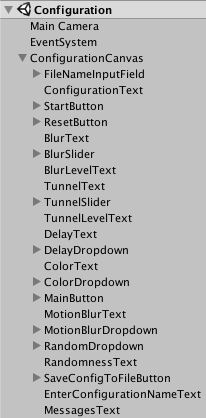
## The Simulating Substances App GUI

Unity uses the metaphor „Scene“ in order to refer to a file, which contains the objects of your game. Scenes can be used to create a main menu, individual levels, and anything else. Think of each unique Scene file as a unique level. In each Scene, you will place your environments, obstacles, and decorations, essentially designing and building your game in pieces.

The Hierarchy Window in Unity is a hierarchical text representation of every object in the scene. The hierarchy reveals the structure of how objects are attached to one another and can became very complex.

Below we explain the four Simulating Substances application scenes ordered by growing complexity. Each scene has a corresponding C sharp script responsible of its control following the Model-View-Controller Pattern.

For a better understanding of the GUI it’s necessary to explain before how is data preserved between scenes.

## Communication between scenes

A scene is an independent entity with its own lifecycle. That means, that when a new scene is loaded (f. e. a new level in a game) all objects from the previous scene are destroyed.

In order to preserve data Unity provides the PlayerPrefs class, which stores and accesses player preferences between game sessions. This class uses a simple Key/Value system to save the data to a file:

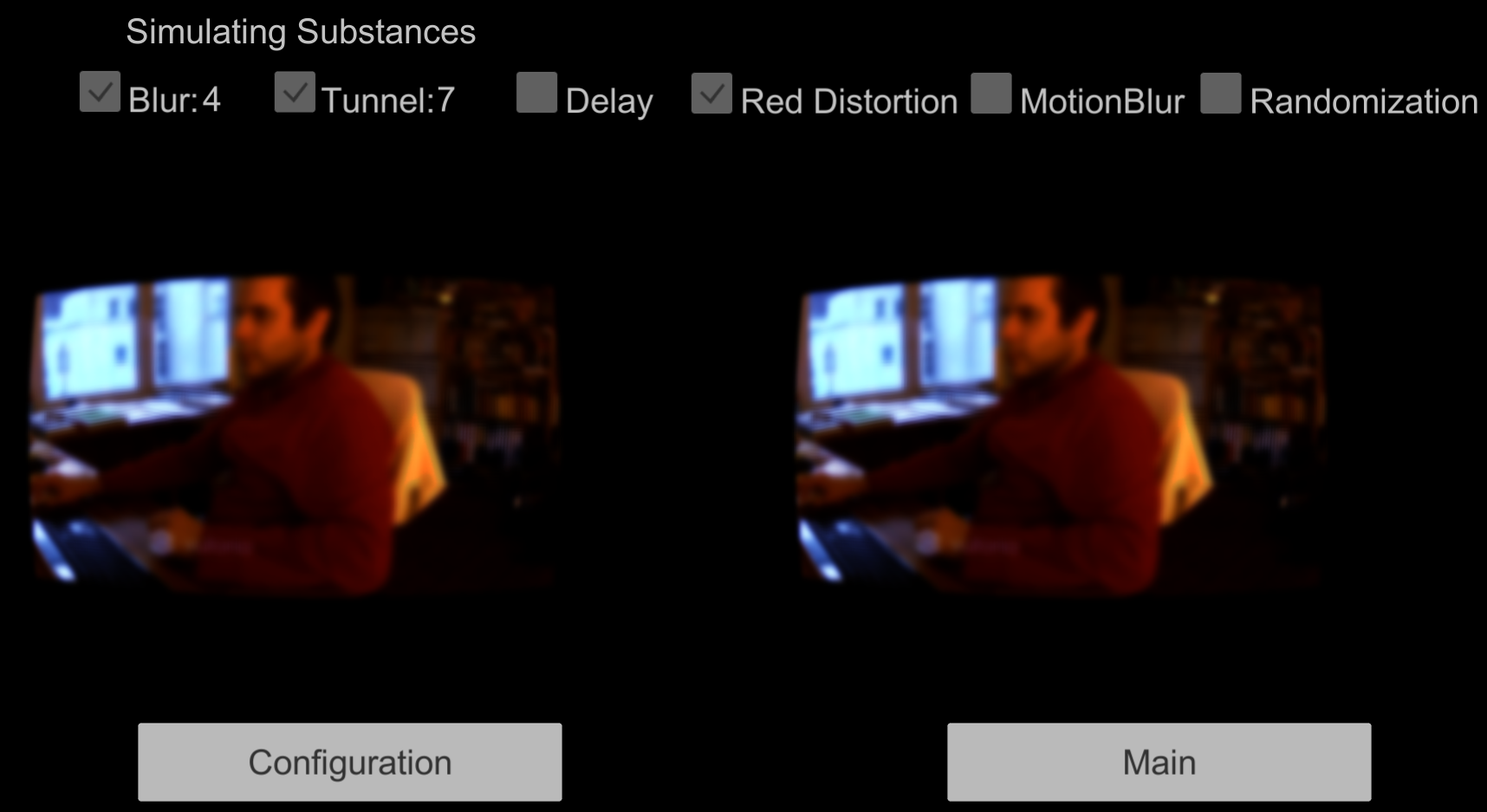
PlayerPrefs.SetInt("Player Score", 10);

var score = PlayerPref.GetInt("Player Score")

We employ this class to preserve the configuration values of the image effects selected by the user. So the user can switch between scenes or even shut down the application without loosing the configuration.

## Simulating Substances scene

This is the scene where all happens. The Vuforia Plugin provides de Augmented Reality functionality and the different Unity Image Filters create the desired effects as blur or tunnel view.



The effect values shown in the upper part are loaded from the PlayerPrefs when the scene starts. In the SimulatingViewModel class:

  UpdateBlurValue(PlayerPrefs.GetFloat(PlayerPreferences.BlurLevel));  
 UpdateTunnelValue(PlayerPrefs.GetFloat(PlayerPreferences.TunnelLevel));  
 UpdateDelay(PlayerPrefs.GetInt(PlayerPreferences.DelayLevel));

 private void UpdateBlurValue(float value)  
        {  
            if (value > 0)  
            {  
                blurValueText.text = value.ToString(); //Shows 4 in the image

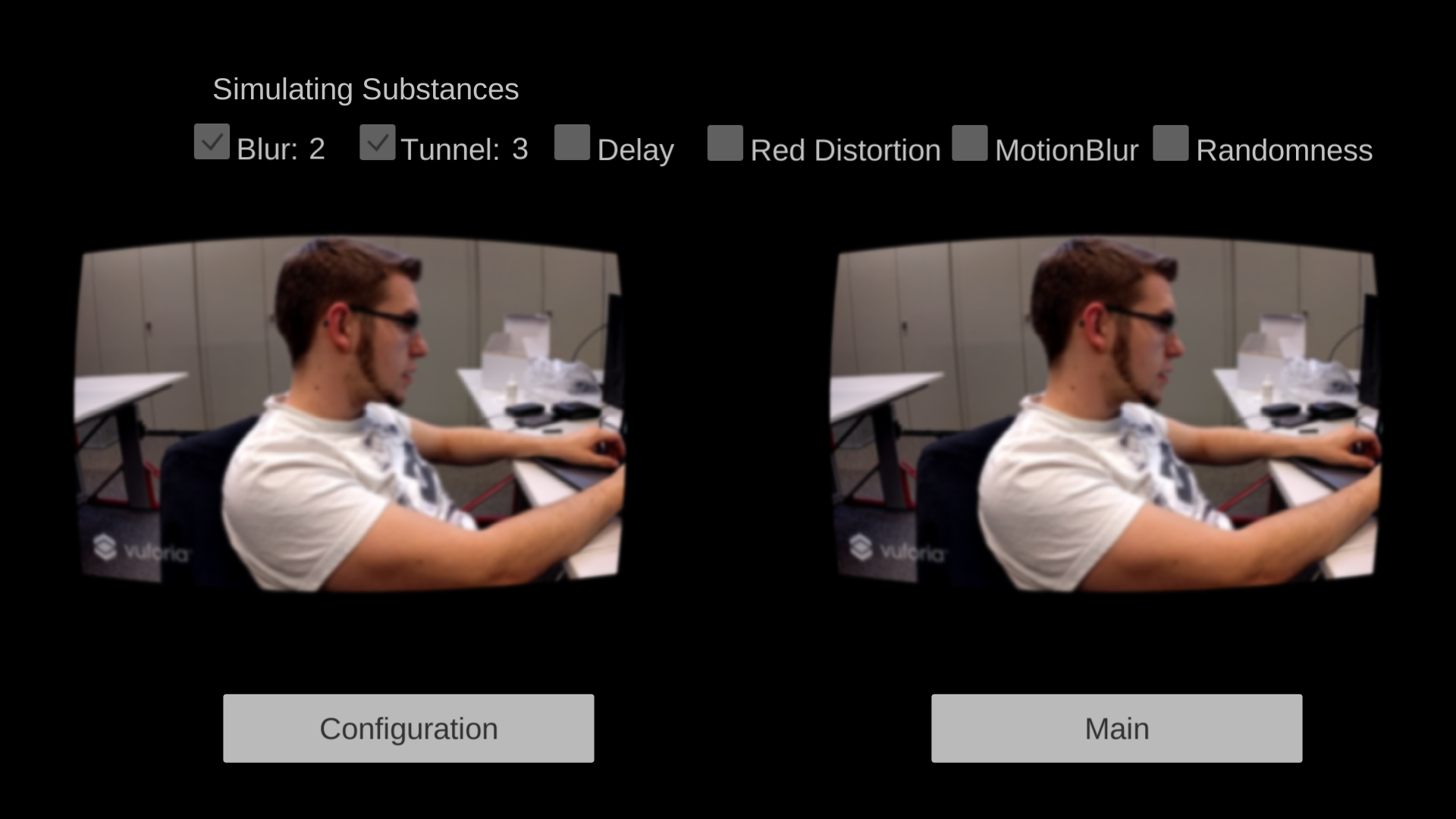
                blurToggle.isOn = true; //Shows the toggle as selected  
            }  
        }

[TODO: Ask Koni if he wants to explain here the Randomization class or in a separated section. If separated, here there should be a reference to the section]

As you can see in the image, this scene contains the application’s principal functionality; it shows the user the augmented reality.

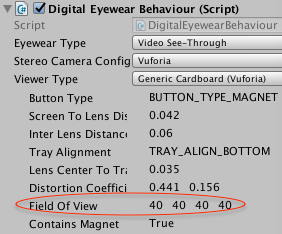
### Problems in the Simulating Substance scene

The main problem we discovered in this scene is the size of the augmented reality windows. A comparison of a Virtual Reality application (left) with the augmented reality:



Although the images are not in scale you can see that the size of the augmented reality windows is much smaller. After long research we could determine that this is a limitation of the Vuforia Plugin used for augmented reality.

The Vuforia Augmented Reality Camera has only a 40 degrees Field of View[[1]](#footnote-1) capacity, and this conditions the windows size.



## Configuration scene

Here you can configure the image effects strength. The configuration is saved automatically when you play start. The scene shows always the last configuration used.

Here is the true augmented reality. You can see the chosen configuration values above the camera and navigate back to the main or configuration scenes.

The navigation between scenes

1. https://docs.unity3d.com/Manual/class-Camera.html [↑](#footnote-ref-1)