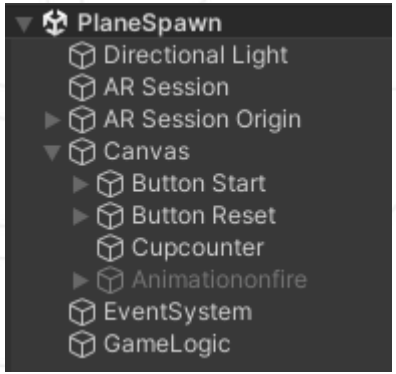
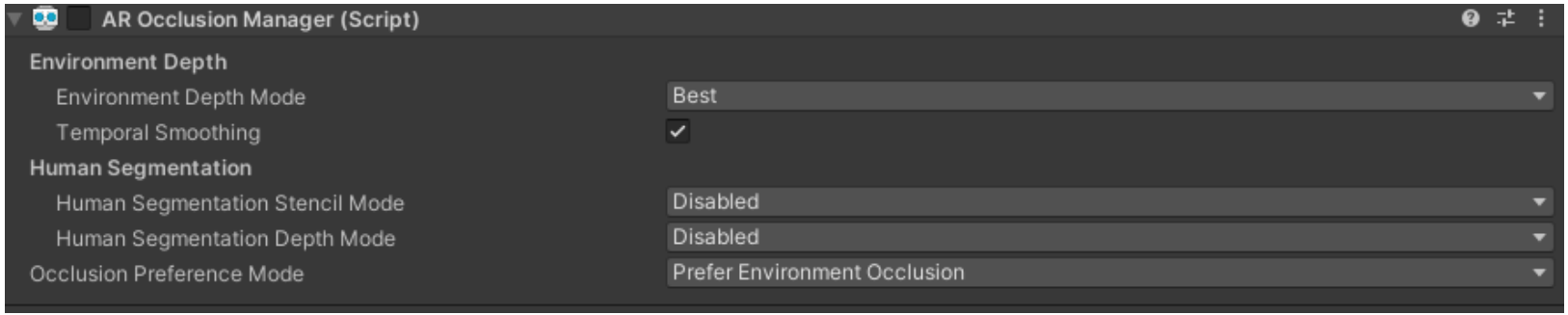
1. Implemented Features

- Placement of Cups with Plane Detection
- Orientation of Cups automatically towards camera
- Spawning/Despawning of Balls/Cups and Throwing Physics
- Reset and Start Button
- Beerpong Rules (On-Fire, Repositioning of Cups)
- UI Animations
- MQTT Connection for multiplayer mode

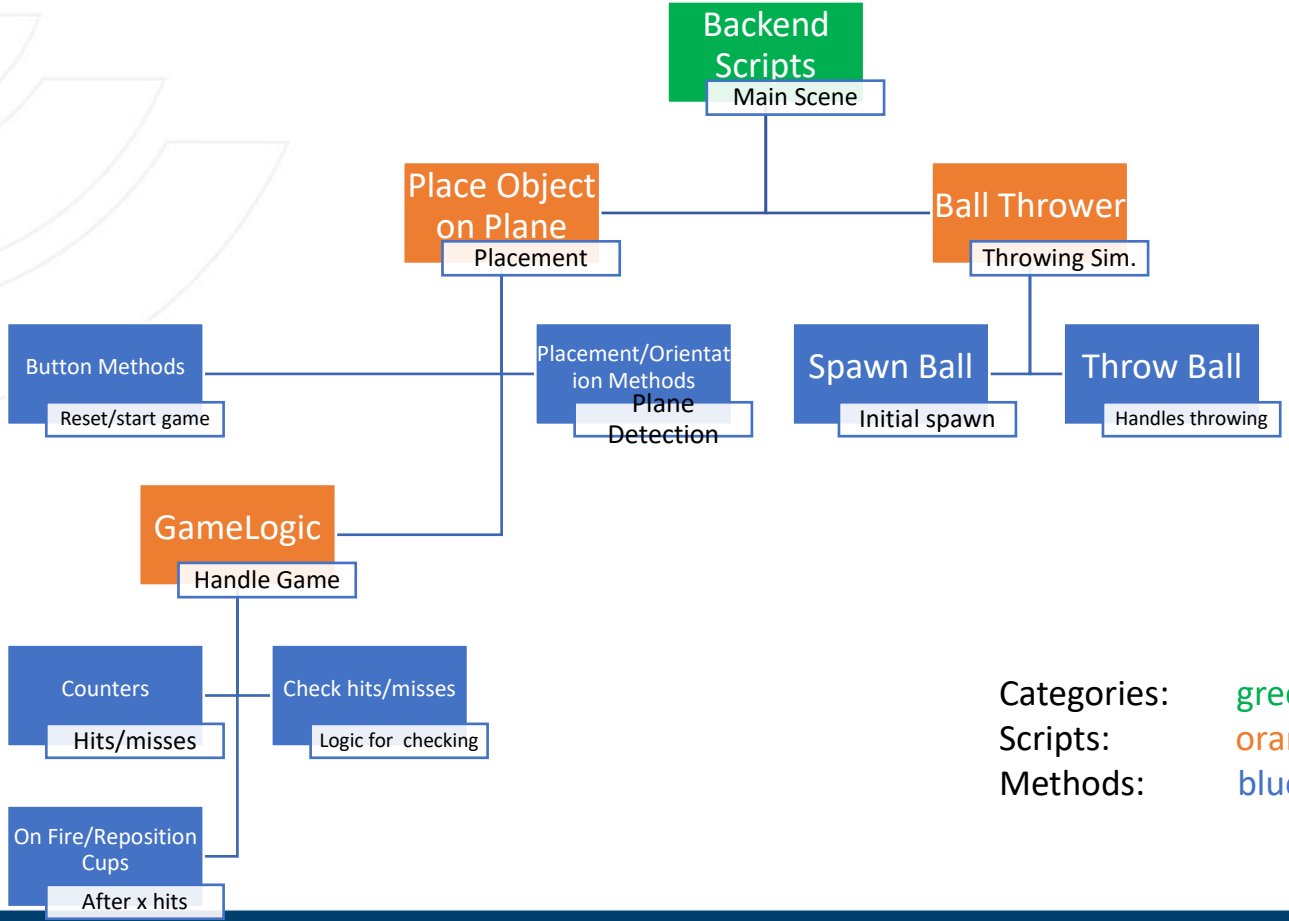
2. Main Scene



- Main Scene runs plane detection
- Occlusion added to AR camera



3. Script Structure Overview



Categories: green
Scripts: orange
Methods: blue

3.1 Backend Examples: Throwing

```
0 references
void Update()
{
    // Access the cupsHitCount from the TriggerAction script
    int beersets = PlaceObjectonPlane.spawnedObjectCount;
    Debug.Log("Beersets" +beersets);

    if (gamestarted)
    {
        if (ballSpawned)
        {
            HandleBallThrow();
        }
        else
        {
            HandleBallSpawn();
        }
    }

}
```

3.1 Backend Examples: GameLogic

```
0 references
void Update()
{
    // Access the cupsHitCount from the TriggerAction script
    int cupsHit = TriggerAction.cupsHitCount;

    // Update the Text element in the UI with the cupsHit count
    if (cupsHitText != null)
    {
        cupsHitText.text = "Hits: " + cupsHit;
    }

    repositioncups();
    OnFire();
    // For testing, print the count to the console
    // Debug.Log("Cups hit: " + cupsHit);
}
```

3.1 Backend Examples: GameLogic

```
1 reference
void OnFire()
{
    Missandhit();
    Debug.Log("misses:"+ misscounter);
    Debug.Log("consecutivehits"+ consecutivehits);
    if(consecutivehits >= 3)
    {
        Debug.Log("on fire!");
        Fireanimation.SetActive(true);
        spriteAnimation.Func_PlayUIAnim();
    }
    else
    {
        spriteAnimation.Func_StopUIAnim();
        Fireanimation.SetActive(false);
    }
}
```

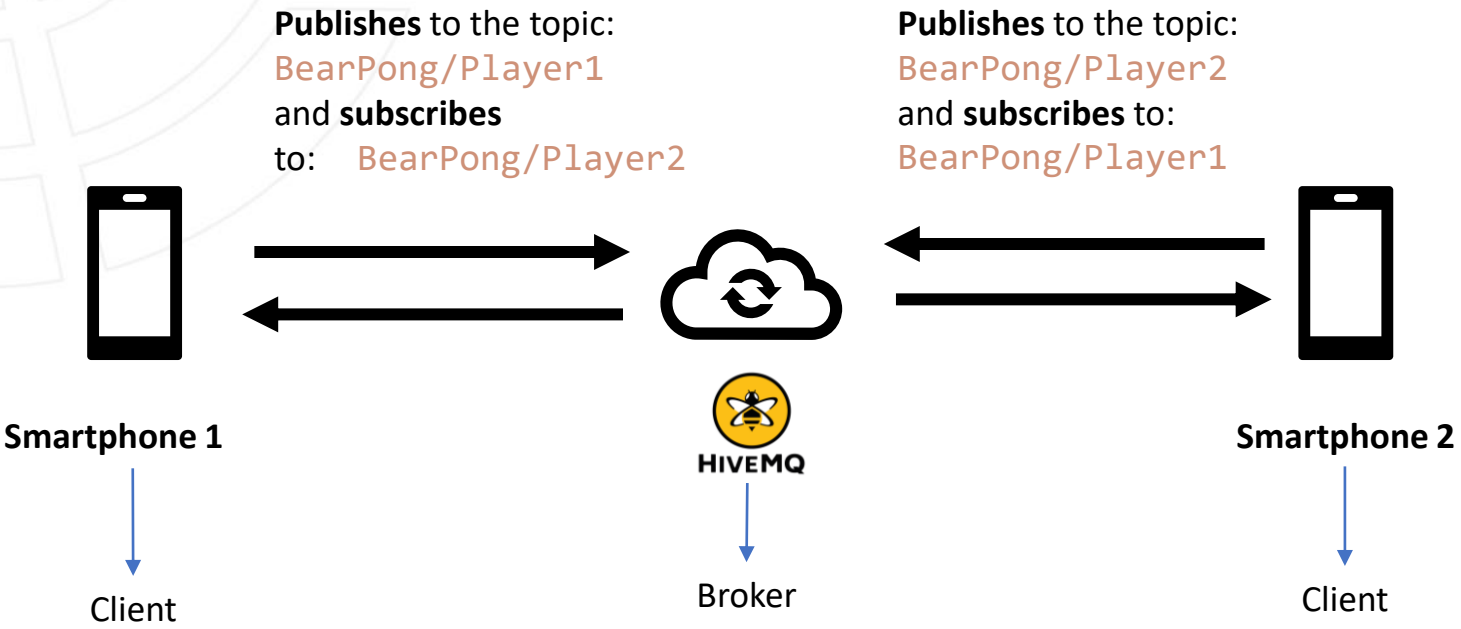
```
1 reference
void repositioncups()
{
    // Access the cupsHitCount from the TriggerAction script
    int cupsHit = TriggerAction.cupsHitCount;

    if(cupsHit >= 5)
    {
        placeObjectonPlane.umstellen();
    }
}
```

3.2 MQTT Connection

- MQTT – **M**essage **Q**ueuing **T**elemetry **T**ransport
- Works in a publisher – subscriber pattern
- The different smartphones can be seen as **clients**
- Publisher (in this case e.g: Player 1/ Smartphone 1) sends data to a specific topic
- Subscriber (Player 2) subscribes to the the topic of Player 1 and gets the information about the score
- Publisher and subscriber do not contact each other directly, everything is via a third-party broker
  - In our case: **broker.hivemq.com**
  - Each smartphone is Subscriber and Publisher at the same time

3.2 MQTT Connection



3.3 Sprite Animation

- Simple 2D Animation as UI overlay => pictures are shown after each other
- Attach Script to image texture in UI canvas
- Script handles start, stop and speed of Sprites