



# POST.SPACES

APPENDIX II: SYSTEMS DOCUMENTATION



# 1 Code Documentation

The Unity platform means that much of the design and programming work takes place across various objects, variables, and linkages, not to mention plugin support with other libraries and assets. Much of the programme structure is difficult to document due to the component hierarchies and connections between the various game objects and prefab instances. However, these are the bulk of the raw scripts that have been manually coded to support the VR environment. Note that due to time limitations they have not been fully cleaned, organised and refactored; this appendix exists for reference and as a proof of work.

Due to the several changes in direction over the course of the year, there have been several different systems and custom scripts that have since been scrapped or superseded; those scripts are not included here to avoid more confusion. All scripts here are active in the final release, but there might be some minor ones which I have left out.

## Contents

<b>1</b>	<b>Code Documentation</b>	<b>1</b>
<b>2</b>	<b>Network Scripts</b>	<b>4</b>
2.1	Network Manager . . . . .	4
2.2	Network Player Spawner . . . . .	7
2.3	Network Avatar . . . . .	8
2.4	Avatar Random Customiser . . . . .	12

2.5	Grabbable Photon Ownership . . . . .	13
<b>3</b>	<b>Player and XR Scripts</b>	<b>14</b>
3.1	Player Controller . . . . .	14
3.2	Movement Provider . . . . .	15
3.3	XR Controller Manager . . . . .	20
3.4	Teleportation Anchored Area . . . . .	24
3.5	Comment Frame Teleporter . . . . .	25
3.6	Custom Teleportation Provider . . . . .	26
<b>4</b>	<b>Reddit General Scripts</b>	<b>28</b>
4.1	Reddit Controller . . . . .	28
4.2	Reddit Central Space . . . . .	34
4.3	Reddit Central Sphere . . . . .	40
4.4	Reddit Cube Spawner . . . . .	41
4.5	Reddit Node Block . . . . .	43
4.6	Reddit Node Sphere . . . . .	44
4.7	Subreddit Star . . . . .	46
4.8	Reddit Info Panel . . . . .	47
4.9	Geometry Torus . . . . .	48
<b>5</b>	<b>Reddit Post Scripts</b>	<b>52</b>
5.1	Reddit Post Controller . . . . .	52
5.2	Reddit Comment Frame . . . . .	65
5.3	Reddit Comment Child . . . . .	71
5.4	Vote Controller . . . . .	73

5.5	Refresh Comments . . . . .	76
<b>6</b>	<b>General Support Scripts</b>	<b>78</b>
6.1	Scroll Surface . . . . .	78
6.2	Frame Dynamic . . . . .	80
6.3	Frame Dynamic Planes . . . . .	83
6.4	Text Frame . . . . .	84
6.5	Material Manager . . . . .	87
6.6	Lib Color . . . . .	89
6.7	ColorRandomiser . . . . .	89
6.8	ColorChanger . . . . .	90

## 2 Network Scripts

Network management has been built using the Photon suite of network tools, allowing both object and variable synchronization as well as an additional plugin that works together for voice communication management. Most operations are performed client-side and the relevant data regarding synced objects, players, etc are sent over the network and objects are moved or instantiated on the other client.

### 2.1 Network Manager

The Network Manager handles most of the general network syncing operations, such as joining/leaving the network and updating player stats (i.e. which subreddit they are currently in).

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Photon.Pun;
5 using Photon.Realtime;
6 using ExitGames.Client.Photon;
7 using UnityEngine.XR.Interaction.Toolkit;
8
9
10
11 public class NetworkManager : MonoBehaviourPunCallbacks
12 {
13     public GameObject rig;
14     public RedditController redditController;
15     private string localSubreddit { get; set; }
16     private ExitGames.Client.Photon.Hashtable playerProperties;
17     private Dictionary<Player, NetworkAvatar> playerAvatarDict = new
18         Dictionary<Player, NetworkAvatar>();
19     // Start is called before the first frame update
20     void Start()
```

```

20     {
21         rig = GameObject.Find("XR Rig");
22         playerProperties = PhotonNetwork.LocalPlayer.CustomProperties;
23         ConnectToServer();
24     }
25
26     void ConnectToServer()
27     {
28         PhotonNetwork.ConnectUsingSettings();
29         Debug.Log("Try Connect To Server...");
30     }
31
32     public override void OnConnectedToMaster()
33     {
34         Debug.Log("Connected To Server.");
35
36         base.OnConnectedToMaster();
37         RoomOptions roomOptions = new RoomOptions();
38         roomOptions.MaxPlayers = 16;
39         roomOptions.IsVisible = true;
40         roomOptions.IsOpen = true;
41
42         PhotonNetwork.JoinOrCreateRoom("all", roomOptions, TypedLobby.
            Default);
43     }
44
45     public override void OnJoinedRoom()
46     {
47         Debug.Log("Joined a Room: " + PhotonNetwork.CurrentRoom);
48         base.OnJoinedRoom();
49     }
50
51     public override void OnPlayerEnteredRoom(Player newPlayer)
52     {
53         Debug.Log("A new player joined the room.");
54         base.OnPlayerEnteredRoom(newPlayer);
55     }
56
57     public override void OnDisconnected(DisconnectCause cause)
58     {
59         Debug.Log("PUN DISCONNECTED: " + cause);
60         Invoke("ConnectToServer", 20f);
61     }
62
63
64     public override void OnPlayerPropertiesUpdate (Player targetPlayer,
        ExitGames.Client.Photon.Hashtable changedProps)
65     {
66         Debug.Log("player properties update: " + targetPlayer + " - " +
            changedProps);

```

```

67
68     if(changedProps.ContainsKey("subreddit"))
69     {
70         if(targetPlayer != PhotonNetwork.LocalPlayer)
71         {
72             if(changedProps["subreddit"].ToString() == localSubreddit)
73             {
74                 playerAvatarDict[targetPlayer].EnableSelf();
75                 Debug.Log(targetPlayer + " has entered your subreddit.");
76                 ;
77             }
78             else
79             {
80                 playerAvatarDict[targetPlayer].DisableSelf();
81                 Debug.Log(targetPlayer + " has left your subreddit.");
82             }
83         }
84     }
85
86     public void SetPlayerSubreddit(string subreddit)
87     {
88         localSubreddit = subreddit;
89         rig.GetComponent<XRRig>().MatchRigUpRigForward(new Vector3(0, 3, 0),
90             new Vector3(1, 0, 0));
91         rig.transform.position = new Vector3(0, -3, 0);
92
93         //Set network player property
94         if (!playerProperties.ContainsKey("subreddit"))
95         {
96             playerProperties.Add("subreddit", subreddit);
97             Debug.Log("Property added");
98         }
99         playerProperties["subreddit"] = subreddit;
100         PhotonNetwork.LocalPlayer.SetCustomProperties(playerProperties);
101
102         //Initialise posts
103         redditController.Initialise(subreddit);
104     }
105
106     public void AddPlayerAvatar(Player player, NetworkAvatar avatar)
107     {
108         if(!playerAvatarDict.ContainsKey(player))
109         {
110             playerAvatarDict.Add(player, avatar);
111             Debug.Log("Added " + playerAvatarDict[player] + " to list of
112                 player avatars.");
113         }
114     }

```



114 }

## 2.2 Network Player Spawner

The Network Player Spawner is responsible for spawning in player avatars.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Photon.Pun;
5
6 public class NetworkPlayerSpawner : MonoBehaviourPunCallbacks
7 {
8     private GameObject spawnedPlayerPrefab;
9     private Color shirtColor;
10    private Color pantsColor;
11    private object[] avatarInitData = new object[6];
12
13    void Start()
14    {
15        //Randomise clothes
16        //shirt color
17        avatarInitData[0] = Random.Range(80, 230);
18        avatarInitData[1] = Random.Range(80, 230);
19        avatarInitData[2] = Random.Range(80, 230);
20
21        //pants color
22        avatarInitData[3] = Random.Range(20, 70);
23        avatarInitData[4] = Random.Range(20, 70);
24        avatarInitData[5] = Random.Range(20, 70);
25
26    }
27
28    public override void OnJoinedRoom()
29    {
30        base.OnJoinedRoom();
31        spawnedPlayerPrefab = PhotonNetwork.Instantiate("Network Avatar",
32            transform.position, transform.rotation, 0, avatarInitData);
33    }
34
35    public override void OnLeftRoom()
36    {
37        base.OnLeftRoom();
38    }
39 }
```

```

37     PhotonNetwork.Destroy(spawnedPlayerPrefab);
38 }
39 }

```

## 2.3 Network Avatar

The Network Avatar script is an important script that controls the positioning, animation and behaviour of each player avatar. These avatars are spawned client-side for each player in the server and are synced over the Photon network.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR;
5  using Photon.Pun;
6  using Photon.Realtime;
7  using Photon.Voice.Unity;
8
9  public class NetworkAvatar : MonoBehaviour, IPunInstantiateMagicCallback
10 {
11     public Transform head;
12     public List<Transform> meshes;
13     public Transform character;
14     public bool hideSelf = true;
15     public bool hideHead = true;
16     public float cameraFrontOffset;
17     public GameObject speaker;
18     private Vector3 baseVector;
19     private bool parented = false;
20     private GameObject rig;
21     private Animator animator;
22     private NetworkManager networkManager;
23     //private int interval = 1;
24     //private float animUpdateTime = 0;
25
26     private PhotonView photonView;
27
28     // Start is called before the first frame update
29     void Awake()
30     {
31         photonView = GetComponent<PhotonView>();

```

```

32     rig = GameObject.Find("XR Rig");
33     networkManager = GameObject.Find("NetworkManager").GetComponent<
        NetworkManager>();
34     animator = GetComponentInChildren<Animator>();
35     if(photonView.IsMine) MapCharacterPosition(character, XRNode.Head);
36 }
37
38 public void OnPhotonInstantiate(PhotonMessageInfo info)
39 {
40     object[] data = info.photonView.InstantiationData;
41     if (data != null)
42     {
43         SetColor(new Color((int)data[0]/255f, (int)data[1]/255f, (int)
            data[2]/255f), new Color((int)data[3]/255f, (int)data[4]/255
            f, (int)data[5]/255f));
44     }
45     networkManager.AddPlayerAvatar(photonView.Owner, this);
46 }
47
48 // Update is called once per frame
49 void Update()
50 {
51     if (photonView.IsMine)
52     {
53         if (hideSelf)
54         {
55             foreach (Transform mesh in meshes)
56             {
57                 mesh.gameObject.SetActive(false);
58             }
59         }
60         if (hideHead || hideSelf)
61         {
62             head.gameObject.SetActive(false);
63         }
64
65         if(rig.GetComponent<MovementProvider>().inCenter && rig)
66         {
67             MapCharacterPosition(character, XRNode.Head);
68         }
69         //else if(parented) MapLocalPosition(character, XRNode.Head);
70         Animate();
71     }
72 }
73
74 void MapPosition(Transform target, XRNode node)
75 {
76     InputDevices.GetDeviceAtXRNode(node).TryGetFeatureValue(CommonUsages
        .devicePosition, out Vector3 position);

```

```

77     InputDevices.GetDeviceAtXRNode(node).TryGetFeatureValue(CommonUsages
78         .deviceRotation, out Quaternion rotation);
79
80     target.position = position;
81     target.rotation = rotation;
82 }
83
84 void MapCharacterPosition(Transform characterModel, XRNode node)
85 {
86     InputDevices.GetDeviceAtXRNode(node).TryGetFeatureValue(CommonUsages
87         .devicePosition, out Vector3 position);
88     InputDevices.GetDeviceAtXRNode(node).TryGetFeatureValue(CommonUsages
89         .deviceRotation, out Quaternion rotation);
90
91     Transform rigTransform = rig.GetComponent<Transform>();
92
93     if(rigTransform.rotation.eulerAngles.x == 0)
94     {
95         Quaternion snapTurnOffset = rigTransform.rotation;
96         Vector3 characterControllerOffset = rigTransform.position;
97         //Vector3 headPosition = new Vector3(characterControllerOffset.x
98             , characterControllerOffset.y, characterControllerOffset.z);
99         Vector3 headPosition = new Vector3(characterControllerOffset.x +
100             (snapTurnOffset * position).x, characterControllerOffset.y,
101             characterControllerOffset.z + (snapTurnOffset * position).z
102             );
103         //Vector3 headRotation = new Vector3(0, rotation.eulerAngles.y +
104             (snapTurnOffset.eulerAngles.y), 0);
105         Vector3 headRotation = new Vector3(0, (snapTurnOffset.
106             eulerAngles.y), 0);
107         Vector3 headPositionOffset = Quaternion.Euler(headRotation) *
108             new Vector3(cameraFrontOffset, 0, 0);
109         //Debug.Log(headPositionOffset);
110
111         characterModel.position = headPosition - headPositionOffset;
112         characterModel.rotation = Quaternion.LookRotation(rigTransform.
113             right, rigTransform.up);
114         //Debug.Log(Quaternion.LookRotation(rigTransform.up,
115             rigTransform.forward));
116         //characterModel.rotation = Quaternion.Euler(headRotation);
117
118         //Parent to XR rig and stop moving dynamically
119         transform.SetParent(rig.transform);
120         baseVector = characterModel.localPosition;
121         parented = true;
122         //Invoke("Parented", 5f);
123     }
124 }
125
126 void MapLocalPosition(Transform characterModel, XRNode node)

```

```

115     {
116         InputDevices.GetDeviceAtXRNode(node).TryGetFeatureValue(CommonUsages
            .devicePosition, out Vector3 position);
117         Transform rigTransform = rig.GetComponent<Transform>();
118
119         Vector3 headposition = new Vector3(position.x, 0, position.z);
120
121         characterModel.localPosition = baseVector + headposition;
122     }
123
124     private void Parented()
125     {
126         parented = true;
127     }
128
129     public void SetColor(Color shirt, Color pants)
130     {
131         transform.Find("Torso Male Short Sleeves").GetComponent<
            SkinnedMeshRenderer>().material.SetColor("_BaseColor", shirt);
132         transform.Find("Pants Male").GetComponent<SkinnedMeshRenderer>().
            material.SetColor("_BaseColor", pants);
133     }
134
135     private void Animate()
136     {
137         // Blend between walk/run
138         //float blend = rig.GetComponent<CharacterController>().velocity.
            magnitude;
139         //animator.SetFloat("Move", blend);
140     }
141
142     public void MuteSelf()
143     {
144         speaker.SetActive(false);
145     }
146
147     public void UnmuteSelf()
148     {
149         speaker.SetActive(true);
150     }
151
152     public void DisableSelf()
153     {
154         foreach (Transform mesh in meshes)
155         {
156             mesh.gameObject.SetActive(false);
157         }
158         head.gameObject.SetActive(false);
159         //MuteSelf();
160     }

```

```

161
162     public void EnableSelf()
163     {
164         foreach (Transform mesh in meshes)
165         {
166             mesh.gameObject.SetActive(true);
167         }
168         head.gameObject.SetActive(true);
169         //UnmuteSelf();
170     }
171
172     public Player GetOwner()
173     {
174         return photonView.Owner;
175     }
176
177 }

```

## 2.4 Avatar Random Customiser

Small script to handle randomisation of avatar shirt colors, so players can tell each other apart.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class AvatarRandomCustomiser : MonoBehaviour
6 {
7
8     // Start is called before the first frame update
9     void Start()
10    {
11
12
13    }
14
15    // Update is called once per frame
16    void Update()
17    {
18
19    }
20
21    public void SetColor(Color shirt, Color pants)

```

```

22     {
23         transform.Find("Torso Male Short Sleeves").GetComponent<
                SkinnedMeshRenderer>().material.SetColor("_BaseColor", shirt);
24         transform.Find("Pants Male").GetComponent<SkinnedMeshRenderer>().
                material.SetColor("_BaseColor", pants);
25     }
26 }

```

## 2.5 Grabbable Photon Ownership

This script handles the transfer of ownership when different players want to move synced objects.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR;
5  using UnityEngine.XR.Interaction.Toolkit;
6  using Photon.Realtime;
7  using Photon.Pun;
8
9  [RequireComponent( typeof( PhotonView ) )]
10 public class GrabbablePhotonOwnership : MonoBehaviourPun
11 {
12     private XRGrabInteractable interactable = null;
13     private void Awake()
14     {
15         interactable = GetComponent<XRGrabInteractable>();
16     }
17     private void OnEnable()
18     {
19         interactable.onSelectEnter.AddListener(RequestOwnership);
20     }
21     private void RequestOwnership(XRBaseInteractor interactor)
22     {
23         if( this.photonView.Owner == PhotonNetwork.LocalPlayer )
24         {
25             Debug.Log( "Not requesting ownership. Already mine." );
26             return;
27         }
28
29         this.photonView.RequestOwnership();
30         Debug.Log( "Requesting ownership" );
31     }

```

```
32  
33 }
```

## 3 Player and XR Scripts

The main behaviour for VR/XR comes from the Unity XR plugin and is configured via existing script components and options. However, custom scripts are still needed to support all the additional behaviours available to the player as well as some custom functions for controls and teleportation not supported by the base package.

### 3.1 Player Controller

Simple script that sets the player's base position, should really be merged into another script because the player rig has lots of different scripts attached to it.

```
1 using System.Collections;  
2 using System.Collections.Generic;  
3 using UnityEngine;  
4 using Photon.Pun;  
5  
6  
7 public class PlayerController : MonoBehaviour  
8 {  
9     private Vector3 startPosition;  
10    public int numVoiceRooms = 0;  
11  
12  
13    // Start is called before the first frame update  
14    void Start()  
15    {  
16        startPosition = transform.position;
```



```

17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22
23     }
24
25     private void ResetPlayerPosition()
26     {
27         transform.position = startPosition;
28     }
29
30     public void TeleportPlayer(Vector3 target)
31     {
32
33     }
34
35 }

```

## 3.2 Movement Provider

Main script controlling player movement, gravity, etc.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.XR;
5 using UnityEngine.XR.Interaction.Toolkit;
6
7 public class MovementProvider : LocomotionProvider
8 {
9     public float speed = 1.0f;
10    public float gravityMultiplier = 1.0f;
11    public bool gravity = true;
12    public List<XRController> controllers = null;
13    public TeleportationProvider teleportationProvider;
14    private CharacterController characterController = null;
15    private GameObject head = null;
16    private bool resetClick = true;
17    public bool relativeGravity = true;
18    public bool inCenter = true;
19
20
21    //CUSTOM GRAVITY VARIABLES

```

```

22     float distanceToGround;
23     Vector3 Groundnormal;
24     bool OnGround = false;
25
26     protected override void Awake()
27     {
28         characterController = GetComponent<CharacterController>();
29         head = GetComponent<XRRig>().cameraGameObject;
30     }
31
32     // Start is called before the first frame update
33     private void Start()
34     {
35         PositionController();
36     }
37
38     private void Update()
39     {
40         // Ground Control (GRAVITY SCRIPT NOT WRITTEN BY ME)
41         RaycastHit hit = new RaycastHit();
42         if (Physics.Raycast(transform.position, -transform.up, out hit, 10))
43         {
44
45             distanceToGround = hit.distance;
46             Groundnormal = hit.normal;
47
48             if (distanceToGround <= 0.2f)
49             {
50                 OnGround = true;
51             }
52             else
53             {
54                 OnGround = false;
55             }
56
57         }
58     }
59 }
60
61 // Update is called once per frame
62 private void FixedUpdate()
63 {
64     PositionController();
65     if (!relativeGravity)
66     {
67         if (gravity)
68             ApplyGravity();
69         else if (characterController.transform.position.y <= 9)
70             gravity = true;
71     }

```

```

72     else
73     {
74         if(inCenter)
75             ApplyGravity();
76         //if relative gravity is on
77         //ApplyRelativeGravity();
78     }
79 }
80
81 public void PositionController()
82 {
83     //Get the head in local, playspace ground
84     float headHeight = Mathf.Clamp(head.transform.localPosition.y, 0.5f,
85         2);
86     characterController.height = headHeight;
87
88     //Cut in half, add skin
89     Vector3 newCenter = Vector3.zero;
90     newCenter.y = characterController.height / 2;
91     newCenter.y += characterController.skinWidth;
92
93     //Move the capsule in local space as well
94     newCenter.x = head.transform.localPosition.x;
95     newCenter.z = head.transform.localPosition.z;
96
97     //Apply
98     characterController.center = newCenter;
99 }
100 private void CheckForInput()
101 {
102 }
103 }
104
105 public void StartMove(Vector2 position)
106 {
107     //Add forward vector
108     Vector3 direction = new Vector3(position.x, 0, position.y);
109     Vector3 headRotation = head.transform.eulerAngles; //new Vector3(0,
110         head.transform.eulerAngles.y, 0);
111     Vector3 bodyRotation = transform.up;
112
113     direction = head.transform.rotation * direction; //raw direction
114     pointing in direction player head
115     direction = direction - bodyRotation * Vector3.Dot(direction,
116         bodyRotation); //direction constrained to XY plane of player, by
117     subtracting scalar projection of normal
118
119     //Apply speed and move
120     Vector3 movement = direction.normalized * speed;

```

```

117     characterController.Move(movement * Time.deltaTime);
118     XR_TestVariables.XRMoveCount += Vector3.Magnitude(movement * Time.
        deltaTime);
119 }
120
121 public void StartMove(Vector3 position)
122 {
123     //Add forward vector
124     Vector3 direction = position;
125     Vector3 headRotation = head.transform.eulerAngles; //new Vector3(0,
        head.transform.eulerAngles.y, 0);
126     Vector3 bodyRotation = transform.up;
127
128     direction = head.transform.rotation * direction; //raw direction
        pointing in direction player head
129     //direction = direction - bodyRotation * Vector3.Dot(direction,
        bodyRotation); //direction constrained to XY plane of player, by
        subtracting scalar projection of normal
130
131     //Apply speed and move
132     Vector3 movement = direction.normalized * speed;
133     characterController.Move(movement);
134     XR_TestVariables.XRMoveCount += Vector3.Magnitude(movement * Time.
        deltaTime);
135 }
136
137 private void ApplyGravity()
138 {
139     Vector3 gravity = new Vector3(0, Physics.gravity.y *
        gravityMultiplier, 0);
140     gravity.y *= Time.deltaTime;
141
142     characterController.Move(gravity);
143 }
144
145 private void ApplyRelativeGravity()
146 {
147     //Vector3 sphericalGravDirection = new Vector3( transform.position.
        normalized.z, 0, transform.position.normalized.y);
148     Vector3 relativeGravDirection = -transform.up;
149     Vector3 gravityTime = -relativeGravDirection * Physics.gravity.y *
        gravityMultiplier * Time.deltaTime;
150
151     /*
152     if (OnGround == false)
153     {
154         characterController.Move(gravityTime);
155     }
156     */
157

```

```

158     characterController.Move(gravityTime);
159
160     //Quaternion toRotation = Quaternion.FromToRotation(transform.up,
161         Groundnormal) * transform.rotation;
162     //transform.rotation = toRotation;
163 }
164
165 private void HighJump(bool jump)
166 {
167     Vector3 highJump = new Vector3(0, 11 - characterController.transform
168         .position.y, 0);
169
170     if (jump)
171     {
172         characterController.Move(highJump);
173         gravity = false;
174     }
175     else
176     {
177         characterController.Move(-highJump);
178         gravity = true;
179     }
180 }
181
182 public void VertMove(float magnitude)
183 {
184     //Debug.Log("VertMove " + magnitude);
185     characterController.Move(transform.up * magnitude * Time.deltaTime);
186     //var moveVector3 = new Vector3(0, magnitude, 0);
187     //StartMove(moveVector3);
188 }
189
190 void OnTriggerEnter(Collider collider)
191 {
192     if(collider.gameObject.name == "CenterSphere")
193     {
194         inCenter = true;
195         Debug.Log("in center!");
196     }
197 }
198
199 void OnTriggerExit(Collider collider)
200 {
201     if(collider.gameObject.name == "CenterSphere")
202     {
203         inCenter = false;
204         Debug.Log("out of center!");
205     }
206 }

```

```
206
207 }
```

### 3.3 XR Controller Manager

Custom script I wrote that handles higher-level operations with the XR controllers, allowing behaviour like toggling raycast lines instead of having them on permanently.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 using UnityEngine.XR.Interaction.Toolkit.UI;
6 using UnityEngine.EventSystems;
7
8 namespace UnityEngine.XR.Interaction.Toolkit
9 {
10     public class XRControllerManager : MonoBehaviour
11     {
12         public XRController LeftController;
13         public XRController RightController;
14         public XRController TeleportController;
15         public List<XRController> controllers;
16         private MovementProvider movementProvider;
17         private XRRayInteractor TeleportInteractor;
18         private XRInteractorLineVisual TeleportLine;
19         private XRRayInteractor GrabRayInteractor;
20         private XRInteractorLineVisual GrabRayLine;
21         private GameObject rig;
22         private GameObject cameraOffset;
23         private GameObject teleportReticle;
24         private GameObject teleportPlane;
25         public bool TeleportMode = true;
26         //public Dictionary<string, GameObject> spawnPrefabsList = new
27             Dictionary<string, GameObject>();
28
29         // Start is called before the first frame update
30         void Start()
31         {
32             rig = GameObject.Find("XR Rig");
33             movementProvider = transform.GetComponent<MovementProvider>();
```

```

34     cameraOffset = rig.transform.Find("Camera Offset").gameObject;
35     teleportReticle = cameraOffset.transform.Find("
36         GroundTeleportReticle").gameObject;
37     teleportPlane = cameraOffset.transform.Find("MagicTeleportPlane"
38         ).gameObject;
39
40     TeleportInteractor = TeleportController.GetComponent<
41         XRRayInteractor>();
42     TeleportLine = TeleportController.GetComponent<
43         XRInteractorLineVisual>();
44     GrabRayInteractor = RightController.GetComponent<XRRayInteractor
45         >();
46     GrabRayLine = RightController.GetComponent<
47         XRInteractorLineVisual>();
48
49     CheckTeleportMode();
50
51     //spawnPrefabsList.Add("anchorBush", Resources.Load("Blobs/
52         BlobAnchorDynamic_Bush", typeof(GameObject)) as GameObject);
53 }
54
55 // Update is called once per frame
56 void FixedUpdate()
57 {
58     CheckForInput();
59     CheckForTeleport();
60     CheckInteractorRay();
61 }
62
63 private bool resetClick, resetClick2 = true;
64
65 private void CheckForInput()
66 {
67     if (TeleportController.inputDevice.TryGetFeatureValue(
68         CommonUsages.secondary2DAxisClick, out bool click))
69     {
70         {
71             if (click && resetClick)
72             {
73                 TeleportMode = !TeleportMode;
74                 CheckTeleportMode();
75                 resetClick = false;
76             }
77             else if (!click)
78             {
79                 resetClick = true;
80             }
81         }
82     }
83
84     if (TeleportController.inputDevice.TryGetFeatureValue(
85         CommonUsages.primary2DAxisClick, out bool click2))

```

```

75     {
76         if (click2 && resetClick2)
77         {
78             TeleportMode = !TeleportMode;
79             CheckTeleportMode();
80             resetClick2 = false;
81         }
82         else if (!click2)
83         {
84             resetClick2 = true;
85         }
86     }
87
88     /*
89     foreach (XRController controller in controllers)
90     {
91         if (controller.enableInputActions)
92             CheckForMovement(controller.inputDevice);
93     }
94     */
95
96     CheckForMovement(LeftController.inputDevice);
97 }
98
99 private void CheckForMovement(InputDevice device)
100 {
101     if (device.TryGetFeatureValue(CommonUsages.primary2DAxis, out
102         Vector2 position))
103     {
104         //movementProvider.StartMove(position);
105         //Debug.Log(position);
106     }
107     if (device.TryGetFeatureValue(CommonUsages.secondary2DAxis, out
108         Vector2 position2) && position2.magnitude > 0.2f) //check >
109         0.2f for deadzone
110     {
111         movementProvider.StartMove(position2);
112         //Debug.Log(position2.magnitude);
113     }
114     /*
115     //"High Jump" float function
116     if (device.TryGetFeatureValue(CommonUsages.secondaryButton, out
117         bool click))
118     {
119         if (click && resetClick)
120         {
121             if (characterController.transform.position.y <= 12)
122             {
123                 HighJump(true);
124                 resetClick = false;

```



```

121         }
122         else
123         {
124             HighJump(false);
125             resetClick = false;
126         }
127     }
128     else if (!click && !resetClick)
129         resetClick = true;
130 }
131 */
132 }
133
134 private void CheckForTeleport()
135 {
136     if (TeleportController.inputDevice.TryGetFeatureValue(
137         CommonUsages.triggerButton, out bool triggerValue) &&
138         triggerValue)
139     {
140         TeleportInteractor.enabled = true;
141         TeleportLine.enabled = true;
142         if (TeleportMode == true)
143         {
144             teleportReticle.SetActive(true);
145         }
146         else teleportReticle.SetActive(false);
147     }
148     else if (!triggerValue && TeleportInteractor.enabled)
149     {
150         TeleportInteractor.enabled = false;
151         TeleportLine.enabled = false;
152         teleportReticle.SetActive(false);
153     }
154 }
155
156 private void CheckInteractorRay()
157 {
158     if (RightController.inputDevice.TryGetFeatureValue(CommonUsages.
159         triggerButton, out bool triggerValue) && triggerValue)
160     {
161         GrabRayInteractor.enabled = true;
162         GrabRayLine.enabled = true;
163     }
164     else if (!triggerValue && GrabRayInteractor.enabled)
165     {
166         GrabRayInteractor.enabled = false;
167         GrabRayLine.enabled = false;
168     }
169 }

```

```

168     private void CheckTeleportMode()
169     {
170         //Teleport mode 1 i.e. spatial teleport projectile curve
171         if (TeleportMode == true)
172         {
173             SetTeleportProjectile();
174         }
175         //Teleport mode 2 i.e. straightline select
176         else if (TeleportMode == false)
177         {
178             SetTeleportRaycast();
179         }
180     }
181
182     private void SetTeleportProjectile()
183     {
184         TeleportInteractor.lineType = XRRayInteractor.LineType.
            ProjectileCurve;
185         //TeleportLine.AttachCustomReticle(teleportReticle);
186         //teleportPlane.SetActive(true);
187         teleportReticle.SetActive(true);
188     }
189
190     private void SetTeleportRaycast()
191     {
192         TeleportInteractor.lineType = XRRayInteractor.LineType.
            StraightLine;
193         //TeleportLine.RemoveCustomReticle();
194         //teleportPlane.SetActive(false);
195         teleportReticle.SetActive(false);
196     }
197 }
198 }

```

### 3.4 Teleportation Anchored Area

Another custom XR script that modifies teleportation behaviour, allowing the player to aim at any part of the collision object but always teleport to a fixed point.

```

1 using System;
2 using System.Collections.Generic;
3 using UnityEditor;

```

```

4 using UnityEngine;
5
6
7 namespace UnityEngine.XR.Interaction.Toolkit
8 {
9     public class TeleportationAnchoredArea : BaseTeleportationInteractable
10    {
11        public GameObject TeleportationAnchor;
12
13        protected override bool GenerateTeleportRequest(XRBaseInteractor
14            interactor, RaycastHit raycastHit, ref TeleportRequest
15            teleportRequest)
16        {
17            teleportRequest.destinationPosition = TeleportationAnchor.
18                transform.position;
19            teleportRequest.destinationUpVector = TeleportationAnchor.
20                transform.up; // use the area transform for data.
21            teleportRequest.destinationForwardVector = transform.forward;
22            teleportRequest.destinationRotation = TeleportationAnchor.
23                transform.rotation;
24
25            GameObject.Find("XR Rig").GetComponent<MovementProvider>().
26                PositionController();
27            return true;
28        }
29    }
30 }

```

### 3.5 Comment Frame Teleporter

Script that handles the teleporting to a specific spot on the comment frame.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEditor;
4 using UnityEngine;
5
6 namespace UnityEngine.XR.Interaction.Toolkit
7 {
8     public class CommentFrameTeleporter : BaseTeleportationInteractable
9     {
10        private float teleportBufferDistance = 8.0f;

```

```

11     protected override bool GenerateTeleportRequest(XRBaseInteractor
12         interactor, RaycastHit raycastHit, ref TeleportRequest
13         teleportRequest)
14     {
15         teleportRequest.destinationPosition = raycastHit.point - (
16             transform.forward.normalized * teleportBufferDistance) - (
17             transform.up.normalized * 1);
18         teleportRequest.destinationUpVector = transform.up; // use the
19             area transform for data.
20         teleportRequest.destinationForwardVector = -transform.right;
21         teleportRequest.destinationRotation = Quaternion.identity;//
22             transform.rotation;
23         return true;
24     }
25 }

```

## 3.6 Custom Teleportation Provider

Custom script instance copied from the default teleportation provider that adds some additional functions

```

1 using System;
2 using UnityEngine;
3 using System.Collections.Generic;
4
5 namespace UnityEngine.XR.Interaction.Toolkit
6 {
7
8     public class CustomTeleportationProvider : LocomotionProvider
9     {
10
11         // the current teleportation request
12         TeleportRequest m_CurrentRequest;
13         // whether the current teleportation request is valid.
14         bool m_ValidRequest = false;
15
16
17         /// <summary>
18         /// This function will queue a teleportation request within the
19         provider.
20         /// </summary>
21         /// <param name="teleportRequest">The teleportation request</param>

```

```

21     /// <returns>true if successful.</returns>
22     public bool QueueTeleportRequest(TeleportRequest teleportRequest)
23     {
24         m_CurrentRequest = teleportRequest;
25         m_ValidRequest = true;
26         return true;
27     }
28
29     /// <summary>
30     /// Update function for the Teleportation Provider
31     /// </summary>
32     private void Update()
33     {
34         if(m_ValidRequest && BeginLocomotion())
35         {
36             var xrRig = system.xrRig;
37             if (xrRig != null)
38             {
39                 switch (m_CurrentRequest.matchOrientation)
40                 {
41                     case MatchOrientation.None:
42                         xrRig.MatchRigUp(m_CurrentRequest.
43                             destinationUpVector);
44                         break;
45                     case MatchOrientation.Camera:
46                         xrRig.MatchRigUpCameraForward(m_CurrentRequest.
47                             destinationUpVector, m_CurrentRequest.
48                             destinationForwardVector);
49                         break;
50                     //case MatchOrientation.Rig:
51                     //    xrRig.MatchRigUpRigForward(m_CurrentRequest.
52                     //        destinationUpVector, m_CurrentRequest.
53                     //        destinationForwardVector);
54                     //    break;
55                 }
56
57                 Vector3 heightAdjustment = xrRig.rig.transform.up *
58                     xrRig.cameraInRigSpaceHeight;
59
60                 Vector3 cameraDestination = m_CurrentRequest.
61                     destinationPosition + heightAdjustment;
62
63                 xrRig.MoveCameraToWorldLocation(cameraDestination);
64             }
65             EndLocomotion();
66             m_ValidRequest = false;
67         }
68     }
69 }

```

## 4 Reddit General Scripts

Reddit data is loaded using the Reddit.NET API interface. Probably not the most ideal due to synchronous lag which I haven't been able to solve, but it is relatively easy to understand and implement.

### 4.1 Reddit Controller

The big, main component script that handles the reddit interface and spawns all the subsequent posts and geometry which then execute their own scripts.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Reddit;
5 using Reddit.Controllers;
6 using Reddit.Controllers.EventArgs;
7 using TMPro;
8 using System.Linq;
9 using System.IO;
10
11 public class RedditController : MonoBehaviour
12 {
13     public RedditClient r;
14     //private string subredditTarget = "AskReddit";
15     private GameObject testPrefab;
16     private GameObject redPostPrefab;
17     private GameObject textTemplate;
18     private GameObject textTemplateColumn;
19     private GameObject textTemplateComment;
20     public GameObject torusTemplate;
21     public GameObject subredditStar;
22     public RedCentralSpaceMonobehaviour centralSpace;
23     public GameObject subredditTitleMesh;
24     public NetworkManager networkManager;
25     public bool generatePosts = true;
26     private string targetSubreddit;
27     private string refreshToken;
28     public List<GameObject> postObjects;
29     private List<Post> allPostList;
```

```

30     private List<(float, float)> postPropertiesList = new List<(float
        upvoteScale, float commentWeight)>();
31     private List<string> subredditList;
32
33     //PARAMETERS
34     private float spawnRadius = 100.0f;
35     private int numRings = 3;
36     private int postsPerRing = 18;
37     private int numPosts = 50;
38
39     // Start is called before the first frame update
40     void Start()
41     {
42         //testPrefab = Resources.Load("Blobs/BlobAnchorStatic_Block", typeof
            (GameObject)) as GameObject;
43
44         redPostPrefab = Resources.Load("Reddit/RedPostPlatform", typeof(
            GameObject)) as GameObject;
45         textTemplate = Resources.Load("Generic_Text", typeof(GameObject)) as
            GameObject;
46         textTemplateColumn = Resources.Load("Generic_Text_Column", typeof(
            GameObject)) as GameObject;
47         textTemplateComment = Resources.Load("Generic_Text_Column_Smaller",
            typeof(GameObject)) as GameObject;
48
49         targetSubreddit = File.ReadAllText(Application.streamingAssetsPath +
            "/subreddit.txt");
50         refreshToken = File.ReadAllText(Application.streamingAssetsPath + "/"
            + "refreshToken.txt");
51
52         r = new RedditClient("DHawrgYdt30hPg", refreshToken);
53         Debug.Log("Username: " + r.Account.Me.Name);
54
55         GenerateSubreddits();
56         Initialise(targetSubreddit);
57         networkManager.SetPlayerSubreddit(targetSubreddit);
58
59
60         /*for (int i = 0; i < numRings + 1; i++)
61         {
62             GameObject ringDivider = Instantiate(torusTemplate, transform);
63             ringDivider.transform.localPosition = new Vector3(0, i * 30f, 0)
64             ;
65         }*/
66         //UnityEngine.Debug.Log(r.Account.Me);
67     }
68     public void Initialise(string subreddit = "")
69     {
70         DestroyGeometry();

```

```

71     targetSubreddit = subreddit;
72     centralSpace.Initialise(r.Subreddit(targetSubreddit));
73     allPostList = GetTopPosts(targetSubreddit, numPosts);
74     Debug.Log("Total posts:" + allPostList.Count);
75     ProcessPostData();
76
77     int oneRingCount = Mathf.FloorToInt(postsPerRing);
78
79     //Calculate which posts in the ring first
80     List<int> ringBreakpoints = new List<int>();
81     ringBreakpoints.Add(0);
82     float sumWeights = postPropertiesList.Sum(x => x.Item2 + 1);
83     //Debug.Log("sum weights :" + sumWeights);
84
85     float totalWeight = 0;
86     for (int p = 0; p < allPostList.Count; p++)
87     {
88         if(totalWeight >= sumWeights/(numRings))
89         {
90             totalWeight = 0;
91             ringBreakpoints.Add(p);
92             //Debug.Log("breakpoint: " + p);
93         }
94         totalWeight += postPropertiesList[p].Item2 + 1;
95     }
96     if(ringBreakpoints.Count < numRings + 1)
97     {
98         ringBreakpoints.Add(allPostList.Count - 1);
99     }
100    //Debug.Log(ringBreakpoints.Count);
101
102    for (int i = 0; i < numRings; i++)
103    {
104        //Debug.Log("Generating ring - " + ringBreakpoints[i] + "-" +
105            ringBreakpoints[i+1]);
106        if(generatePosts) GeneratePostGeometry(allPostList.GetRange(
107            ringBreakpoints[i], ringBreakpoints[i+1]-ringBreakpoints[i])
108            ,
109            postPropertiesList.GetRange(
110                ringBreakpoints[i], ringBreakpoints
111                [i+1]-ringBreakpoints[i]), 15f + (i
112                * 30f));
113        //Debug.Log("Ring " + i + " Generated");
114        /*
115        if ((i + 1) * postsPerRing <= postList.Count)
116        {
117            GeneratePostGeometry(postList.GetRange(i * postsPerRing,
118                postsPerRing), 10f + (i * 30f));
119        }
120        */

```



```

114     }
115 }
116
117 public void DestroyGeometry()
118 {
119     foreach (GameObject post in postObjects)
120     {
121         GameObject.Destroy(post);
122     }
123 }
124
125 // Update is called once per frame
126 void Update()
127 {
128
129 }
130
131 public List<Post> GetTopPosts(string subreddit, int num)
132 {
133     return r.Subreddit(subreddit).Posts.Hot.GetRange(0, num);
134 }
135
136 public void GeneratePostGeometry(List<Post> postList, List<(float, float
137 )> postProperties, float yoffset)
138 {
139     int numObjects = postList.Count;
140     var postWeights = new List<float>();
141
142     for (int i = 0; i < numObjects; i++)
143     {
144         //var weightCalc = Mathf.RoundToInt(Mathf.Clamp((postList[i].
145             Listing.NumComments / 200f), 0f, 1f) * 5);
146         postWeights.Add(1f + postProperties[i].Item2);
147     }
148
149     //postWeights = postList.Select(x => (5 + Mathf.RoundToInt(Mathf.
150         Clamp((x.UpVotes / 40000f), 0f, 1f) * 5)) ).ToList<int>();
151     float totalWeight = postWeights.Sum();
152     //Debug.Log(totalWeight);
153
154     var cumulativeAngle = 0f;
155
156     for (int i = 0; i < numObjects; i++)
157     {
158         Post currentPost = postList[i];
159         string postTitle = currentPost.Title;
160
161         //Circle geometry math
162         float circumference = 2 * Mathf.PI * spawnRadius;

```

```

161     float postAngleShare = postWeights[i] / totalWeight;
162     float angle = postAngleShare * 2 * Mathf.PI;
163     int postLengthUnits = Mathf.FloorToInt(2 * spawnRadius * Mathf.
        Sin(angle / 2) / 4) - 1;
164
165     //float angle = i * Mathf.PI * 2 / numObjects;
166     cumulativeAngle += angle / 2; //add half because you actually
        want the center of the angle
167     float x = Mathf.Cos(cumulativeAngle) * spawnRadius;
168     float z = Mathf.Sin(cumulativeAngle) * spawnRadius;
169     Vector3 pos = transform.position + new Vector3(x, yoffset, z);
170     float angleDegrees = -cumulativeAngle * Mathf.Rad2Deg;
171     Quaternion rot = Quaternion.Euler(270, angleDegrees, 90);
172
173
174     //Instantiate platform geometry
175     GameObject currentPlatform = Instantiate(redPostPrefab, pos, rot
        );
176     currentPlatform.GetComponent<RedPostMonobehaviour>().
        InitialisePost(currentPost, currentPost.Comments,
        postLengthUnits, postProperties[i].Item1);
177
178     cumulativeAngle += angle / 2; //add half again to get the
        actual cumulative angle
179
180
181     postObjects.Add(currentPlatform);
182
183 }
184 }
185
186 public void ProcessPostData()
187 {
188     postPropertiesList.Clear();
189     for(int i = 0; i < allPostList.Count; i++)
190     {
191         Post tempPost = allPostList[i];
192         int maxUpvotes = allPostList.Max(x => x.UpVotes);
193         int maxComments = allPostList.Max(x => x.Listing.NumComments);
194
195         float scaledUpvotes = (float)tempPost.UpVotes/(float)maxUpvotes;
196         float scaledComments = tempPost.Listing.NumComments/(float)
            maxComments;
197
198         scaledUpvotes = Mathf.Ceil(scaledUpvotes * 100) / 100f;
199
200         postPropertiesList.Add((scaledUpvotes, scaledComments));
201     }
202 }
203

```

```

204 public void GenerateSubreddits()
205 {
206     List<Subreddit> subredditList = r.GetSubreddits("popular", 100);
207     int num = subredditList.Count;
208     List<string> subredditNames = subredditList.Select(x => x.Name.
        ToString()).ToList();
209
210     Vector3 center = new Vector3(0, 0, 0);
211     for (int i = 0; i < num; i++)
212     {
213         //Vector sphere math to position stars at random point on
        constrained sphere
214         float thetaAngle = Random.Range(5, 45);
215         float phiAngle = Random.Range(0, 360);
216         Quaternion rotationAngle = Quaternion.Euler(thetaAngle, phiAngle
            , 0);
217
218         Vector3 pos = rotationAngle * new Vector3(0,2000,0);
219
220         //Vector3 pos = Random.onUnitSphere * 2000.0f;
221
222         Quaternion rot = Quaternion.FromToRotation(Vector3.up, center -
            pos);
223         GameObject star = Instantiate(subredditStar, pos, rot);
224
225         //Debug.Log(subredditList[i].Name);
226         string subredditName = subredditList[i].Name;
227         star.GetComponent<SubredditStarMonobehaviour>().InitialiseStar(
            subredditName);
228
229     }
230
231     GameObject allstar = Instantiate(subredditStar, new Vector3(0,2000,0
        ), Quaternion.FromToRotation(Vector3.up, new Vector3(0,-2000,0))
        );
232     allstar.GetComponent<SubredditStarMonobehaviour>().InitialiseStar("
        all");
233 }
234
235 public void GenerateOAuth()
236 {
237     //AuthTokenRetrieverLib authTokenRetrieverLib = new
        AuthTokenRetrieverLib(appId, appSecret, port);
238 }
239 }

```

## 4.2 Reddit Central Space

Script controlling the behaviour of the central space, including scaling management.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Reddit;
5 using Reddit.Controllers;
6 using TMPro;
7 using System;
8 using System.Linq;
9
10 public class RedCentralSpaceMonobehaviour : MonoBehaviour
11 {
12
13     public List<GameObject> nodeList = new List<GameObject>();
14     public GameObject centralPlatform;
15     public GameObject subredditTitleMesh;
16     public GameObject boundaryRingPrefab;
17     public GameObject boundarySphere;
18     public float distanceMargin = 10;
19     public float minScale = 20;
20
21     //Boundary Parameters
22     public float ringWidth = 3;
23     public float ringHeight = 1;
24     public int numSteps = 3;
25     public float stepWidth = 1;
26     private GameObject boundaryRing;
27     private List<GameObject> stepList = new List<GameObject>();
28
29     //adjustment helper parameters
30     private float maxDistance;
31     private float targetScale;
32     private bool startTransform = true;
33     private float scaleRate = 0.005f;
34     private float nextUpdateTime = 0.0f;
35     private float updateInterval = 2f;
36
37     //Subreddit Data
38     private Subreddit sr;
39     private string subredditName = "all";
40     //private List<string> infoModerators;
41     private string infoDescription;
42     private string infoSidebar;
```

```

43     private List<string> sidebarPanelStrings;
44     private List<GameObject> sidebarPanelObjects = new List<GameObject>();
45     public GameObject subredditInfoParent;
46     public GameObject sidebarPanelPrefab;
47     private Reddit.Controllers.Internal.Lists infoLists;
48
49     // Start is called before the first frame update
50     void Start()
51     {
52         CreateBoundaryRing();
53         GetAllNodes();
54         InstantScale();
55         //targetScale = centralPlatform.transform.localScale.x;
56     }
57
58     public void Initialise(Subreddit subreddit)
59     {
60         ClearAll();
61
62         if (subreddit.Name != "all")
63         {
64             sr = subreddit.About();
65             subredditName = sr.Name;
66
67             if (sr.Description != null) infoDescription = sr.Description;
68             if (sr.Sidebar != null)
69             {
70                 infoSidebar = sr.Sidebar;
71                 sidebarPanelStrings = new List<string>(infoSidebar.Split(new
72                     string[] { "***" }, StringSplitOptions.None));
73                 GeneratePanels();
74             }
75             if (sr.Lists != null) infoLists = sr.Lists;
76         }
77         else
78         {
79             subredditName = "all";
80         }
81
82         //Set geometry stuff
83         RefreshGeometry();
84     }
85
86     public void RefreshGeometry()
87     {
88         subredditTitleMesh.GetComponent<TextMeshPro>().text = "r/" +
89             subredditName;
90     }

```

```

91
92 public void ClearAll()
93 {
94     if (sidebarPanelStrings != null) sidebarPanelStrings.Clear();
95     DestroyPanels();
96 }
97
98 public void GeneratePanels()
99 {
100     DestroyPanels();
101
102     var cumulativeAngle = Mathf.PI / 6;
103     var radius = 6.5f;
104
105     for (int i = 0; i < sidebarPanelStrings.Count; i++)
106     {
107         float angleIncrease = Mathf.PI / 12;
108
109         cumulativeAngle = -cumulativeAngle; //flip sides back and forth
110
111         float x = Mathf.Cos(cumulativeAngle) * radius;
112         float z = Mathf.Sin(cumulativeAngle) * radius;
113         Vector3 pos = new Vector3(x, -2.25f, z);
114
115         float angleDegrees = -cumulativeAngle * Mathf.Rad2Deg;
116         Quaternion rot = Quaternion.Euler(0, angleDegrees + 90, 0);
117
118         GameObject panel = GameObject.Instantiate(sidebarPanelPrefab,
119             pos, rot);
120         panel.transform.SetParent(subredditInfoParent.transform);
121
122         panel.GetComponent<RedInfoPanelMonobehaviour>().SetText(
123             sidebarPanelStrings[i]);
124
125         sidebarPanelObjects.Add(panel);
126
127         if (i % 2 == 1)
128         {
129             cumulativeAngle += angleIncrease;
130         }
131     }
132 }
133
134 public void DestroyPanels()
135 {
136     for (int i = 0; i < sidebarPanelObjects.Count; i++)
137     {
138         GameObject.Destroy(sidebarPanelObjects[i]);
139     }
140 }

```

```

139
140 // Update is called once per frame
141 void Update()
142 {
143
144     if (Time.time > nextUpdateTime)
145     {
146         nextUpdateTime += updateInterval;
147
148         CheckDistances();
149
150         var platformRadius = (maxDistance + distanceMargin + stepWidth *
151             numSteps);
152
153         if (platformRadius * 2 > minScale && Mathf.Abs(platformRadius *
154             2 - centralPlatform.transform.localScale.x) > 1 &&
155             platformRadius < 80)
156         {
157             ChangeTargetScale(Mathf.Round((maxDistance + distanceMargin
158                 + stepWidth * numSteps) * 2));
159         }
160
161         scaleRate = 0.01f * Mathf.Clamp01(Mathf.Abs(platformRadius * 2 -
162             centralPlatform.transform.localScale.x) / (distanceMargin *
163                 2));
164     }
165
166     if (startTransform)
167     {
168         if (centralPlatform.transform.localScale.x - targetScale < -1f)
169         {
170             centralPlatform.transform.localScale += new Vector3(1, 0, 1)
171                 * scaleRate;
172             float sphereScale = centralPlatform.transform.localScale.x +
173                 2 * ringWidth + 2;
174             boundarySphere.transform.localScale = new Vector3(
175                 sphereScale + 1f, sphereScale + 1f, sphereScale + 1f);
176             UpdateBoundaryRing();
177         }
178         else if (centralPlatform.transform.localScale.x - targetScale >
179             1f)
180         {
181             centralPlatform.transform.localScale += new Vector3(-1, 0, -
182                 1) * scaleRate;
183             float sphereScale = centralPlatform.transform.localScale.x +
184                 2 * ringWidth + 2;
185             boundarySphere.transform.localScale = new Vector3(
186                 sphereScale + 1f, sphereScale + 1f, sphereScale + 1f);
187             UpdateBoundaryRing();
188         }
189     }
190 }

```

```

176         }
177         else
178         {
179             startTransform = false;
180         }
181     }
182 }
183
184 void CheckDistances()
185 {
186     float tempMax = minScale/4;
187     Vector3 centerCoord = new Vector3(transform.position.x, 0, transform
        .position.z);
188
189     nodeList.RemoveAll(i => i == null);
190
191     foreach (GameObject i in nodeList)
192     {
193         Vector3 tempCoord = new Vector3(i.transform.position.x, 0, i.
            transform.position.z);
194         float tempDist = Vector3.Distance(centerCoord, tempCoord);
195         if (tempDist > tempMax)
196         {
197             tempMax = tempDist;
198         }
199     }
200
201     maxDistance = tempMax;
202 }
203
204 void ChangeTargetScale(float target)
205 {
206     targetScale = target;
207     startTransform = true;
208 }
209
210 void UpdateBoundaryRing()
211 {
212     boundaryRing.GetComponent<GeometryTorus>().SetParameters(
        centralPlatform.transform.localScale.x / 2 + ringWidth / 2,
        ringWidth, 64, 4, 0.5f, true);
213     boundaryRing.transform.localScale = new Vector3(1, ringHeight /
        ringWidth, 1);
214
215     for (int i = 0; i < stepList.Count; i++)
216     {
217         stepList[i].GetComponent<GeometryTorus>().SetParameters(
            centralPlatform.transform.localScale.x / 2 - (i * stepWidth
                * 1.3f) - ringWidth / 2, stepWidth, 64, 4, 0.5f, true);

```



```

218         stepList[i].transform.localScale = new Vector3(1, (1 - (float)(i
                + 1) / (numSteps + 1)) * ringHeight / stepWidth, 1);
219     }
220 }
221
222 void CreateBoundaryRing()
223 {
224     boundaryRing = GameObject.Instantiate(boundaryRingPrefab);
225     boundaryRing.GetComponent<GeometryTorus>().SetParameters((
        centralPlatform.transform.localScale.x / 2) + (ringWidth / 2),
        ringWidth, 64, 4, 0.5f, true);
226     boundaryRing.transform.localScale = new Vector3(1, ringHeight /
        ringWidth, 1);
227     boundaryRing.transform.position = this.transform.position;
228     boundaryRing.transform.SetParent(this.transform);
229
230     for (int i = 0; i < numSteps; i++)
231     {
232         GameObject stepObject = GameObject.Instantiate(
            boundaryRingPrefab);
233         stepObject.GetComponent<GeometryTorus>().SetParameters(
            centralPlatform.transform.localScale.x / 2 - (i * stepWidth
                * 1.3f) - ringWidth / 2, stepWidth, 64, 4, 0.5f, true);
234         stepObject.transform.localScale = new Vector3(1, (1 - (float)(i
            + 1) / (numSteps + 1)) * ringHeight / stepWidth, 1);
235         stepObject.transform.position = this.transform.position;
236         stepObject.transform.SetParent(this.transform);
237
238         stepList.Add(stepObject);
239     }
240
241 }
242
243 public void InstantScale()
244 {
245     CheckDistances();
246
247     var platformRadius = (maxDistance + distanceMargin + stepWidth *
        numSteps);
248
249     if (platformRadius * 2 > minScale && Mathf.Abs(platformRadius * 2 -
        centralPlatform.transform.localScale.x) > 1 && platformRadius <
        80)
250     {
251         ChangeTargetScale(Mathf.Round((maxDistance + distanceMargin +
            stepWidth * numSteps) * 2));
252     }
253
254     centralPlatform.transform.localScale = new Vector3(targetScale, 0.3f
        , targetScale);

```

```

255     float sphereScale = centralPlatform.transform.localScale.x + 2 *
        ringWidth + 2;
256     boundarySphere.transform.localScale = new Vector3(sphereScale + 1f,
        sphereScale + 1f, sphereScale + 1f);
257
258     UpdateBoundaryRing();
259
260 }
261
262 public void GetAllNodes()
263 {
264     nodeList.Clear();
265     nodeList = GameObject.FindGameObjectsWithTag("NodeBlock").ToList();
266     Debug.Log(nodeList.Count);
267 }
268
269 public void NodeListAdd(GameObject node)
270 {
271     nodeList.Add(node);
272 }
273
274 public void NodeListRemove(GameObject node)
275 {
276     nodeList.Remove(node);
277 }
278
279 }

```

### 4.3 Reddit Central Sphere

Script controlling the collision events for the sphere area to determine if player is still in center area.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class RedCenterSphereCollision : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }

```

```

12
13 // Update is called once per frame
14 void Update()
15 {
16
17 }
18 }

```

## 4.4 Reddit Cube Spawner

Script for the pedestal that spawns the node-block.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Photon.Realtime;
5 using Photon.Pun;
6
7 public class RedCubeSpawner : MonoBehaviour
8 {
9     private bool touchingNode;
10    public RedCentralSpaceMonoBehaviour centralControl;
11    public ColorChanger colorChanger;
12    private List<Collision> collidingNodes = new List<Collision>();
13    private List<GameObject> nodeBlockList = new List<GameObject>();
14    // Start is called before the first frame update
15    void Start()
16    {
17
18    }
19
20    // Update is called once per frame
21    void Update()
22    {
23
24    }
25
26    public void SpawnCube()
27    {
28        if(!IsTouchingNode())
29        {
30            object[] id = new object[1];
31            id[0] = Random.Range(1, 255);
32

```

```

33         GameObject cube = PhotonNetwork.Instantiate("Reddit/NodeBlock",
34             transform.position + new Vector3(0, 1, 0), transform.
35             rotation, 0, id);
36         nodeBlockList.Add(cube);
37         centralControl.NodeListAdd(cube);
38     }
39 }
40
41 void OnCollisionEnter(Collision collision)
42 {
43     if(collision.gameObject.CompareTag("NodeBlock"))
44     {
45         collidingNodes.Add(collision);
46         colorChanger.SetColor(1);
47     }
48 }
49
50 void OnCollisionExit(Collision collision)
51 {
52     if(collision.gameObject.CompareTag("NodeBlock"))
53     {
54         collidingNodes.Remove(collision);
55         if(!IsTouchingNode())
56         {
57             colorChanger.SetColor(0);
58         }
59     }
60 }
61
62 public bool IsTouchingNode()
63 {
64     if(collidingNodes.Count > 0)
65     {
66         return true;
67     }
68     else
69     {
70         return false;
71     }
72 }
73
74 public void HoverIn()
75 {
76     if(!IsTouchingNode())
77     {
78         colorChanger.SetColor(2);
79     }
80 }
81
82 public void HoverOut()

```

```

81     {
82         if(!IsTouchingNode())
83         {
84             colorChanger.SetColor(0);
85         }
86     }
87
88
89 }

```

## 4.5 Reddit Node Block

Script controlling the node block - just simple sync functions and destroy if it falls off the map.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Photon.Pun;
5  using Photon.Realtime;
6  using Photon.Voice.Unity;
7
8  public class RedNodeBlock : MonoBehaviour, IPunInstantiateMagicCallback
9  {
10     public int voiceRoomID = 0;
11     public RedNodeSphere sphere;
12
13
14     // Start is called before the first frame update
15     void Start()
16     {
17
18     }
19
20     public void OnPhotonInstantiate(PhotonMessageInfo info)
21     {
22         object[] data = info.photonView.InstantiationData;
23         if (data != null)
24         {
25             voiceRoomID = (int)data[0];
26             sphere.voiceRoomID = voiceRoomID;
27         }
28         else
29         {

```

```

30         voiceRoomID = this.GetComponent<PhotonView>().ViewID;
31         sphere.voiceRoomID = voiceRoomID;
32     }
33
34 }
35
36 // Update is called once per frame
37 void Update()
38 {
39     if(transform.position.y <= -100)
40     {
41         Destroy();
42     }
43 }
44
45 void ConnectToRoom()
46 {
47
48 }
49
50 public void Destroy()
51 {
52
53     PhotonNetwork.Destroy(this.GetComponent<PhotonView>());
54 }
55 }

```

## 4.6 Reddit Node Sphere

Handles collision events for the projected sphere around each node block.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Photon.Pun;
5 using Photon.Realtime;
6 using Photon.Voice.Unity;
7 using Photon.Voice.PUN;
8
9 public class RedNodeSphere : MonoBehaviour
10 {
11     public int voiceRoomID = 0;
12     public PhotonVoiceNetwork voiceNetwork;

```

```

13 public Recorder recorder;
14 public PlayerController playerController;
15 private List<GameObject> collidingPlayers = new List<GameObject>();
16
17 // Start is called before the first frame update
18 void Start()
19 {
20     playerController = GameObject.Find("XR Rig").GetComponent<
21         PlayerController>();
22     voiceNetwork = GameObject.Find("VoiceManager").GetComponent<
23         PhotonVoiceNetwork>();
24     recorder = GameObject.Find("VoiceManager").GetComponent<Recorder>();
25 }
26
27 // Update is called once per frame
28 void Update()
29 {
30 }
31
32 void OnTriggerEnter(Collider collider)
33 {
34     if(collider.gameObject.CompareTag("Player"))
35     {
36         playerController.numVoiceRooms += 1;
37
38         byte[] room = new byte[1];
39         room[0] = (byte)voiceRoomID;
40         voiceNetwork.Client.OpChangeGroups(null, room);
41         recorder.InterestGroup = room[0];
42
43         Debug.Log("Joined group " + room[0]);
44     }
45 }
46
47 void OnTriggerExit(Collider collider)
48 {
49     if(collider.gameObject.CompareTag("Player"))
50     {
51         playerController.numVoiceRooms -= 1;
52
53         byte[] room = new byte[1];
54         room[0] = (byte)voiceRoomID;
55         voiceNetwork.Client.OpChangeGroups(room, null);
56
57         if(playerController.numVoiceRooms == 0 && recorder.InterestGroup
58             != 0)
59         {
60             recorder.InterestGroup = 0;
61             Debug.Log("Recorder joined back group 0.");
62         }
63     }
64 }

```

```

60
61         Debug.Log("Disconnected from group " + room[0]);
62     }
63 }
64 }

```

## 4.7 Subreddit Star

Script to handle the behaviour of individual subreddit links.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5  using Photon.Pun;
6  using UnityEngine.XR.Interaction.Toolkit;
7
8  public class SubredditStarMonoBehaviour : MonoBehaviour
9  {
10     public GameObject titleMesh;
11     private string subreddit;
12     //private RedditController redditController;
13     private NetworkManager networkManager;
14
15     // Start is called before the first frame update
16     void Start()
17     {
18         //InitialiseStar("");
19         //redditController = GameObject.Find("RedditManager").GetComponent<
20             RedditController>();
21         networkManager = GameObject.Find("NetworkManager").GetComponent<
22             NetworkManager>();
23         HideTitle();
24     }
25
26     public void InitialiseStar(string subredditName)
27     {
28         subreddit = subredditName;
29         //Debug.Log("subreddit = " + subreddit);
30         titleMesh.GetComponent<TextMeshPro>().text = string.Concat("r/",
31             subreddit);
32     }
33
34     public void ChangeSubreddit()
35     {

```



```

33     //networkManager.rig.GetComponent<XRRig>().MatchRigUpRigForward(new
        Vector3(0, 3, 0), new Vector3(1, 0, 0));
34     networkManager.SetPlayerSubreddit(subreddit);
35     //redditController.Initialise(subreddit);
36 }
37
38 public void ShowTitle()
39 {
40     //titleMesh.SetActive(true);
41     titleMesh.GetComponent<TextMeshPro>().fontSize = 15;
42 }
43
44 public void HideTitle()
45 {
46     //titleMesh.SetActive(false);
47     titleMesh.GetComponent<TextMeshPro>().fontSize = 6;
48 }
49
50 // Update is called once per frame
51 void Update()
52 {
53     titleMesh.transform.rotation = Camera.main.transform.rotation;
54 }
55 }

```

## 4.8 Reddit Info Panel

Handles the info panel boards

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using TMPro;
5 using LogicUI.FancyTextRendering;
6 using NaughtyAttributes;
7
8 public class RedInfoPanelMonobehaviour : MonoBehaviour
9 {
10
11     public GameObject mainText;
12     private Camera XRcamera;
13     public Canvas cameraCanvas;
14
15     // Start is called before the first frame update
16     void Start()

```

```

17     {
18         XRcamera = GameObject.Find("VR Camera Controller").GetComponent<
            Camera>();
19         cameraCanvas.GetComponent<Canvas>().worldCamera = XRcamera;
20         Invoke("FitContents", 0.1f);
21     }
22
23     public void Initialise()
24     {
25         Invoke("FitContents", 0.1f);
26     }
27
28     // Update is called once per frame
29     void Update()
30     {
31
32     }
33
34     public void SetText(string str)
35     {
36         mainText.GetComponent<MarkdownRenderer>().Source = str;
37         Invoke("FitContents", 0.1f);
38     }
39
40     void FitContents()
41     {
42         RectTransform textRect = mainText.GetComponent<RectTransform>();
43
44         textRect.localPosition = new Vector3(0, - textRect.sizeDelta.y / 2,
            0);
45     }
46 }

```

## 4.9 Geometry Torus

Script that generates a torus-shape dynamically at runtime, because otherwise scaling a pre-made torus would be uniform and not give the expected results. Used for the giant rings as well as the cylindrical boundaries of the center space.

```

1 /**

```

```

2 * Based on a script by Steffen (http://forum.unity3d.com/threads/torus-in-unity.8487/) (in $primitives_966_104.zip, originally named "Primitives.cs")
3 *
4 * Edited by Michael Zoller on December 6, 2015.
5 * It was shortened by about 30 lines (and possibly sped up by a factor of 2) by consolidating math & loops and removing intermediate Collections.
6 */
7 using UnityEngine;
8
9 [RequireComponent(typeof(MeshFilter), typeof(MeshRenderer))]
10 public class GeometryTorus : MonoBehaviour
11 {
12
13     public float segmentRadius = 1f;
14     public float tubeRadius = 0.1f;
15     public int numSegments = 32;
16     public int numTubes = 12;
17     public float tubeRotation = 0.0f;
18     public bool hardEdges = false;
19
20     void Start()
21     {
22         RefreshTorus();
23     }
24
25     public void SetParameters(float circleRadius, float thickness, int circleResolution, int tubeResolution, float tubeRotate, bool hardShading)
26     {
27         segmentRadius = circleRadius;
28         tubeRadius = thickness;
29         numSegments = circleResolution;
30         numTubes = tubeResolution;
31         tubeRotation = tubeRotate;
32         hardEdges = hardShading;
33
34         RefreshTorus();
35     }
36
37     public void RefreshTorus()
38     {
39         // Total vertices
40         int totalVertices = numSegments * numTubes;
41
42         // Total primitives
43         int totalPrimitives = totalVertices * 2;
44
45         // Total indices
46         int totalIndices = totalPrimitives * 3;

```

```

47
48 // Init the mesh
49 Mesh mesh = new Mesh();
50
51 // Init the vertex and triangle arrays
52 Vector3[] vertices = new Vector3[totalVertices];
53 int[] triangleIndices = new int[totalIndices];
54
55 // Calculate size of a segment and a tube
56 float segmentSize = 2 * Mathf.PI / (float)numSegments;
57 float tubeSize = 2 * Mathf.PI / (float)numTubes;
58
59 // Create floats for our xyz coordinates
60 float x, y, z;
61
62 // Begin loop that fills in both arrays
63 for (int i = 0; i < numSegments; i++)
64 {
65     // Find next (or first) segment offset
66     int n = (i + 1) % numSegments; // changed segmentList.Count to
        numSegments
67
68     // Find the current and next segments
69     int currentTubeOffset = i * numTubes;
70     int nextTubeOffset = n * numTubes;
71
72     for (int j = 0; j < numTubes; j++)
73     {
74         // Find next (or first) vertex offset
75         int m = (j + 1) % numTubes; // changed currentTube.Count to
        numTubes
76
77         // Find the 4 vertices that make up a quad
78         int iv1 = currentTubeOffset + j;
79         int iv2 = currentTubeOffset + m;
80         int iv3 = nextTubeOffset + m;
81         int iv4 = nextTubeOffset + j;
82
83         // Calculate X, Y, Z coordinates.
84         x = (segmentRadius + tubeRadius * Mathf.Cos((j +
            tubeRotation) * tubeSize)) * Mathf.Cos(i * segmentSize);
85         z = (segmentRadius + tubeRadius * Mathf.Cos((j +
            tubeRotation) * tubeSize)) * Mathf.Sin(i * segmentSize);
86         y = tubeRadius * Mathf.Sin((j + tubeRotation) * tubeSize);
87
88         // Add the vertex to the vertex array
89         vertices[iv1] = new Vector3(x, y, z);
90
91         // "Draw" the first triangle involving this vertex
92         triangleIndices[iv1 * 6] = iv1;

```

```

93         triangleIndices[iv1 * 6 + 1] = iv2;
94         triangleIndices[iv1 * 6 + 2] = iv3;
95         // Finish the quad
96         triangleIndices[iv1 * 6 + 3] = iv3;
97         triangleIndices[iv1 * 6 + 4] = iv4;
98         triangleIndices[iv1 * 6 + 5] = iv1;
99     }
100 }
101 mesh.vertices = vertices;
102 mesh.triangles = triangleIndices;
103
104 mesh.RecalculateBounds();
105 mesh.RecalculateNormals(); // added on suggestion of Eric5h5 &
    joaeba in the forum thread
106
107 //HARD SHADING
108 if(hardEdges)
109 {
110     Vector3[] oldVerts = mesh.vertices;
111     int[] triangles = mesh.triangles;
112     Vector3[] newVertices = new Vector3[triangles.Length];
113     for (int i = 0; i < triangles.Length; i++)
114     {
115         newVertices[i] = oldVerts[triangles[i]];
116         triangles[i] = i;
117     }
118     mesh.vertices = newVertices;
119     mesh.triangles = triangles;
120     mesh.RecalculateBounds();
121     mesh.RecalculateNormals();
122 }
123
124
125 mesh.Optimize();
126 MeshFilter mFilter = GetComponent<MeshFilter>(); // tweaked to
    Generic
127 mFilter.mesh = mesh;
128 MeshCollider mCollider = GetComponent<MeshCollider>();
129 if(mCollider != null)
130 {
131     mCollider.sharedMesh = mesh;
132 }
133 }
134 }

```

## 5 Reddit Post Scripts

Sub-scripts that handle various aspects of the reddit object behaviours such as posts, comments, scrolling, etc.

### 5.1 Reddit Post Controller

Script handling all the procedural generation of post space geometry, as well as interactions with interface and other stuff.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Net;
4 using System.IO;
5 using UnityEngine;
6 using UnityEngine.Networking;
7 using Reddit;
8 using Reddit.Controllers;
9 using TMPro;
10 using Vuplex.WebView;
11 using System.Globalization;
12
13 public class RedPostMonobehaviour : MonoBehaviour
14 {
15
16     //REDDIT POST DATA
17     private Post post;
18     private int numUpvotes;
19     private string postText;
20     private string postURL;
21     private string postTitle;
22     private string postSubreddit;
23     private Comments postComments;
24     private WebClient webClient;
25
26
27     //GEOMETRY
28     public GameObject platformGeometry;
29     public GameObject platformCenter;
30     public GameObject platformLeft;
31     public GameObject platformRight;
32     public GameObject platformSimple;
33     public GameObject platformWall;
```

```

34     public GameObject platformBase;
35     public GameObject platformAreaCollider;
36     public GameObject mainContent;
37     public GameObject mainContentText;
38     public GameObject mainTitle;
39     public GameObject mainTitleText;
40     public GameObject interfaceButtons;
41     public GameObject commentFrame;
42     public GameObject cylinderPlatform;
43     public GameObject externalThumbnail;
44     public GameObject externalTitle;
45     public GameObject externalSubreddit;
46     public GameObject externalUpvotes;
47
48     private List<GameObject> commentFramesList = new List<GameObject>();
49     private List<GameObject> cylinderPlatforms = new List<GameObject>();
50     //private GameObject textTemplateColumn;
51     //private GameObject textTemplateComment;
52     public GameObject imagePlane;
53     public GameObject webViewObject;
54     public GameObject webView;
55     public GameObject webViewFrame;
56     public GameObject webViewLoadingText;
57     public float frameTextureValue = 0f;
58     WebViewPrefab webViewPrefab;
59     private Texture thumbnailTexture;
60     private Texture imageTexture;
61     private Color platformColor;
62
63     //PARAMETERS
64     //private (int votesCeil, int minRadius, int maxRadius) upvoteScaling =
65     (20000, 6, 12);
66     private bool instantiateCommentFrames = true;
67     private bool loadComments = true;
68     private bool linkPost = false;
69     private int platformLength;
70     private float platformLengthUnits;
71     private float unitLength = 4f;
72     private float collisionHeight = 0f;
73
74     // Start is called before the first frame update
75     void Start()
76     {
77         //platformGeometry = transform.Find("Platform").gameObject;
78         //platformAreaCollider = transform.Find("Collider").GetComponent<
79         Collider>();
80         //mainText = transform.Find("MainText").gameObject;
81         //imagePlane = transform.Find("ImagePlane").gameObject;
82         webViewPrefab = webView.GetComponent<WebViewPrefab>();

```

```

82     platformColor = LibColor.RandomiseHSV(new Color(0.5f, 0.5f, 0.7f, 1.
      Of), 0.3f, 0.1f, 0.1f);
83     RefreshGeometry();
84     HideInterior();
85
86 }
87
88 // Update is called once per frame
89 void Update()
90 {
91     var video = post.Listing.Media;
92 }
93
94 public void InitialisePost(Post p, Comments c, int length, float
  upvoteScale)
95 {
96     //POST DATA
97     post = p;
98     numUpvotes = p.UpVotes;
99     postTitle = p.Title;
100    postSubreddit = p.Subreddit;
101    postComments = c;
102
103    //PLATFORM GEOMETRY
104    platformLength = length;
105    platformLengthUnits = platformLength * unitLength;
106    frameTextureValue = upvoteScale;
107
108    if (post.Listing.IsSelf)
109    {
110        postText = ((SelfPost)post).SelfText;
111        if (postText.Length == 0)
112        {
113            //postText = postTitle;
114        }
115    }
116    else
117    {
118        postText = postTitle;
119        postURL = ((LinkPost)post).URL;
120        //Debug.Log("Not self post!");
121        StartCoroutine(GetThumbnailTexture(p.Listing.Thumbnail));
122        StartCoroutine(GetImageTexture(postURL));
123        if (!post.Listing.IsRedditMediaDomain && postURL != null)
124        {
125            //Debug.Log(postURL);
126            linkPost = true;
127        }
128    }
129

```



```

130
131     RefreshGeometry();
132     //RefreshComments();
133
134 }
135
136 void OnTriggerEnter(Collider collider)
137 {
138     //Debug.Log("Collision");
139     if (collider.gameObject.tag == "Player")
140     {
141         ShowInterior();
142         Debug.Log("Player entered post - refreshing comments...");
143         RefreshComments();
144         if (linkPost) InitialiseWebView(postURL);
145     }
146 }
147
148 void OnTriggerExit(Collider collider)
149 {
150     //Debug.Log("Collision");
151     if (collider.gameObject.tag == "Player")
152     {
153         HideInterior(true);
154     }
155 }
156
157 private void RefreshGeometry()
158 {
159     //Calculate size of platform
160     //platformRadius = upvoteScaling.minRadius + Mathf.Clamp((numUpvotes
161         / upvoteScaling.votesCeil), 0, 1) * (upvoteScaling.maxRadius -
162         upvoteScaling.minRadius);
163
164     //Platform geometry
165     platformCenter.transform.localScale = new Vector3(
166         platformLengthUnits / 2, 0.5f, 9); //old transform without frame
167     prefab
168     platformLeft.transform.localScale = new Vector3(platformLengthUnits,
169         1, 4);
170     platformRight.transform.localScale = new Vector3(platformLengthUnits
171         , 1, 4);
172     platformSimple.transform.localScale = new Vector3(
173         platformLengthUnits, 18, 0.5f);
174
175     platformGeometry.GetComponent<MeshRenderer>().material.color =
176         platformColor;
177
178     //Platform Text
179     //var imagePos = new Vector3(platformLengthUnits / 3, 0, 0);

```

```

172     //var textPos = new Vector3(platformLengthUnits / 3 + 0.2f, 3.5f, 0)
173     ;
174     var rot = Quaternion.identity;
175     //imagePlane.transform.localPosition = imagePos;
176     //mainText.transform.localPosition = textPos;
177
178     mainContentText.GetComponent<TextMeshProUGUI>().text = postText;
179     mainContent.GetComponent<RedTextFrameMonobehaviour>().Initialise();
180     mainTitleText.GetComponent<TextMeshProUGUI>().text = postTitle;
181     mainTitle.GetComponent<RedTextFrameMonobehaviour>().Initialise();
182
183     externalTitle.GetComponent<TextMeshPro>().text = postTitle;
184     externalSubreddit.GetComponent<TextMeshPro>().text = numUpvotes.
185         ToString() + " | r/" + postSubreddit;
186
187     externalTitle.GetComponent<RectTransform>().sizeDelta = new Vector2(
188         platformLengthUnits - 3, 4);
189     externalSubreddit.GetComponent<RectTransform>().sizeDelta = new
190         Vector2(platformLengthUnits - 3, 4);
191     externalUpvotes.GetComponent<RectTransform>().sizeDelta = new Vector
192         2(platformLengthUnits - 3, 4);
193
194     MaterialManager matManager = GameObject.Find("MaterialManager").
195         GetComponent<MaterialManager>();
196     Color frameColor = Color.HSVToRGB(0.1f, 0.7f, 0.8f);
197     Material frameMaterial = matManager.InstantiateMaterialWithEmission(
198         string.Concat("FrameMaterial", frameTextureValue), frameColor, 0
199         .5f + frameTextureValue * 2);
200     platformSimple.GetComponent<FrameDynamic>().ReplaceBaseMaterial(
201         frameMaterial);
202     platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial();
203
204     if (post.Listing.IsSelf)
205     {
206         //Resize title to be larger in absence of thumbnail
207         externalTitle.GetComponent<RectTransform>().sizeDelta = new
208             Vector2(platformLengthUnits - 3, 10);
209         externalTitle.transform.localPosition = new Vector3(0, 1.3f, 1.8
210             f);
211     }
212
213     if (instantiateCommentFrames)
214     {
215         foreach (GameObject i in commentFramesList)
216         {
217             GameObject.Destroy(i);
218         }
219
220         for (int i = 0; i < platformLength; i++)
221         {

```

```

211     float xpos = (platformLengthUnits / 2) - (i * unitLength) -
                (unitLength / 2);
212
213     GameObject newComment = Instantiate(commentFrame, transform)
                ;
214     newComment.transform.localPosition = new Vector3(xpos, 1f, -
                (18f / 2 + 3));
215     newComment.transform.rotation *= Quaternion.Euler(new Vector
                3(0, 180, 0));
216     commentFramesList.Add(newComment);
217
218     //GameObject newPlatform = Instantiate(cylinderPlatform,
                transform);
219     //newPlatform.transform.localPosition = new Vector3(xpos, 0.
                6f, -5.5f);
220     //cylinderPlatforms.Add(newPlatform);
221
222     newComment = Instantiate(commentFrame, transform);
223     newComment.transform.localPosition = new Vector3(xpos, 1f, 1
                8f / 2 + 3);
224     //newComment.transform.rotation *= Quaternion.Euler(new
                Vector3(0, 180, 0));
225     commentFramesList.Add(newComment);
226
227     //newPlatform = Instantiate(cylinderPlatform, transform);
228     //newPlatform.transform.localPosition = new Vector3(xpos, 0.
                6f, 5.5f);
229     //cylinderPlatforms.Add(newPlatform);
230 }
231
232     instantiateCommentFrames = false;
233 }
234
235     RefreshCollisionHeight(20f);
236
237
238     Invoke("RetrieveVotes", 1.0f);
239     RefreshImage();
240     RefreshThumbnail();
241
242 }
243
244 private void ShowInterior()
245 {
246     platformSimple.SetActive(false);
247     externalThumbnail.SetActive(false);
248     externalTitle.SetActive(false);
249     externalSubreddit.SetActive(false);
250     externalUpvotes.SetActive(false);
251

```

```

252     platformCenter.SetActive(true);
253     platformLeft.SetActive(true);
254     platformRight.SetActive(true);
255     platformWall.SetActive(true);
256     platformBase.SetActive(true);
257
258     if (linkPost) webViewObject.SetActive(true);
259     else if (imageTexture != null) imagePlane.SetActive(true);
260     else if (postText != "") mainContent.SetActive(true);
261
262     mainTitle.SetActive(true);
263     interfaceButtons.SetActive(true);
264
265     foreach (GameObject i in commentFramesList)
266     {
267         i.SetActive(true);
268     }
269     foreach (GameObject i in cylinderPlatforms)
270     {
271         i.SetActive(true);
272     }
273 }
274
275 private void HideInterior(bool hideComments = true)
276 {
277     platformSimple.SetActive(true);
278     externalThumbnail.SetActive(true);
279     externalTitle.SetActive(true);
280     externalSubreddit.SetActive(true);
281     externalUpvotes.SetActive(true);
282
283     platformCenter.SetActive(false);
284     platformLeft.SetActive(false);
285     platformRight.SetActive(false);
286     platformWall.SetActive(false);
287     platformBase.SetActive(false);
288
289     imagePlane.SetActive(false);
290     webViewObject.SetActive(false);
291     mainContent.SetActive(false);
292     mainTitle.SetActive(false);
293     interfaceButtons.SetActive(false);
294
295     if (hideComments)
296     {
297         foreach (GameObject i in commentFramesList)
298         {
299             i.SetActive(false);
300         }
301     }

```

```

302     foreach (GameObject i in cylinderPlatforms)
303     {
304         i.SetActive(false);
305     }
306 }
307
308 private void RefreshComments()
309 {
310     if (!loadComments) return;
311
312     int numComments = platformLength * 2;
313     var topComments = RetrieveComments(numComments);
314
315     for (int c = 0; c < commentFramesList.Count; c++)
316     {
317         if (c < topComments.Count)
318         {
319             commentFramesList[c].GetComponent<
320                 RedCommentFrameMonobehaviour>().InitialiseComments(
321                 topComments[c]);
322         }
323         else
324         {
325             //commentFramesList[c].GetComponent<
326                 RedCommentFrameMonobehaviour>().Deactivate();
327             commentFramesList[c].SetActive(false);
328         }
329     }
330
331     loadComments = false;
332
333     RefreshGeometry();
334 }
335
336 public void DeepRefreshAllComments()
337 {
338     if (!loadComments)
339     {
340         for (int c = 0; c < commentFramesList.Count; c++)
341         {
342             if (commentFramesList[c].activeSelf)
343             {
344                 commentFramesList[c].GetComponent<
345                     RedCommentFrameMonobehaviour>().DeepRefreshComments(
346                     );
347             }
348         }
349     }
350 }

```

```

347 private async void InitialiseWebView(string url)
348 {
349     webViewPrefab = webView.GetComponent<WebViewPrefab>();
350
351     //Set initial url of webview and initialise
352     webViewPrefab.InitialUrl = url;
353     webViewPrefab.Init();
354
355     await webViewPrefab.WaitUntilInitialized();
356     //bool initialLoad = true;
357     //Wait until webpage is fully loaded, then check height of page and
358     //resize webview window accordingly.
359     webViewPrefab.WebView.LoadProgressChanged += async (sender,
360     EventArgs) =>
361     {
362         //Resize when finished
363         if (EventArgs.Type == ProgressChangeType.Finished)
364         {
365             var heightString = await webViewPrefab.WebView.
366             ExecuteJavaScript("document.documentElement.scrollHeight
367             ");
368             var heightInPixels = float.Parse(heightString, CultureInfo.
369             InvariantCulture);
370             float heightInUnityUnits;
371             bool tooLong = false;
372
373             if (webViewPrefab.WebView.Size.x * webViewPrefab.WebView.
374             Resolution * heightInPixels > 7000000)
375             {
376                 //Throw error if webpage too large to load
377                 webViewLoadingText.GetComponent<TextMeshPro>().text = "
378                 WEBPAGE TOO LONG";
379                 heightInPixels = 7000000 / (webViewPrefab.WebView.Size.x
380                 * webViewPrefab.WebView.Resolution);
381                 tooLong = true;
382             }
383
384             ResizeWebView(heightInPixels, out heightInUnityUnits);
385
386             //Hide scrollbar and loading text
387             await webViewPrefab.WebView.ExecuteJavaScript("document.body
388             .style.overflow = 'hidden'");
389             if (!tooLong)
390             {
391                 webViewLoadingText.GetComponent<TextMeshPro>().text = ""
392                 ;
393                 webViewLoadingText.gameObject.SetActive(false);
394             }
395         }
396     }

```

```

386         webViewLoadingText.transform.localPosition = new Vector3(0,
387             -heightInUnityUnits, 0);
388     }
389 };
390
391 }
392
393 private void ResizeWebView(float heightInPixels, out float
394     heightInUnityUnits)
395 {
396     //Resize webview window to accommodate full webpage height
397     Debug.Log("Webpage Height: " + heightInPixels);
398     var existingWidth = webViewPrefab.WebView.Size.x;
399     heightInUnityUnits = heightInPixels / webViewPrefab.WebView.
400         Resolution;
401
402     webViewPrefab.Resize(existingWidth, heightInUnityUnits);
403
404     //Resize backing frame accordingly
405     webViewFrame.transform.localScale = new Vector3(existingWidth + 0.2f
406         , heightInUnityUnits + 0.2f, 0.09f);
407     webViewFrame.transform.localPosition = new Vector3(0.051f, -(
408         heightInUnityUnits / 2 - 0.5f), 0);
409
410     RefreshCollisionHeight(heightInUnityUnits);
411 }
412
413 IEnumerator GetImageTexture(string url)
414 {
415     UnityWebRequest www = UnityWebRequestTexture.GetTexture(url);
416     yield return www.SendWebRequest();
417
418     if (www.isNetworkError || www.isHttpError)
419     {
420         //Debug.Log("Image retrieval error");
421     }
422     else
423     {
424         imageTexture = DownloadHandlerTexture.GetContent(www);
425         //Debug.Log("Image downloaded");
426         Invoke("RefreshImage", 5.0f);
427     }
428 }
429
430 IEnumerator GetThumbnailTexture(string url)
431 {
432     UnityWebRequest www = UnityWebRequestTexture.GetTexture(url);

```

```

431     yield return www.SendWebRequest();
432
433     if (www.isNetworkError || www.isHttpError)
434     {
435         //Debug.Log("Thumbnail retrieval error");
436     }
437     else
438     {
439         thumbnailTexture = DownloadHandlerTexture.GetContent(www);
440         //Debug.Log("Thumbnail downloaded");
441         Invoke("RefreshThumbnail", 5.0f);
442     }
443
444 }
445
446 private void RefreshImage()
447 {
448     //SET IMAGE TEXTURE
449     if (imageTexture != null && imageTexture.height > 0 && imageTexture.
450         width > 0)
451     {
452         imagePlane.GetComponent<MeshRenderer>().enabled = true;
453         float imageRatio = (float)imageTexture.height / (float)
454             imageTexture.width;
455         imagePlane.transform.localScale = new Vector3(5f, 5f *
456             imageRatio, 0.1f);
457         imagePlane.transform.localPosition = new Vector3(7f, -(5f *
458             imageRatio - 5f) / 2 - 2, 0);
459         imagePlane.GetComponent<MeshRenderer>().material.SetTexture("_BaseMap", imageTexture);
460         imagePlane.GetComponent<FrameDynamic>().RefreshFrame();
461         //Debug.Log("Setting Texture");
462     }
463     else
464     {
465         imagePlane.GetComponent<MeshRenderer>().enabled = false;
466     }
467 }
468
469 private void RefreshThumbnail()
470 {
471     //SET THUMBNAIL TEXTURE
472     if (thumbnailTexture != null && thumbnailTexture.height > 0 &&
473         thumbnailTexture.width > 0)
474     {
475         externalThumbnail.GetComponent<MeshRenderer>().enabled = true;
476         float thumbnailRatio = (float)thumbnailTexture.height / (float)
477             thumbnailTexture.width;
478         //if (thumbnailRatio == 0) thumbnailRatio = 1;
479         float platformRatio = 12f / platformLengthUnits;

```



```

474         if (thumbnailRatio >= platformRatio)
475         {
476             externalThumbnail.transform.localScale = new Vector3(10f, 10
                f, 1);
477         }
478         else
479         {
480             externalThumbnail.transform.localScale = new Vector3(
                platformLengthUnits - 2, (platformLengthUnits - 2) *
                thumbnailRatio, 1);
481         }
482         externalThumbnail.GetComponent<MeshRenderer>().material.
                SetTexture("_BaseMap", thumbnailTexture);
483         //Debug.Log("Setting Texture");
484
485         //Resize title to default with thumbnail
486         externalTitle.GetComponent<RectTransform>().sizeDelta = new
                Vector2(platformLengthUnits - 3, 4);
487         externalTitle.transform.localPosition = new Vector3(0, 1.3f, 4.3
                f);
488     }
489     else
490     {
491         externalThumbnail.GetComponent<MeshRenderer>().enabled = false;
492
493         //Resize title to be larger in absence of thumbnail
494         externalTitle.GetComponent<RectTransform>().sizeDelta = new
                Vector2(platformLengthUnits - 3, 10);
495         externalTitle.transform.localPosition = new Vector3(0, 1.3f, 1.8
                f);
496     }
497 }
498
499 private List<Comment> RetrieveComments(int numComments)
500 {
501     //var numComments = 4; //Mathf.CeilToInt(2 * Mathf.PI *
                platformRadius / 8);
502     var topComments = postComments.GetComments(sort: "top", 0, 0, true,
                false, true, 2, 1000);
503     return topComments;
504 }
505
506 public void RefreshCollisionHeight(float yheight)
507 {
508     //only fire if yheight is greater than current height
509     if (yheight > collisionHeight)
510     {
511         yheight = Mathf.Ceil(yheight);
512         platformWall.transform.localScale = new Vector3(
                platformLengthUnits + 4, 18 + 8, yheight + 20 + 4);

```

```

513     platformWall.transform.localPosition = new Vector3(0, -((yheight
514         + 20)/2), 0);
515     platformBase.transform.localScale = new Vector3(
516         platformLengthUnits + 4, 18 + 8, yheight + 20 + 4);
517     platformBase.transform.localPosition = new Vector3(0, -(yheight
518         + 20 + 2));
519     platformAreaCollider.transform.localScale = new Vector3(
520         platformLengthUnits + 4, yheight + 20 + 20, 25);
521     platformAreaCollider.transform.localPosition = new Vector3(0, -(
522         (yheight + 20)/2), 0);
523
524     foreach(GameObject c in commentFramesList)
525     {
526         c.GetComponent<RedCommentFrameMonobehaviour>().
527             ResizeVerticalFX(yheight + 20 + 4 - 1);
528     }
529
530     collisionHeight = yheight;
531 }
532
533 }
534
535 public void Vote(int vote)
536 {
537     if(vote == 1)
538     {
539         post.UpvoteAsync();
540         platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial(1);
541         imagePlane.GetComponent<FrameDynamic>().SetFrameMaterial(1);
542         mainTitle.GetComponent<RedTextFrameMonobehaviour>().
543             frameGeometry.GetComponent<ColorChanger>().SetColor(1);
544         mainContent.GetComponent<RedTextFrameMonobehaviour>().
545             frameGeometry.GetComponent<ColorChanger>().SetColor(1);
546         webViewFrame.GetComponent<ColorChanger>().SetColor(1);
547     }
548     else if(vote == -1)
549     {
550         post.DownvoteAsync();
551         platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial(2);
552         imagePlane.GetComponent<FrameDynamic>().SetFrameMaterial(2);
553         mainTitle.GetComponent<RedTextFrameMonobehaviour>().
554             frameGeometry.GetComponent<ColorChanger>().SetColor(2);
555         mainContent.GetComponent<RedTextFrameMonobehaviour>().
556             frameGeometry.GetComponent<ColorChanger>().SetColor(2);
557         webViewFrame.GetComponent<ColorChanger>().SetColor(2);
558     }
559     else if(vote == 0)
560     {
561         post.UnvoteAsync();
562         platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial(0);

```

```

553     imagePlane.GetComponent<FrameDynamic>().SetFrameMaterial(0);
554     mainTitle.GetComponent<RedTextFrameMonobehaviour>().
555         frameGeometry.GetComponent<ColorChanger>().SetColor(0);
556     mainContent.GetComponent<RedTextFrameMonobehaviour>().
557         frameGeometry.GetComponent<ColorChanger>().SetColor(0);
558     webViewFrame.GetComponent<ColorChanger>().SetColor(0);
559 }
560
561 public void RetrieveVotes()
562 {
563     if(post.IsUpvoted)
564     {
565         interfaceButtons.GetComponentInChildren<
566             VoteControllerMonobehaviour>().SetVote(1);
567         platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial(1);
568         imagePlane.GetComponent<FrameDynamic>().SetFrameMaterial(1);
569         mainTitle.GetComponent<RedTextFrameMonobehaviour>().
570             frameGeometry.GetComponent<ColorChanger>().SetColor(1);
571         mainContent.GetComponent<RedTextFrameMonobehaviour>().
572             frameGeometry.GetComponent<ColorChanger>().SetColor(1);
573         webViewFrame.GetComponent<ColorChanger>().SetColor(1);
574     }
575     else if(post.IsDownvoted)
576     {
577         interfaceButtons.GetComponentInChildren<
578             VoteControllerMonobehaviour>().SetVote(-1);
579         platformSimple.GetComponent<FrameDynamic>().SetFrameMaterial(2);
580         imagePlane.GetComponent<FrameDynamic>().SetFrameMaterial(2);
581         mainTitle.GetComponent<RedTextFrameMonobehaviour>().
582             frameGeometry.GetComponent<ColorChanger>().SetColor(2);
583         mainContent.GetComponent<RedTextFrameMonobehaviour>().
584             frameGeometry.GetComponent<ColorChanger>().SetColor(2);
585         webViewFrame.GetComponent<ColorChanger>().SetColor(2);
586     }
587 }
588 }

```

## 5.2 Reddit Comment Frame

Script handling all the procedural generation and calculation of comment frames, as well as associated interactions.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Linq;
4 using UnityEngine;
5 using Reddit;
6 using Reddit.Controllers;
7 using TMPro;
8 using UnityEngine.UI;
9 using NaughtyAttributes;
10 using LogicUI.FancyTextRendering;
11
12 public class RedCommentFrameMonobehaviour : MonoBehaviour
13 {
14
15     //PLAYER
16     private GameObject rig;
17     private MovementProvider playerMovementProvider;
18
19     //GAMEOBJECT / GEOMETRY
20     private Camera XRcamera;
21     public GameObject canvas;
22     public GameObject frameGeometry;
23     public GameObject textContainer;
24     public GameObject mainText;
25     public GameObject refreshButton;
26     public GameObject voteArrows;
27     public GameObject subTextPrefab;
28     public GameObject helperContainer;
29     public Material subCommentMaterial;
30     public GameObject verticalFX;
31     public int totalReplies = 0;
32     private RedPostMonobehaviour parentPostController;
33     private ScrollRect scrollRect;
34     private MaterialManager materialManager = null;
35     private List<string> subCommentStrings = new List<string>();
36     private List<int> subCommentDepth = new List<int>();
37     private List<GameObject> subCommentObjectList = new List<GameObject>();
38     private List<Comment> subCommentList = new List<Comment>();
39
40     //REDDIT COMMENT DATA
41     private Comment mainComment = null;
42     private List<Comment> directReplies;
43
44
45     //PARAMETERS
46     private float frameWidth = 2.4f;
47     private float frameMargin = 0.2f;
48     private float yMargin = 0.05f;
49
50

```

```

51     private bool initialised = false;
52     private bool deepRefresh = false;
53     private Vector2 previousVector2 = new Vector2(0, 0);
54
55
56     public void InitialiseComments(Comment c)
57     {
58         mainComment = c;
59
60         if (!initialised)
61         {
62             //directReplies = RetrieveChildComments(c);
63             RetrieveAllChildComments(c.replies, 1);
64             totalReplies = c.NumReplies.Value;
65
66             UpdateText();
67             RetrieveVote();
68             //FitContents();
69             Invoke("FitContents", 0.1f); //need to delay fitcontents so '
              content fitter' has time to work
70             initialised = true;
71         }
72
73         parentPostController = transform.parent.GetComponent<
              RedPostMonobehaviour>();
74     }
75
76     // Start is called before the first frame update
77     void Start()
78     {
79         XRcamera = GameObject.Find("VR Camera Controller").GetComponent<
              Camera>();
80         rig = GameObject.Find("XR Rig");
81         playerMovementProvider = rig.GetComponent<MovementProvider>();
82         canvas.GetComponent<Canvas>().worldCamera = XRcamera;
83         scrollRect = textContainer.GetComponent<ScrollRect>();
84
85         materialManager = GameObject.Find("MaterialManager").GetComponent<
              MaterialManager>();
86
87         //canvas.SetActive(false);
88     }
89
90     // Update is called once per frame
91     void Update()
92     {
93
94     }
95
96     void UpdateText()

```

```

97     {
98         mainText.GetComponent<MarkdownRenderer>().Source = "";
99         mainText.GetComponent<MarkdownRenderer>().Source = mainComment.
            UpVotes + " | u/" + mainComment.Author + "\n" + mainComment.Body
            ;
100
101
102         for (int i = 0; i < subCommentStrings.Count; i++)
103         {
104             GameObject subComment = Instantiate(subTextPrefab, textContainer
                .transform);
105             //subComment.GetComponent<TextMeshProUGUI>().text =
                subCommentStrings[i];
106             subComment.GetComponent<RedCommentChildMonobehaviour>().
                Initialise(subCommentList[i], subCommentStrings[i], this);
107
108             Color subCommentColor = Color.HSVToRGB(0.4f, 0.3f, 0.2f + 0.6f/
                subCommentDepth[i]);
109             string subCommentColorName = string.Concat("CommentMaterial",
                subCommentDepth[i]);
110             //Debug.Log(subCommentColorName + " " + subCommentColor + " " +
                subCommentMaterial);
111
112             if(materialManager == null)
113             {
114                 materialManager = GameObject.Find("MaterialManager").
                    GetComponent<MaterialManager>();
115             }
116             Material subCommentBackgroundMaterial = materialManager.
                InstantiateMaterialWithColor(subCommentColorName,
                subCommentColor, subCommentMaterial);
117             subComment.transform.Find("ChildBackground").GetComponent<Image>
                ().material = subCommentBackgroundMaterial;
118             subCommentObjectList.Add(subComment);
119         }
120
121     }
122
123     void FitContents()
124     {
125         //canvas.SetActive(true);
126
127
128         var mainRectTransform = mainText.GetComponent<RectTransform>();
129         var mainHeight = mainRectTransform.sizeDelta[1];
130         mainRectTransform.anchoredPosition = new Vector3(0, -
            mainRectTransform.sizeDelta[1] / 2, 0);
131
132
133         var yOffsetTotal = 0.0f;

```

```

134     for (int i = 0; i < subCommentList.Count; i++)
135     {
136         var subRectTransform = subCommentObjectList[i].GetComponent<
            RectTransform>();
137         var subHeight = subRectTransform.sizeDelta[1];
138         subRectTransform.anchoredPosition = new Vector3(0.1f, -(
            mainHeight + (subHeight / 2) + yMargin + yOffsetTotal), 0);
139
140         yOffsetTotal += subHeight + yMargin;
141     }
142
143     var textContainerRect = textContainer.GetComponent<RectTransform>();
144     var columnHeight = mainHeight + yOffsetTotal + 0.1f;
145     textContainerRect.sizeDelta = new Vector2(frameWidth, columnHeight);
146     textContainerRect.anchoredPosition = new Vector3(0, -columnHeight /
        2, 0);
147
148     frameGeometry.transform.localScale = new Vector3(frameWidth +
        frameMargin, columnHeight + frameMargin, 0.1f);
149     frameGeometry.transform.localPosition = new Vector3(0, -columnHeight
        / 2, 0.1f);
150
151     //Refresh total height
152     parentPostController.RefreshCollisionHeight(yOffsetTotal);
153 }
154
155 List<Comment> RetrieveChildComments(Comment comment)
156 {
157     return comment.Replies;
158 }
159
160 void RetrieveAllChildComments(IList<Comment> comments, int depth)
161 {
162     if (comments != null)
163     {
164         foreach (Comment comment in comments)
165         {
166             subCommentStrings.Add(comment.UpVotes + " | u/" + comment.
                Author + "\n" + string.Concat(Enumerable.Repeat<string>(
                ">", depth)) + " " + comment.Body);
167             subCommentList.Add(comment);
168             subCommentDepth.Add(depth);
169             //Debug.Log(comment.replies.Count);
170             if (comment != null) RetrieveAllChildComments(comment.replies
                , (depth + 1));
171         }
172     }
173 }
174
175 public void DeepRefreshComments()

```

```

176     {
177         if(!deepRefresh)
178         {
179             foreach(GameObject c in subCommentObjectList)
180             {
181                 GameObject.Destroy(c);
182             }
183             subCommentObjectList.Clear();
184             subCommentList.Clear();
185             subCommentDepth.Clear();
186             subCommentStrings.Clear();
187
188             deepRefresh = true;
189
190             RetrieveAllChildComments(mainComment.Comments.GetTop(limit:500),
191                                     1);
192             UpdateText();
193             Invoke("FitContents", 0.1f);
194
195             refreshButton.GetComponent<RefreshCommentMonobehaviour>().
196                 SetRefreshed(true);
197         }
198
199     public void Deactivate()
200     {
201         canvas.SetActive(false);
202     }
203
204     public void MovePlayer(Vector2 vector)
205     {
206         var velocity = scrollRect.velocity;
207         playerMovementProvider.VertMove(-velocity.y);
208     }
209
210     public void ResizeVerticalFX(float height)
211     {
212         verticalFX.transform.localScale = new Vector3(2f, height, 1);
213         verticalFX.transform.localPosition = new Vector3(0, -height/2, 0.1f)
214         ;
215     }
216
217     public bool Refreshed()
218     {
219         return deepRefresh;
220     }
221
222     public void Vote(int vote)
223     {

```



```

223     if(vote == 1)
224     {
225         mainComment.UpvoteAsync();
226         frameGeometry.GetComponent<ColorChanger>().SetColor(1);
227     }
228     else if(vote == -1)
229     {
230         mainComment.DownvoteAsync();
231         frameGeometry.GetComponent<ColorChanger>().SetColor(2);
232     }
233     else if(vote == 0)
234     {
235         mainComment.UnvoteAsync();
236         frameGeometry.GetComponent<ColorChanger>().SetColor(0);
237     }
238 }
239
240 public void RetrieveVote()
241 {
242     VoteControllerMonobehaviour voteController = voteArrows.GetComponent
243         <VoteControllerMonobehaviour>();
244     if(mainComment.IsUpvoted)
245     {
246         voteController.SetVote(1);
247         frameGeometry.GetComponent<ColorChanger>().SetColor(1);
248     }
249     else if(mainComment.IsDownvoted)
250     {
251         voteController.SetVote(-1);
252         frameGeometry.GetComponent<ColorChanger>().SetColor(2);
253     }
254 }
255 public int GetHiddenComments()
256 {
257     int numComments = subCommentList.Sum(x => x.NumReplies).Value;
258     return numComments;
259 }
260 }

```

### 5.3 Reddit Comment Child

Script attached to each child comment for finer control of attributes.

```

1 using System.Collections;

```

```

2 using System.Collections.Generic;
3 using UnityEngine;
4 using Reddit;
5 using Reddit.Controllers;
6 using TMPro;
7 using NaughtyAttributes;
8 using LogicUI.FancyTextRendering;
9
10 public class RedCommentChildMonobehaviour : MonoBehaviour
11 {
12     private RedCommentFrameMonobehaviour parentCommentController;
13     private Comment comment;
14     private string commentString;
15
16     //Geometry
17     public TextMeshProUGUI textMeshComponent;
18     public MarkdownRenderer markdownRendererComponent;
19     public VoteControllerMonobehaviour voteController;
20
21     // Start is called before the first frame update
22     void Start()
23     {
24
25     }
26
27     // Update is called once per frame
28     void Update()
29     {
30
31     }
32
33     public void Initialise(Comment c, string s, RedCommentFrameMonobehaviour
34         parent)
35     {
36         comment = c;
37         commentString = s;
38         parentCommentController = parent;
39
40         UpdateText(commentString);
41         RetrieveVote();
42     }
43
44     public void UpdateText(string s)
45     {
46         markdownRendererComponent.Source = s;
47     }
48
49     public void Vote(int vote)
50     {
51         if(vote == 1)

```

```

51     {
52         comment.UpvoteAsync();
53     }
54     else if (vote == -1)
55     {
56         comment.DownvoteAsync();
57     }
58     else if (vote == 0)
59     {
60         comment.UnvoteAsync();
61     }
62 }
63
64 public void RetrieveVote()
65 {
66     if (comment.IsUpvoted)
67     {
68         voteController.SetVote(1);
69     }
70     else if (comment.IsDownvoted)
71     {
72         voteController.SetVote(-1);
73     }
74 }
75 }

```

## 5.4 Vote Controller

Script attached to the upvote/downvote arrows that handle the API operation and toggle the arrow colors.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class VoteControllerMonoBehaviour : MonoBehaviour
6 {
7     public RedPostMonoBehaviour parentPostController;
8     public RedCommentChildMonoBehaviour parentCommentChildController;
9     public RedCommentFrameMonoBehaviour parentCommentController;
10    public GameObject arrowUpvote;
11    public GameObject arrowDownvote;
12    private int vote = 0;
13

```

```

14 // Start is called before the first frame update
15 void Start()
16 {
17
18 }
19
20 public void ToggleUpvote(bool live = true)
21 {
22     if (vote != 1)
23     {
24         vote = 1;
25         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
26         arrowUpvote.GetComponent<ColorChanger>().SetColor(1);
27
28         if (parentPostController != null && live) parentPostController.
                Vote(1);
29         if (parentCommentChildController != null && live)
                parentCommentChildController.Vote(1);
30         if (parentCommentController != null && live)
                parentCommentController.Vote(1);
31     }
32     else
33     {
34         vote = 0;
35         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
36         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
37
38         if (parentPostController != null && live) parentPostController.
                Vote(0);
39         if (parentCommentChildController != null && live)
                parentCommentChildController.Vote(0);
40         if (parentCommentController != null && live)
                parentCommentController.Vote(0);
41     }
42 }
43
44 public void ToggleDownvote(bool live = true)
45 {
46     if (vote != -1)
47     {
48         vote = -1;
49         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
50         arrowDownvote.GetComponent<ColorChanger>().SetColor(1);
51
52         if (parentPostController != null && live) parentPostController.
                Vote(-1);
53         if (parentCommentChildController != null && live)
                parentCommentChildController.Vote(-1);
54         if (parentCommentController != null && live)
                parentCommentController.Vote(-1);

```

```

55     }
56     else
57     {
58         vote = 0;
59         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
60         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
61
62         if(parentPostController != null && live) parentPostController.
63             Vote(0);
64         if(parentCommentChildController != null && live)
65             parentCommentChildController.Vote(0);
66         if(parentCommentController != null && live)
67             parentCommentController.Vote(0);
68     }
69 }
70
71 public void HoverUpvote()
72 {
73     //Debug.Log("hover upvote");
74     if(vote != 1)
75     {
76         arrowUpvote.GetComponent<ColorChanger>().SetColor(2);
77     }
78 }
79
80 public void HoverDownvote()
81 {
82     //Debug.Log("hover downvote");
83     if(vote != -1)
84     {
85         arrowDownvote.GetComponent<ColorChanger>().SetColor(2);
86     }
87 }
88
89 public void HoverOutUpvote()
90 {
91     if(vote == 1)
92     {
93         arrowUpvote.GetComponent<ColorChanger>().SetColor(1);
94     }
95     else
96     {
97         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
98     }
99 }
100
101 public void HoverOutDownvote()
102 {
103     if(vote == -1)
104     {

```

```

102         arrowDownvote.GetComponent<ColorChanger>().SetColor(1);
103     }
104     else
105     {
106         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
107     }
108 }
109
110 public void ResetColors()
111 {
112     if(vote == 0)
113     {
114         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
115         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
116     }
117     else if(vote == 1)
118     {
119         arrowUpvote.GetComponent<ColorChanger>().SetColor(1);
120         arrowDownvote.GetComponent<ColorChanger>().SetColor(0);
121     }
122     else if(vote == -1)
123     {
124         arrowUpvote.GetComponent<ColorChanger>().SetColor(0);
125         arrowDownvote.GetComponent<ColorChanger>().SetColor(1);
126     }
127 }
128
129 public void SetVote(int i)
130 {
131     vote = i;
132     Invoke("ResetColors", 0.1f);
133 }
134 }

```

## 5.5 Refresh Comments

Script attached to the refresh arrow that handles the refresh operation.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using TMPro;
5
6 public class RefreshCommentMonoBehaviour : MonoBehaviour
7 {

```

```

8
9 public RedPostMonobehaviour parentPostController;
10 public RedCommentFrameMonobehaviour parentCommentFrameController;
11 public GameObject refreshText;
12 public GameObject refreshArrow;
13 private bool refreshed = false;
14
15
16 // Start is called before the first frame update
17 void Start()
18 {
19     if(parentCommentFrameController != null)
20     {
21         //refreshText.GetComponent<TextMeshPro>().text = "Load More
22             Comments\n>" + parentCommentFrameController.
23             GetHiddenComments();
24     }
25     else if(parentPostController != null)
26     {
27         //refreshText.GetComponent<TextMeshPro>().text = "Load All
28             Comments";
29     }
30     refreshText.SetActive(false);
31 }
32
33 public void Refresh()
34 {
35     if(parentCommentFrameController != null)
36     {
37         RefreshComments();
38         SetRefreshed(true);
39     }
40     else if(parentPostController != null)
41     {
42         RefreshAllPostComments();
43         SetRefreshed(true);
44     }
45 }
46
47 public void RefreshComments()
48 {
49     if(!parentCommentFrameController.Refreshed())
50     {
51         parentCommentFrameController.DeepRefreshComments();
52     }
53 }
54 public void RefreshAllPostComments()
55 {
56     parentPostController.DeepRefreshAllComments();
57 }

```

```

55
56 public void SetRefreshed(bool x)
57 {
58     refreshed = x;
59     refreshArrow.GetComponent<ColorChanger>().SetColor(2);
60     refreshText.SetActive(false);
61     refreshArrow.SetActive(false);
62 }
63
64 public void HoverIn()
65 {
66     if(!refreshed)
67     {
68         refreshText.SetActive(true);
69         refreshArrow.GetComponent<ColorChanger>().SetColor(1);
70     }
71 }
72
73 public void HoverOut()
74 {
75     refreshText.SetActive(false);
76     if(!refreshed)
77     {
78         refreshArrow.GetComponent<ColorChanger>().SetColor(0);
79     }
80 }
81 }

```

## 6 General Support Scripts

Some general scripts and libraries to handle miscellaneous operations, mostly regarding procedural geometry.

### 6.1 Scroll Surface

Key script that handles the reverse-scrolling of the player - uses an invisible helper scroll rectangle that moves the player in the opposite direction when scrolled.



```

1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Linq;
4 using UnityEngine;
5 using Reddit;
6 using Reddit.Controllers;
7 using TMPro;
8 using UnityEngine.UI;
9
10 public class RedScrollSurface : MonoBehaviour
11 {
12
13     //PLAYER
14     private GameObject rig;
15     private MovementProvider playerMovementProvider;
16
17     //GAMEOBJECT / GEOMETRY
18     private Camera XRcamera;
19     public GameObject cameraCanvas;
20     public GameObject scrollRectContainer;
21     public GameObject scaleReference = null;
22     private ScrollRect scrollRect;
23
24     //PARAMETERS
25     private float frameWidth = 2.4f;
26     private float frameMargin = 0.2f;
27     private float yMargin = 0.2f;
28
29
30     private Vector2 previousVector2 = new Vector2(0, 0);
31
32
33     // Start is called before the first frame update
34     void Start()
35     {
36         XRcamera = GameObject.Find("VR Camera Controller").GetComponent<
37             Camera>();
38         rig = GameObject.Find("XR Rig");
39         playerMovementProvider = rig.GetComponent<MovementProvider>();
40         cameraCanvas.GetComponent<Canvas>().worldCamera = XRcamera;
41         scrollRect = scrollRectContainer.GetComponent<ScrollRect>();
42     }
43
44     // Update is called once per frame
45     void Update()
46     {
47
48     }
49

```

```

50     public void MovePlayer(Vector2 vector)
51     {
52         var velocity = scrollRect.velocity;
53         if(scaleReference != null)
54         {
55             velocity = velocity * scaleReference.transform.localScale.y;
56         }
57         playerMovementProvider.VertMove(-velocity.y);
58     }
59
60     public void SnapPlayer()
61     {
62
63     }
64 }

```

## 6.2 Frame Dynamic

Creates the 3D frame around certain UI elements to make them less flat.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5
6  public class FrameDynamic : MonoBehaviour
7  {
8      public float frameThickness = 0.1f;
9      public float frameDepth = 0f;
10     public Material frameMaterial;
11     public List<Material> frameMaterials;
12     private List<GameObject> frameObjects;
13     private GameObject frameUpper;
14     private GameObject frameRight;
15     private GameObject frameLower;
16     private GameObject frameLeft;
17
18     // Start is called before the first frame update
19     void Start()
20     {
21         CreateFrame();
22     }
23
24     // Update is called once per frame
25     void Update()

```

```

26     {
27
28     }
29
30     void CreateFrame()
31     {
32         if (frameUpper != null || frameRight != null || frameLower != null
33             || frameLeft != null)
34         {
35             return;
36         }
37
38         frameMaterials.Insert(0, frameMaterial);
39
40         frameUpper = GameObject.CreatePrimitive(PrimitiveType.Cube);
41         frameRight = GameObject.CreatePrimitive(PrimitiveType.Cube);
42         frameLower = GameObject.CreatePrimitive(PrimitiveType.Cube);
43         frameLeft = GameObject.CreatePrimitive(PrimitiveType.Cube);
44
45         frameUpper.transform.SetParent(transform);
46         frameRight.transform.SetParent(transform);
47         frameLower.transform.SetParent(transform);
48         frameLeft.transform.SetParent(transform);
49
50         RefreshFrame();
51     }
52
53     public void RefreshFrame()
54     {
55         if (frameUpper == null || frameRight == null || frameLower == null
56             || frameLeft == null)
57         {
58             CreateFrame();
59         }
60
61         frameUpper.transform.localRotation = Quaternion.identity;
62         frameRight.transform.localRotation = Quaternion.identity;
63         frameLower.transform.localRotation = Quaternion.identity;
64         frameLeft.transform.localRotation = Quaternion.identity;
65
66         frameUpper.transform.localPosition = new Vector3(0, (transform.
67             localScale.y / 2 + frameThickness / 2) / transform.localScale.y,
68             0);
69         frameRight.transform.localPosition = new Vector3(- (transform.
70             localScale.x / 2 + frameThickness / 2) / transform.localScale.x
71             , 0, 0);
72         frameLower.transform.localPosition = new Vector3(0, - (transform.
73             localScale.y / 2 + frameThickness / 2) / transform.localScale.y
74             , 0);
75         frameLeft.transform.localPosition = new Vector3((transform.
76             localScale.x / 2 + frameThickness / 2) / transform.localScale.x

```

```

        , 0, 0);
67
68     frameUpper.transform.localScale = new Vector3( (transform.localScale
        .x + 2 * frameThickness) / transform.localScale.x,
        frameThickness / transform.localScale.y, (transform.localScale.z
        + frameDepth) / transform.localScale.z);
69     frameRight.transform.localScale = new Vector3(frameThickness /
        transform.localScale.x, (transform.localScale.y) / transform.
        localScale.y, (transform.localScale.z + frameDepth) / transform.
        localScale.z);
70     frameLower.transform.localScale = new Vector3( (transform.localScale
        .x + 2 * frameThickness) / transform.localScale.x,
        frameThickness / transform.localScale.y, (transform.localScale.z
        + frameDepth) / transform.localScale.z);
71     frameLeft.transform.localScale = new Vector3(frameThickness /
        transform.localScale.x, (transform.localScale.y) / transform.
        localScale.y, (transform.localScale.z + frameDepth) / transform.
        localScale.z);
72
73     SetFrameMaterial(0);
74 }
75
76 public void SetFrameMaterial(int i = 0)
77 {
78     if (frameUpper == null || frameRight == null || frameLower == null
79         || frameLeft == null)
80     {
81         CreateFrame();
82     }
83     frameUpper.GetComponent<MeshRenderer>().material = frameMaterials[i]
84     ;
85     frameRight.GetComponent<MeshRenderer>().material = frameMaterials[i]
86     ;
87     frameLower.GetComponent<MeshRenderer>().material = frameMaterials[i]
88     ;
89     frameLeft.GetComponent<MeshRenderer>().material = frameMaterials[i];
90 }
91
92 public void ReplaceBaseMaterial(Material mat)
93 {
94     if (frameUpper == null || frameRight == null || frameLower == null
95         || frameLeft == null)
96     {
97         CreateFrame();
98     }
99     frameMaterials[0] = mat;
100 }

```

## 6.3 Frame Dynamic Planes

Same as frame dynamic but for planar elements.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5
6 public class FrameDynamicPlanes : MonoBehaviour
7 {
8     public float frameDepth = 0f;
9     public Material frameMaterial;
10    private List<GameObject> frameObjects;
11    private GameObject frameUpper;
12    private GameObject frameRight;
13    private GameObject frameLower;
14    private GameObject frameLeft;
15
16    // Start is called before the first frame update
17    void Start()
18    {
19        CreateFrame();
20    }
21
22    // Update is called once per frame
23    void Update()
24    {
25
26    }
27
28    public void RefreshFrame()
29    {
30        if (frameUpper == null || frameRight == null || frameLower == null
31            || frameLeft == null)
32        {
33            return;
34        }
35        frameUpper.transform.localRotation = Quaternion.Euler(270, 90, 0);
36        frameRight.transform.localRotation = Quaternion.Euler(180, 90, 0);
37        frameLower.transform.localRotation = Quaternion.Euler(90, 90, 0);
38        frameLeft.transform.localRotation = Quaternion.Euler(0, 90, 0);
39
40        frameUpper.transform.localPosition = new Vector3(0, (transform.
41            localScale.y / 2) / transform.localScale.y, 0);
42        frameRight.transform.localPosition = new Vector3(- (transform.
43            localScale.x / 2) / transform.localScale.x, 0, 0);
```

```

41     frameLower.transform.localPosition = new Vector3(0, - (transform.
42         localScale.y / 2 ) / transform.localScale.y, 0);
43
44     frameUpper.transform.localScale = new Vector3( (transform.localScale
45         .x + frameDepth) / transform.localScale.x, 1, 1);
46     frameRight.transform.localScale = new Vector3( (transform.localScale
47         .x + frameDepth) / transform.localScale.x, 1, 1);
48     frameLower.transform.localScale = new Vector3( (transform.localScale
49         .x + frameDepth) / transform.localScale.x, 1, 1);
50     frameLeft.transform.localScale = new Vector3( (transform.localScale.
51         x + frameDepth) / transform.localScale.x, 1, 1);
52
53     frameUpper.GetComponent<MeshRender>().material = frameMaterial;
54     frameRight.GetComponent<MeshRender>().material = frameMaterial;
55     frameLower.GetComponent<MeshRender>().material = frameMaterial;
56     frameLeft.GetComponent<MeshRender>().material = frameMaterial;
57 }
58
59 void CreateFrame()
60 {
61     frameUpper = GameObject.CreatePrimitive(PrimitiveType.Quad);
62     frameRight = GameObject.CreatePrimitive(PrimitiveType.Quad);
63     frameLower = GameObject.CreatePrimitive(PrimitiveType.Quad);
64     frameLeft = GameObject.CreatePrimitive(PrimitiveType.Quad);
65
66     frameUpper.transform.SetParent(transform);
67     frameRight.transform.SetParent(transform);
68     frameLower.transform.SetParent(transform);
69     frameLeft.transform.SetParent(transform);
70
71     RefreshFrame();
72 }
73 }

```

## 6.4 Text Frame

Some stuff to handle text panels.

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class RedTextFrameMonoBehaviour : MonoBehaviour

```

```

6 {
7
8     public GameObject frameGeometry;
9     public GameObject textContainer;
10    public GameObject mainText;
11    public AlignEnum alignment;
12
13    //PARAMETERS
14    public float frameWidth = 4f;
15    public float frameMargin = 0.2f;
16
17    public enum AlignEnum
18    {
19        Top,
20        Bottom
21    };
22
23    // Start is called before the first frame update
24    void Start()
25    {
26
27    }
28
29    // Update is called once per frame
30    void Update()
31    {
32
33    }
34
35    public void Initialise()
36    {
37        Invoke("FitContents", 0.1f);
38    }
39
40    void FitContents()
41    {
42
43        var mainTextRect = mainText.GetComponent<RectTransform>();
44        var mainHeight = mainTextRect.sizeDelta[1];
45
46        var textContainerRect = textContainer.GetComponent<RectTransform>();
47        textContainerRect.sizeDelta = new Vector2(frameWidth, mainHeight);
48
49        frameGeometry.transform.localScale = new Vector3(frameWidth +
50            frameMargin, mainHeight + frameMargin, 0.1f);
51
52        if(alignment.ToString() == "Bottom")
53        {

```

```

54         textContainerRect.anchoredPosition = new Vector3(0, mainHeight /
55             2, 0);
56         mainTextRect.anchoredPosition = new Vector3(0, mainHeight / 2, 0
57             );
58         frameGeometry.transform.localPosition = new Vector3(0, - (1 -
59             mainHeight / 2), 0.051f);
60         //transform.localPosition += new Vector3(0, 2f, 0); //moves
61         //entire gameobject to correct position
62     }
63     else
64     {
65         textContainerRect.anchoredPosition = new Vector3(0, - mainHeight
66             / 2, 0);
67         mainTextRect.anchoredPosition = new Vector3(0, - mainHeight / 2,
68             0);
69         frameGeometry.transform.localPosition = new Vector3(0, (1 -
70             mainHeight / 2), 0.051f);
71     }
72
73     /*
74     var yOffsetTotal = 0.0f;
75     for (int i = 0; i < subCommentList.Count; i++)
76     {
77         var subRectTransform = subCommentList[i].GetComponent<
78             RectTransform>();
79         var subHeight = subRectTransform.sizeDelta[1];
80         subRectTransform.anchoredPosition = new Vector3(0, -(mainHeight
81             + (subHeight / 2) + yMargin + yOffsetTotal), 0);
82
83         yOffsetTotal += subHeight + yMargin;
84     }
85
86     var textContainerRect = textContainer.GetComponent<RectTransform>();
87     var columnHeight = mainHeight + yOffsetTotal + yMargin;
88     textContainerRect.sizeDelta = new Vector2(frameWidth, columnHeight);
89     textContainerRect.anchoredPosition = new Vector3(0, -columnHeight /
90         2, 0);
91
92     frameGeometry.transform.localScale = new Vector3(frameWidth +
93         frameMargin, columnHeight + frameMargin, 0.1f);
94     frameGeometry.transform.localPosition = new Vector3(0, -columnHeight
95         / 2, 0.1f);
96
97     //transform.localPosition += new Vector3(0, mainHeight + 0.5f, 0);
98     if(!deepRefresh) transform.localPosition += new Vector3(0, 2f, 0);
99     */
100 }

```



## 6.5 Material Manager

Important script handling all dynamic material instances - ensures that procedural material generation goes through a central dictionary and uses the same materials if already instantiated, instead of creating a new instance for 10000s of comments.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System;
4 using UnityEngine;
5
6 public class MaterialManager : MonoBehaviour
7 {
8
9     public Material defaultMaterial;
10    public Material defaultNeon;
11    public Material skyBox;
12    public int dayInterval;
13    private Dictionary<string, Material> materials = new Dictionary<string,
14        Material>();
15
16    // Start is called before the first frame update
17    void Start()
18    {
19
20    }
21
22    void Update()
23    {
24        AdjustDayNight();
25    }
26
27    public Material InstantiateMaterialWithColor(string materialName, Color
28        materialColor, Material material = null)
29    {
30        //Debug.Log("Instantiating material - " + materialName);
31        if(material == null)
32        {
33            material = defaultMaterial;
34        }
35
36        if(!materials.ContainsKey(materialName))
37        {
38            Material newMaterial = Material.Instantiate(material);
```

```

37         newMaterial.SetColor("_BaseColor", materialColor);
38         Debug.Log("Instantiating new material - " + materialName);
39
40         materials.Add(materialName, newMaterial);
41     }
42
43     return materials[materialName];
44 }
45
46 public Material InstantiateMaterialWithEmission(string materialName,
47     Color materialColor, float intensity, Material material = null)
48 {
49     //Debug.Log("Instantiating material - " + materialName);
50     if(material == null)
51     {
52         material = defaultNeon;
53     }
54
55     if(!materials.ContainsKey(materialName))
56     {
57         Color emissionColor = new Vector4(materialColor.r * intensity,
58             materialColor.g * intensity, materialColor.b * intensity, 1.
59             0f);
60         Material newMaterial = Material.Instantiate(material);
61         newMaterial.SetColor("_EmissionColor", emissionColor);
62         //Debug.Log("Instantiating new material - " + materialName);
63
64         materials.Add(materialName, newMaterial);
65     }
66
67     return materials[materialName];
68 }
69
70 public void AdjustDayNight()
71 {
72     float currentTime = DateTime.Now.Minute + DateTime.Now.Second/60f;
73     float scaledDay = (Mathf.Abs(currentTime / 59f - 0.5f) - 0.25f) * 3;
74
75     float exposure = 1.1f + scaledDay;
76
77     RenderSettings.skybox.SetFloat("_Exposure", exposure);
78 }
79 }

```

## 6.6 Lib Color

I wanted more stuff in here but it's just a random color function at the moment that takes in HSV values (easier to control result).

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public static class LibColor
6 {
7
8     public static Color RandomiseHSV(Color inputColor, float varianceHue=0.0
9         5f, float varianceSaturation=0.05f, float varianceValue=0.05f )
10    {
11        float[] HSV = { 0, 0, 0 };
12        Color.RGBToHSV(inputColor, out HSV[0], out HSV[1], out HSV[2]);
13        HSV[0] = Random.Range(HSV[0] - varianceHue, HSV[0] + varianceHue);
14        HSV[1] = Random.Range(HSV[1] - varianceSaturation, HSV[1] +
15            varianceSaturation);
16        HSV[2] = Random.Range(HSV[2] - varianceValue, HSV[2] + varianceValue
17            );
18
19        Color newColor = Color.HSVToRGB(HSV[0], HSV[1], HSV[2]);
20        return newColor;
21    }
22 }
```

## 6.7 ColorRandomiser

Monobehaviour script that makes use of the aforementioned color randomising function.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ColorRandomiser : MonoBehaviour
6 {
```

```

7   public Color platformColor;
8   public float varianceHue = 0.05f;
9   public float varianceSaturation = 0.05f;
10  public float varianceValue = 0.05f;
11  private float[] HSV = { 0, 0, 0 };
12
13
14  // Start is called before the first frame update
15  void Start()
16  {
17      Color.RGBToHSV(platformColor, out HSV[0], out HSV[1], out HSV[2]);
18      HSV[0] = Random.Range(HSV[0] - varianceHue, HSV[0] + varianceHue);
19      HSV[1] = Random.Range(HSV[1] - varianceSaturation, HSV[1] +
20                          varianceSaturation);
21      HSV[2] = Random.Range(HSV[2] - varianceValue, HSV[2] + varianceValue
22                          );
23
24      //Change platform material color
25      Color newColor = Color.HSVToRGB(HSV[0], HSV[1], HSV[2]);
26      GetComponent<MeshRenderer>().material.color = newColor;
27  }
28
29  // Update is called once per frame
30  void Update()
31  {
32  }

```

## 6.8 ColorChanger

Clever universal script that allows you to set specific materials to an array that is unique to this component instance, then call their indices through another script to change object materials.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ColorChanger : MonoBehaviour
6  {
7      Material redMat;
8      Material blueMat;

```

```

9     Material greenMat;
10    Material clearMat;
11    Material defaultMat;
12    public List<Material> materialList = new List<Material>();
13
14    void Start()
15    {
16        defaultMat = GetComponent<MeshRenderrer>().material;
17        materialList.Insert(0, defaultMat);
18
19        redMat = Resources.Load("Materials/Select-Red", typeof(Material)) as
20            Material;
21        blueMat = Resources.Load("Materials/Select-Blue", typeof(Material))
22            as Material;
23        greenMat = Resources.Load("Materials/Select-Green", typeof(Material))
24            as Material;
25        clearMat = Resources.Load("Materials/Select-Clear", typeof(Material))
26            as Material;
27    }
28
29    public void SetMaterialDefault()
30    {
31        GetComponent<MeshRenderrer>().material = defaultMat;
32    }
33
34    public void SetColorRed()
35    {
36        GetComponent<MeshRenderrer>().material = redMat;
37    }
38
39    public void SetColorBlue()
40    {
41        GetComponent<MeshRenderrer>().material = blueMat;
42    }
43
44    public void SetColorGreen()
45    {
46        GetComponent<MeshRenderrer>().material = greenMat;
47    }
48
49    public void SetColorClear()
50    {
51        GetComponent<MeshRenderrer>().material = clearMat;
52    }
53
54    public void SetColor(int colorIndex)
55    {
56        if(colorIndex < materialList.Count)
57        {
58            GetComponent<MeshRenderrer>().material = materialList[colorIndex]
59            ;
60        }
61    }

```

```
54     }  
55   }  
56 }
```



