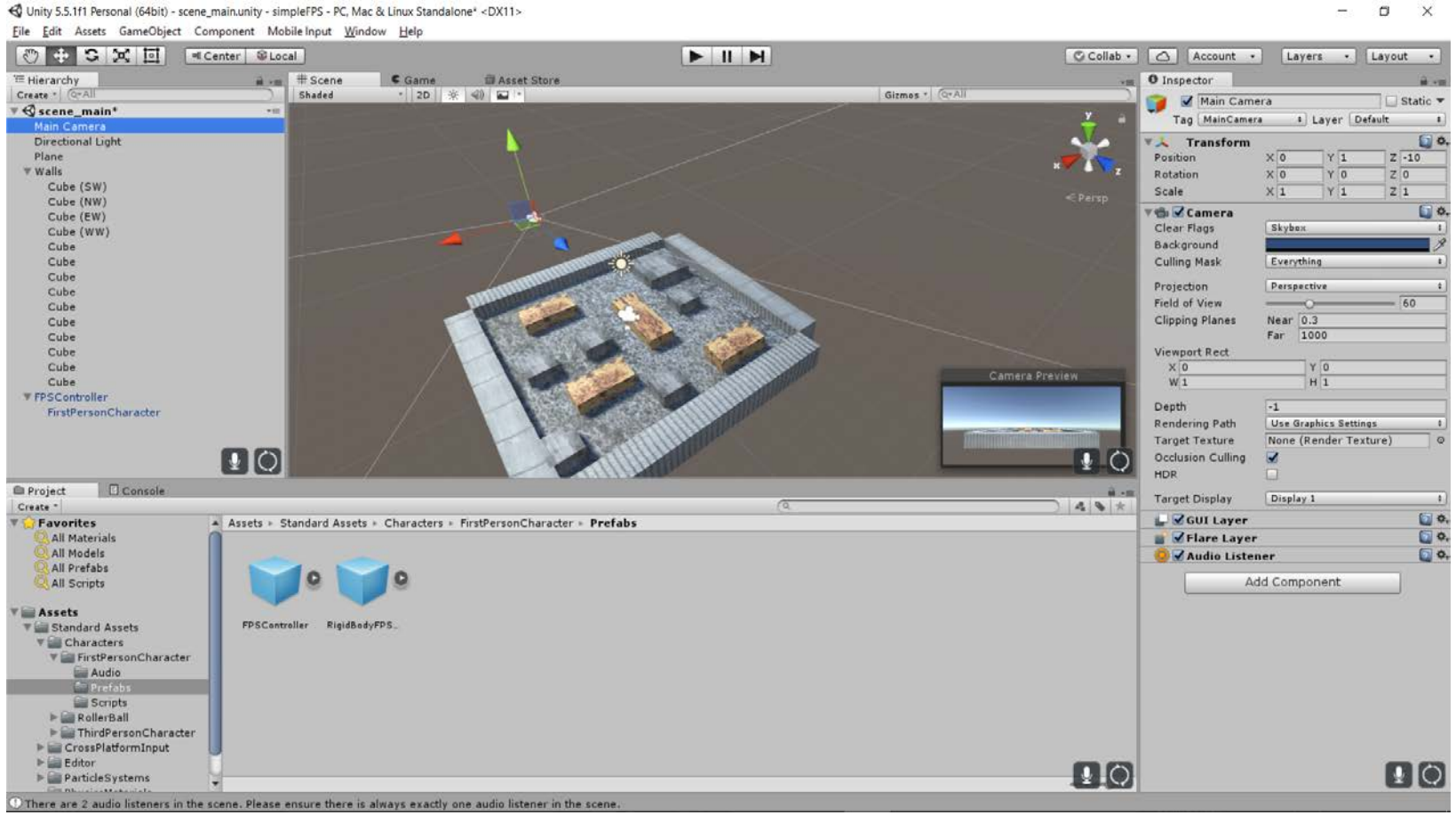
# Føj en firstperson controller til

## Standard assets > Characters > FirstPersoncharacter > Prefabs

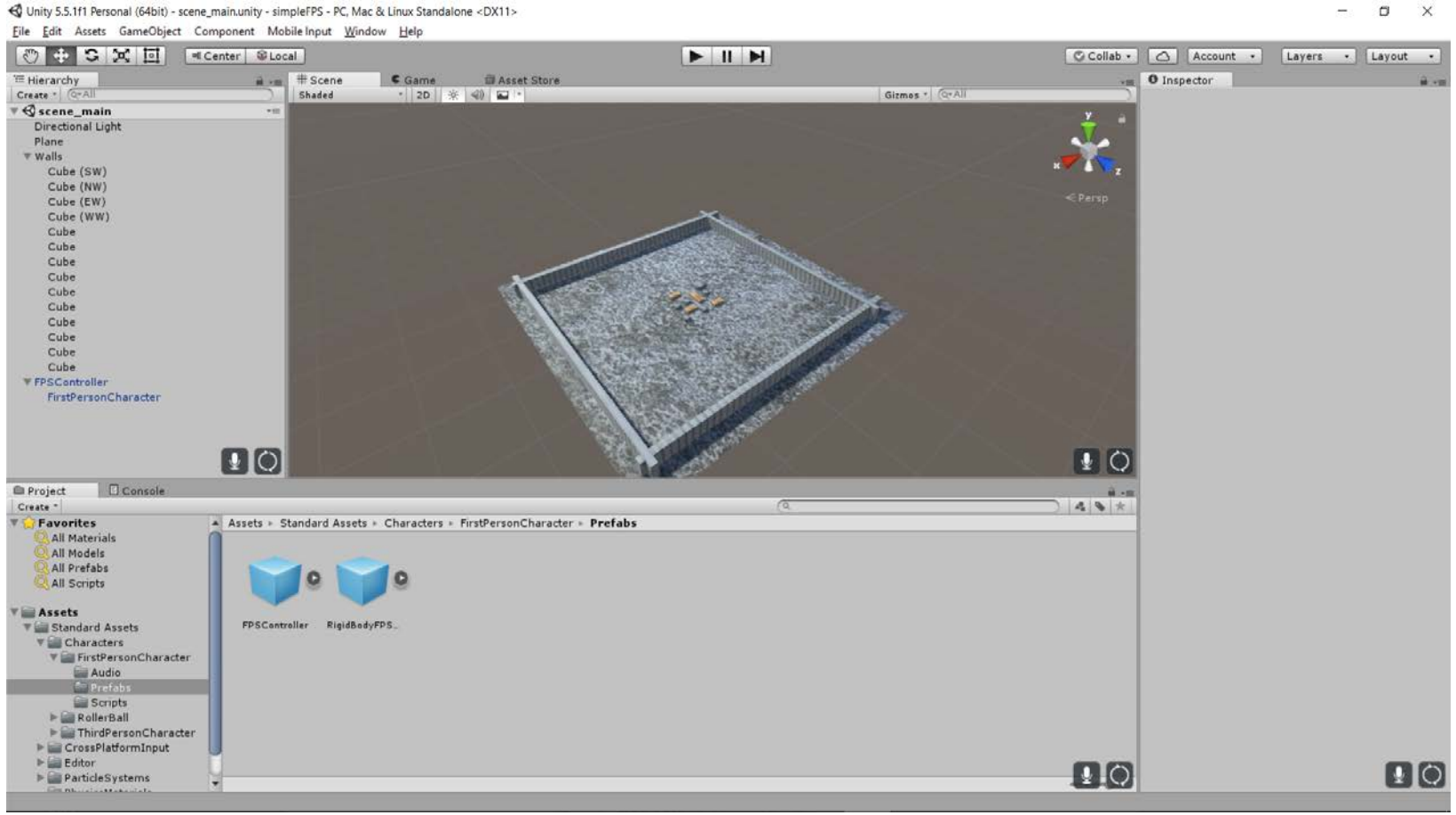
Den giver os WASD bevægelse og first person kamera på en let måde



# Siden fpscontroller kommer med et kamera kan vi slette det automatiske main camera, da det ikke bruges

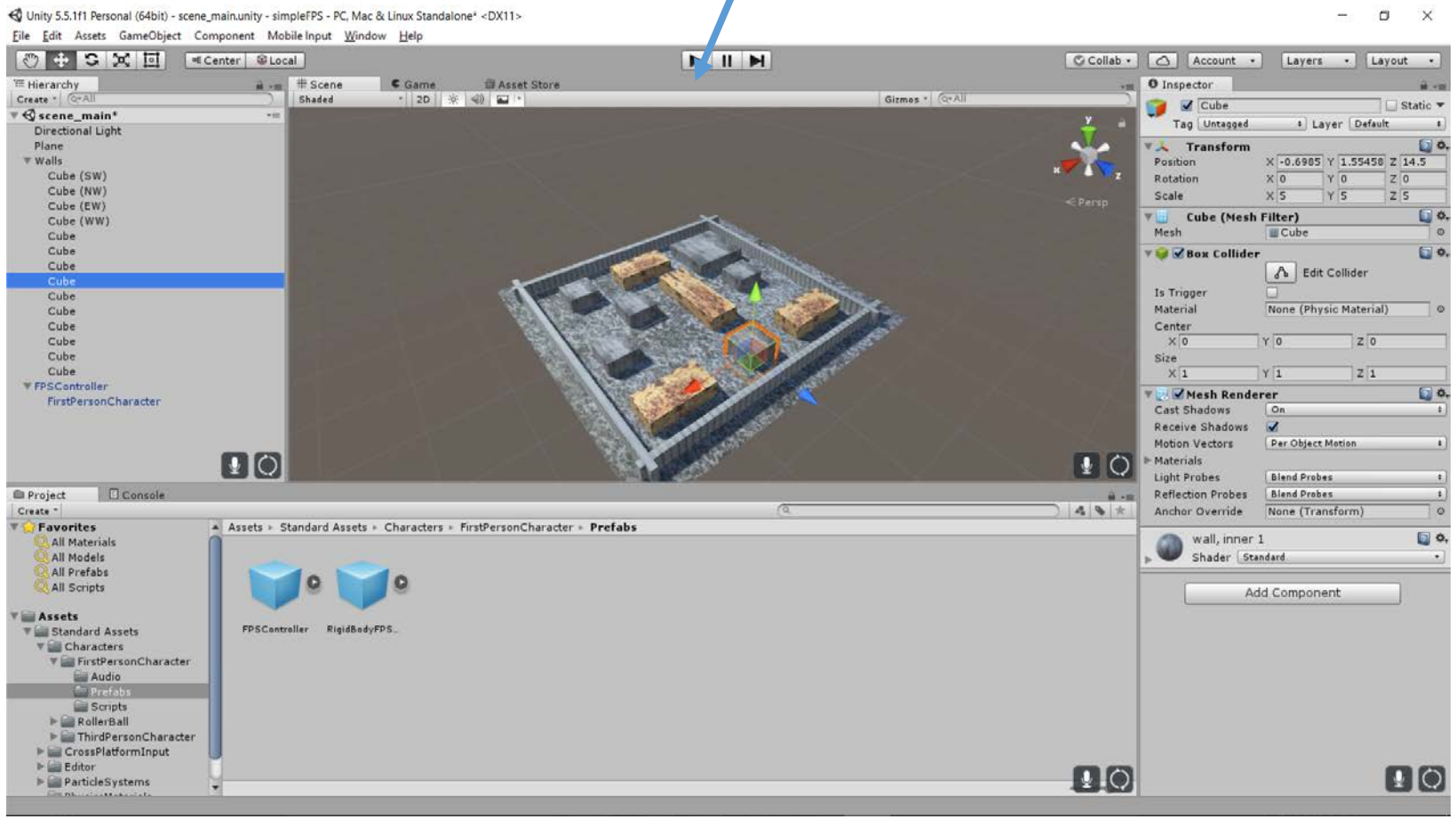


# Hvis I har brugt de samme tal som mig skal vi lige gøre alting større før det begynder at virke interessant

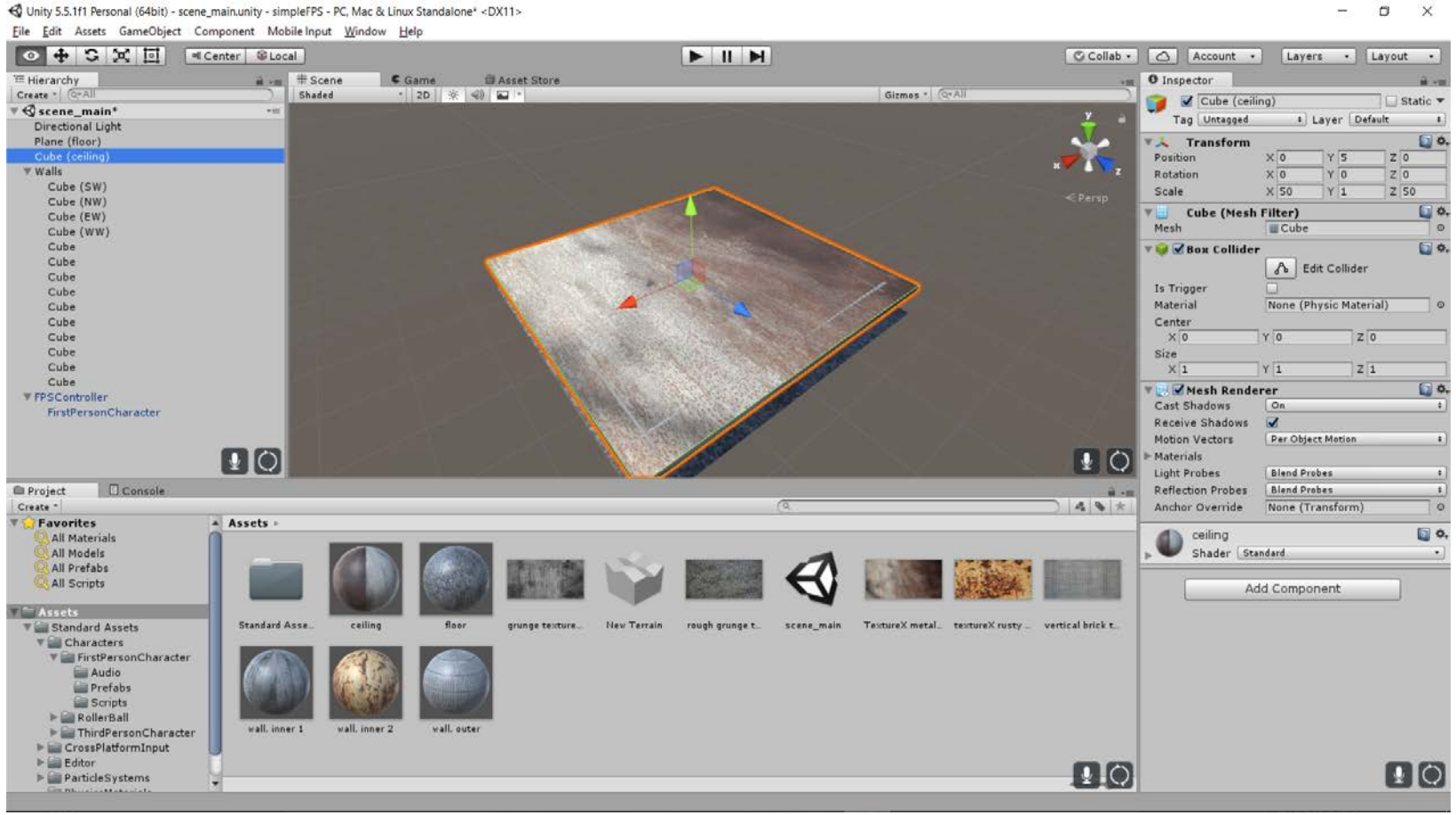


# Sådan!

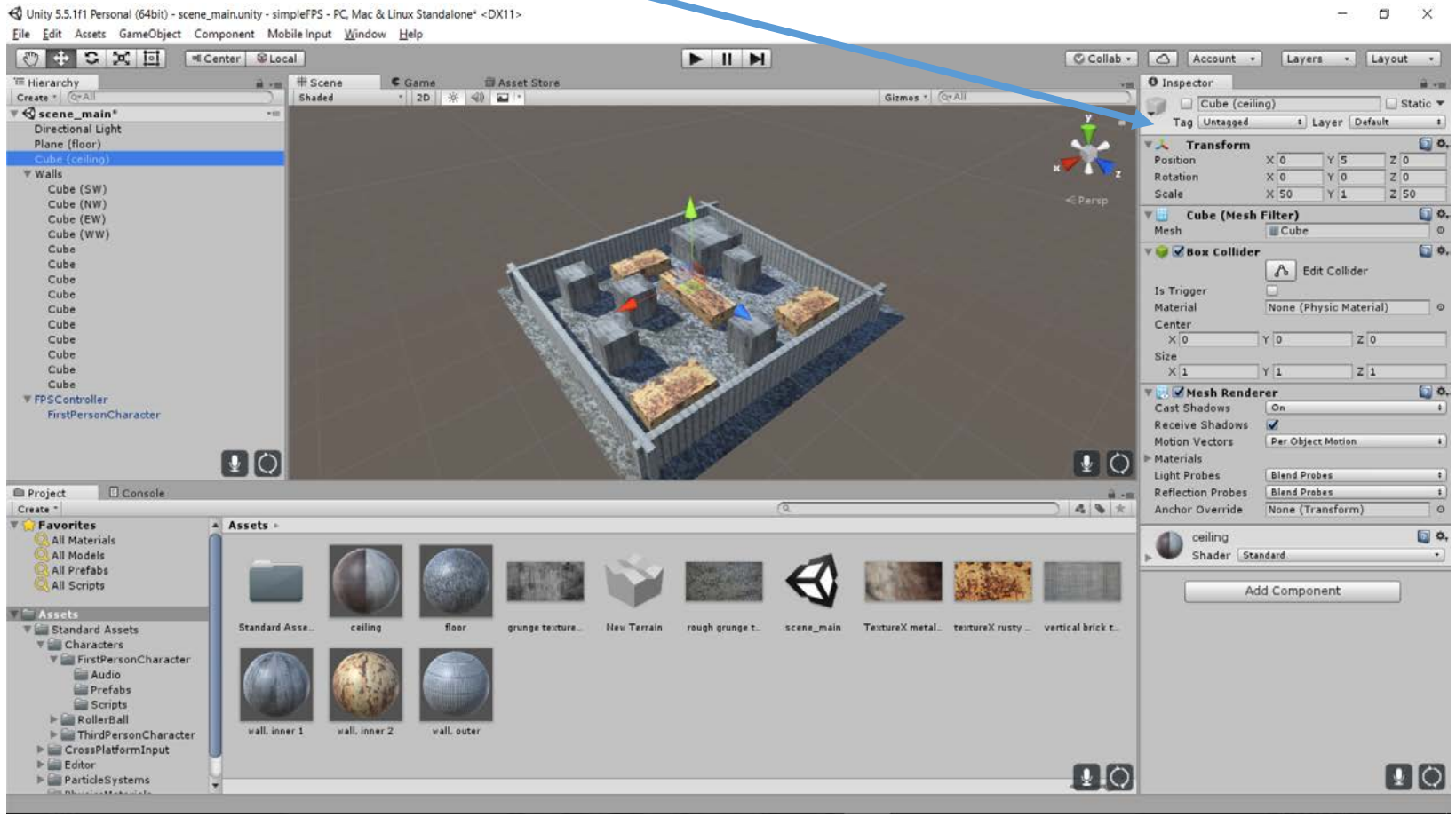
Husk hele tiden at afprøve i preview mode for at tjekke at dine ideer virker i praksis



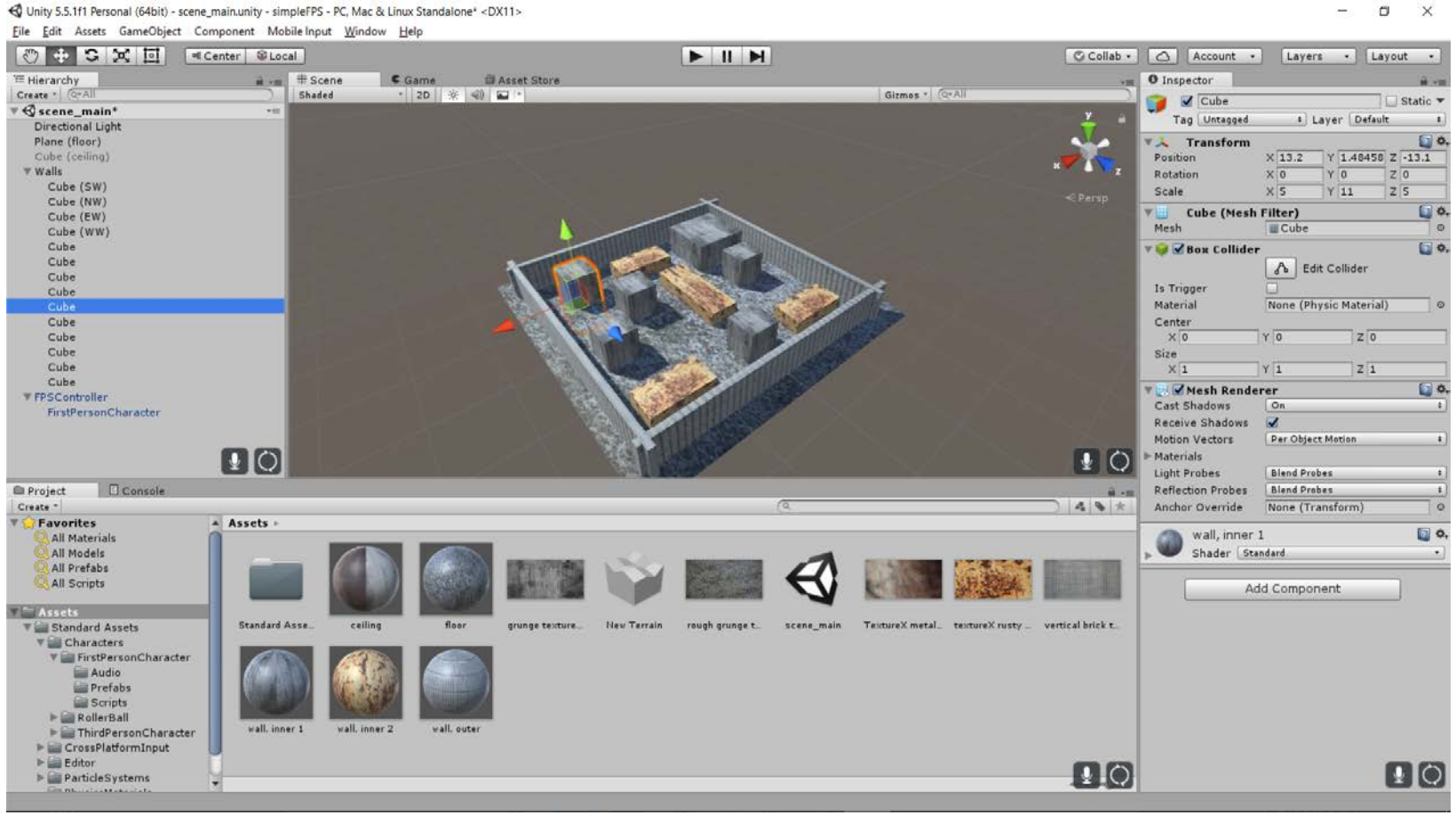
# Vi føjer et loft til så vi får en mere klaustrofobisk spille oplevelse



# Husk at du kan slå objekter fra så de bliver usynlige så man kan se det der ligger under

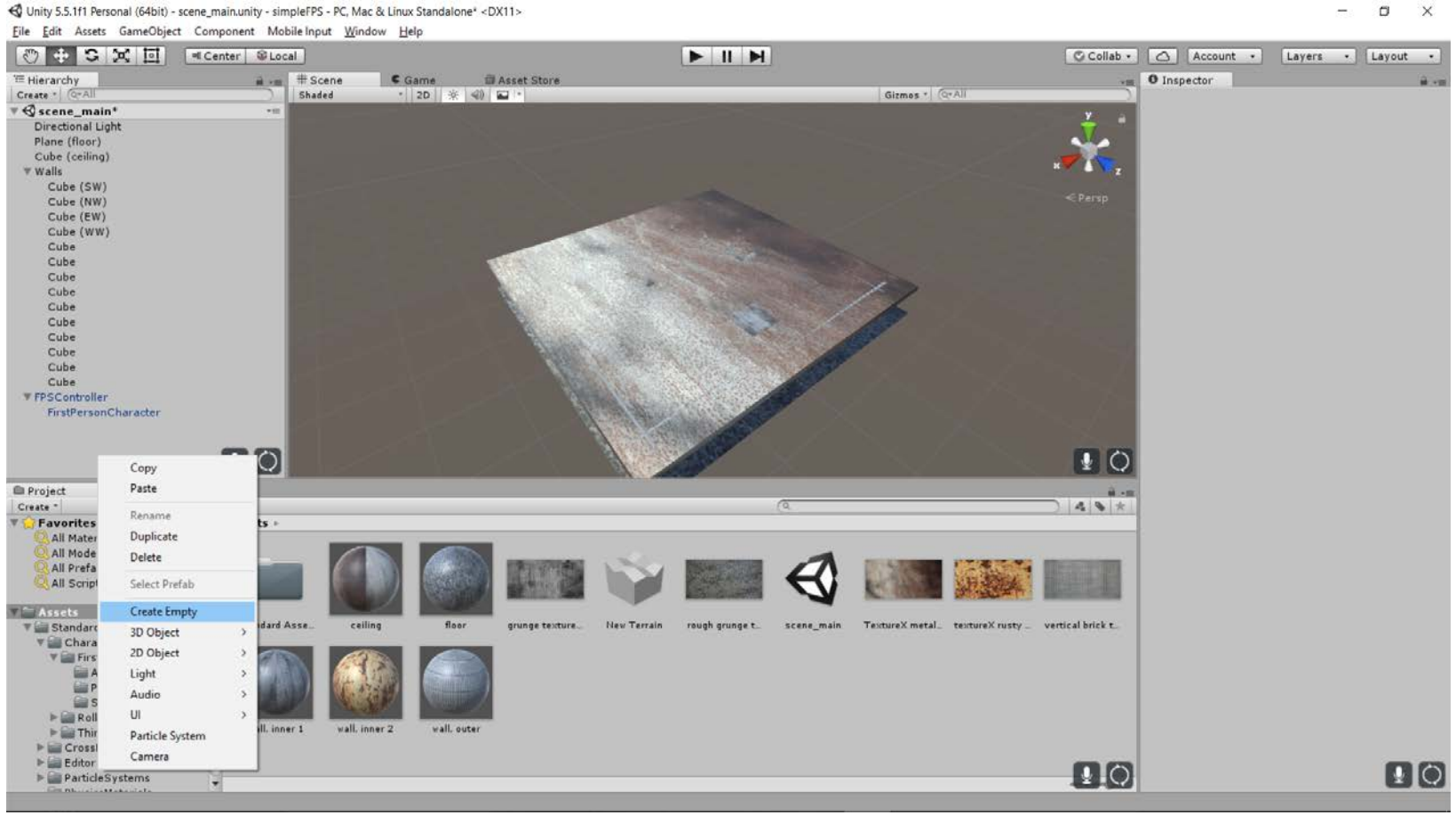


# Vi gør de omkringliggende vægge og nogle af de indre højere nu vi er igang

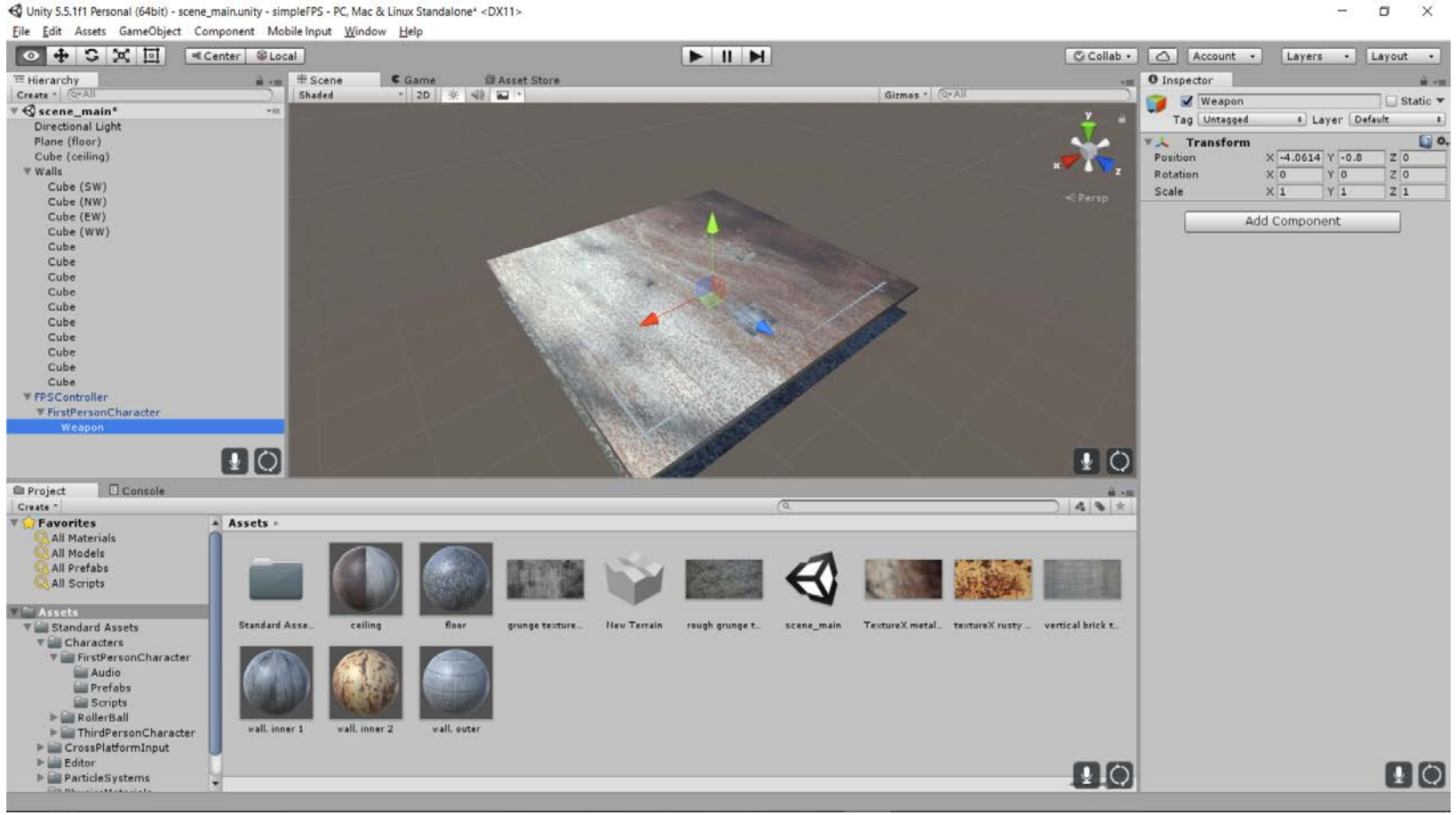


# Vi lægger ud med at lave et våben af unitys primitives

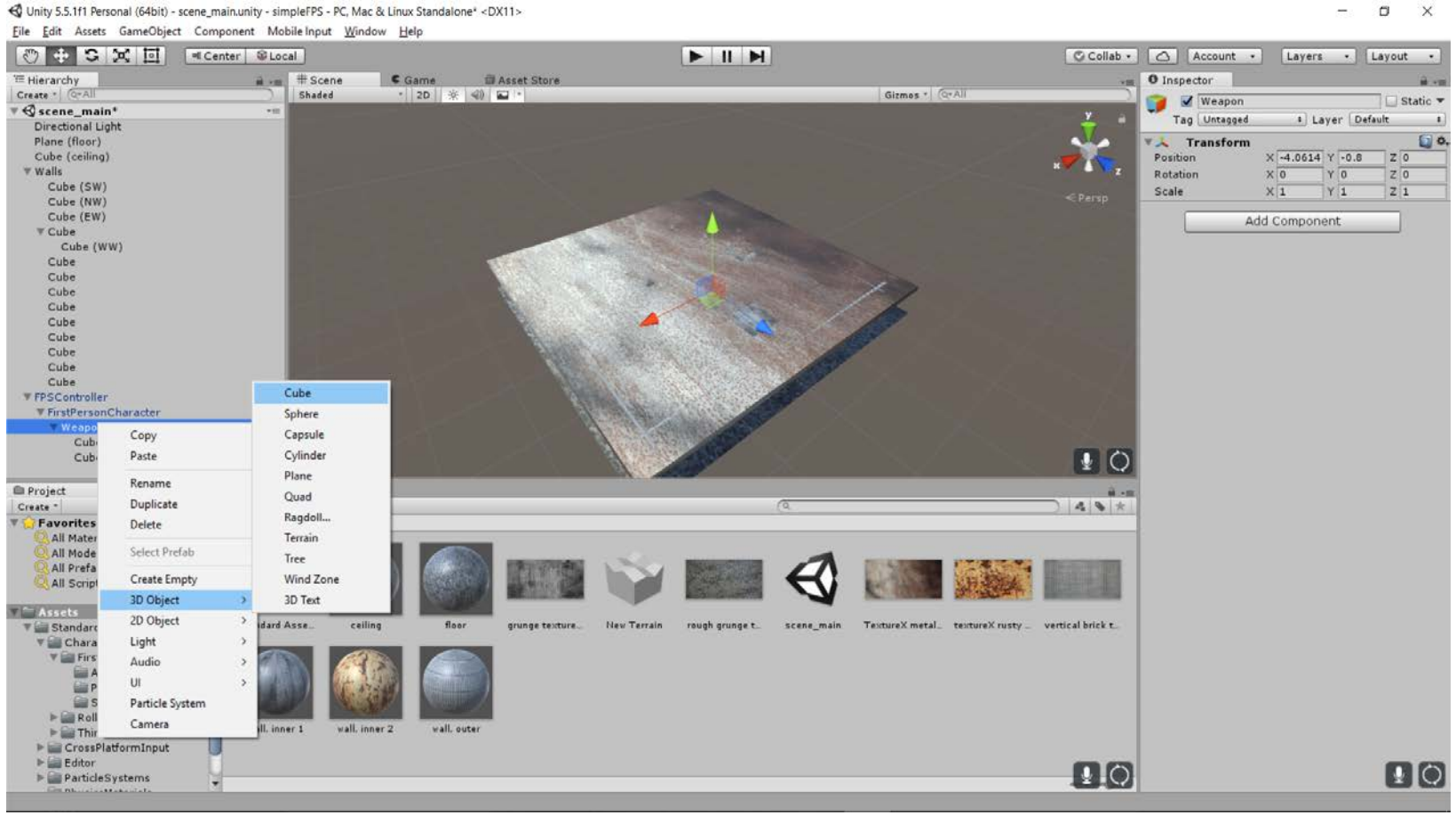
## Vi lægger ud med et tomt gameobject som forælder til delene



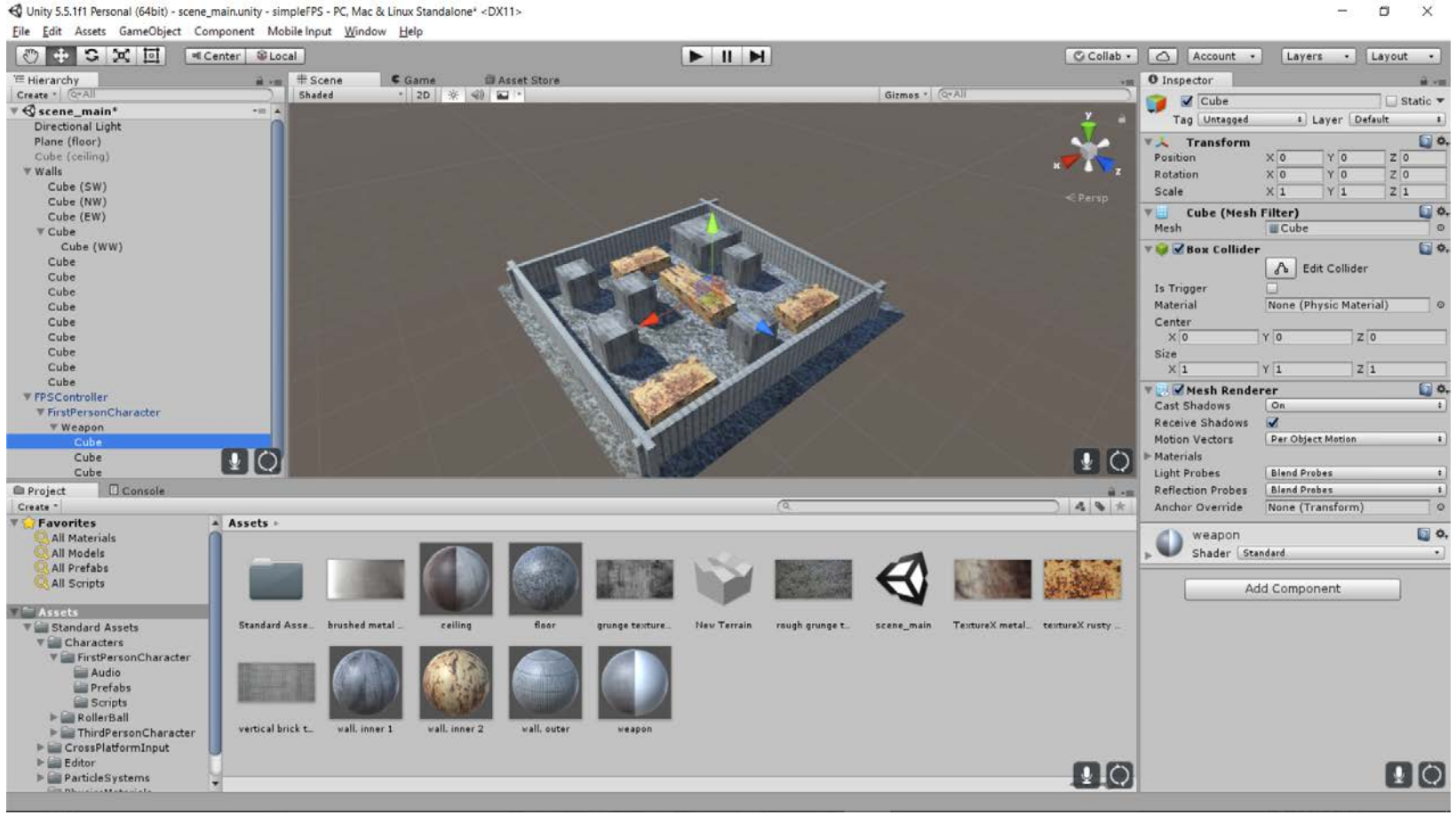
# Siden det skal følge first person gør vi det til et barn deraf gennem hierarkiet



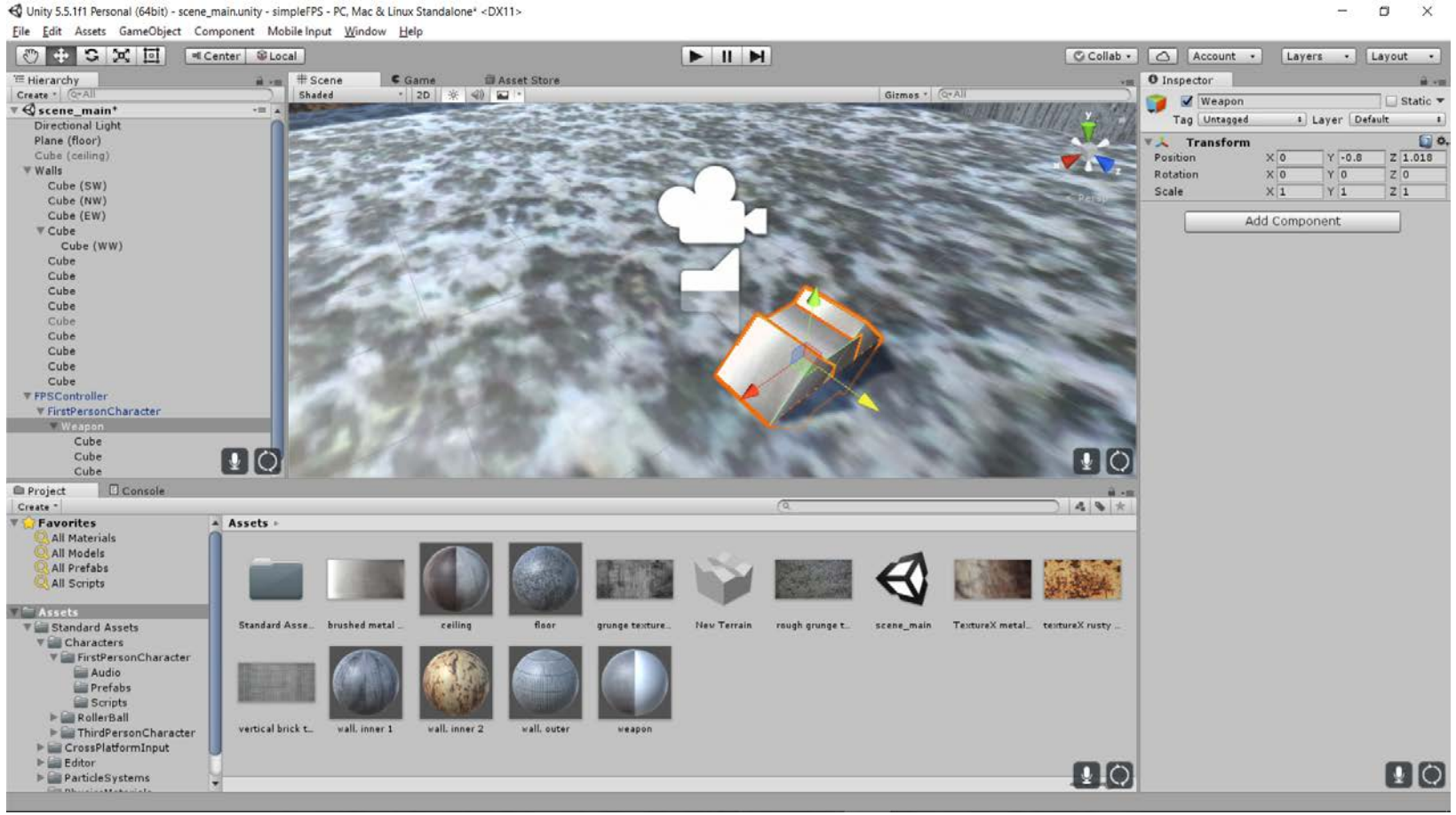
# Vi laver tre cubes der skal danne våbnet



# Vi laver et materiale til våbnet ligesom vi gjorde med væggene

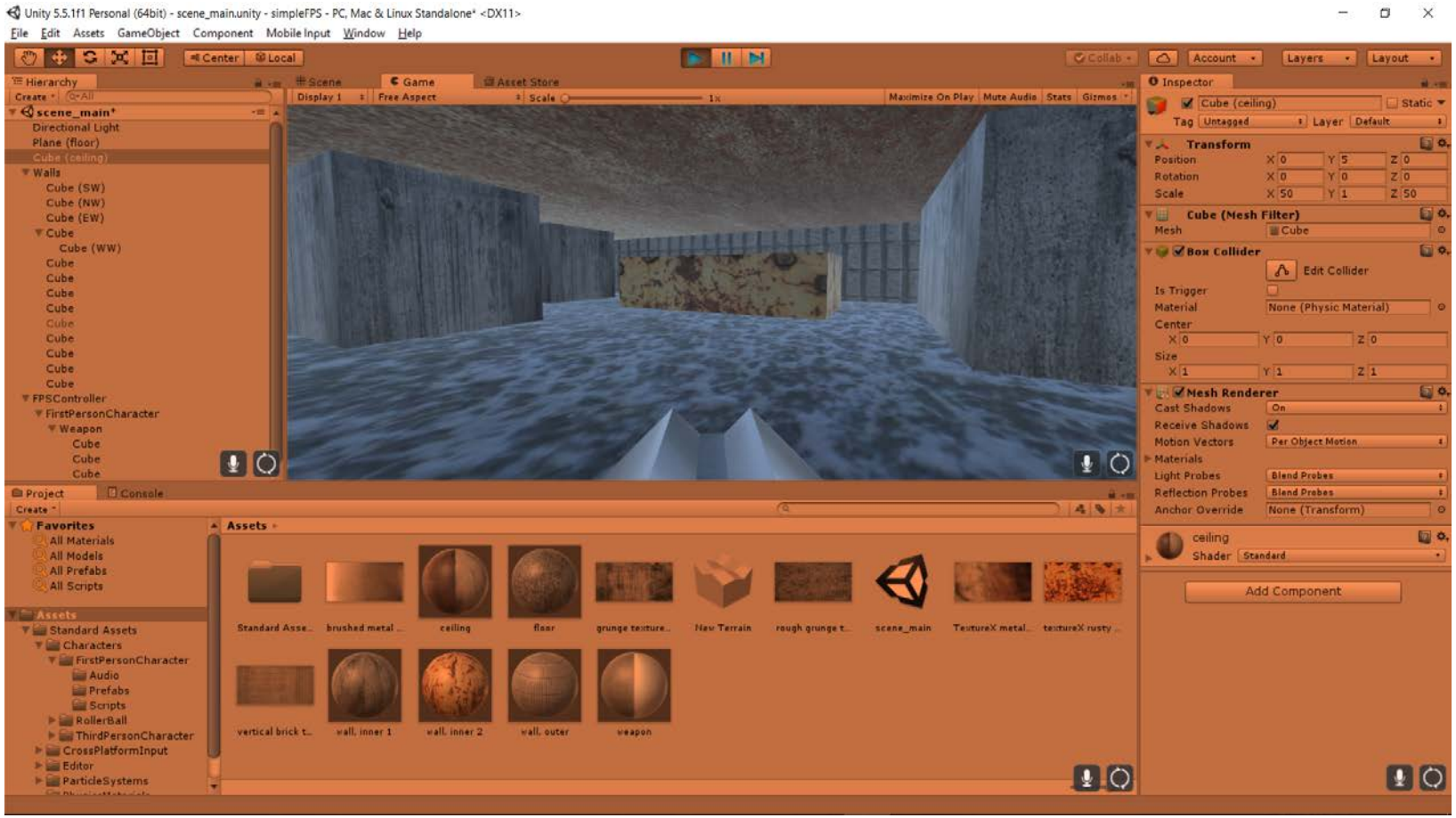


# Ryk rundt på våben delene indtil de passer med kameraet



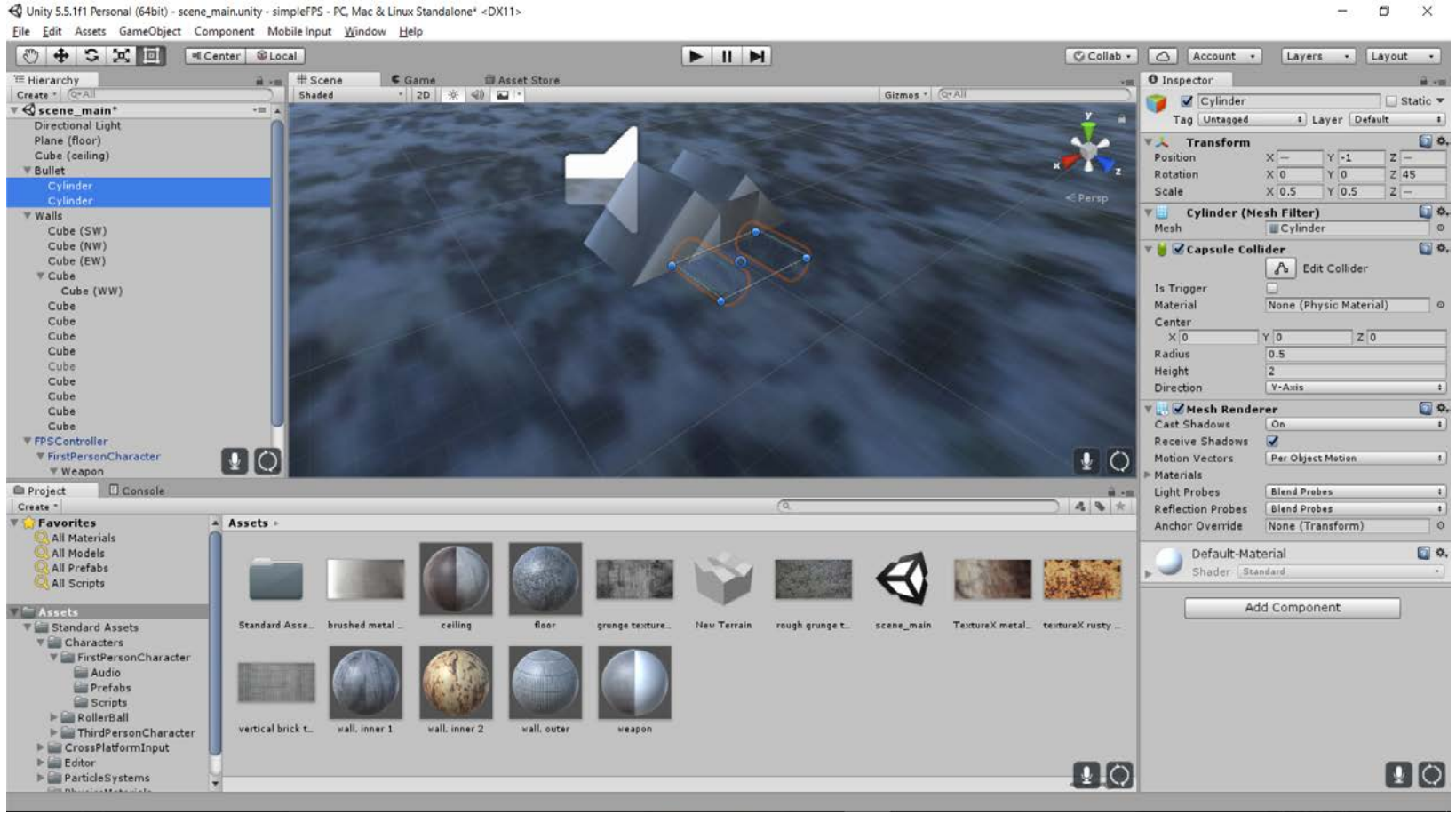
# Sådan!

## Bemærk hvordan de følger med rundt når spilleren går



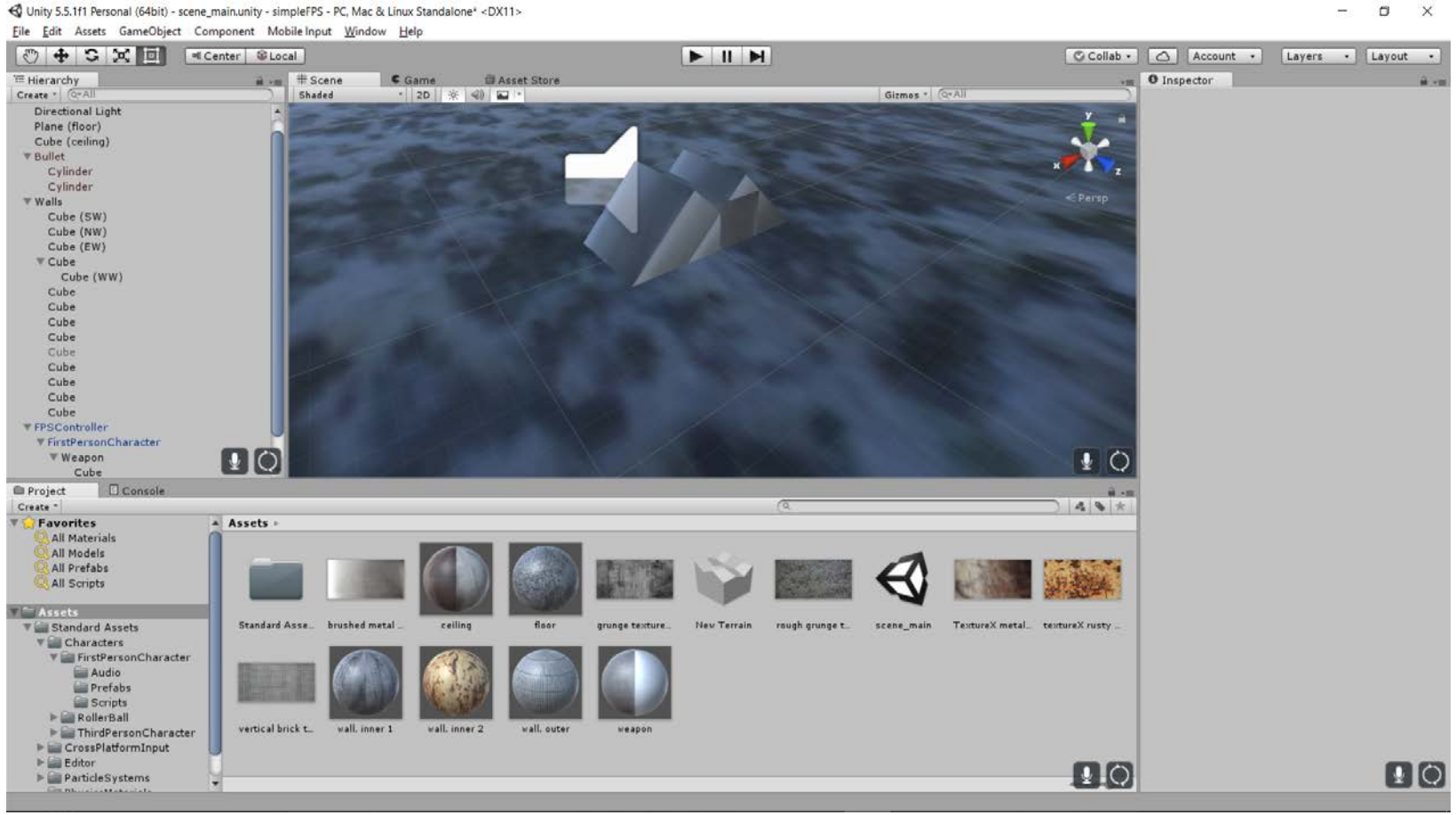

Jeg har sat playmode for mig til at have en overlay farve så jeg kan se når jeg er i playmode.
Dette gøres under Edit > Preferences > Colors > Playmode tint

# Lavet et tomt gameobject der indeholder to cylindre til kugler



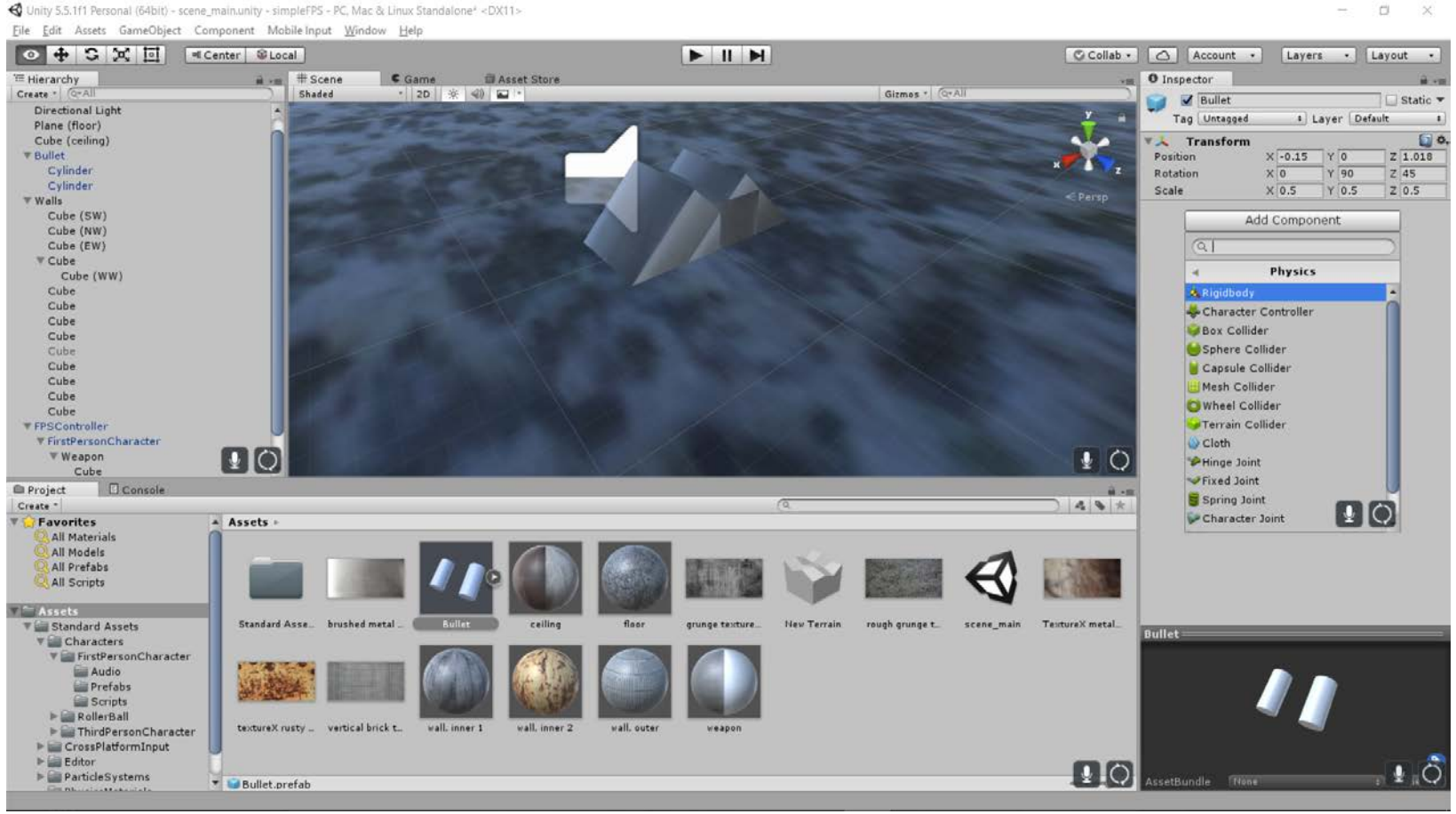
# Træk kuglen fra hierarkiet ned i assets området for at gøre det til en prefab

Slet den derefter fra hierarkiet, da vi ikke skal bruge den fast



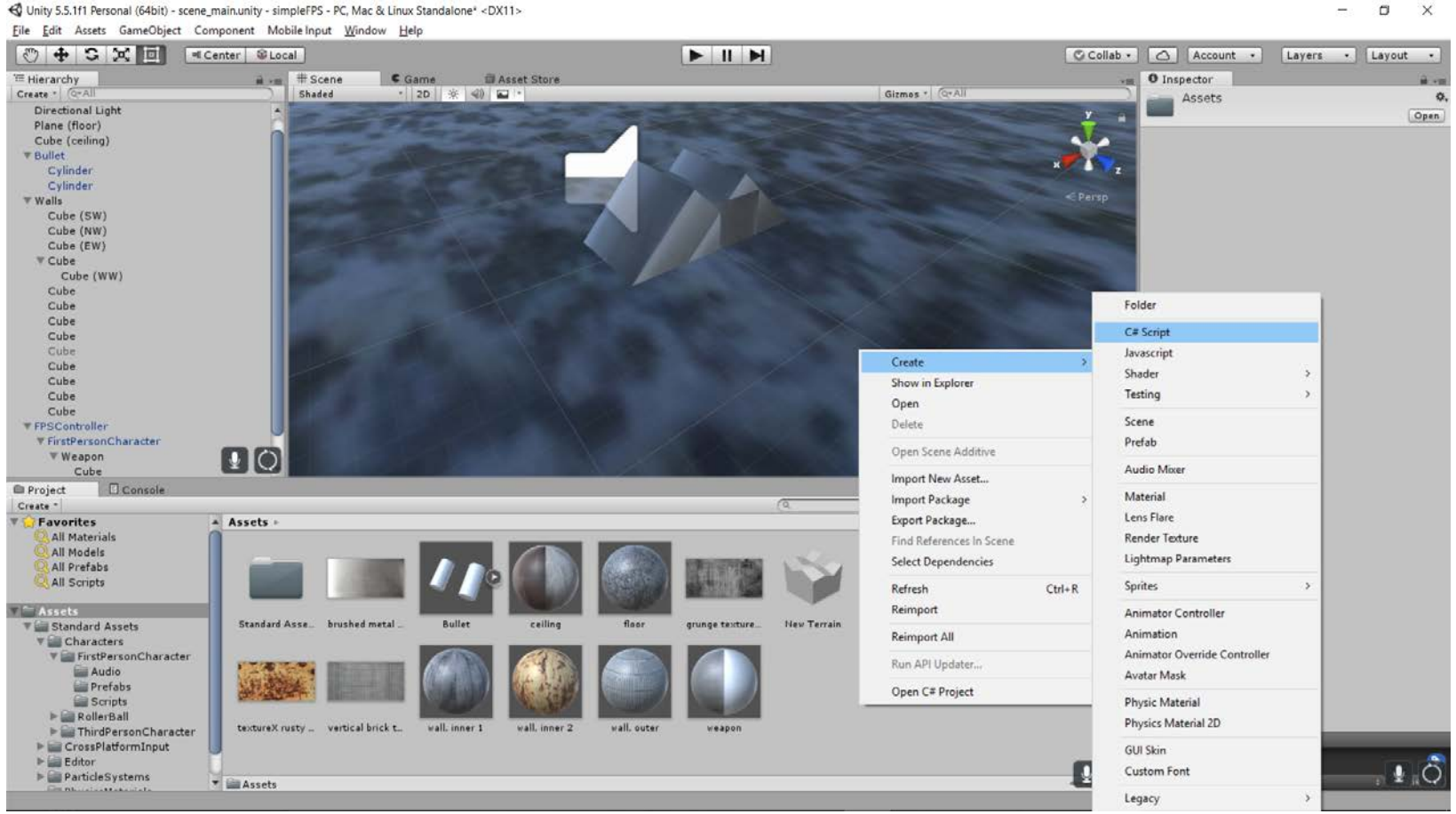
# Kuglerne skal have fysiske egenskaber, så vi giver dem en rigidbody

Add component > physics > rigidbody



# Vi skal have et script der gør at kuglerne eksploderer når de rammer

Create > C# Script



# Føj koden herunder til script filen

Assembly-CSharp - boom.cs - MonoDevelop-Unity

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

boom.cs

```
using System.Collections;
using UnityEngine;

public class boom : MonoBehaviour {
    public float speed = 2000.0f;
    // The fly speed (used by the weapon later)

    public GameObject explosionPrefab;
    // explosion prefab (particles)

    void OnCollisionEnter(Collision c) {
        // find out when it hit something

        Instantiate(explosionPrefab, transform.position, Quaternion.identity);
        // show explosion - transform.position because it should be where the bullet is - Quaternion.identity because it should have default rotation

        Destroy(gameObject);
        // destroy the bullet. Destroy(this) would just destroy the script attached to it, Destroy(gameObject) destroys the whole thing
    }
}
```

D:\Unity\simpleFPS\Assets\boom.cs - MonoDevelop-Unity

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

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

boom.cs

boom ▸ No selection

```
1 using System.Collections;
2 using UnityEngine;
3
4 public class boom : MonoBehaviour {
5     public float speed = 2000.0f;
6     // The fly speed (used by the weapon later)
7
8     public GameObject explosionPrefab;
9     // explosion prefab (particles)
10
11    void OnCollisionEnter(Collision c) {
12        // find out when it hit something
13
14        Instantiate(explosionPrefab, transform.position, Quaternion.identity);
15        // show explosion - transform.position because it should be where the bullet is - Quaternion.identity because it should have default rotation.
16
17        Destroy(gameObject);
18        // destroy the bullet. Destroy(this) would just destroy the script attached to it, Destroy(gameObject) destroys the whole thing
19    }
20 }
```

Watch | Locals | Breakpoints | Threads | Call Stack | Immediate

| Name | Value | Type |
|---|---|---|

Click here to add a new watch

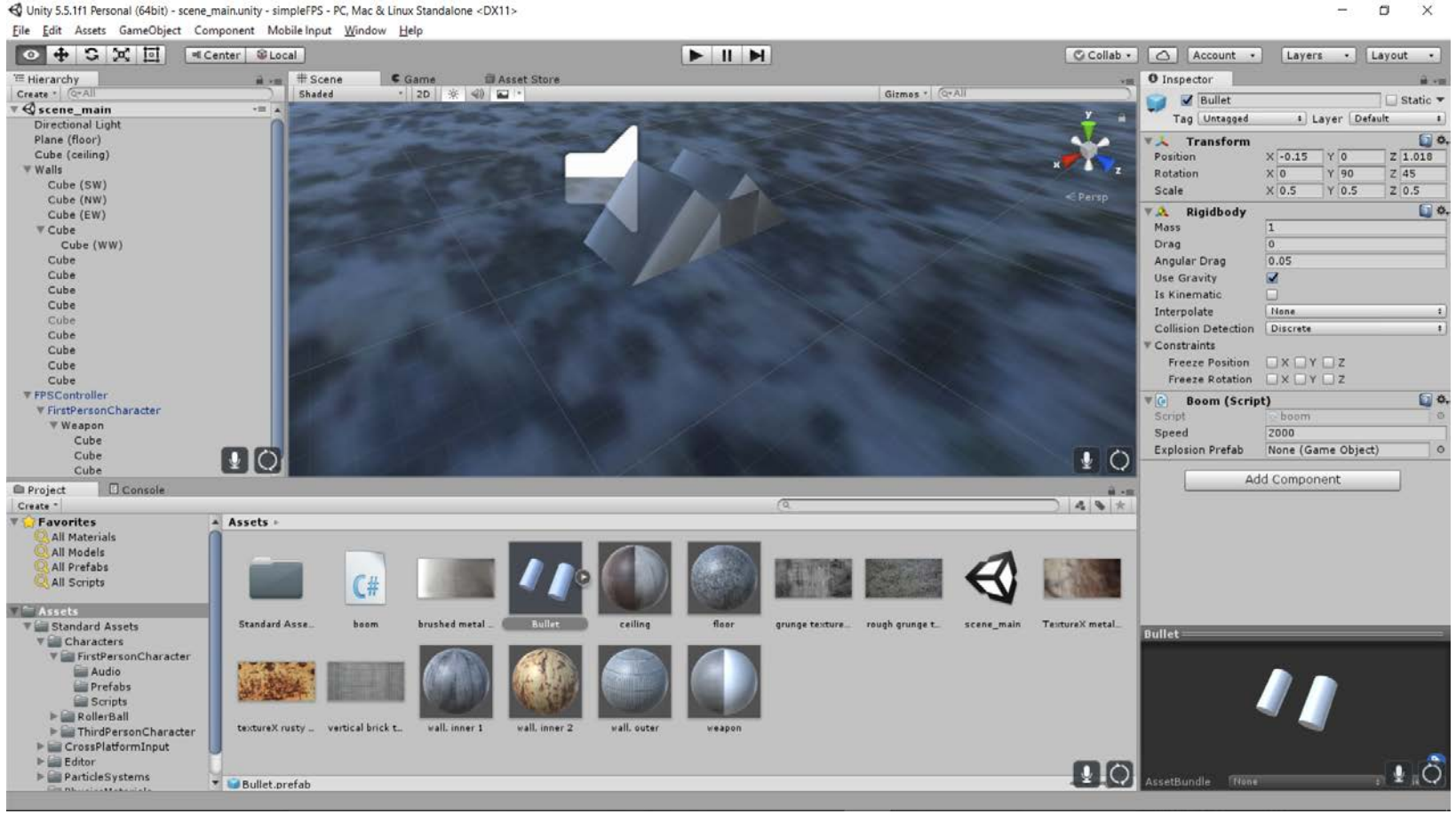
| Name | File | Language | Address |
|---|---|---|---|

Toolbox | Unit Tests | Properties | Document Outline

Errors | Tasks | Application Output

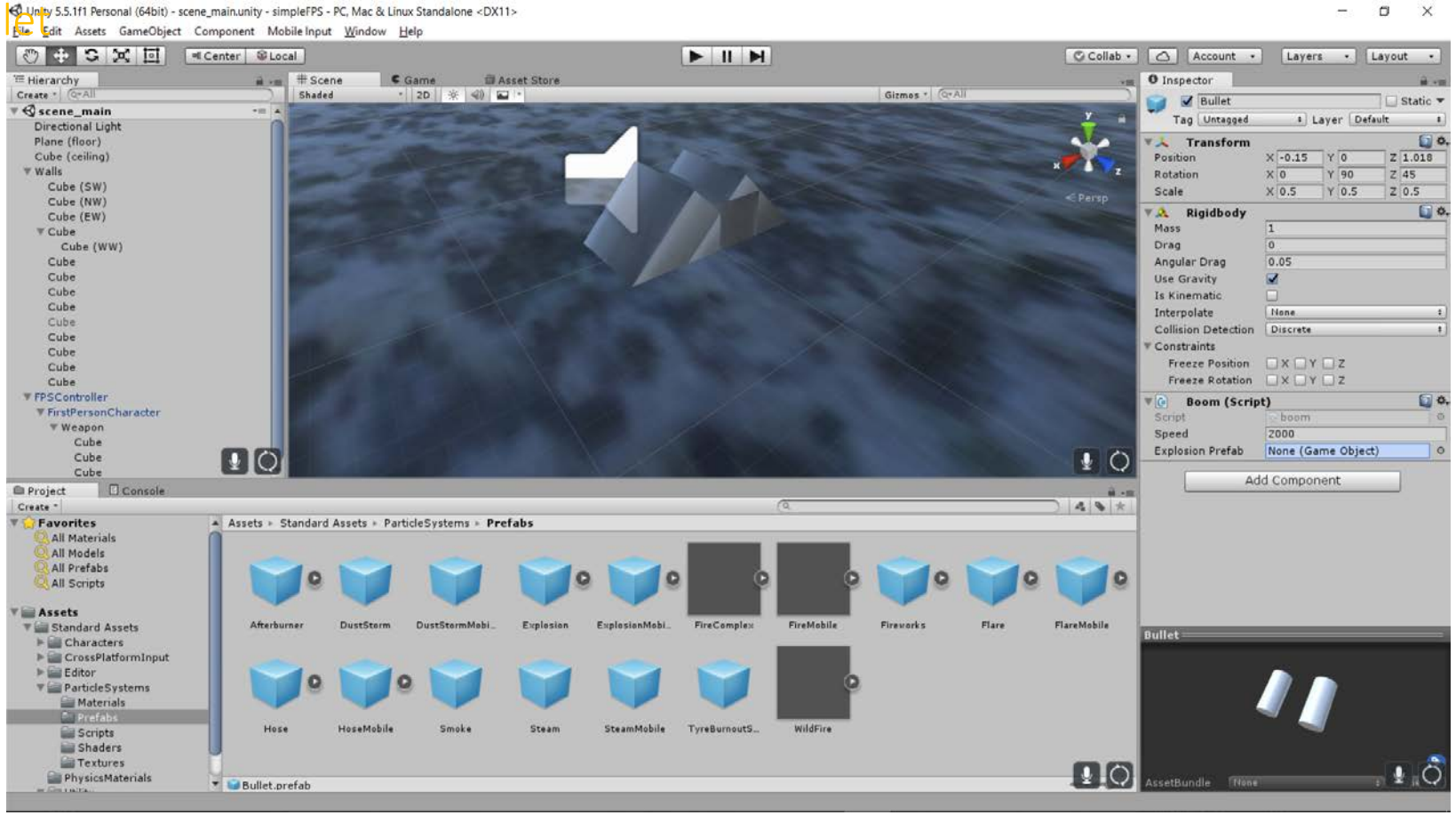
# Vi instantierer eksplosionen og ødelægger det ved at tilføje scriptet

Bare træk det fra assets til indstillings-Panelet for Bullet prefab



# Eksplosionen får vi gennem unitys partikler, som vi importerede i starten

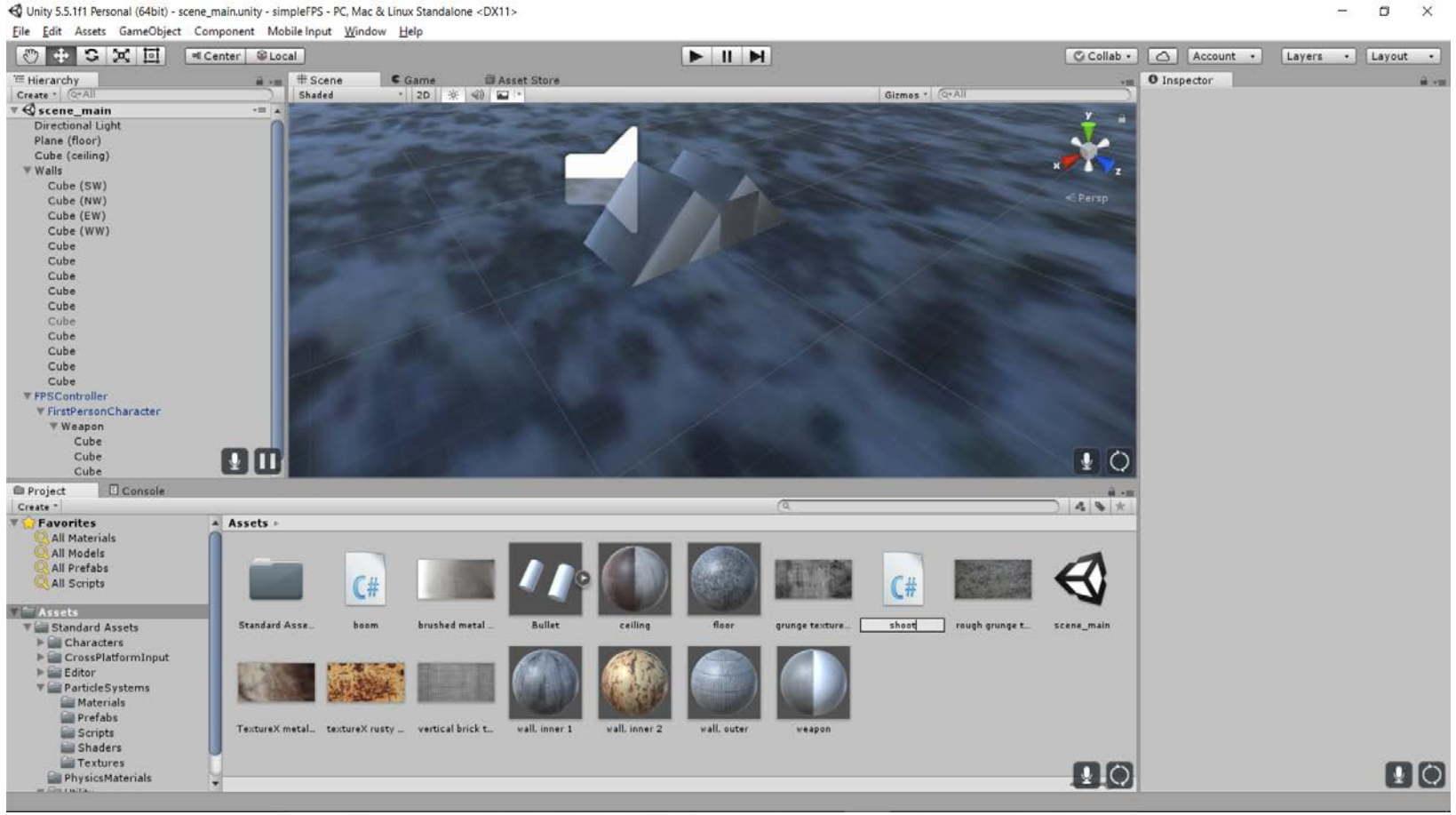
Standard Assets > Particlesystems > prefabs og træk så explosion ind i booms scriptets explosion felt på vores bullet




Sæt eventuelt scale ned til 0.2 da man ellers stort set kun kan skyde opad. Prøv dig frem med det senere når vi har spillet oppe at køre.

# For at kuglerne kan flyve skal vi have et script til

Create > C# Script



# Føj koden herunder til script filen

## Vi bruger rigidbodys Addforce metode til at lade det flyve fremad

Assembly-CSharp - shoot.cs - MonoDevelop-Unity

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

bullet.cs | shoot.cs

shoot ▸ No selection

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class shoot : MonoBehaviour {
6
7     public GameObject bulletPrefab;
8
9     void Update () {
10        if (Input.GetMouseButtonDown (0)) {
11
12            GameObject g = (GameObject)Instantiate (bulletPrefab, transform.position, transform.parent.rotation);
13            float force = g.GetComponent<bullet> ().speed;
14            g.GetComponent<Rigidbody> ().AddForce (g.transform.forward * force);
15        }
16    }
17 }
```

Watch | Locals | Breakpoints | Threads | Call Stack | Immediate

Name | Value | Type

Click here to add a new watch

Name | File | Language | Address

Errors | Tasks | Application Output

Assembly-CSharp - shoot.cs - MonoDevelop-Unity

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

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

bullet.cs | shoot.cs

shoot ▸ No selection

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

public class shoot : MonoBehaviour {

    public GameObject bulletPrefab;

    void Update () {
        if (Input.GetMouseButtonDown (0)) {

            GameObject g = (GameObject)Instantiate (bulletPrefab, transform.position, transform.parent.rotation);
            float force = g.GetComponent<bullet> ().speed;
            g.GetComponent<Rigidbody> ().AddForce (g.transform.forward * force);
        }
    }
}
```

Watch | Locals | Breakpoints | Threads

| Name | Value | Type |
|---|---|---|

Click here to add a new watch

Call Stack | Immediate

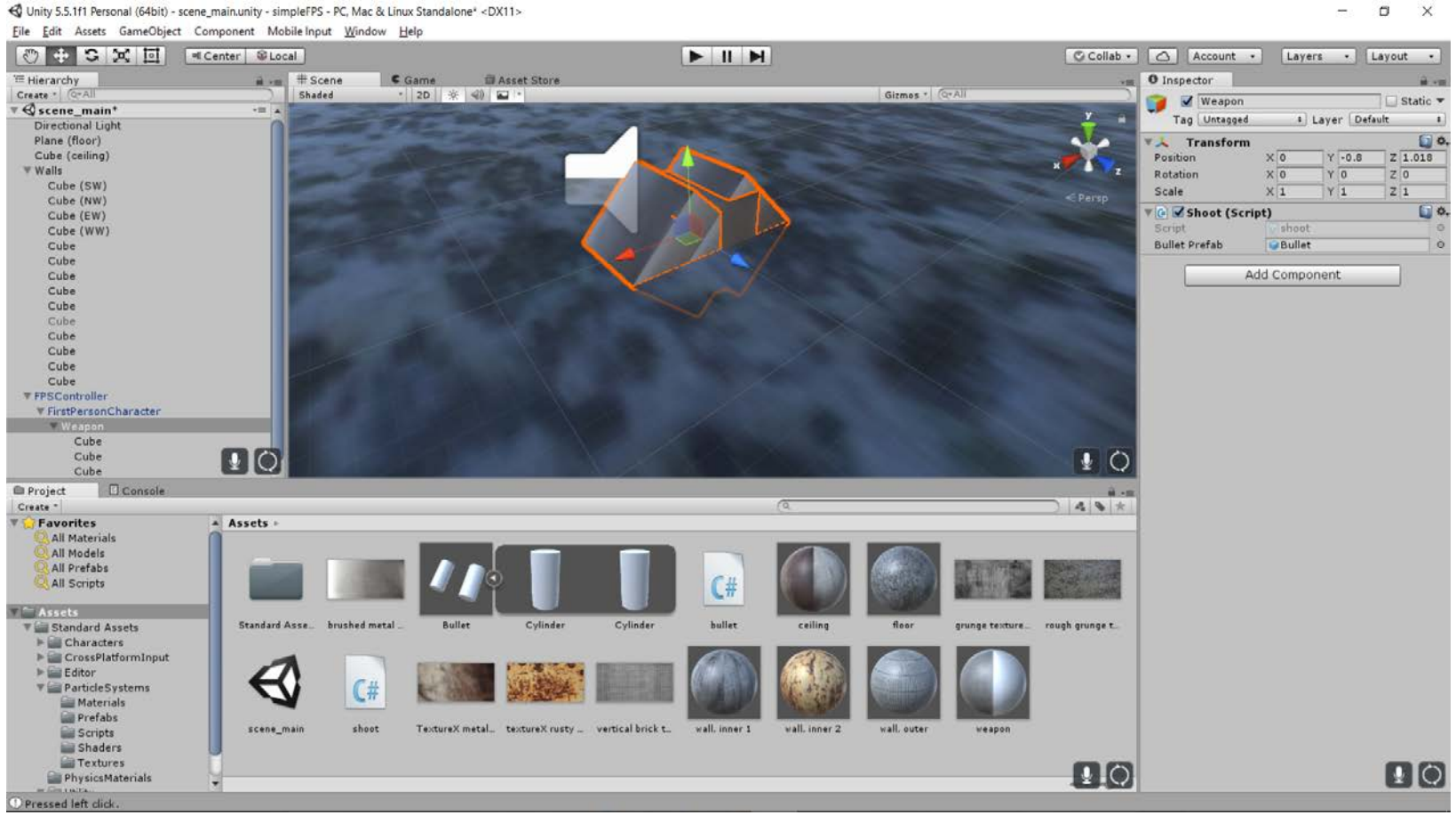
| Name | File | Language | Address |
|---|---|---|---|

Toolbox | Unit Tests | Properties | Document Outline

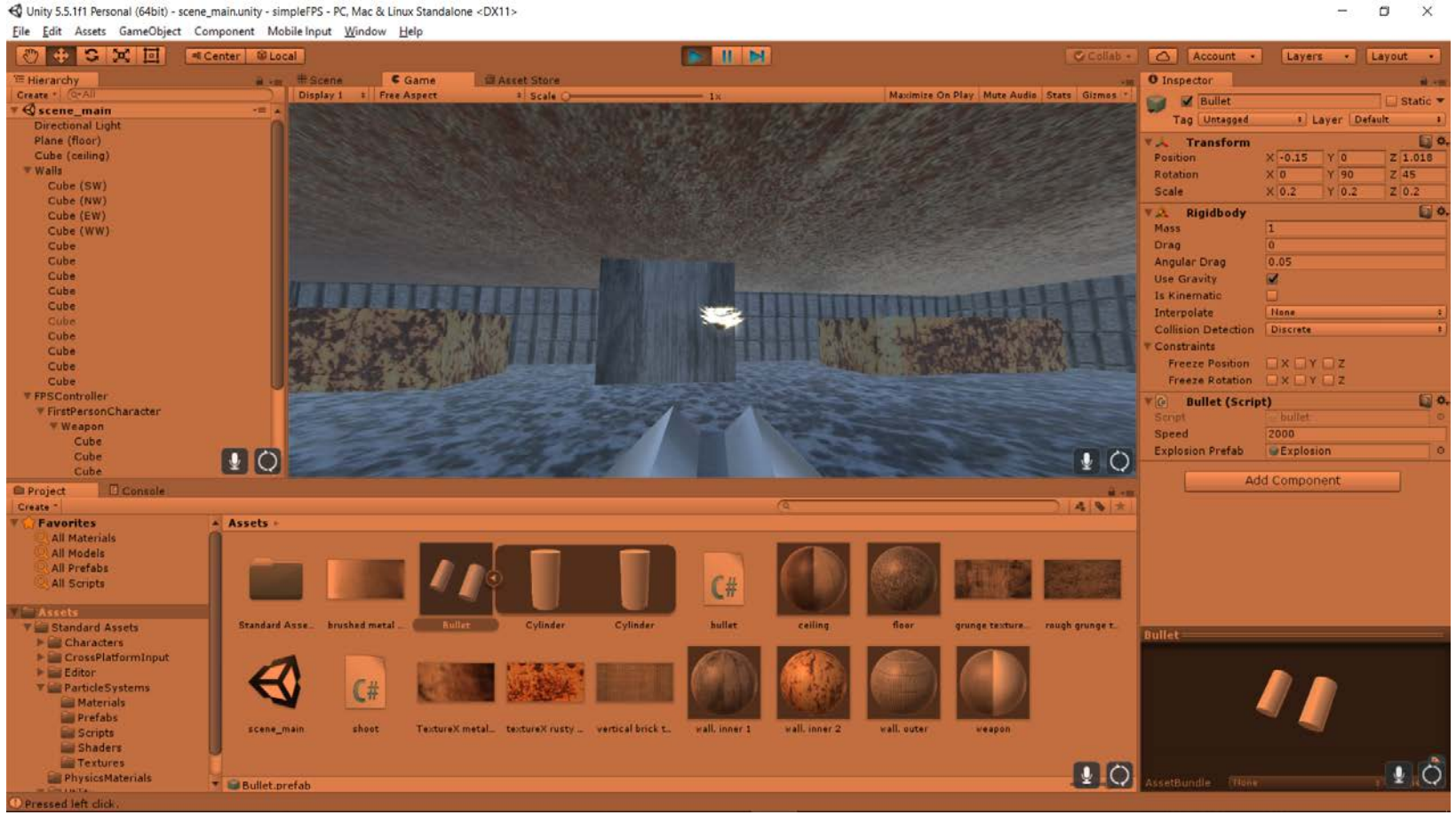
Errors | Tasks | Application Output

# Føj shoot scriptet til vores våben

husk at trække vores Bullet prefab over i feltet

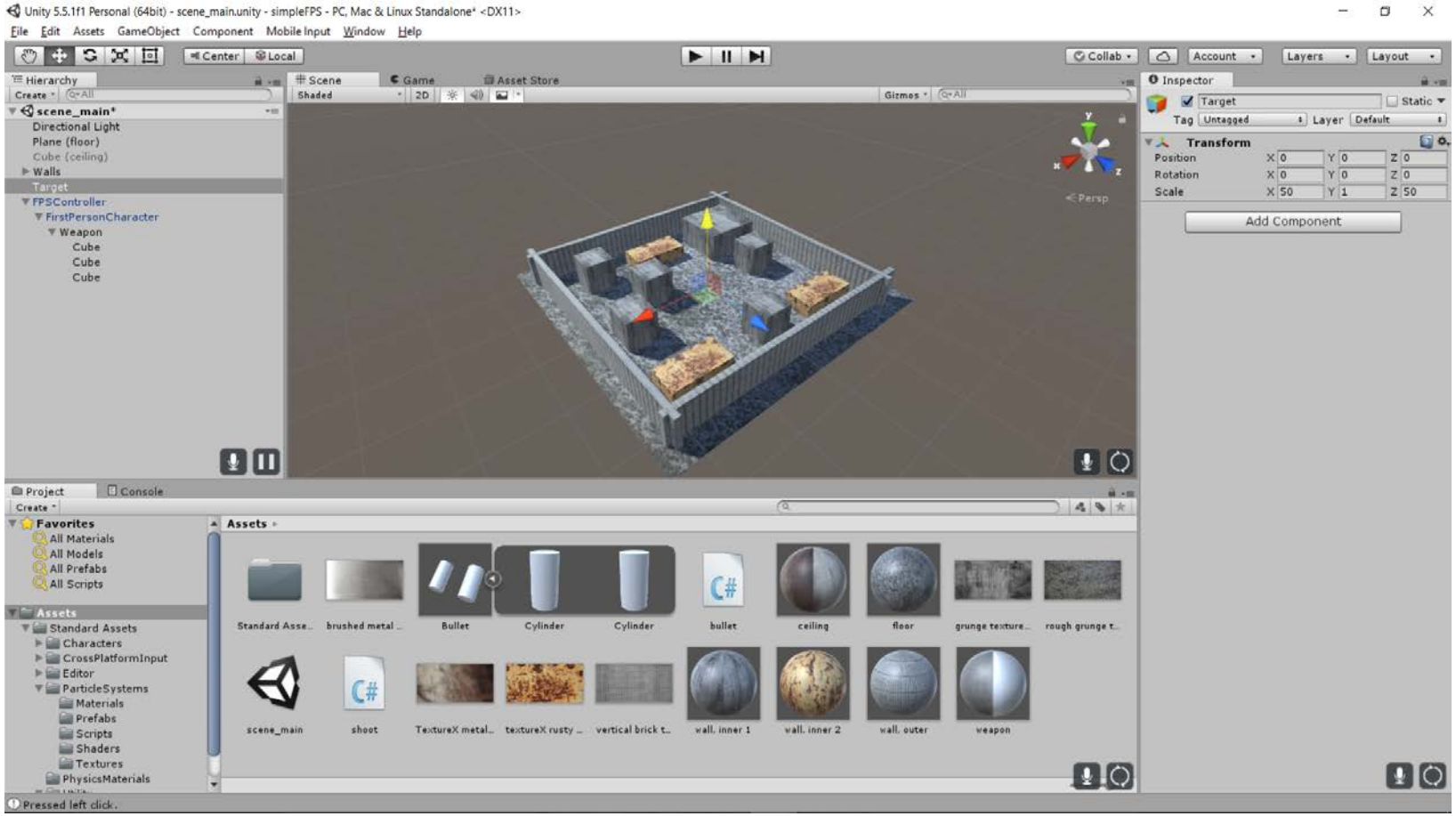


# Sådan!


Unity 5.5.1f1 Personal (64bit) - scene_main.unity - simpleFPS - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Mobile Input Window Help
Hierarchy
scene_main
Directional Light
Plane (floor)
Cube (ceiling)
Walls
FPSController
FirstPersonCharacter
Weapon
Inspector
Bullet
Transform
Position
Rotation
Scale
Rigidbody
Mass
Drag
Angular Drag
Use Gravity
Is Kinematic
Interpolate
Collision Detection
Constraints
Freeze Position
Freeze Rotation
Bullet (Script)
Speed
Explosion Prefab
Add Component
Project
Console
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Standard Assets
Characters
CrossPlatformInput
Editor
ParticleSystems
Materials
Prefabs
Scripts
Shaders
Textures
PhysicsMaterials
Bullet.prefab
Pressed left click.

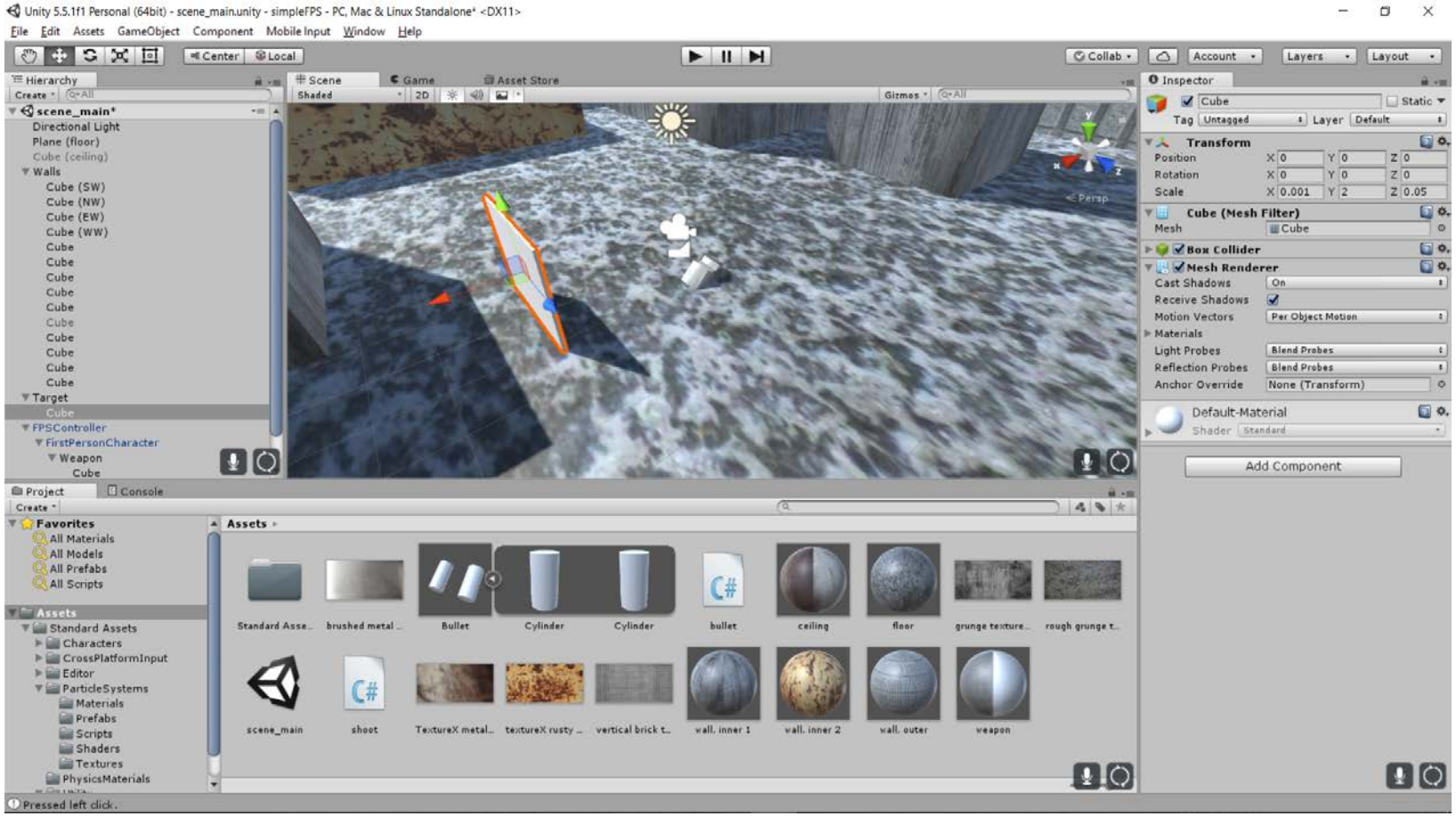
# Las os føje nogle skydeskiver til

## Create empty taget som overordnet



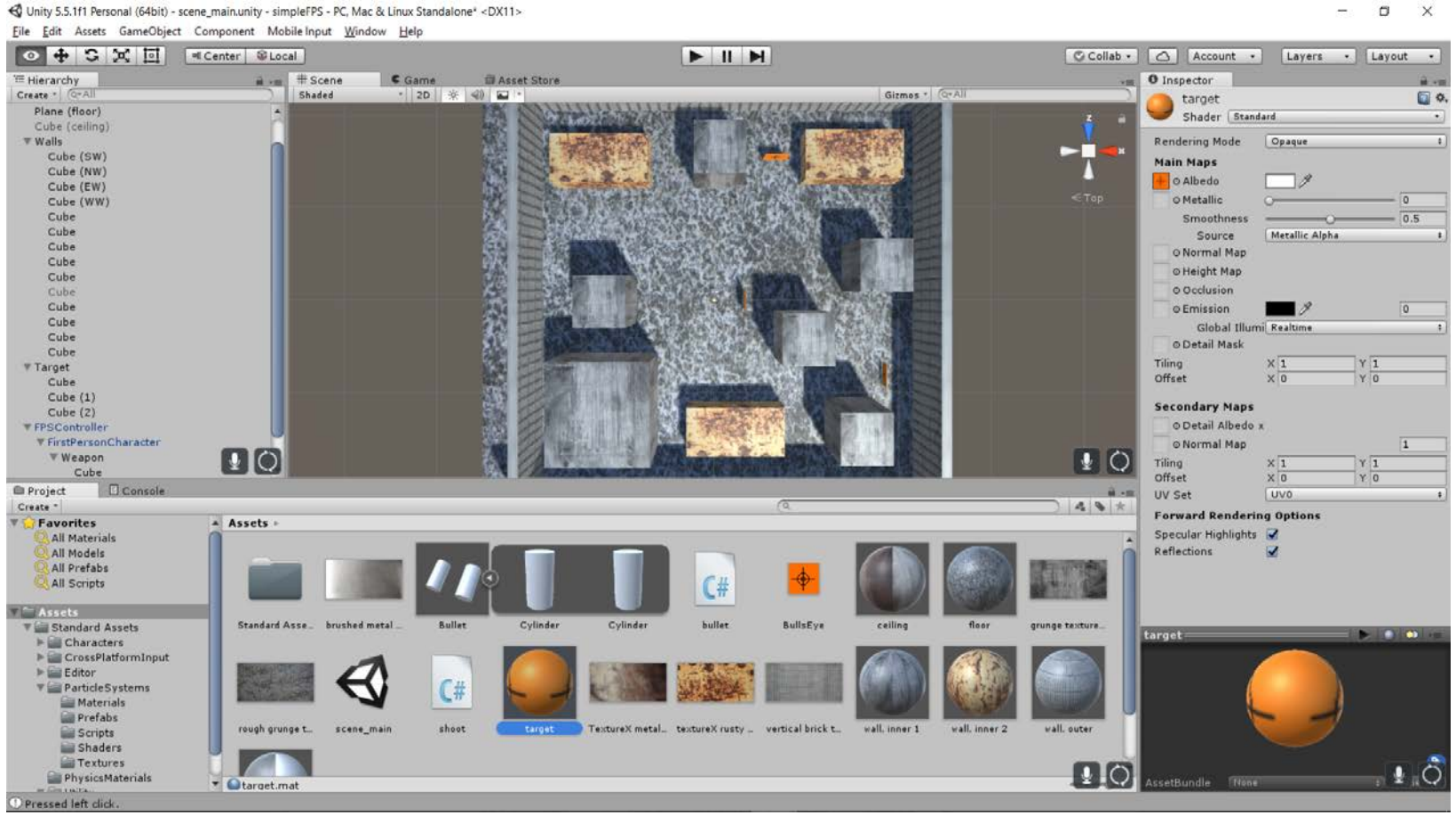
# Las os føje nogle skydeskiver til

## lav cubes derefter og placer dem rundt om på banen



# Las os føje nogle skydeskiver til

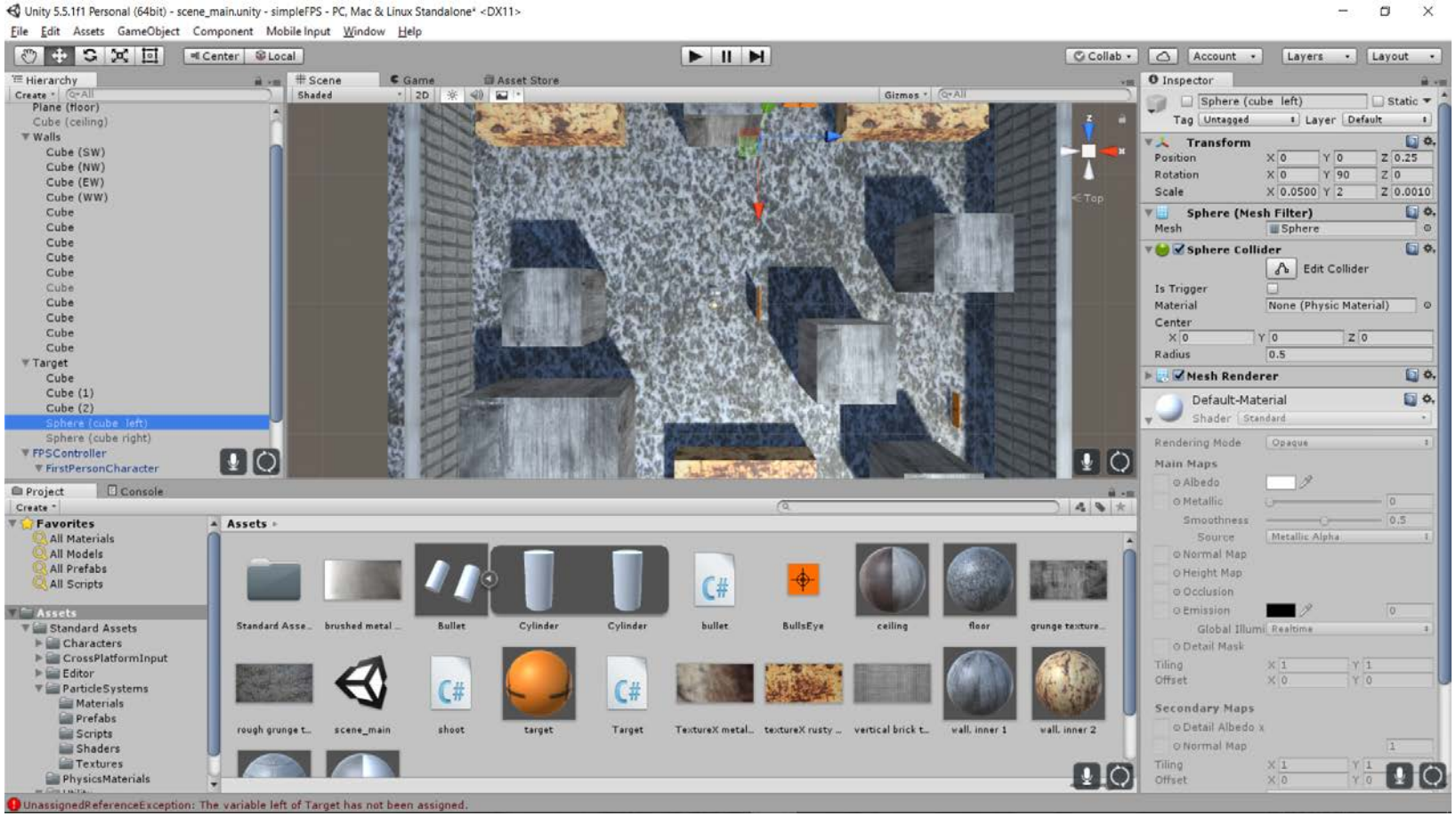
lav et passende materiale og føj det til vores skydeskiver



# Lad os føje nogle skydeskiver til

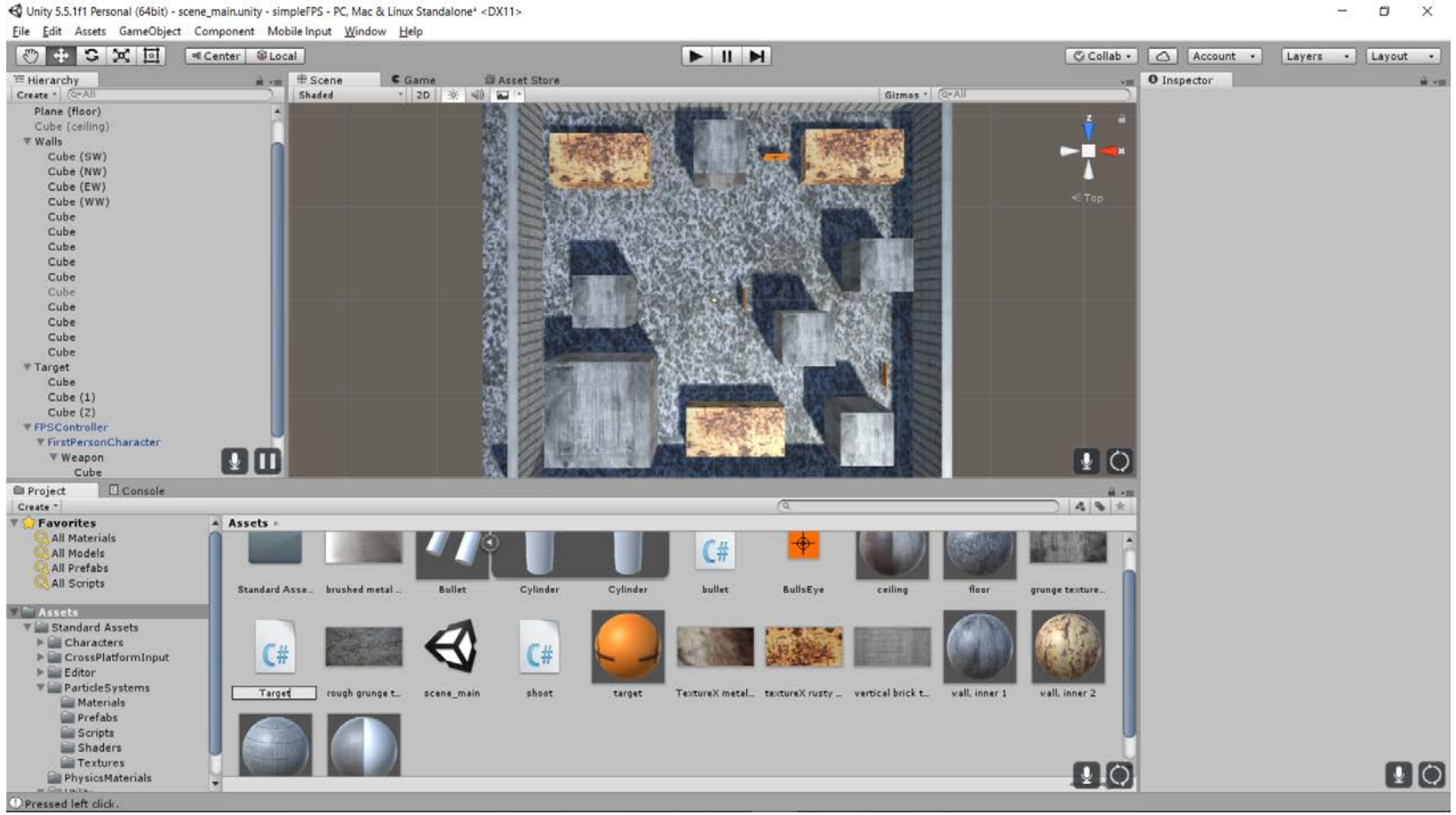
Vi skal have nogle punkter de kan bevæge sig mellem, til højre og venstre for. De behøves ikke være synlige for spilleren.

ikke



# Las os føje nogle skydeskiver til

Statiske skydeskiver er kedelige, så vi laver et script der får dem til at bevæge sig



Assembly-CSharp - Target.cs* - MonoDevelop-Unity

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

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

bullet.cs | shoot.cs | Target.cs

target ▸ Update ()

```
using UnityEngine;
using System.Collections;

public class target : MonoBehaviour {
    // left and right marks
    public Transform left;
    public Transform right;

    public float speed = 1.0f;  // speed

    bool dir = false;   // current direction (false means to the left, true means to the right)

    void Update () {
        if (dir) {
            // go closer to the right one
            transform.position = Vector3.MoveTowards(transform.position, right.position, Time.deltaTime * speed);

            // reached it?
            if (transform.position == right.position)
                dir = !dir; // go to opposite direction next time
        } else {
            // go closer to the left one
            transform.position = Vector3.MoveTowards(transform.position, left.position, Time.deltaTime * speed);

            // reached it?
            if (transform.position == left.position)
                dir = !dir; // go to opposite direction next time
        }
    }
}
```

Watch | Locals | Breakpoints | Threads

| Name | Value | Type |
|---|---|---|

Click here to add a new watch

Call Stack | Immediate

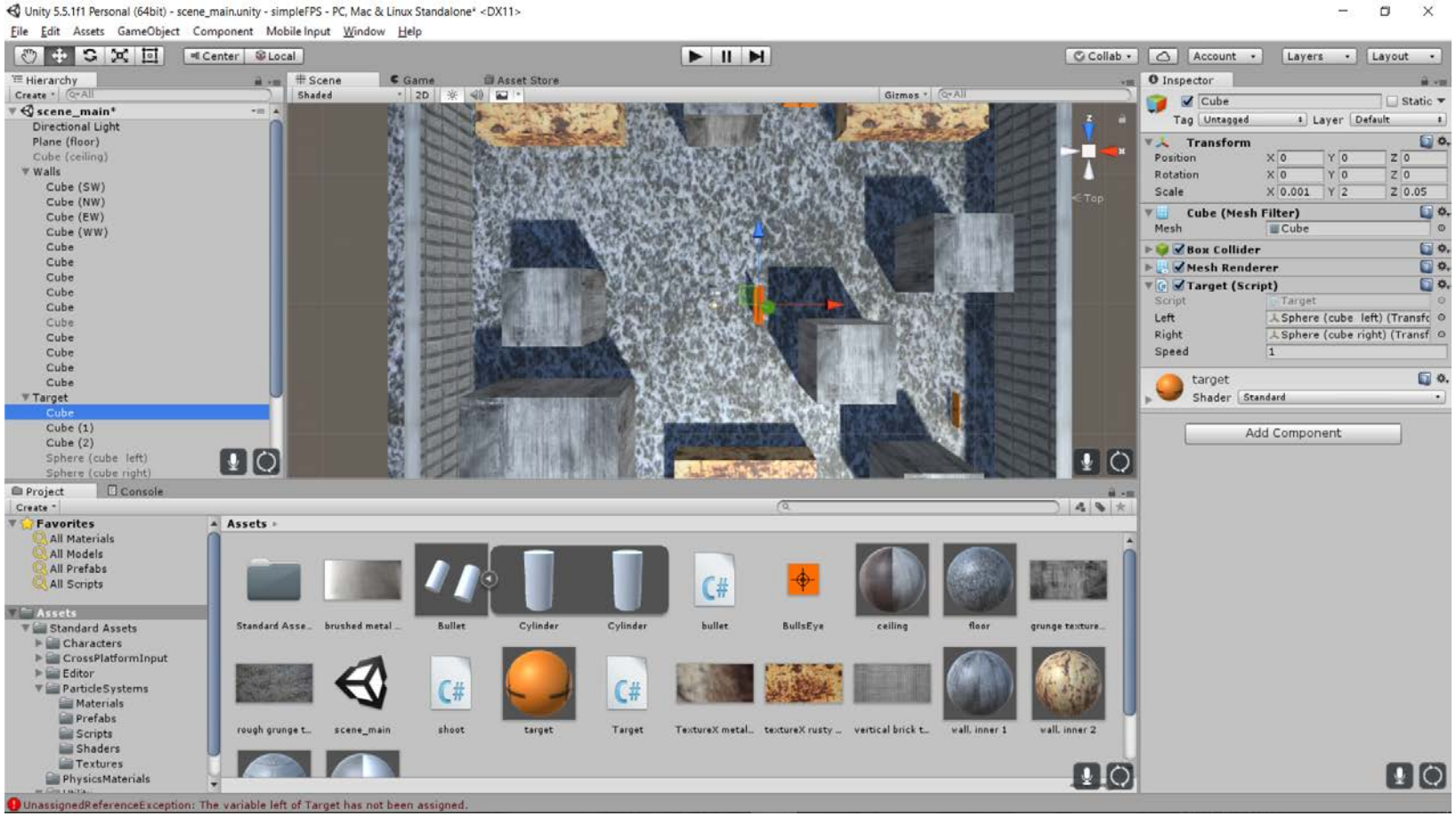
| Name | File | Language | Address |
|---|---|---|---|

Toolbox | Unit Tests | Properties | Document Outline

Errors | Tasks | Application Output

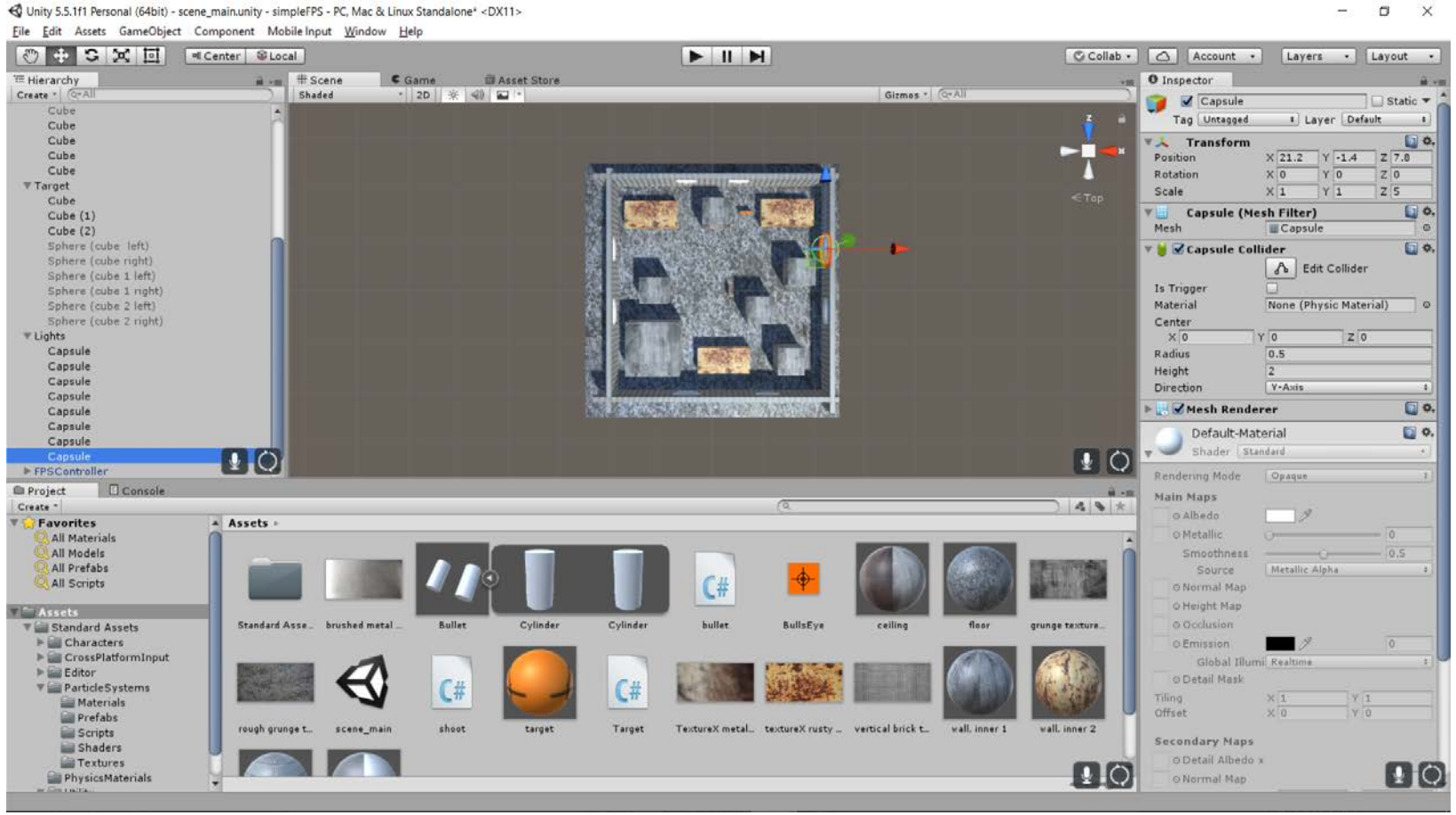
# Las os føje nogle skydeskiver til

## Føj scriptet til skydeskiverne og sæt højre og venstre mærkerne



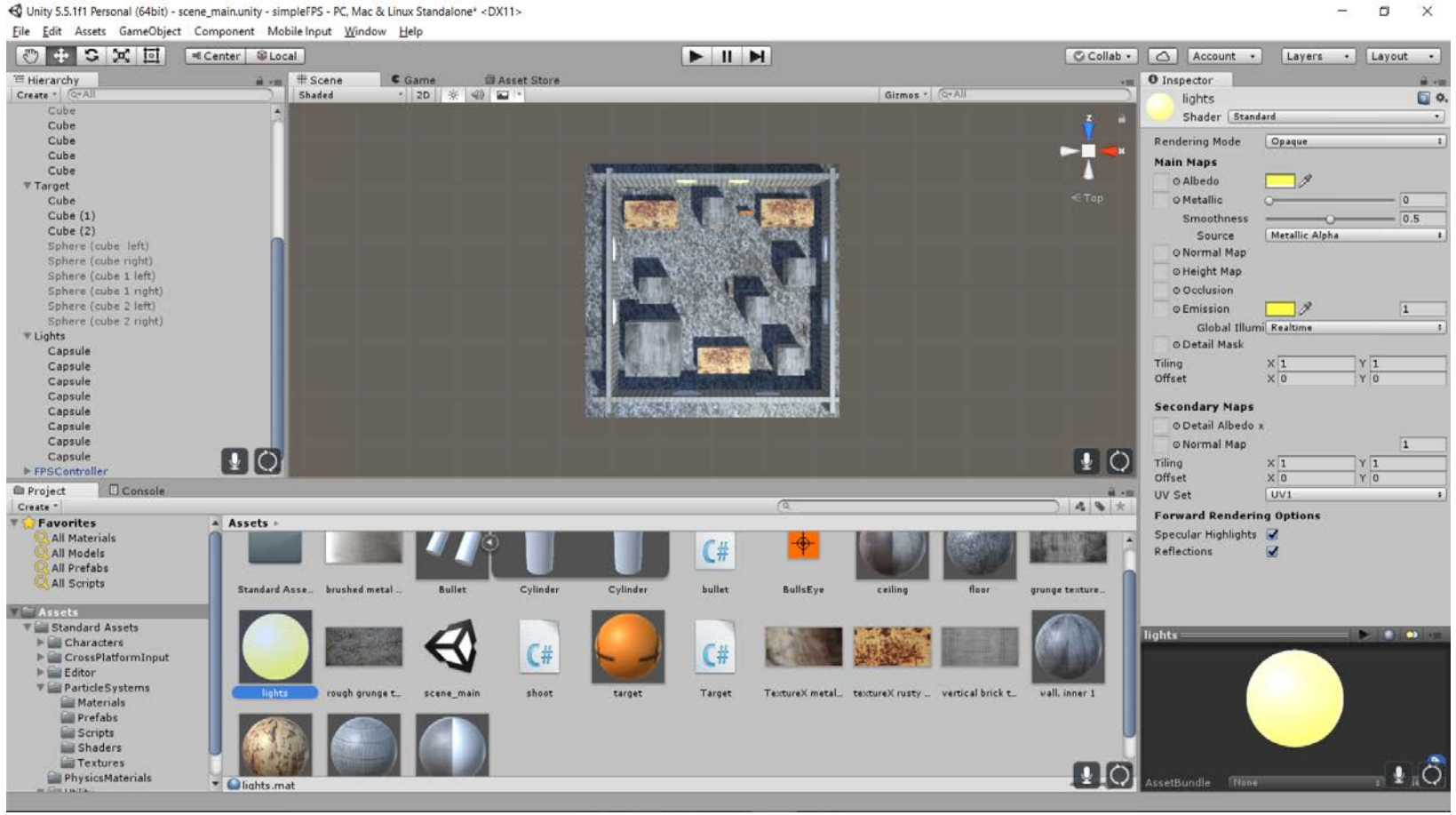
# Med loftet aktivt er scenen lidt mørk

Vi laver en gruppe, lights, der indeholder capsules til lysstofrør



# Med loftet aktivt er scenen lidt mørk

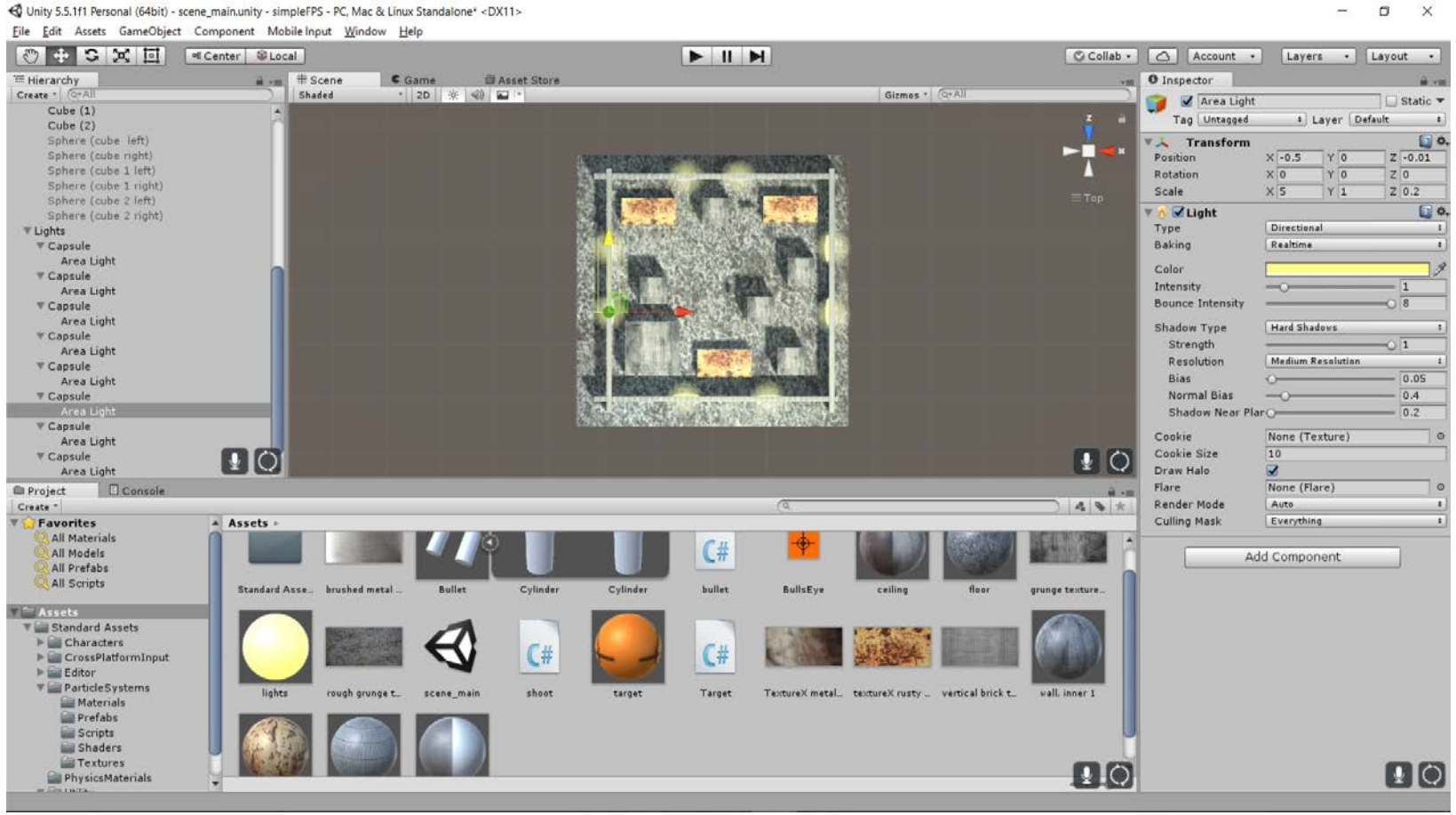
## Vi laver et nyt materiale så vores lysstofrør er mere gule



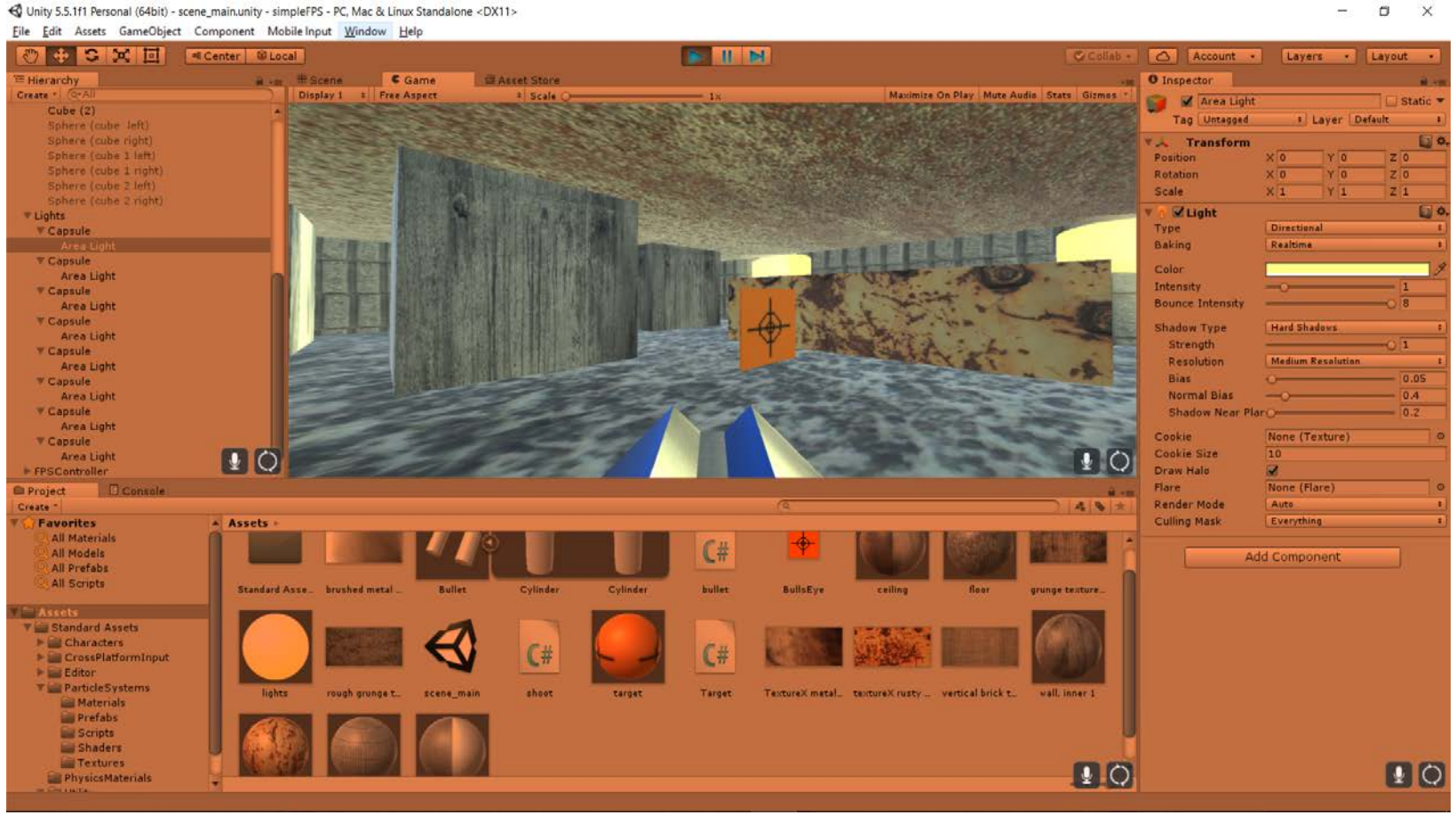
# Med loftet aktivt er scenen lidt mørk

Vi føjer area lights til og lægger dem oveni vores ”lysstofrør”

Eksperimenter med indstillingerne for det bedste resultat



# Måske lidt meget men sådan!



# Kilder

# Kilder

## UML

- https://www.tutorialspoint.com/uml/uml_class_diagram.htm
- https://msdn.microsoft.com/en-us/library/dd409416.aspx
- http://creately.com/blog/diagrams/umldiagram-types-examples/
- http://modeling-languages.com/best-uml-cheatsheets-and-reference-guides/

## Objekt Orienteret Programmering

- https://youtu.be/lbXsrHGhBAU