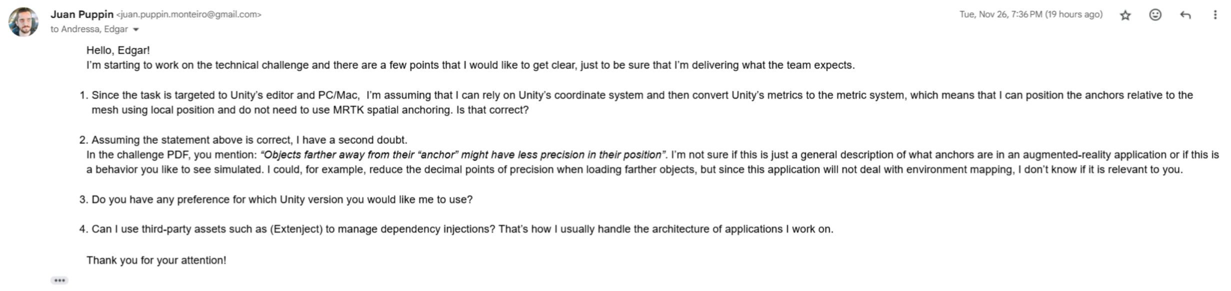
Juan Puppin - Code Challenge

Hello, in this dev log, I'm reporting the process of developing this technical challenge step-by-step. I intend to make the whole process of planning and delivering this application as clear as possible. I hope you enjoy it!

Understanding the requirements

After reading the challenge document, I had some doubts about whether or not my interpretation was correct. I was worried that I might be underestimating the demand since spatial anchorings in the AR context have special implications due to the environment mapping. So, I decided to email Edgar to get my doubts clear. It turned out that the requirements were indeed quite simple, and I could start the development.



Juan Puppin <juan.puppin.monteiro@gmail.com>
to Andressa, Edgar
Tue, Nov 26, 7:36 PM (19 hours ago)

Hello, Edgar!
I'm starting to work on the technical challenge and there are a few points that I would like to get clear, just to be sure that I'm delivering what the team expects.

1. Since the task is targeted to Unity's editor and PC/Mac, I'm assuming that I can rely on Unity's coordinate system and then convert Unity's metrics to the metric system, which means that I can position the anchors relative to the mesh using local position and do not need to use MRTK spatial anchoring. Is that correct?

2. Assuming the statement above is correct, I have a second doubt.
In the challenge PDF, you mention: *"Objects farther away from their "anchor" might have less precision in their position"*. I'm not sure if this is just a general description of what anchors are in an augmented-reality application or if this is a behavior you like to see simulated. I could, for example, reduce the decimal points of precision when loading farther objects, but since this application will not deal with environment mapping, I don't know if it is relevant to you.

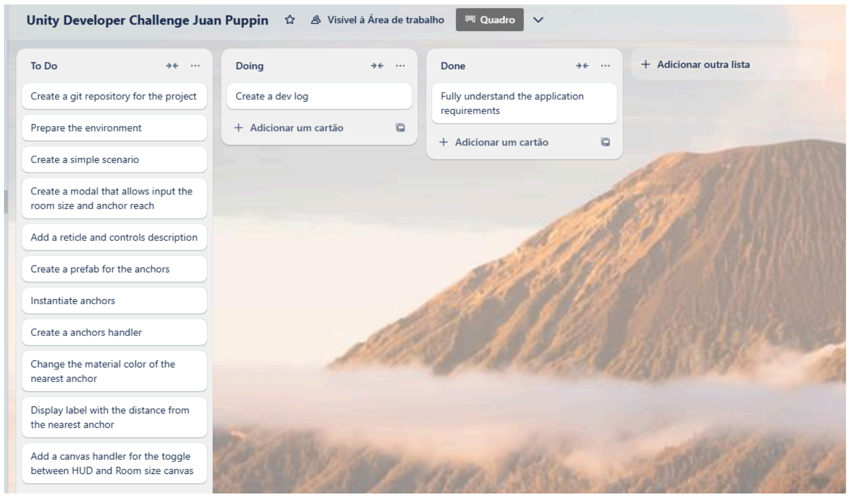
3. Do you have any preference for which Unity version you would like me to use?

4. Can I use third-party assets such as (Extenject) to manage dependency injections? That's how I usually handle the architecture of applications I work on.

Thank you for your attention!

Planning

With my doubts clear, it was time to plan and develop this application, in a short description, I'm creating an application where the user will be able to control a camera and instantiate anchors, the nearest anchor should be displayed in green and show a label with the distance from the user position, open a modal where it will be possible to change the room size and anchors reach at runtime.
With that in mind, I planned my tasks and started the development.



Unity Developer Challenge Juan Puppin

To Do
- Create a git repository for the project
- Prepare the environment
- Create a simple scenario
- Create a modal that allows input the room size and anchor reach
- Add a reticle and controls description
- Create a prefab for the anchors
- Instantiate anchors
- Create a anchors handler
- Change the material color of the nearest anchor
- Display label with the distance from the nearest anchor
- Add a canvas handler for the toggle between HUD and Room size canvas

Doing
- Create a dev log

Done
- Fully understand the application requirements

Architecture and organization

Architecture: In terms of architecture, I like to keep things as decoupled as possible. To achieve this I rely on dependency injections for handling references across the scene and I heavily use the observer pattern to enable dependency inversion.

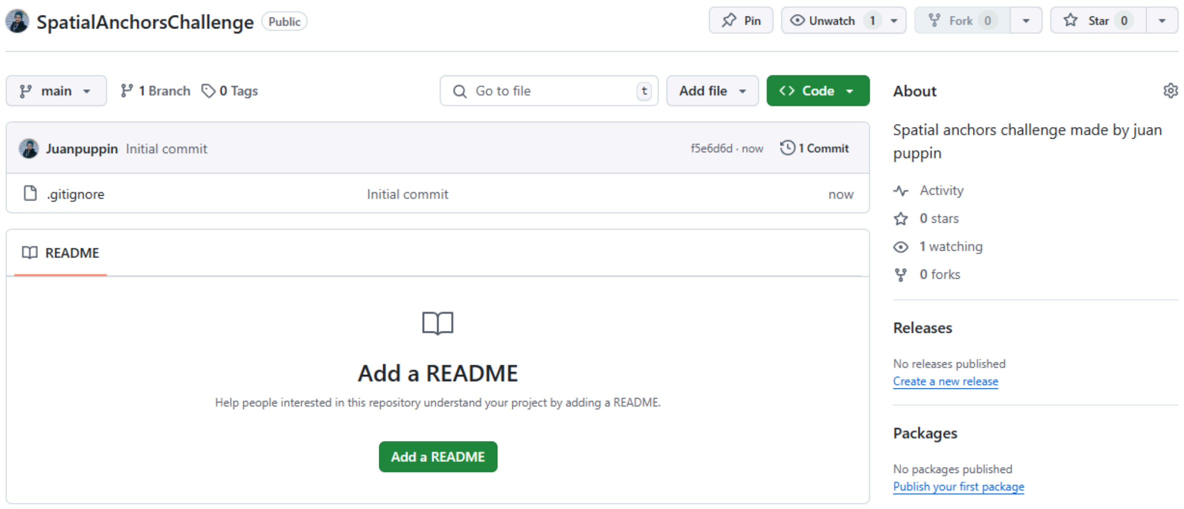
Naming Convention: To keep code readable, I like to use Pascal with camelCase for local variables and arguments. I also like to write every access modifier explicitly.

Folders structure: I'm using a quite conventional approach for folder structure, with two small details, I like to use an underline on the scripts folder "_Scripts" so it is always the first folder on the project view and also, I moved the plugins folder to inside the scripts, so all the classes are coverer by the scripts dll.

Code comments: Since the intention is to have a second interview where I will explain my code decisions, and I'm already writing this document, I decided to make as few comments as possible in the classes.

Preparing the environment

As asked, I set up a new project on GitHub and added a Unity .gitgnore that comes as one of the options when creating a new repository on the platform. After that, I created a Unity project, imported **Zenject**, **TextMeshPro**, **EditorCools, Free Fly Camera**, and made my first commit.


SpatialAnchorsChallenge Public

Juanpuppin Initial commit f5e6d6d · now 1 Commit

.gitignore Initial commit now

README

Add a README

Help people interested in this repository understand your project by adding a README.

About

Spatial anchors challenge made by juan puppin

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

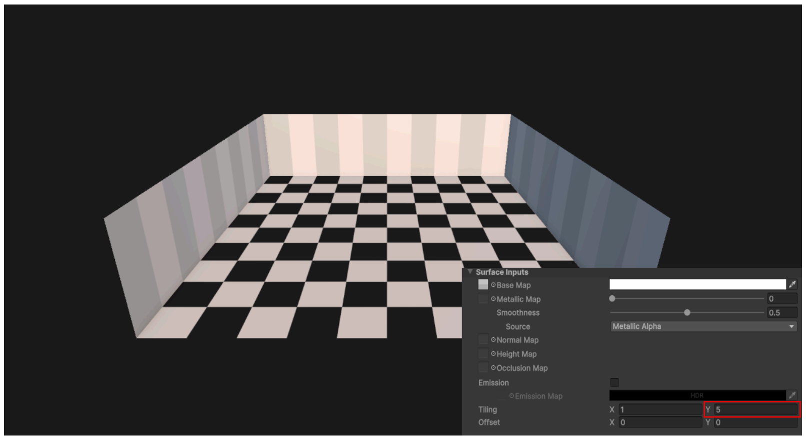
Packages

No packages published

Publish your first package

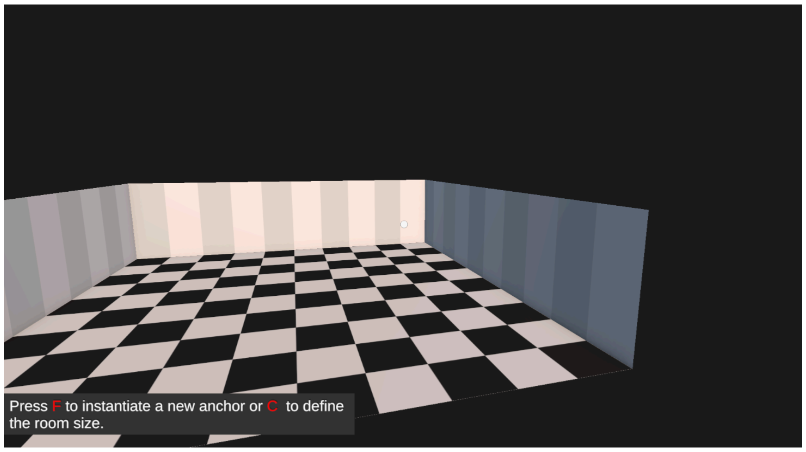
Setting up the scene

With the project ready, I created a quite simple scenario, with some tiled textures for the floor and walls, each tile represents one square meter so it’s easy to visualize the room proportions.



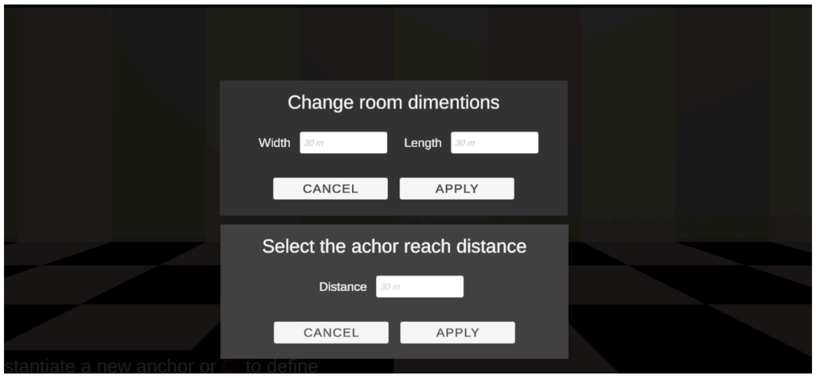
```
▼ Surface Inputs
  Base Map
  Metallic Map              0
  Smoothness                0.5
  Source                    Metallic Alpha
  Normal Map
  Height Map
  Occlusion Map
  Emission
  Emission Map              HDR
  Tiling                    X 1    Y 5
  Offset                    X 0    Y 0
```

Then I added a label anchored on the bottom left of the screen to display the inputs.



Press F to instantiate a new anchor or C to define the room size.

For last, I added two modals, one for changing the room dimension and the second one set the anchor reach on instantiate



Change room dimentions

Width 30 m Length 30 m

CANCEL APPLY

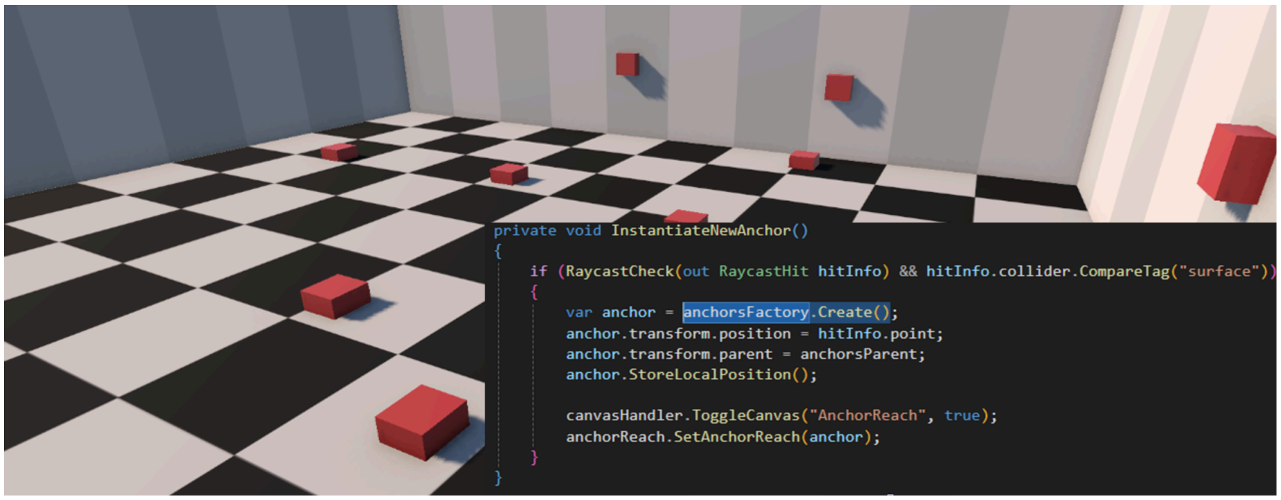
Select the achor reach distance

Distance 30 m

CANCEL APPLY

Instantiating Objects

For instantiating anchors, I decided to use a Raycast to get the hit point of the surface where the cursor is pointing, and I used Zenject factories to create each one, it allows me to inject references into the instantiated prefabs.

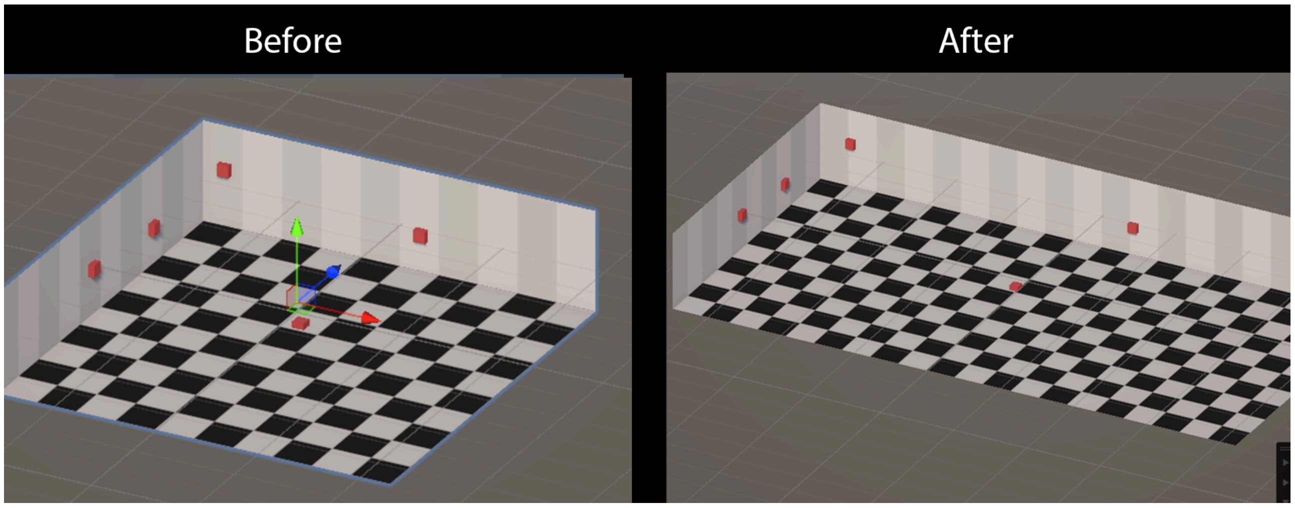


```
private void InstantiateNewAnchor()
{
    if (RaycastCheck(out RaycastHit hitInfo) && hitInfo.collider.CompareTag("surface"))
    {
        var anchor = anchorsFactory.Create();
        anchor.transform.position = hitInfo.point;
        anchor.transform.parent = anchorsParent;
        anchor.StoreLocalPosition();

        canvasHandler.ToggleCanvas("AnchorReach", true);
        anchorReach.SetAnchorReach(anchor);
    }
}
```

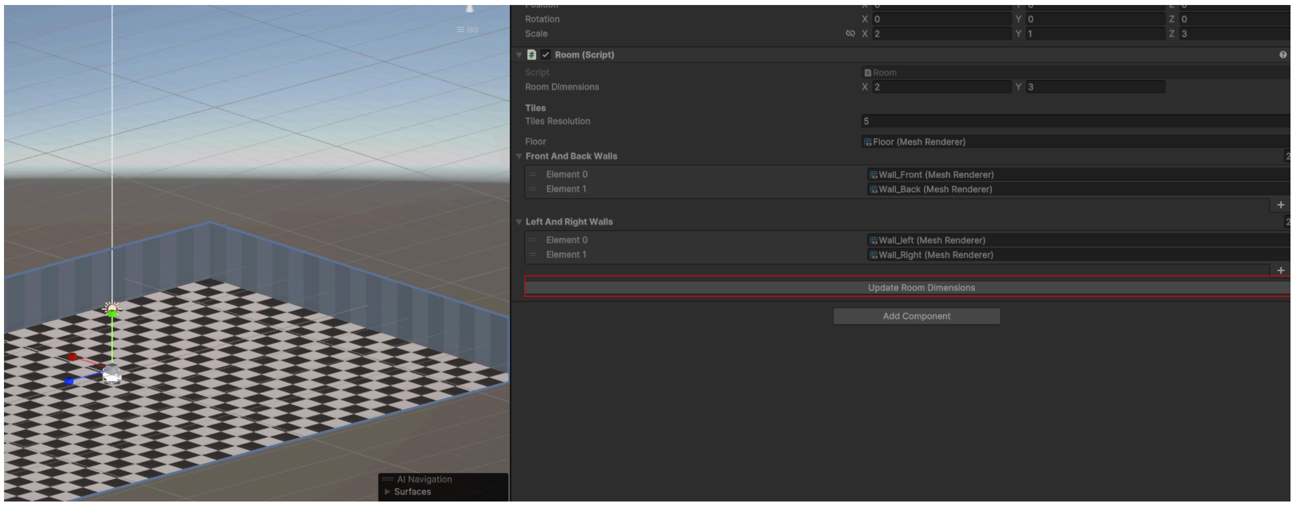
Anchoring points

For anchoring the position of each object, I created a method called **UpdateAnchorPosition** that multiplies the local position for room size, making the anchors stay at a fixed position relative to the room, so if an object is placed at the left wall, it will keep it's no position no matter the room dimension.



Update Room dimensions

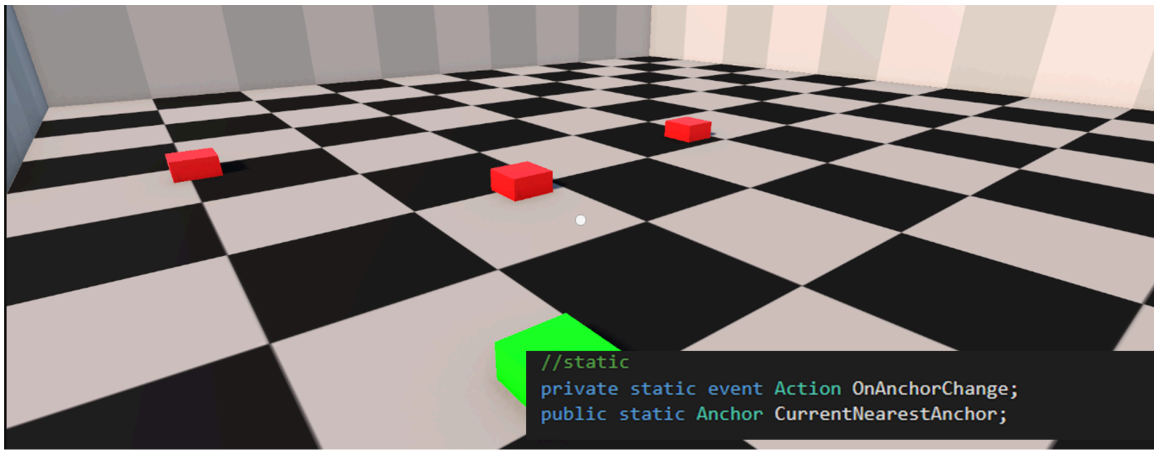
For debugging, I added a button to the room class that allows changing the room size through the inspector. At the first interview, I mentioned that I like to use a tool called Editor Cools, so I implemented this for the update room size method.



```
Rotation            X 0      Y 0      Z 0
Scale               X 2      Y 1      Z 3
Room (Script)
Script              Room
Room Dimensions     X 2      Y 3
Tiles
Tiles Resolution    5
Floor               Floor (Mesh Renderer)
Front And Back Walls
  Element 0         Wall_Front (Mesh Renderer)
  Element 1         Wall_Back (Mesh Renderer)
Left And Right Walls
  Element 0         Wall_left (Mesh Renderer)
  Element 1         Wall_Right (Mesh Renderer)
```

Finding nearest anchor

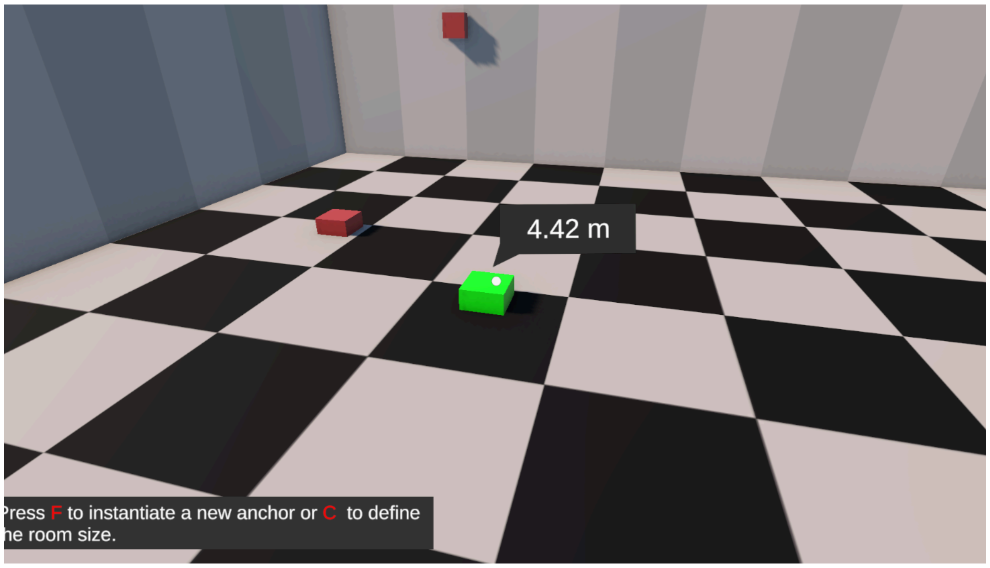
To find the nearest anchor I created a static event and a static reference to the instance of the object, I decided to not pass the Anchor as an argument of the event because it gives me the flexibility to check the nearest anchor at any moment without the need to wait for an event. Another important aspect was the optimization, for making the anchors perform as few checks as possible I made it listen for the anchor's handler **UpdateAnchorPosition** events, these checks are performed only 3 times per second, instead of every frame, and only if the camera is moving.



```
//static
private static event Action OnAnchorChange;
public static Anchor CurrentNearestAnchor;
```

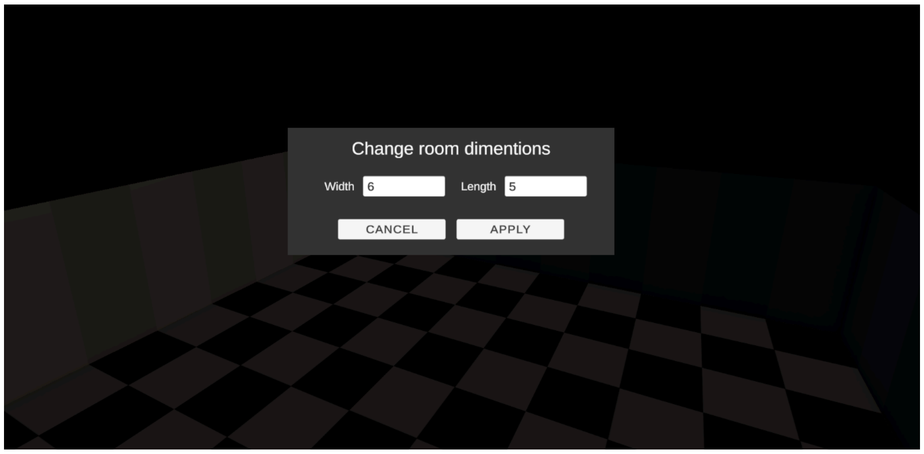
Label

With the public static reference to the nearest anchor, I added a small label in 3D space that always faces the camera and displays the distance from the user formatted in metter.


4.42 m
Press F to instantiate a new anchor or C to define the room size.

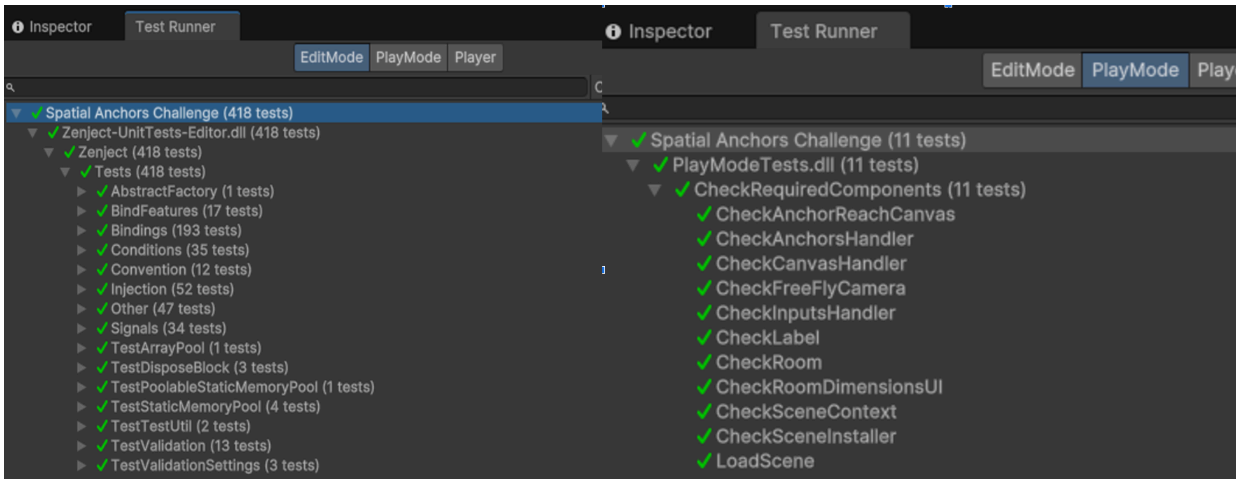
Handling Canvas

The canvas is handled by two main classes, **CanvasHandler** and **CanvasInstance,** the canvas handler looks for all the canvas instances in its child and then adds it to a list of objects. With that, it is possible to toggle canvas visibility easily and add as much canvas as necessary.
The next step was to implement the two modals I created before, one is displayed every time a new anchor is created and the second one listens for a key press and allows to change of the room dimension in playmode.



Unity tests

One of the advantages of using zenject is that it already performs tests that validate all it's dependencies, and ensure the plugin is performing as intended.
To extend it a little bit further, I added a Test that validates if all the needed components are present in the scene. The verifications are simple FindObjectsByType that asserts the presence of the component in the scene. I tried an approach of validating the container binding first, but I had some issues and decided to go with this alternative route.

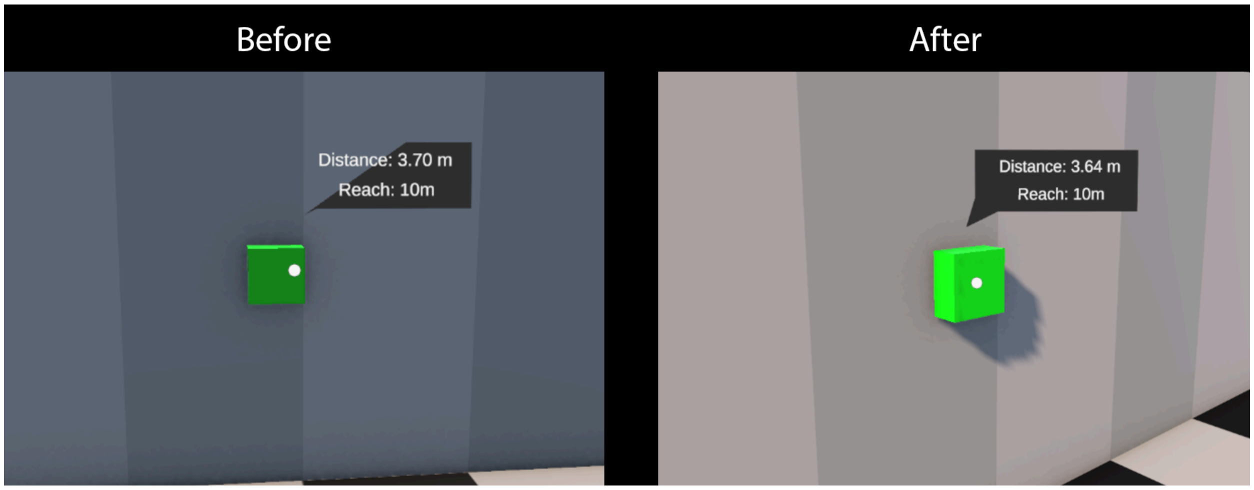


```
▼ ✓ Spatial Anchors Challenge (418 tests)
  ▼ ✓ Zenject-UnitTests-Editor.dll (418 tests)
    ▼ ✓ Zenject (418 tests)
      ▼ ✓ Tests (418 tests)
        ▶ ✓ AbstractFactory (1 tests)
        ▶ ✓ BindFeatures (17 tests)
        ▶ ✓ Bindings (193 tests)
        ▶ ✓ Conditions (35 tests)
        ▶ ✓ Convention (12 tests)
        ▶ ✓ Injection (52 tests)
        ▶ ✓ Other (47 tests)
        ▶ ✓ Signals (34 tests)
        ▶ ✓ TestArrayPool (1 tests)
        ▶ ✓ TestDisposeBlock (3 tests)
        ▶ ✓ TestPoolableStaticMemoryPool (1 tests)
        ▶ ✓ TestStaticMemoryPool (4 tests)
        ▶ ✓ TestTestUtil (2 tests)
        ▶ ✓ TestValidation (13 tests)
        ▶ ✓ TestValidationSettings (3 tests)
```

```
▼ ✓ Spatial Anchors Challenge (11 tests)
  ▼ ✓ PlayModeTests.dll (11 tests)
    ▼ ✓ CheckRequiredComponents (11 tests)
        ✓ CheckAnchorReachCanvas
        ✓ CheckAnchorsHandler
        ✓ CheckCanvasHandler
        ✓ CheckFreeFlyCamera
        ✓ CheckInputsHandler
        ✓ CheckLabel
        ✓ CheckRoom
        ✓ CheckRoomDimensionsUI
        ✓ CheckSceneContext
        ✓ CheckSceneInstaller
        ✓ LoadScene
```

Finishing Touches

To wrap up the project I corrected some UI texts that were misspelled and fixed a small issue with the labels not being visible through the walls. This was fixed by adding special shaders for both the label and the TPM.



Thank you for your attention!