Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

TREBALL FINAL DE GRAU

ANNEX

VIRTUAL REALITY TO ENHANCE SAFETY AND HEALTH IN CONSTRUCTION

ONLINE MULTIPLAYER SERIOUS GAME.

Treball realitzat per:

Marc Arnau Navarro

Dirigit per:

Francisco Javier Mora Serrano

Ignacio Valero López

Grau en:

Enginyeria Civil

Barcelona, setembre de 2018

Departament d'Enginyeria Civil i Ambiental

This Annex is my explanation of all my written scripts of this project.

It goes from most important scripts to less important (although the order could be arguable). In here I try to explain how C# works while programming in Unity3D, in what I think is an understandable and easy way.

I hope that the person who reads this takes in account the number of late night hours put in this and that will be able to forgive me if there are typos or any other kind of mistakes.

Enjoy.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking; //We type this to tell this script that it'll
     have Networked components.
5
6  /*Probably the most important script of all. THIS WILL CONTROL ALL THE
     SCRIPTS ATTATCHED TO THE PLAYER.
7  Note that when you are using Networking your player prefab gets a copy of
     itself
8  in every client/host. So, you may be player1 in your computer and you are
     seeing the copy of
9  player2 in the screen and visceversa for the person that is playing with
     you.
10
11  Since you would like to control your player and ONLY your player, and you
      don't want other
12  persons to control yours, we apply this script, that enables all the
      scripts attatched to the
13  prefab of the player only if it isLocalPlayer.
14
15  That means that when you are a client the software will run through all the
       copies of player being
16  displayed and will enable the scripts only for the copy that you are
      supposed to control. Note that
17  in order for this to work you have to disable them manually in the
      inspector first.
18
19  And how am I going to see what other players do then? Well, thats why you
      have the
20  Network Identity -which basically says that the gameObject it is attatched
      to exists in the network,
21  and has to be checked as LocalPlayerAuthority for the player prefab -, the
      Network Transform -that updates
22  the position of the gameObject it is attatched to to all the other clients -
       and the Network Animator
23  -that in case you have animations in your game it will update them too -.
      Note how this 3 components
24  have to be enabled in the inspector!*/
25
26
27
28 public class NetworkedPlayerScript : NetworkBehaviour //NetworkBehaviour
     instead of Monobehaviour.
29 {
30     public UnityStandardAssets.Characters.FirstPerson.FirstPersonController
         fpsController; /*
31     We are defining the script FirstPersonController under the name
         "fpsController". That fact
32     that it is "public" means that we will have to drag it into the
         inspector. Usually the way to declare things
33     for the script is not so long and messy, it is just because here we're
         using the Character Controller provided
```

```
34      by unity and it is done like this.*/
35      public Camera fpsCamera; //Declare our Camera named fpsCamera.
36      public AudioListener audioListener; //Declare our AudioListener.
37      public LiniaDeVida liniaDeVida; //Goes on with several other
          scripts...Easy uh?
38      public Helmet helmetScript;
39      public FallDamage2 fallDamage2;
40      public Player_Net player_net;
41      public GameObject UI;
42      public Earmuffs earmuffs;
43      public FirstAid firstAid;
44      public BuildWall donaldTrump;
45      public BuildWall2 donaldTrump2;
46      public BuildWall3 donaldTrump3;
47      public BuildWall4 donaldTrump4;
48      public BuildWall5 donaldTrump5;
49      public BuildWall6 donaldTrump6;
50      public BuildWall7 donaldTrump7;
51      public BuildWall8 donaldTrump8;
52      public BuildWall9 donaldTrump9;
53      public BuildWall10 donaldTrump10;
54      public BuildWall11 donaldTrump11;
55      public BuildWall12 donaldTrump12;
56      public PickUpBlock block;
57      public PickUpBoxRoof blockRoof;
58      public BuildWallWest21 build1;
59      public BuildWallWest22 build12;
60      public BuildWallWest23 build13;
61      public BuildWallWest24 build14;
62      public BuildWallWest6 build15;
63      public BuildWallWest27 build16;
64      public BuildWall28 build17;
65      public BuildWallWest29 build18;
66      public BuildWallWest210 build19;
67      public BuildWallWest211 build10;
68      public BuildWallWest213 build11;
69      public BuildWallWest214 build112;
70      public BuildWallWest15 build113;
71      public BuildWallWest216 build114;
72      public BuildWallWest217 build115;
73      public BuildWallWest218 build116;
74      public BuildWallWest219 build117;
75      public BuildWallEast21 build221;
76      public BuildWallEast22 build222;
77      public BuildWallEast3 build223;
78      public BuildWallEast24 build224;
79      public BuildWallEast26 build225;
80      public BuildWallEast27 build226;
81      public BuildWallEast28 build227;
82      public BuildWallEast29 build228;
83      public BuildWallEast210 build229;
84      public BuildWallEast211 build2210;
85      public BuildWallEast213 build2211;
```

```
 86      public BuildWallEast214 build2212;
 87      public BuildWallEast215 build2213;
 88      public BuildWallEast216 build2214;
 89      public BuildWallEast217 build2215;
 90      public BuildWallEast218 build2216;
 91      public BuildWallEast219 build2217;
 92      public BuildWallSouth21 buildsouth1;
 93      public BuildWallSouth22 buildsouth2;
 94      public BuildWallSouth23 buildsouth3;
 95      public BuildWallSouth4 buildsouth4;
 96      public BuildWallSouth25 buildsouth5;
 97      public BuildWallSouth26 buildsouth6;
 98      public BuildWallSouth27 buildsouth7;
 99      public BuildWallSouth28 buildsouth8;
100      public BuildWallSouth29 buildsouth9;
101      public BuildWallSouth210 buildsouth10;
102      public BuildWallSouth211 buildsouth11;
103      public BuildWallSouth212 buildsouth12;
104      public PickUpArnes pickUpArnes;
105      public ActivateTextWall textWall;
106      public TextBlocks textBlocks;
107      public EndingInfo endInfo;
108
109
110
111      public override void OnStartLocalPlayer() /*Here we use the function
           that will enable
112          all the scripts and components declared before when we are the local
               player*/
113      {
114          UI.gameObject.SetActive(true); /*Enable he UI for only my player,
               this way I will see
115          only my points, my life, etc*/
116          fpsController.enabled = true; //REALLY IMPORTANT. Enable the
               controls only for my player.
117          fpsCamera.enabled = true; /*REALLY IMPORTANT. Enable the camera only
                for my player
118          (otherwise we would see through the "eyes" of other players who are
               not us.*/
119          audioListener.enabled = true; //REALLY IMPORTANT. This way we will
               listen only through our "ears".
120          liniaDeVida.enabled = true;
121          helmetScript.enabled = true;
122          fallDamage2.enabled = true;
123          player_net.enabled = true;
124          earmuffs.enabled = true;
125          firstAid.enabled = true;
126          donaldTrump.enabled = true; //All this trump stuff is because these
               scripts build walls. Humor programming at 4am I guess...
127          donaldTrump2.enabled = true;
128          donaldTrump3.enabled = true;
129          donaldTrump4.enabled = true;
130          donaldTrump5.enabled = true;
```

```
131         donaldTrump6.enabled = true;
132         donaldTrump7.enabled = true;
133         donaldTrump8.enabled = true;
134         donaldTrump9.enabled = true;
135         donaldTrump10.enabled = true;
136         donaldTrump11.enabled = true;
137         donaldTrump12.enabled = true;
138         block.enabled = true;
139         blockRoof.enabled = true;
140         build1.enabled = true;
141         build12.enabled = true;
142         build13.enabled = true;
143         build14.enabled = true;
144         build15.enabled = true;
145         build16.enabled = true;
146         build17.enabled = true;
147         build18.enabled = true;
148         build19.enabled = true;
149         build10.enabled = true;
150         build11.enabled = true;
151         build112.enabled = true;
152         build113.enabled = true;
153         build114.enabled = true;
154         build115.enabled = true;
155         build116.enabled = true;
156         build117.enabled = true;
157         build221.enabled = true;
158         build222.enabled = true;
159         build223.enabled = true;
160         build224.enabled = true;
161         build225.enabled = true;
162         build226.enabled = true;
163         build227.enabled = true;
164         build228.enabled = true;
165         build229.enabled = true;
166         build2210.enabled = true;
167         build2211.enabled = true;
168         build2212.enabled = true;
169         build2213.enabled = true;
170         build2214.enabled = true;
171         build2215.enabled = true;
172         build2216.enabled = true;
173         build2217.enabled = true;
174         buildsouth1.enabled = true;
175         buildsouth2.enabled = true;
176         buildsouth3.enabled = true;
177         buildsouth4.enabled = true;
178         buildsouth5.enabled = true;
179         buildsouth6.enabled = true;
180         buildsouth7.enabled = true;
181         buildsouth8.enabled = true;
182         buildsouth9.enabled = true;
183         buildsouth10.enabled = true;
```

```
184         buildsouth11.enabled = true;
185         buildsouth12.enabled = true;
186         pickUpArnes.enabled = true;
187         textWall.enabled = true;
188         textBlocks.enabled = true;
189         endInfo.enabled = true;
190
191
192         gameObject.name = "LOCAL Player"; //Change the name of the Local
              Player in the inspector to "LOCAL Player" for easier recognition.
193         //Not really necessary but useful.
194         base.OnStartLocalPlayer(); //End the function here.
195     }
196
197     void Start()
198     {
199
200     }
201
202
203 }
204
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI; //Type this every time you'll be working with UI's in
     your script.
6
7  /*THIS SCRIPT WILL CONTROL MOST OF OUR ANIMATIONS, INCLUDING THE DEATH
     ANIMATION AND SOME CAMERA
8   EFFECT FOR IT.*/
9
10 public class Player_Net : NetworkBehaviour
11 {
12     NetworkAnimator anim; //Declare a variable "anim" that will be of the
         class "NetworkAnimator".
13     //With this we'll control our animations! (some of them).
14
15     GameObject background; //Declare several GameObject type variables.
         They'll work as UI's.
16     GameObject helmetOn;
17     GameObject arnesOn;
18     GameObject arnesOff;
19
20     public GameObject player; //Public GameObject that we'll have to drag in
          the inspector. It'll be the player prefab.
21     public CharacterController controller; //Declare our CharacterController
          -gives the player a collider, velocity, etc-.
22     AudioSource audio_; //Variable audio_ that'll be an AudioSource.
23     public UnityStandardAssets.Characters.FirstPerson.FirstPersonController
         fpsController;//Again the fpsController...
24     public Camera fpsCamera;//And our Camera. We'll use it to make a cool
         effect when we die.
25
26     [SyncVar] bool dead;/*Boolean variables are "true or false" variables.
         So... it'll be dead=true or dead= false.
27     The [syncVar] thing is to sync certain variables from the server to the
         clients. Obiously this script will be run
28     on a client so that doesn't make much sense here... I couldn't have time
          to get more into syncvars, sadly.
29     Note that although we don't specify it, since it is not public it won't
         be seen in the inspector.*/
30
31     float rotation; /*Float variables are basically numbers with decimals.
         Remember that every number with decimal needs
32     to be ended with an "f", just like 1.5f as an example.*/
33     float temps;
34
35
36
37     // Use this for initialization. Start function runs when the game starts
          and only then.
38     void Start()
39     {
```

```
40          anim = GetComponent<NetworkAnimator>(); /*Define where does this
              anim varible comes form.
41          The GetComponent helps us find components in gameObjects. In this
              case anim will be the NetworkAnimator
42          component in the gameObject this script is attatched to. Similar
              with the next ones, the "player." here is redundant sometimes*/
43          player.GetComponent<Health2>();
44          controller = player.GetComponent<CharacterController>();
45          audio_ = player.GetComponent<AudioSource>();
46          fpsCamera.GetComponent<Transform>();
47
48          background = GameObject.FindGameObjectWithTag("Background");/*A good
               way to find GameObjects in the Network is with
49          GameObject.FindGameObjectWithTag("[insertTag]"). It is a "slow" way
              to define objects, but certainly confortable.
50          Remember to put the tag in the gameobject too.*/
51          helmetOn = GameObject.FindGameObjectWithTag("CascOn");
52          arnesOn = GameObject.FindGameObjectWithTag("Text");
53          arnesOff = GameObject.FindGameObjectWithTag("Text1");

54      }
55
56
57
58      // Update is called once per frame.
59      void Update()
60      {
61          if (Input.GetKey(KeyCode.LeftShift)) //In case we press the left
              Shift
62          {
63              if (Input.GetKey(KeyCode.W)) //If we also press the W (in the
                  First Person Controller that is making us run forward)
64              {
65                  anim.animator.SetBool("isWalking", false); /*Get the anim
                      variable, access its animator and set a Boolean value.
66                  In the animator we will have previously created those tags
                      as booleans and made the connections between animations.*/
67                  anim.animator.SetBool("isRunning", true);
68                  anim.animator.SetBool("isIdle", false);
69                  anim.animator.SetBool("isWBackwards", false);
70                  anim.animator.SetBool("isRBackwards", false);
71
72              }
73
74              //Same for S.
75              else if (Input.GetKey(KeyCode.S))
76              {
77                  anim.animator.SetBool("isWalking", false);
78                  anim.animator.SetBool("isRunning", false);
79                  anim.animator.SetBool("isIdle", false);
80                  anim.animator.SetBool("isWBackwards", false);

81                  anim.animator.SetBool("isRBackwards", true);
```

```
82
83              }
84
85              //If we're not pressing W nor S, the animation played will be
                  the Idle animation.
86              else
87              {
88                  anim.animator.SetBool("isWalking", false);
89                  anim.animator.SetBool("isRunning", false);
90                  anim.animator.SetBool("isIdle", true);
91                  anim.animator.SetBool("isWBackwards", false);

92                  anim.animator.SetBool("isRBackwards", false);
93
94              }
95          }
96
97          else if (Input.GetKey(KeyCode.W))
98          {
99              //Set speed to walking speed.
100             //Play the animation of walking through the "isWalking" string.
101             anim.animator.SetBool("isWalking", true);
102             anim.animator.SetBool("isRunning", false);
103             anim.animator.SetBool("isIdle", false);
104             anim.animator.SetBool("isWBackwards", false);
105             anim.animator.SetBool("isRBackwards", false);
106         }
107
108         //Same for S.
109         else if (Input.GetKey(KeyCode.S))
110         {
111             anim.animator.SetBool("isWalking", false);
112             anim.animator.SetBool("isRunning", false);
113             anim.animator.SetBool("isIdle", false);
114             anim.animator.SetBool("isWBackwards", true);
115             anim.animator.SetBool("isRBackwards", false);
116         }
117
118
119
120         //If we don't press any key, switch the state to the Idle animation.

121         else
122         {
123             anim.animator.SetBool("isWalking", false);
124             anim.animator.SetBool("isRunning", false);
125             anim.animator.SetBool("isIdle", true);
126             anim.animator.SetBool("isWBackwards", false);
127             anim.animator.SetBool("isRBackwards", false);

128         }
129
130         if (dead) //The dead boolean will be set true when the function
```

```
            below -see Die()- sets it true;
131         {
132             rotation += 5 * Time.deltaTime; /*+= adds to the variable
                  rotation 5*Time.deltaTime every frame that "dead" is true,
133             so rotation gets bigger and bigger. Time.deltaTime is the time
                  that passes from frame to frame updates.*/
134             if (rotation >= 360f)
135                 rotation -= 360f; /*if rotation gets equal or bigger than
                      360 degrees, substract 360 from it. See the difference
136             between doing that and setting rotation to 0, because if you set
                     the roatation to 0 when it eas 360.45, it will cause
137             the camera to snap, instead of following the swift rotation!*/
138
139             temps += Time.deltaTime; /*Add Time.deltaTime to our variable
                  temps.*/
140
141             //Rotate the camera above the dead player.
142             fpsCamera.transform.position = player.transform.position + new
                  Vector3(0.5f, 2 + 0.1f*temps, -0.5f);
143             /*Take the fpsCamera gameObject that is our Camera, then its
                  transform and inside its transform get its position.
144              Make the camera's position equal to the players position + a
                    certain vector to center it in aproximately the
145              middle of its dead lying body and that slowly goes up*/
146             fpsCamera.transform.rotation = Quaternion.Euler(90f, rotation,
                  0f); /*Rotate the camera downwards (x=90) and make it
147             spin around the y axis with the increasing rotation degree
                  value.*/
148
149
150
151         }
152     }
153
154     public void Die() /*Function die. Another REALLY IMPORTANT thing about
          public variables and functions is that they can be accessed
155         from other scripts! Specifically this one will be accessed from the
              Health2 script.*/
156     {
157         dead = true; //Set the dead bool to true.
158         anim.SetTrigger("Died"); /*The death animation is an animation that
              is played once something happens -press a key, get out of life,
159         whatever-. That means it is a trigger, and the propper way to tell
              the animator to run triggers is anim.SetTrigger("[insertTag]").*/
160
161         audio_.enabled = false; //We cannot make sounds with our player. It
              is, indeed, dead.
162         fpsController.enabled = false; //Also we can't move obiously.
163
164         //Disable any of these UI that may have been active at that moment.
165         background.GetComponent<Image>().enabled = false;
166         helmetOn.GetComponent<Text>().enabled = false;
167         arnesOn.GetComponent<Text>().enabled = false;
```

```
168         arnesOff.GetComponent<Text>().enabled = false;
169         CmdDead();/*REALLY IMPORTANT!!! So, the way information goes form
              client to client is the following. The player does something,
170         and then it sends a Command to the Server saying "Hey Server! I've
              done something!". Then if the command asks for it, the server
171         sends whatever update that the command asks for to the rest of the
              clients.
172         Example: I want to click on a helmet and make it disappear. If I
              script it but I dont make a command, I will see the helmet
              disappear,
173         because the script will have been run localy, but all the other
              clients won't see the helmet disappear, because the server won't
              have told
174         them to make it disappear!
175
176          All commands have to start to the prefix Cmd, as all the
               instructions from Server to Client (ClientRpc's) have to start
               with Rpc.
177
178          So basically writing CmdDead() we said: "Hey Server! I did the
               command!"*/
179     }
180
181     [Command]
182     void CmdDead() //And what is the thing the command wants to do?
183     {
184         RpcDead(); //It wants the server to run the ClientRpc function
              (RpcDead) to all the clients!
185     }
186
187
188
189     [ClientRpc]
190     void RpcDead() //And what is the RpcDead doing? Well...
191     {
192         gameObject.tag = "Dead"; /*The gameObject that died changes his tag
              to "Dead". Remember to set it before. All the copies of
193         that certain object will be updated on all the clients. Wohoo!*/
194         controller.height = 0.2f;
195         controller.radius = 0.15f;
196         controller.center = new Vector3(0.2f, 0.14f, -0.9f);
197         controller.transform.position = player.transform.position + new
              Vector3(0.5f, 0, -0.5f);
198         //Also change a bunch of properties of the controller collider of
              that gameObject such as height, radius, center and position.
199     }
200
201 }
202
203
204
205
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  /*THIS SCRIPT WILL ALLOW OUR PLAYER TO CONNECT TO THE LIFE LINE. THE SCRIPT
     INVOLVES IMPORTANT INPUTS
8   ON RAYCAST AND COMMANDS/RPC's.*/
9
10 /*DISCLAIMER: In this script there is a bit of redundancy due to last minute
     changes. What this script does is to
11  let you connect to the life line if you have equiped the harness
     previously. The harness is equiped through the
12  script "PickUpArnes", so when here we declare the boolean variable "arnes"
     it should really say "lifeLine". Sorry bro.*/
13 public class LiniaDeVida : NetworkBehaviour {
14
15     public Transform cameraTransform; /*We'll set a Transform variable
         called cameraTransform, and it'll be public,
16     so we'll have to set the gameObject whose transform we're talking about
         manually in the inspector.
17     SPOILER ALERT: It will be our camera's transform. Yeah, the one that is
         only enabled for us. It will
18     shoot a ray forward at every frame. If this ray collides with something
         we will gain a really useful information
19     that will allow us to interact with it!*/
20
21     GameObject arnesStatus; //Some private gameObjects that will act as UIs.
22     GameObject arnesStatus1;
23     GameObject myText2;
24
25
26     GameObject[] Wall; //An ARRAY of GameObjects tha will be called Wall. To
         declare an array we have to write "[]" after the class.
27     BoxCollider[] boxColliders; //Same for out BoxCollider class.
28
29     bool arnes;
30     float waitTime = 3f;
31     float timer;
32
33     // Use this for initialization
34     void Start () {
35
36         arnesStatus = GameObject.FindGameObjectWithTag("Text"); //What is
             arnes? It is the GameObject with the tag "Text".
37         arnesStatus1 = GameObject.FindGameObjectWithTag("Text1"); //You get
             the point right?
38         myText2 = GameObject.FindGameObjectWithTag("Text2"); //Same here.
39
40     }
41
42     // Update is called once per frame
```

```
43      void Update () {
44
45          RaycastHit hit; /*REALLY IMPORTANT! We declare a variable "hit" of
              the RaycastHit class. This is the ray I was talking
46          about before.*/
47          Vector3 rayPos = cameraTransform.position + cameraTransform.forward
              * 1f; /*Vector 3 class variablt called rayPos
48          that will tell us from where the ray starts (the camera position + 1
               unit forward).*/
49          Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
              0.5f);/*This I used for debugging purposes, like seeing where
50          the ray goes, what hits, how far... I make it start from rayPos, go
              forward from the camera position 5 units, paint it green
51          and make it last 0.5 seconds. Remember this is NOT the real ray,
              just a represantation of what it may be.*/
52
53          if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f)) /
              *HERE we shoot the ray. Enter the physics class, and send a
54              Raycast starting from rayPos, going forward, saving the
                  information of any collisions in the variable hit, and go for
                  5 units.*/
55          {
56              if (hit.transform.tag == "LiniaDeVida")/*If the raycast hits the
                   transform of a gameobject (it really hits its collider, but
                  the info
57                  that the hit variable stores is its transform), and its tag
                      is "LiniaDeVida"...*/
58              {
59                  PickUpArnes script = gameObject.GetComponent<PickUpArnes>
                      ();/*...MAGIC! We declare a class that is PickUpArnes,
                      under the
60                  variable script. And what is that script of the class
                      PickUpArnes? Well its the component called PickUpArnes of
                      the gameobject
61                  that this script is attatched to. It could be another
                      gameObject, but since we haven't specified it by default
                      is the gameObject
62                  this script is attatched to. And then we can access all its
                      public variables and functions, and more specifically...*/
63                  if (script.hasArnes) //If the variable "hasArnes" of script
                      (remember that it is just the name for PickUpArnes
                      script)...
64                  {
65                      if (!arnes)//...if the boolean variable "arnes" that we
                      had declared before is false (so if we're not connected
                      to the life line)...
66                      {
67                          myText2.GetComponent<Text>().enabled = true; //Go to
                       myText gameObject, get its component Text and set it
                      true.
68                          //In the game this text is giving us the option to
                      pick up the harness.
69                      }
```

```
70                      if (Input.GetMouseButtonDown(0)) //If we press the left mouse button...
71                      {
72                          arnesStatus.GetComponent<Text>().enabled = true; //Again, set some text active that says that we have the harness and stuff.
73                          arnesStatus1.GetComponent<Text>().enabled = false;
74                          myText2.GetComponent<Text>().enabled = false;
75                          CmdWall(); //Tell the Server "Hey Server! I did something!"
76                          arnes = true; //Set our boolen arnes to true, so we won't be able to connect ourselves to the life line once we're already connected
77                      }
78                  }
79              }
80
81              else
82              {
83                  myText2.GetComponent<Text>().enabled = false;
84                  //If our ray doesn't hit anything tagged as "LiniaDeVida" then don't show the text option to connect to it.
85              }
86
87          }
88
89          if (Input.GetMouseButtonDown(1) && arnes)  /*If we're connected to the life line and we press the right mouse button then...*/
90          {
91              arnes = false; /*Our boolean value says that we're not connected to the life line anymore. Obviously the fact of changing
92              a boolean value doesn't make us connect or disconnect, this is made with the command set before.*/
93              arnesStatus.GetComponent<Text>().enabled = false; //Text saying we're connected is disabled.
94              arnesStatus1.GetComponent<Text>().enabled = true; //Text saying we're disconnected it enabled.
95              timer = 0; //Set our float variable timer to 0. We'll use it to make the arnesStatus1 text disappear after a few seconds.
96
97              CmdOffWall(); //More "Notice me senpai" server stuff...
98          }
99
100         if (!arnes) //If indeed we're not connected to the life line anymore...
101         {
102             timer += Time.deltaTime; //Add Time.deltaTime to the timer variable at every frame.
103             if (timer > waitTime) //If timer is larger than waitTime, that we set at the beggining of the script as 3 seconds.
104             {
105                 arnesStatus1.GetComponent<Text>().enabled = false; //Make disappear the text saying that we're not connected
```

```
                    anymore.
106             }
107         }
108     }
109
110
111
112     [Command]
113     void CmdWall() //What does this command want the Server to do?
114     {
115         RpcCollide(); //Run the RpcCollide.
116     }
117
118     [Command]
119     void CmdOffWall()
120     {
121         RpcOffCollide();
122     }
123
124     [ClientRpc]
125     void RpcCollide() //What does the RpcCollide do?
126     {
127         if (isLocalPlayer) /*REALLY IMPORTANT! The Server sends the
              RpcCollide to all the clients and says "Hey Clients, the player in

128             control here is the player who sent the command? If it is
                then...*/
129         {
130             Wall = GameObject.FindGameObjectsWithTag("Wall"); /*Our array of
                   gameObjects "Wall" that we declared at the begging will be
131             all the gameobjects tagged as wall. Note how there is an "s" at
                FindGameObjectSWithTag.*/
132             boxColliders = new BoxCollider[Wall.Length]; /*The BoxCollider
                array called as boxColliders that we declared before will
133             be an array of BoxColliders with the same lenght as the array of
                 walls.*/
134             for(int i=0; i < Wall.Length; i++) /*So for an integer i=0,
                while i < to the lenght of the Wall array, run the "for"
                function and
135                 add +1 to "i".*/
136             {
137                 boxColliders[i] = Wall[i].GetComponent<BoxCollider>(); /*The
                     boxCollider in the position "i" of the array will be
                    equal
138                 to the BoxCollider component attatched to the Wall in the
                    position [i] of its array.*/
139                 boxColliders[i].enabled = true; //Enable that collider, that
                     originally was disabled.
140                 /*What this Rpc does is to enable the colliders of some
                    invisible walls put at the edges of the building, so when
141                  you connect to the life line you can't fall. The fact that
                    we use the "if(isLocalPlayer) condition is to make the
                    box colliders
```

```
142                 of the wall active ONLY in the player thats connected to
                      the life line network copy of the walls. So basically he
                      won't be able
143                 to fall, whereas all the copies of the walls' box colliders
                       won't be active for the other players and they'll be
                      able to fall,
144                 if they're not connected to the life line. Pretty cool,
                      uh? ;)*/
145             }
146         }
147     }
148
149     [ClientRpc]
150     void RpcOffCollide() //Reverse process of the RpcCollide for when you
          disconnect from the life line.
151     {
152         if (isLocalPlayer)
153         {
154             Wall = GameObject.FindGameObjectsWithTag("Wall");
155             boxColliders = new BoxCollider[Wall.Length];
156             for (int i = 0; i < Wall.Length; i++)
157             {
158                 boxColliders[i] = Wall[i].GetComponent<BoxCollider>();
159                 boxColliders[i].enabled = false;
160             }
161         }
162     }
163
164
165 }
166
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  /*THIS SCRIPT WILL ALLOW US TO PICK UP A HELMET AND PUT IT ON OUR HEAD FOR
8   OTHER PLAYERS TO SEE. ALSO SETS THE TRIGGER FOR THE CPR ANIMATION. AGAIN
9   WE'LL WORK WITH RAYCASTS AND COMMANDS/RPC's.*/
10
11 public class Helmet : NetworkBehaviour {
12
13     public Transform cameraTransform; //Again the public transform of our
         camera.
14
15     GameObject helmetFloor; //Declare several gameObjects that we'll define
         later.
16     GameObject cascOn;
17     GameObject myText;
18     GameObject myText2;
19
20     public GameObject cPRText; //Declare several gameObjects that will be
         set in the inspector.
21     public GameObject helmetHead;
22
23     public bool hasHelmet; //Public bool variable hasHelmet.
24
25     NetworkAnimator anim; //Again we'll be working with the Network Animator
          class that we'll call anim.
26
27     // Use this for initialization
28     void Start () {
29
30         cascOn = GameObject.FindGameObjectWithTag("CascOn");//Define some of
             the GameObjects through tags.
31         myText = GameObject.FindGameObjectWithTag("Text3");
32         myText2 = GameObject.FindGameObjectWithTag("Text2");
33
34         anim = GetComponent<NetworkAnimator>();//Define the anim variable.
35
36     }
37
38     // Update is called once per frame
39     void Update () {
40         /*Again a raycast with a similar pattern as the life line script.*/
41         RaycastHit hit;
42         Vector3 rayPos = cameraTransform.position + cameraTransform.forward
             * 1f;
43         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
44
45         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 2f))
46         {
```

```
47              if (hit.transform.tag == "Helmet")
48              {
49                  if (!hasHelmet)
50                  {
51                      myText.GetComponent<Text>().enabled = true;
52                  }
53
54                  helmetFloor = hit.transform.gameObject; /*Remember how we
                      didn't define our gameObject variable helmetFloor?
55                  This is because we can make this variable be applied to
                      different gameObjects, as long as they are tagged as
                      "Helmet",
56                  by saying: "What will helmetFloor be? It will be the
                      gameObject whose transform has been hit by the raycast".*/
57                  if (Input.GetMouseButtonDown(0))
58                  {
59                      myText.GetComponent<Text>().enabled = false;
60                      cascOn.GetComponent<Text>().enabled = true;
61                      CmdHelmet(helmetFloor, hasHelmet); //"Hey Server! Make
                        this stuff... Careful that now I give you some inputs!"
62                      if (!hasHelmet)
63                      {
64                          PointSystem point =
                        gameObject.GetComponent<PointSystem>(); /*Access the
                        PointSystem script that we'll call point,
65                          located in this gameObject as a component named
                        PointSystem*/
66                          point.Points(10);/*Inside the point script there is
                        a public function called Points, that accepts integer
                        numbers.
67                          Sum 10 to that variable*/
68                      }
69                      hasHelmet = true; //We have a helmet already, so next
                        time we hit a helmet with our raycast we won't be able to
                         pick it up.
70                  }
71              }
72
73
74
75              else if (hit.transform.tag == "Dead") //If the raycast doesn't
                  hit a Helmet but hits a player that died (therefore tagged as
                  dead):
76              {
77                  Debug.Log("PrepareCPR"); //In the console window say
                      "PrepareCpr". Again not important, just for debuggin
                      reasons.
78                  cPRText.GetComponent<Text>().enabled = true; //Enable a text
                       allowing to make the CPR.
79                  if (Input.GetMouseButtonDown(0))
80                  {
81                      EndingInfo script = gameObject.GetComponent<EndingInfo>
                        (); //Access the EndingInfo script attatched to this
```

```
                    gameObject.
82                   script.EndGame(); //Run the EndingInfo script EndGame()
                    function.
83                   anim.SetTrigger("CPR"); //Set the trigger for the
                    animator to make the CPR animation.
84              }
85
86          }
87
88
89
90          else
91          {
92              myText.GetComponent<Text>().enabled = false;
93              cPRText.GetComponent<Text>().enabled = false;
94          }
95
96
97      }
98
99  }
100
101 [Command]
102 void CmdHelmet(GameObject helmetFloor, bool hasHelmet) /*Okay so in this
       Command we send some info to the Server, more specifically
103     which gameObject is the helmet that we have hit with our raycast and
           if we have a helmet already on or no.*/
104 {
105     RpcHelmet(helmetFloor, hasHelmet); //The Rpc function will use those
           parameters.
106 }
107
108 [ClientRpc]
109 void RpcHelmet(GameObject helmetFloor, bool hasHelmet)/*When we declare
      the inputs that can go inside a function, we have to declare which
110     class they belong to).*/
111 {
112     if (!hasHelmet) //If the player that sent the command doesn't have a
           helmet:
113     {
114         helmetFloor.SetActive(false); //Make the helmet that we hit with
               the floor disappear (SetActive to false).
115         helmetHead.GetComponent<Renderer>().enabled = true; //Set the
              renderers of the "invisible" helmet that the player wore on
              his head true;
116     }
117     /*Note the huge difference between this Rpc and the one in the life
          line script. Here what the server does is say to all the clients:
118      "Hey Clients. If the your copy of the player that sent the command
          doesn't have a helmet, make the renderers of his helmet active
119      so everyone can see how safe and fabulous he looks in it!"
120
121      So in this script all copies of the object that we are being
```

```
            refered to are affected by the Rcp, whereas in the Life Line
            script
122         the only copies of the walls that got its colliders activated where
             the ones of the player that sent the Command, thanks to the
123         "(isLocalPlayer)" condition.*/
124     }
125
126
127 }
128
129
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.Networking;
6
7  /*THIS SCRIPT WILL CONTROL THE HEALTH OF OUR PLAYER.*/
8
9  public class Health2 : NetworkBehaviour {
10
11     public const int maxHealth = 100;/*Maximum health at the start.*/
12     public float currentHealth = maxHealth; //At the start the current health
          is equal the to the max health.
13     public bool dead;
14
15     Player_Net player; //Referencing the Player_Net script and calling it
         "player".
16     public RectTransform healthbar2; //RectTransform is a class that will
         allow us to easily change the size of our health bar.
17     //It is a public RectTransform that we will have to drag into the
         inspector.
18
19     void Start()
20     {
21         player = gameObject.GetComponent<Player_Net>(); /*We tell the script
             to find the Player_Net script named as "player" that
22         we had declared before. Where will the Health2 script find it? In
             this gameObject, in the component called Player_Net.*/
23     }
24     public void TakeDamage(float amount) /*We create a function called take
         damage. Note certain things:
25         - It is public. This script will run all the health updates, so it
             will be accessed by all the scripts that control the different
             damages.
26         - It is not in Update(). This function will run only when other
             scripts tell it to run.
27         - It has an input, that will be a float number, and that we'll call
             "amount" to work with it in the function.*/
28     {
29         currentHealth -= amount; //Everytime this function runs, substract
             the number corresponding to amount from currentHealth;
30         int roundAmount = Mathf.RoundToInt(amount); //Create the integer
             variable roundAmount and make it be the amount variable rounded.
31         PointSystem script = gameObject.GetComponent<PointSystem>(); //Access
              the PointSystem script attatched to this same gameObject.
32         script.Points(-1*roundAmount);//IMPORTANT. From the PointSystem
             script, access its function Points and pass -1*roundAmount.
33         //We just accessed a function from another script and passed a value
             to it!
34
35         if (currentHealth <= 0) //If the variable currentHealth goes below
             0...
36         {
```

```
37              currentHealth = 0; //Make it 0. We don't want negative health values, do we?
38              dead = true; //Set the boolean value "dead" to true;
39              Debug.Log("Dead");
40              player.Die(); //Run the function "Die()" located in the Player_Net script!
41              EndingInfo script2 = gameObject.GetComponent<EndingInfo>(); /*Also declare another script "EndingInfo" and run its function EndGame().*/
42              script2.EndGame();
43          }
44
45          if (currentHealth > 100) //If health is larger than 100...
46          {
47              currentHealth = 100; //Make it 100. This could happen for example if we lost some health and we picked up the FirstAidKit.
48          }
49
50          healthbar2.sizeDelta = new Vector2(currentHealth * 2, healthbar2.sizeDelta.y); /*Access the RectTransform healthBar, its property
51          sizeDelta and make its x value equal to currentHealth*2 (that's only because the healthBar has a length of 200 in the inspector.
52          Its y component remains the same.*/
53      }
54  }
55
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //THIS SCRIPT WILL CONTROL THE DAMAGE TAKEN AFTER FALLS TO DIFFERENT LEVEL.
     NOTE THAT IT IS MONOBEHAVIOUR
6  //SINCE IT DOESN'T HAVE ANYTHING TO DO WITH NETWORKING.
7
8  public class FallDamage2 : MonoBehaviour {
9
10     public float startYPos;
11     public float endYPos;
12     public float damageThershold = 2f;
13     public bool damageMe = false;
14     public bool firstCall = true;
15     public CharacterController player; //Declare the CharacterController
         property. It is public because we'll drag it in the inspector.
16     //It is really important because the CC has the variable "isGrounded"
         that returns if the player is grounded or not.
17
18     public AudioClip[] clips; //Define a public array of audio clips
         called... clips. Not very original, I know.
19
20
21     // Update is called once per frame
22     void Update () {
23         if (!player.isGrounded) //If the CC (that we called "player") says
             that the player is not gorunded...
24         {
25             if (gameObject.transform.position.y > startYPos) //If the y
                 position of our player is higher than startYPos...
26             {
27                 firstCall = true;
28             }
29             if (firstCall)
30             {
31                 startYPos = gameObject.transform.position.y; //Make the
                     startYPos equal to our player y position.
32                 firstCall = false; //Then we won't be able to update this
                     startYPos again until our position is higher than
                     startYPos.
33                 damageMe = true; //Since we're not grounded, we can take
                     damage!
34             }
35         }
36
37         if (player.isGrounded) //So if our player is hitting the ground...
38         {
39             endYPos = gameObject.transform.position.y; //endYPos is equal to
                 our gameObject position.
40             if (startYPos - endYPos > damageThershold) //If the difference
                 between our start posiiton and our end position is higher than
41                 //our damage threshold (distance of fall that won't make us
```

```
                    take damage)
42              {
43                  if (damageMe) //If damage me is true...
44                  {
45                      Health2 health = gameObject.GetComponent<Health2>(); //
                        Access the Health2 script attatched to this same game
                        object.
46                      if (health != null) //If the script is enabled (there's
                        different ways to say this, this is one of them...)
47                      {
48                          PlaySound(); //Run the function PlaySound() stated
                        below.
49                          health.TakeDamage(20 * (startYPos - endYPos -
                        damageThershold)); //Send a value to the TakeDamage
                        function in the Health2
50                          //script, equal to 20 times the difference between
                        the startYPos, and endYPos and damageThreshold.
51                      }
52                      damageMe = false; //DamageMe is false, we cannot take
                        more damage in this frame.
53                      firstCall = true; //We can update again our startYPos.
54                  }
55
56              }
57          }
58      }
59
60      private void OnControllerColliderHit(ControllerColliderHit hit) /*This
          function was written because I was having issues going down stairs.
61          Since the player collider was bigger than the width of the steps, the
               CC didn't count it as isGrounded, because the bottom of the
            collider
62          wasn't colliding with any horizontal surface, and therfore when I got
               to the ground, the script thought that I had jumped, and I took
63          a damage that had no sense and it was stupid and frustrating. The
            OnControllerColliderHit function checks if your CC hits something
64          every frame. In this case I told it that if it hit a collider that in
               this case was tagged as "Stair", it set the startYPos to 0. This
65          way when the player was going down the stairs its startYPos was
            always 0, and as soon as it hit the ground or a floor, the
            startYPos got
66          updated to its real  value again. This could be a problem in case we
            fell from a stair to a huge gap, but in our experience this is not
67          a problem.*/
68      {
69          if (hit.collider.tag == "Stair")
70          {
71              startYPos = 0;
72          }
73      }
74
75      void PlaySound() //We'll play some cool clips to make sure that the
          person playing knows he has taken damage.
```

```
76      {
77          int randomClip = Random.Range(0, clips.Length); //randomClip will be
              an integer number chosen randomly from 0 the the length of the
              clips array.
78          AudioSource source = gameObject.AddComponent<AudioSource>(); //We
              define an audio source called "source" that will be Added to our
              gameObject.
79          source.clip = clips[randomClip]; //The clip played by the source will
               be defined as the clip that corresponds to the index "randomClip"
              of the "clips" array.
80          source.Play(); //Play the source!
81          Destroy(source, clips[randomClip].length); //Destroy the source
              (because after all we don't want a thousand sources in our
              inspector)
82          //after a certain amount of time. What time you say? The amount of
              time will be the length of the clip equal to the index randomClip
              of the clips Array.
83      }
84
85
86  }
87
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  /*THIS SCTIPT WILL CONTROL THE PICK UP OF THE EARMUFFS AND IT WORKS SIMILARLY
       TO THE HELMET SCRIPT.
8   THERE IS AN EXTRA FEATURE THAT IS THE TURN DOWN OF THE VOLUME WHEN WE PUT
       THE EARMUFFS ON.*/
9
10 public class Earmuffs : NetworkBehaviour {
11     public Transform cameraTransform;//Public camera transform for the
         raycast.
12     GameObject earmuffsFloor; //Some GameObjects that will act as text.
13     GameObject earmuffsText;
14     GameObject earmuffsOn;
15     public GameObject earmuffsHead; //The invisible earmuffs that we have on
         the player's head and that we will drag in the inspector.
16
17     bool hasEarmuffs; //Boolean value "hasEarmuffs".
18
19     // Use this for initialization
20     void Start () {
21         earmuffsText = GameObject.FindGameObjectWithTag("EarmuffsText"); //
             Define the GameObjects.
22         earmuffsOn = GameObject.FindGameObjectWithTag("EarmuffsOn");
23     }
24
25     // Update is called once per frame
26     void Update () {
27
28         RaycastHit hit;
29         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
               1f;
30         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
31
32         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f)) //
             Shoot the ray.
33         {
34             if (hit.transform.tag == "EarmuffsFloor") //If collides with the
                 gameObject tagged as "EarmuffsFloor"...
35             {
36                 if (!hasEarmuffs) //If we don't have the earmuffs on yet...
37                 {
38                     earmuffsText.GetComponent<Text>().enabled = true; //Show
                       the text option to pick up the earmuffs.
39                 }
40
41                 earmuffsFloor = hit.transform.gameObject; //Make the
                   earmuffsFloor gameObject be the object which the ray has
                   hit.
```

```
42                  if (Input.GetMouseButtonDown(0)) //If we press the left mouse
                        button...
43                  {
44                      earmuffsText.GetComponent<Text>().enabled = false; //
                        Disable the text option to pick them up.
45                      earmuffsOn.GetComponent<Text>().enabled = true; //Show
                        the text that says "Earmuffs equiped".
46                      CmdEarmuffs(earmuffsFloor, hasEarmuffs); //Command to
                        make the server update the new state of the earmuffs to
                        everyone.
47                      if (!hasEarmuffs) //If we don't have earmuffs... (we
                        actually have them, but the script doesn't know because
                        we havent told it).
48                      {
49                          PointSystem point =
                        gameObject.GetComponent<PointSystem>();
50                          point.Points(10); //Access the PointSystem script and
                         pass a value of 10 to its function "Points()".
51                      }
52                      hasEarmuffs = true; //NOW the script knows we have the
                        earmuffs!
53                  }
54              }
55
56              else //If our raycast is not hitting an object tagged as
                  "EarmuffsFloor" don't show the option to pick the earmuffs up.
57              {
58                  earmuffsText.GetComponent<Text>().enabled = false;
59              }
60          }
61
62      }
63
64      [Command]
65      void CmdEarmuffs(GameObject earmuffsFloor, bool hasEarmuffs) //Hey
          server! Take the GameObject earmuffsFloor and the bool hasEarmufs and
          do an Rpc!
66      {
67          RpcEarmuffs(earmuffsFloor, hasEarmuffs); //Okay Client, and what
              should I do?
68      }
69
70      [ClientRpc]
71      void RpcEarmuffs(GameObject earmuffsFloor, bool hasEarmuffs) //Take those
           inputs and...
72      {
73          if (!hasEarmuffs) //If hasEarmuffs is false...
74          {
75              earmuffsFloor.SetActive(false); //Make the earmuffsFloor
                  gameObject (the one that the raycast hit) invisible.
76              earmuffsHead.GetComponent<Renderer>().enabled = true; //Make the
                  invisible earmuffs in our player's head visible.
77          }
```

```
78
79          if (isLocalPlayer)
80          {
81              AudioListener.volume = 0.1f; //If the Command was sent by the
                  LocalPlayer, turn the volume of his Audio Listener down to 0.1.
82              //Those look like a hell of an earmuffs don't they?
83          }
84      }
85  }
86
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Audio;
5
6  //THIS SCRIPT WILL CONTROL THE COLLISION OF OUR PLAYER WITH THE EXCAVATOR,
     AND PLAY SOME AUDIO OF COMPLAINT.
7
8  public class CollisionExcavator : MonoBehaviour {
9
10     public AudioClip clip1; //Public AudioClip called "clip1" that we'll set
         in the inspector.
11     // Use this for initialization
12     void Start () {
13
14     }
15
16     // Update is called once per frame
17     void Update () {
18
19     }
20
21
22
23     private void OnControllerColliderHit(ControllerColliderHit collision) /
         *Again, when our Controller Collider hits another collider...*/
24     {
25         if (collision.gameObject.tag == "Excavadora") //If it hits a
             gameObject tagged as "Excavadora"...
26         {
27             Debug.Log("collided");
28             Health2 health = gameObject.GetComponent<Health2>(); //Access the
                Health2 script attatched to the player.
29             AudioSource source = gameObject.AddComponent<AudioSource>(); //
                Access the audio source attatched to the player.
30             source.clip = clip1; //The clip thay the source will have to play
                is "clip1"
31             source.Play(); //Play the clip!
32             Destroy(source, clip1.length); //Destroy the clip once it is
                finished.
33             if (health != null)
34             {
35                 health.TakeDamage(5); //Pass a value of 5 to the TakeDamage
                    function of the health script. Note that this function is
                    run every
36                 //frame, so if you keep touching the excavator you'll keep
                    passing 5, after 5, after 5, and you'll die pretty quick...
37             }
38         }
39     }
40 }
41
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  //THIS SCRIPT WILL ALLOW US TO PICK UP THE FIRST AID KIT.
8
9  public class FirstAid : NetworkBehaviour {
10
11     public Transform cameraTransform; //Yaaaaaay raycasts
12     GameObject firstAid; //GameObject that will be our FirstAid box.
13     GameObject firstAidText; //Text allowing us to pick it up.
14     // Use this for initialization
15     void Start () {
16
17         firstAidText = GameObject.FindGameObjectWithTag("FirstAidText"); //
             Define the firstAidText GameObject declared before.
18     }
19
20     // Update is called once per frame
21     void Update () {
22
23         RaycastHit hit;
24         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
              1f;
25         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
26
27         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
28         {
29             if (hit.transform.tag == "FirstAid")
30             {
31                 firstAid = hit.transform.gameObject; //The firstAid
                    gameObject will be the raycasted object if it is tagged as
                    "FirstAid"
32                 firstAidText.GetComponent<Text>().enabled = true;
33
34                 if (Input.GetMouseButtonDown(0))
35                 {
36                     Health2 health = gameObject.GetComponent<Health2>();
37                     if (health != null)
38                     {
39                         firstAidText.GetComponent<Text>().enabled = false;
40                         health.TakeDamage(-25); //We pass a negative value,
                       so we're actually recovering life!
41                         CmdFirstAid(firstAid); //Command passing the
                       raycasted gameObject.
42                     }
43                 }
44             }
45
46             else
```

```
47              {
48                  firstAidText.GetComponent<Text>().enabled = false;
49              }
50
51          }
52
53      }
54
55      [Command]
56      void CmdFirstAid(GameObject firstAid)
57      {
58          RpcFirstAid(firstAid);
59      }
60
61      [ClientRpc]
62      void RpcFirstAid(GameObject firstAid)
63      {
64          firstAid.SetActive(false); //The server tells the clients to set
              unactive all the copies of the gameObject "firstAid".
65      }
66  }
67
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  //THIS SCRIPT WILL CONTROL THE TIME IN THE GAME.
7
8  public class Timer : MonoBehaviour {
9
10     public Text gameTimerText; //Again some text.
11
12     public float gameTimer = 901f; //Seconds of game.
13     public int seconds;
14     public int minutes;
15
16
17     // Use this for initialization
18     void Start () {
19
20     }
21
22     // Update is called once per frame
23     void Update () {
24         gameTimer -= Time.deltaTime; //At every frame substract
             Time.deltaTime from our gameTimer.
25
26         seconds = (int)(gameTimer % 60); //Aply these operation to transform
             form gameTimer to seconds and minutes.
27         minutes = (int)(gameTimer / 60) % 60;
28
29         string timerString = string.Format("{0:00}:{1:00}", minutes,
             seconds); //Same process as in the script PointSystem.
30
31         gameTimerText.text = timerString;
32
33         if (gameTimer <= 0) //If the time gets to 0, set the gameTimer to 0
             and run the function EndGame() from the EndingInfo script.
34         {
35             gameTimer = 0;
36             EndingInfo script2 = gameObject.GetComponent<EndingInfo>();
37             script2.EndGame();
38         }
39     }
40 }
41
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  //THIS SCRIPT WILL CONTROL THE POINT SYSTEM.
7
8  public class PointSystem : MonoBehaviour {
9
10     public Text pointText; //Public text that will show us our points on
         sceen.
11     public int points; //Our points.
12
13     // Use this for initialization
14     void Start()
15     {
16
17     }
18
19     public void Points(int pointsExtra) //Function Points that has an integer
          argument called pointsExtra. It is not in Update(), so will only
20         //run when another script tells it to run.
21     {
22         points += pointsExtra; //Add to our current points the pointsExtra
             amount.
23         if (points <= 0)
24         {
25             points = 0; //If points is equal or minor than 0, make it 0;
26         }
27         string pointString = string.Format("{0}pts", points); /*This line
             controls how the variable points in the script actually turns
28         in something we see on our screen. We declare the string called
             "pointString", and we say that it will be a string, in the format
29         of a number {} followed by the "word" pts. The number inside the
             brackets will be the variable points.*/
30         pointText.text = pointString; //Then the text of our gameObject's
             text will be, indeed, the pointString string.
31     }
32 }
33
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.Networking;
6
7  //THIS SCRIPT WILL CONTROL THE PICK UP OF THE BLOCKS TO BUILD THE WALLS
8  //WHEN THEY ARE PILED. ALSO IT WILL CONTROL THE LOAD OF THOSE BLOCKS THAT WE
9  //PICKED UP ONTO THE PLATFORM OF THE CRANE AND WE'LL SET THE CRANE'S MOVENT
     IN MOTION.
10
11 public class PickUpBlock : NetworkBehaviour {
12
13     public Transform cameraTransform; //Camera transform for the raycast.
14     public GameObject myBlocks; //The public gameObject to be dragged in the
          inspector that will be our blocks.
15
16     GameObject myTakeText; //Text allowing us to pick up the blocks.
17
18     Renderer[] blockRenderer; //Array of renderers of the blocks that we'll
         define later.
19     GameObject[] blocks; //Array of GameObjects that will correspond to the
         blocks that we'll define later.
20
21
22     public bool hasBlocks;
23     NetworkAnimator anim; //We'll be using a new animation when we pick the
         blocks, so we declare thr NetworkAnimator component.
24
25     // Use this for initialization
26     void Start () {
27         anim = GetComponent<NetworkAnimator>(); //What is anim? It is the
             Component Network Animtor of the gameObject this script is
             attatched to.
28         myTakeText = GameObject.FindGameObjectWithTag("TakeBlocks"); //We
             define what gameObject "myTakeText" is.
29     }
30
31     // Update is called once per frame
32     void Update () {
33         RaycastHit hit;
34         Vector3 rayPos = cameraTransform.position + cameraTransform.forward
             * 1f;
35         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
36
37
38
39         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit,
             5f)) //Raycast.
40         {
41             if (hit.transform.tag == "Block") //If we hit a collider with
                 the tag Block...
```

```
42              {
43                  myTakeText.GetComponent<Text>().enabled = true; //Show the
                    option to pick blocks up.
44                  Debug.Log("Can pick up blocks!");
45                  if (Input.GetMouseButtonDown(0)) //If we press the left
                    mouse button...
46                  {
47                      PointSystem script =
                        gameObject.GetComponent<PointSystem>();
48                      script.Points(5); //Add 5 points to the Point() function
                        in the PointSystem script attatched to this gameObject.
49                      hasBlocks = true;
50                      anim.animator.SetBool("isMovingBlock", true); //Play the
                        animation related to the "isMovingBlock" bool.
51                      //It'll be updated through the Network thanks to the
                        Network animator.
52                      CmdBlock(); //"Hey Server! Do something!
53                  }
54              }
55
56              else
57              {
58                  myTakeText.GetComponent<Text>().enabled = false; //If our
                    Raycast doesn't hit anything tagged as "Block", don't
                    enable gameObject myTakeText.
59              }
60          }
61
62          if (Physics.Raycast(rayPos, cameraTransform.forward, out hit,
              5f)) //If the Raycast hits a collider tagged as "Box"...
63          {
64              if (hit.transform.tag == "Box")
65              {
66
67                  if (hasBlocks) //If we previously had picked up the
                    blocks...
68                  {
69                      Debug.Log("Can Load");
70                      if (Input.GetMouseButtonDown(0)) //If we press the left
                        mouse button...
71                      {
72                          PointSystem script =
                            gameObject.GetComponent<PointSystem>();
73                          script.Points(5); //Add 5 points again.
74                          hasBlocks = false; //We don't have the blocks
                            anymore, we put them on the platform.
75                          anim.animator.SetBool("isMovingBlock", false); //We
                            stop playing the animation.
76                          CmdPlatform(); //Hey Server! Do something else!
77                      }
78                  }
79              }
80          }
```

```
81
82      }
83
84      [Command]
85      void CmdBlock()
86      {
87          RpcBlock();
88      }
89
90      [ClientRpc]
91      void RpcBlock()
92      {
93          myBlocks.SetActive(true); //Set the invisible gameObject (blocks)
              parented to the player active for all the clients.
94      }
95
96      [Command]
97      void CmdPlatform()
98      {
99          RpcPlatform();
100     }
101
102     [ClientRpc]
103     void RpcPlatform()
104     {
105         blocks = GameObject.FindGameObjectsWithTag("Box"); //The array of
              GameObjects we declared as blocks is the amount of GameObjects
              tagged as "Box".
106         blockRenderer = new Renderer[blocks.Length]; //blockRenderer is an
              array of Renderers that has the length of the blocks' array.
107         for (int i = 0; i < blocks.Length; i++) //For an integer number i=0,
               if its smaller than the length of the array blocks, do something
              and add i+1;
108         {
109             blocks[i].layer = 0; /*Put that block in the layer 0. That layer
                   is the default one. This is because in the scene they were
                  defined
110             in a custom layer that didn't collide with the players  -so they
                  were invisible and "uncollidable". Now we make them
                  collidable.*/
111             blockRenderer[i] = blocks[i].GetComponent<Renderer>(); //Now we
                  define what the elements in the blockRenderer array are
                  exactly.
112             blockRenderer[i].enabled = true; //We enable them, so now we see
                   the blocks!
113             blocks[i].tag = "BlockRoof"; //We'll tag them as BlockRoof.
                  That's for the next script, when we pick them up in the roof.
114         }
115         myBlocks.SetActive(false); //Set the blocks parented to my player
              unactive for all the clients.
116         GameObject crane = GameObject.FindGameObjectWithTag("Crane"); //
              Declare a gameObject called Crane. It'll be the crane.
117         crane_animate1 script = crane.GetComponent<crane_animate1>(); //We
```

```
            get a script attatched to the gameObject crane, not ourselves!
118         //This script will control the upwards movement of the crane. The
            crane's movement will be done through 2 scripts.
119         script.isMoving = true; //We activate 2 boolean variables from that
            script, so it starts running and therefore moving the crane.
120         script.enabled = true; //We enable the script. This is because both
            scripts that control the crane enable and disable themselves,
121         //But for the first movement we have to tell it when to start going
            upwards!
122         script.Start(); //This is also to restart the position of the crane.
             This way the movement between the two stages that the crane has
            is smooth.
123     }
124 }
125
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5
6  //THIS SCRIPT IS REALLY SIMILAR TO THE PICKUPBLOCKS ONE.
7  //WE'LL PICK UP THE BLOCKS FROM THE PLATFORM ON THE ROOF.
8
9  public class PickUpBoxRoof : NetworkBehaviour {
10
11     public Transform cameraTransform;
12     public GameObject myBlocks;
13
14
15     Renderer[] blockRenderer;
16     GameObject[] blocks;
17
18     public bool canBuild;
19     public int clicks = 0; //Integer that will control we've run out of
         blocks to build and we need another load of blocks.
20
21     NetworkAnimator anim;
22
23     // Use this for initialization
24     void Start()
25     {
26         anim = GetComponent<NetworkAnimator>();
27     }
28
29     // Update is called once per frame
30     void Update()
31     {
32         RaycastHit hit;
33         Vector3 rayPos = cameraTransform.position + cameraTransform.forward
             * 1f;
34         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
35
36
37         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
38         {
39             if (hit.transform.tag == "BlockRoof")
40             {
41                 GameObject crane = GameObject.FindGameObjectWithTag
                     ("Crane");
42                 crane_animate1 script = crane.GetComponent<crane_animate1>
                     ();
43                 if (script.canPickBlocksUp) //In the crane script
                     "crane_animate1" there is a boolean that is set to true
                     when the platform
44                     //touches the roof.
45                 {
46                     if (Input.GetMouseButtonDown(0))
```

```
47                      {
48                          PointSystem script2 =
                        gameObject.GetComponent<PointSystem>();
49                          script2.Points(5);
50                          canBuild = true; //We have blocks now, so... We can
                        build!
51                          anim.animator.SetBool("isMovingBlock", true); //Also
                         start again the animation holding the blocks.
52                          CmdCanBuild();
53                      }
54                  }
55              }
56          }
57
58          if (clicks > 11 && canBuild) //If we put 12 blocks we run out of
              them. This click variable will come from the
59              //several BuildWall scripts every time we build a wall.
60          {
61              anim.animator.SetBool("isMovingBlock", false); //Animation
                  holding blocks is over.
62              CmdJobDone();
63              clicks = 0; //Set clicks back to 0 so next time we have 12
                  blocks again.
64              canBuild = false; //We can't build anymore :(
65          }
66      }
67
68
69
70      [Command]
71      void CmdCanBuild()
72      {
73          RpcCanBuild();
74      }
75
76      [Command]
77      void CmdJobDone()
78      {
79          RpcJobDone();
80      }
81
82      [ClientRpc]
83      void RpcCanBuild()
84      {
85          myBlocks.SetActive(true); //Set the blocks parented to our
              gameObject active for all the clients.
86          blocks = GameObject.FindGameObjectsWithTag("BlockRoof"); //Similar
              procedure as with PickUpBlocks.
87          blockRenderer = new Renderer[blocks.Length];
88          for (int i = 0; i < blocks.Length; i++)
89          {
90              blocks[i].layer = 12; //Bring the blocks back to the layer where
                    they can't collide with the player, because they don't exist
```

```
                in the platform.
91              blockRenderer[i] = blocks[i].GetComponent<Renderer>();
92              blockRenderer[i].enabled = false;
93              blocks[i].tag = "Box"; //Bring the tag of the blocks back to
                  Box.
94          }
95          GameObject crane = GameObject.FindGameObjectWithTag("Crane"); //Find
              the GameObject crane.
96          Crane script = crane.GetComponent<Crane>(); //Find the second script
              in crane that controlls its movement downwards.
97          script.isMoving = true; //We allow it to start moving.
98          script.enabled = true; //We enable it. At this point the first crane
              script will have disbled itself, so both scripts won't overlap.
99          script.Start(); //Restart so the transition of positions between
              scripts is smooth.
100     }
101
102     [ClientRpc]
103     void RpcJobDone()
104     {
105         myBlocks.SetActive(false); //Make the blocks parented to our
              gameObject unactive for all the clients.
106     }
107 }
108
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Audio;
5
6  //THIS SCRIPT WILL CONTROL THE COLLISION OF OUR PLAYER WITH THE PLATFORM.
7
8  public class CollisionPlatform : MonoBehaviour
9  {
10     public AudioClip clips; //Public audio clip to play when we collide with
         the platform.
11
12     // Use this for initialization
13     void Start()
14     {
15
16     }
17
18     // Update is called once per frame
19     void Update()
20     {
21
22     }
23
24     private void OnControllerColliderHit(ControllerColliderHit collision)
25     {
26         if (collision.gameObject.tag == "Platform") //If we collide with the
             Crane (gameObject tagged as "Platform")...
27         {
28             GameObject crane = GameObject.FindGameObjectWithTag("Crane");
29             crane_animate1 script = crane.GetComponent<crane_animate1>();
30             if (script.isMoving) //If the variable isMoving from the
                 crane_animate1 script is true (so, if the platform is going
                 upwards)...
31             {
32                 Debug.Log("collided");
33                 Health2 health = gameObject.GetComponent<Health2>();
34                 if (health != null)
35                 {
36                     PlaySound(); //Run the function PlaySound() declared
                       below.
37                     health.TakeDamage(5); //Take Damage in the healty script.
38                 }
39             }
40
41             Crane script2 = crane.GetComponent<Crane>();
42             if (script2.isMoving) //If the variable isMoving from the Crane
                 script is true (so, if the platform is going downwards)...
43             {
44                 Debug.Log("collided");
45                 Health2 health = gameObject.GetComponent<Health2>();
46                 if (health != null)
47                 {
```

```
48                      PlaySound(); //Run the function PlaySound() declared
                        below.
49                      health.TakeDamage(5); //Take Damage in the healty script.
50                  }
51              }
52          }
53      }
54
55      void PlaySound() //Same function PlaySound as some of the described
          before.
56      {
57          AudioSource source = gameObject.AddComponent<AudioSource>();
58          source.clip = clips;
59          source.Play();
60          Destroy(source, clips.length);
61      }
62  }
63
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  //THIS SCRIPT WILL ALLOW US TO PICK UP THE HARNESS. IT'S EXACTLY THE SAME
8  //PROCESS AS PICKING UP THE HELMET, SO THERE WON'T BE AN EXPLANATION TO IT.
9
10 public class PickUpArnes : NetworkBehaviour
11 {
12
13     public Transform cameraTransform;
14
15     GameObject arnesFloor;
16     public GameObject arnesBody;
17     GameObject myText4;
18     GameObject arnesOn;
19
20     GameObject myText2;
21     public bool hasArnes;
22     // Use this for initialization
23     void Start()
24     {
25         myText4 = GameObject.FindGameObjectWithTag("Text4");
26         arnesOn = GameObject.FindGameObjectWithTag("ArnesOn");
27         myText2 = GameObject.FindGameObjectWithTag("Text2");
28     }
29
30     // Update is called once per frame
31     void Update()
32     {
33         RaycastHit hit;
34         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
               1f;
35         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
36
37         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
38         {
39             if (hit.transform.tag == "Arnes")
40             {
41                 if (!hasArnes)
42                 {
43                     myText4.GetComponent<Text>().enabled = true;
44                 }
45                 arnesFloor = hit.transform.gameObject;
46                 if (Input.GetMouseButtonDown(0))
47                 {
48                     myText4.GetComponent<Text>().enabled = false;
49                     arnesOn.GetComponent<Text>().enabled = true;
50                     CmdArnes(arnesFloor, hasArnes);
51                     if (!hasArnes)
```

```
52                      {
53                          PointSystem point =
                        gameObject.GetComponent<PointSystem>();
54                          point.Points(10);
55                      }
56                      hasArnes = true;
57                  }
58              }
59
60              else
61              {
62                  myText4.GetComponent<Text>().enabled = false;
63              }
64          }
65      }
66
67      [Command]
68      void CmdArnes(GameObject arnesFloor, bool hasArnes)
69      {
70          RpcArnes(arnesFloor, hasArnes);
71      }
72
73      [ClientRpc]
74      void RpcArnes(GameObject arnesFloor, bool hasArnes)
75      {
76          if (!hasArnes)
77          {
78              arnesFloor.SetActive(false);
79              arnesBody.SetActive(true);
80          }
81      }
82  }
83
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.Networking;
6
7  //THIS SCRIPT WILL ALLOW US TO ACTIVATE THE TEXT TO ALLOWS THE PLAYER TO
8  //BUILD EACH SECTION OF WALL. ALSO STOPS THE
9
10 public class ActivateTextWall : NetworkBehaviour
11 {
12     NetworkAnimator anim; //We'll be working with animations again.
13     public int count = 0; //Public integer that we'll use to count how many
          blocks have we built.
14     public Transform cameraTransform; //Camera transform for the raycast.
15     public GameObject blocks; //Our public gameObject blocks that will be
         the "Invisible" blocks to be built.
16     GameObject myBuildText; //Text that will allow us to pick up the
         object.
17     bool canSumPoints1; //Bool value for every block to be built in the
         scene. It shows if it can sum punts or not.
18     bool canSumPoints2;
19     bool canSumPoints3;
20     bool canSumPoints4;
21     bool canSumPoints5;
22     bool canSumPoints6;
23     bool canSumPoints7;
24     bool canSumPoints8;
25     bool canSumPoints9;
26     bool canSumPoints10;
27     bool canSumPoints11;
28     bool canSumPoints12;
29     bool canSumPoints13;
30     bool canSumPoints14;
31     bool canSumPoints15;
32     bool canSumPoints16;
33     bool canSumPoints17;
34     bool canSumPoints18;
35     bool canSumPoints19;
36     bool canSumPoints20;
37     bool canSumPoints121;
38     bool canSumPoints122;
39     bool canSumPoints123;
40     bool canSumPoints124;
41     bool canSumPoints125;
42     bool canSumPoints126;
43     bool canSumPoints127;
44     bool canSumPoints128;
45     bool canSumPoints129;
46     bool canSumPoints130;
47     bool canSumPoints131;
48     bool canSumPoints132;
49     bool canSumPoints133;
```

```
50      bool canSumPoints134;
51      bool canSumPoints135;
52      bool canSumPoints136;
53      bool canSumPoints137;
54      bool canSumPoints138;
55      bool canSumPoints139;
56      bool canSumPoints140;
57      bool canSumPoints141;
58      bool canSumPoints142;
59      bool canSumPoints143;
60      bool canSumPoints144;
61      bool canSumPoints145;
62      bool canSumPoints146;
63      bool canSumPoints147;
64      bool canSumPoints148;
65      bool canSumPoints149;
66      bool canSumPoints150;
67      bool canSumPoints151;
68      bool canSumPoints152;
69      bool canSumPoints153;
70      bool canSumPoints154;
71      bool canSumPoints155;
72      bool canSumPoints156;
73      bool canSumPoints157;
74      bool canSumPoints158;
75
76      // Use this for initialization
77      void Start()
78      {
79          canSumPoints1 = true; //Since no block is built yet all of them can ↵
               sum points.
80          canSumPoints2 = true;
81          canSumPoints3 = true;
82          canSumPoints4 = true;
83          canSumPoints5 = true;
84          canSumPoints6 = true;
85          canSumPoints7 = true;
86          canSumPoints8 = true;
87          canSumPoints9 = true;
88          canSumPoints10 = true;
89          canSumPoints11 = true;
90          canSumPoints12 = true;
91          canSumPoints13 = true;
92          canSumPoints14 = true;
93          canSumPoints15 = true;
94          canSumPoints16 = true;
95          canSumPoints17 = true;
96          canSumPoints18 = true;
97          canSumPoints19 = true;
98          canSumPoints20 = true;
99          canSumPoints121 = true;
100         canSumPoints122 = true;
101         canSumPoints123 = true;
```

```
102         canSumPoints124 = true;
103         canSumPoints125 = true;
104         canSumPoints126 = true;
105         canSumPoints127 = true;
106         canSumPoints128 = true;
107         canSumPoints129 = true;
108         canSumPoints130 = true;
109         canSumPoints131 = true;
110         canSumPoints132 = true;
111         canSumPoints133 = true;
112         canSumPoints134 = true;
113         canSumPoints135 = true;
114         canSumPoints136 = true;
115         canSumPoints137 = true;
116         canSumPoints138 = true;
117         canSumPoints139 = true;
118         canSumPoints140 = true;
119         canSumPoints141 = true;
120         canSumPoints142 = true;
121         canSumPoints143 = true;
122         canSumPoints144 = true;
123         canSumPoints145 = true;
124         canSumPoints146 = true;
125         canSumPoints147 = true;
126         canSumPoints148 = true;
127         canSumPoints149 = true;
128         canSumPoints150 = true;
129         canSumPoints151 = true;
130         canSumPoints152 = true;
131         canSumPoints153 = true;
132         canSumPoints154 = true;
133         canSumPoints155 = true;
134         canSumPoints156 = true;
135         canSumPoints157 = true;
136         canSumPoints158 = true;
137         myBuildText = GameObject.FindGameObjectWithTag("TextBuild"); //We
              find our gameObject acting as text.
138         anim = gameObject.GetComponent<NetworkAnimator>(); //Define anim.
139     }
140
141     // Update is called once per frame
142     void Update()
143     {
144         PickUpBoxRoof script2 = gameObject.GetComponent<PickUpBoxRoof>();
145         RaycastHit hit;
146         Vector3 rayPos = cameraTransform.position + cameraTransform.forward
              * 1f;
147         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
              0.5f);
148
149
150         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit,
              5f)) //Raycast.
```

```
151         {
152             if (script2.canBuild) //If the variable canBuild from the PickUpBoxRoof is true, then...
153             {
154                 if (hit.transform.tag == "WallDef") //If the raycast hit a gameObject tagged as "WallDef"...
155                 {
156 
157                     if (canSumPoints1) //If it canSumPoints1 (so, if it's not built yet)...
158                     {
159                         myBuildText.GetComponent<Text>().enabled = true; //Show the text allowing us to pick it up.
160                         if (Input.GetMouseButtonDown(0)) //If we press the left mouse button...
161                         {
162                             PointSystem script = gameObject.GetComponent<PointSystem>();
163                             script.Points(1); //Add 1 to the points in the PointsSystem script function Points().
164                             canSumPoints1 = false; //We cannot build this concrete block anymore.
165                             myBuildText.GetComponent<Text>().enabled = false; //Disable our text inviting to build.
166                             count += 1; //Add one to the count of blocks built.
167                         }
168 
169                     }
170                 }
171 
172                 else if (hit.transform.tag == "WallDef2") //Eat, drink, sleep and repeat...
173                 {
174 
175                     if (canSumPoints2)
176                     {
177                         myBuildText.GetComponent<Text>().enabled = true;
178                         if (Input.GetMouseButtonDown(0))
179                         {
180                             PointSystem script = gameObject.GetComponent<PointSystem>();
181                             script.Points(1);
182                             canSumPoints2 = false;
183                             myBuildText.GetComponent<Text>().enabled = false;
184                             count += 1;
185                         }
186 
187                     }
188                 }
189 
190                 else if (hit.transform.tag == "WallDef3")
```

```
191                 {
192
193                     if (canSumPoints3)
194                     {
195                         myBuildText.GetComponent<Text>().enabled = true;
196                         if (Input.GetMouseButtonDown(0))
197                         {
198                             PointSystem script =
                       gameObject.GetComponent<PointSystem>();
199                             script.Points(1);
200                             canSumPoints3 = false;
201                             myBuildText.GetComponent<Text>().enabled =
                       false;
202                             count += 1;
203                         }
204
205                     }
206                 }
207
208                 else if (hit.transform.tag == "WallDef4")
209                 {
210
211                     if (canSumPoints4)
212                     {
213                         myBuildText.GetComponent<Text>().enabled = true;
214                         if (Input.GetMouseButtonDown(0))
215                         {
216                             PointSystem script =
                       gameObject.GetComponent<PointSystem>();
217                             script.Points(1);
218                             canSumPoints4 = false;
219                             myBuildText.GetComponent<Text>().enabled =
                       false;
220                             count += 1;
221                         }
222
223                     }
224                 }
225
226                 else if (hit.transform.tag == "WallDef5")
227                 {
228
229                     if(canSumPoints155)
230                     {
231                         myBuildText.GetComponent<Text>().enabled = true;
232                         if (Input.GetMouseButtonDown(0))
233                         {
234                             PointSystem script =
                       gameObject.GetComponent<PointSystem>();
235                             script.Points(1);
236                             canSumPoints155 = false;
237                             myBuildText.GetComponent<Text>().enabled =
                       false;
```

```
238                             count += 1;
239                         }
240
241                     }
242                 }
243                 else if (hit.transform.tag == "WallDef6")
244                 {
245
246                     if(canSumPoints5)
247                     {
248                         myBuildText.GetComponent<Text>().enabled = true;
249                         if (Input.GetMouseButtonDown(0))
250                         {
251                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
252                             script.Points(1);
253                             canSumPoints5 = false;
254                             myBuildText.GetComponent<Text>().enabled =
                    false;
255                             count += 1;
256                         }
257
258                     }
259                 }
260                 else if (hit.transform.tag == "WallDef7")
261                 {
262
263                     if(canSumPoints6)
264                     {
265                         myBuildText.GetComponent<Text>().enabled = true;
266                         if (Input.GetMouseButtonDown(0))
267                         {
268                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
269                             script.Points(1);
270                             canSumPoints6 = false;
271                             myBuildText.GetComponent<Text>().enabled =
                    false;
272                             count += 1;
273                         }
274
275                     }
276                 }
277                 else if (hit.transform.tag == "WallDef8")
278                 {
279
280                     if (canSumPoints7)
281                     {
282                         myBuildText.GetComponent<Text>().enabled = true;
283                         if (Input.GetMouseButtonDown(0))
284                         {
285                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
```

```
286                             script.Points(1);
287                             canSumPoints7 = false;
288                             myBuildText.GetComponent<Text>().enabled =
                        false;
289                             count += 1;
290                         }
291
292                     }
293                 }
294                 else if (hit.transform.tag == "WallDef9")
295                 {
296
297                     if (canSumPoints8)
298                     {
299                         myBuildText.GetComponent<Text>().enabled = true;
300                         if (Input.GetMouseButtonDown(0))
301                         {
302                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
303                             script.Points(1);
304                             canSumPoints8 = false;
305                             myBuildText.GetComponent<Text>().enabled =
                        false;
306                             count += 1;
307                         }
308
309                     }
310                 }
311                 else if (hit.transform.tag == "WallDef10")
312                 {
313
314                     if (canSumPoints9)
315                     {
316                         myBuildText.GetComponent<Text>().enabled = true;
317                         if (Input.GetMouseButtonDown(0))
318                         {
319                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
320                             script.Points(1);
321                             canSumPoints9 = false;
322                             myBuildText.GetComponent<Text>().enabled =
                        false;
323                             count += 1;
324                         }
325
326                     }
327                 }
328                 else if (hit.transform.tag == "WallDef11")
329                 {
330
331                     if (canSumPoints157)
332                     {
333                         myBuildText.GetComponent<Text>().enabled = true;
```

```
334                         if (Input.GetMouseButtonDown(0))
335                         {
336                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
337                             script.Points(1);
338                             canSumPoints157 = false;
339                             myBuildText.GetComponent<Text>().enabled =
                        false;
340                             count += 1;
341                         }
342
343                     }
344                 }
345                 else if (hit.transform.tag == "WallDef12")
346                 {
347
348                     if (canSumPoints158)
349                     {
350                         myBuildText.GetComponent<Text>().enabled = true;
351                         if (Input.GetMouseButtonDown(0))
352                         {
353                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
354                             script.Points(1);
355                             canSumPoints158 = false;
356                             myBuildText.GetComponent<Text>().enabled =
                        false;
357                             count += 1;
358                         }
359
360
361                     }
362                 }
363
364                 else if (hit.transform.tag == "South1")
365                 {
366
367                     if (canSumPoints10)
368                     {
369                         myBuildText.GetComponent<Text>().enabled = true;
370                         if (Input.GetMouseButtonDown(0))
371                         {
372                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
373                             script.Points(1);
374                             canSumPoints10 = false;
375                             myBuildText.GetComponent<Text>().enabled =
                        false;
376                             count += 1;
377                         }
378
379                     }
380                 }
```

```
381
382                 else if (hit.transform.tag == "South2")
383                 {
384
385                     if (canSumPoints11)
386                     {
387                         myBuildText.GetComponent<Text>().enabled = true;
388                         if (Input.GetMouseButtonDown(0))
389                         {
390                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
391                             script.Points(1);
392                             canSumPoints11 = false;
393                             myBuildText.GetComponent<Text>().enabled =
                    false;
394                             count += 1;
395                         }
396
397                     }
398                 }
399
400                 else if (hit.transform.tag == "South3")
401                 {
402
403                     if (canSumPoints12)
404                     {
405                         myBuildText.GetComponent<Text>().enabled = true;
406                         if (Input.GetMouseButtonDown(0))
407                         {
408                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
409                             script.Points(1);
410                             canSumPoints12 = false;
411                             myBuildText.GetComponent<Text>().enabled =
                    false;
412                             count += 1;
413                         }
414
415                     }
416                 }
417
418                 else if (hit.transform.tag == "South4")
419                 {
420
421                     if (canSumPoints13)
422                     {
423                         myBuildText.GetComponent<Text>().enabled = true;
424                         if (Input.GetMouseButtonDown(0))
425                         {
426                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
427                             script.Points(1);
428                             canSumPoints13 = false;
```

```
429                             myBuildText.GetComponent<Text>().enabled =
                    false;
430                             count += 1;
431                         }
432                     }
433                 }
434
435                 else if (hit.transform.tag == "South5")
436                 {
437
438                     if (canSumPoints14)
439                     {
440                         myBuildText.GetComponent<Text>().enabled = true;
441                         if (Input.GetMouseButtonDown(0))
442                         {
443                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
444                             script.Points(1);
445                             canSumPoints14 = false;
446                             myBuildText.GetComponent<Text>().enabled =
                    false;
447                             count += 1;
448                         }
449                     }
450                 }
451
452                 else if (hit.transform.tag == "South6")
453                 {
454
455                     if (canSumPoints15)
456                     {
457                         myBuildText.GetComponent<Text>().enabled = true;
458                         if (Input.GetMouseButtonDown(0))
459                         {
460                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
461                             script.Points(1);
462                             canSumPoints15 = false;
463                             myBuildText.GetComponent<Text>().enabled =
                    false;
464                             count += 1;
465                         }
466                     }
467                 }
468
469                 else if (hit.transform.tag == "South7")
470                 {
471
472                     if (canSumPoints16)
473                     {
474                         myBuildText.GetComponent<Text>().enabled = true;
475                         if (Input.GetMouseButtonDown(0))
476                         {
```

```
477                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
478                             script.Points(1);
479                             canSumPoints16 = false;
480                             myBuildText.GetComponent<Text>().enabled =
                    false;
481                             count += 1;
482                         }
483                     }
484                 }
485
486                 else if (hit.transform.tag == "South8")
487                 {
488
489                     if (canSumPoints17)
490                     {
491                         myBuildText.GetComponent<Text>().enabled = true;
492                         if (Input.GetMouseButtonDown(0))
493                         {
494                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
495                             script.Points(1);
496                             canSumPoints17 = false;
497                             myBuildText.GetComponent<Text>().enabled =
                    false;
498                             count += 1;
499                         }
500                     }
501                 }
502
503                 else if (hit.transform.tag == "South9")
504                 {
505
506                     if (canSumPoints18)
507                     {
508                         myBuildText.GetComponent<Text>().enabled = true;
509                         if (Input.GetMouseButtonDown(0))
510                         {
511                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
512                             script.Points(1);
513                             canSumPoints18 = false;
514                             myBuildText.GetComponent<Text>().enabled =
                    false;
515                             count += 1;
516                         }
517                     }
518                 }
519
520                 else if (hit.transform.tag == "South10")
521                 {
522
523                     if (canSumPoints19)
```

```
524                     {
525                         myBuildText.GetComponent<Text>().enabled = true;
526                         if (Input.GetMouseButtonDown(0))
527                         {
528                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
529                             script.Points(1);
530                             canSumPoints19 = false;
531                             myBuildText.GetComponent<Text>().enabled =
                        false;
532                             count += 1;
533                         }
534                     }
535                 }
536 
537                 else if (hit.transform.tag == "South11")
538                 {
539 
540                     if (canSumPoints20)
541                     {
542                         myBuildText.GetComponent<Text>().enabled = true;
543                         if (Input.GetMouseButtonDown(0))
544                         {
545                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
546                             script.Points(1);
547                             canSumPoints20 = false;
548                             myBuildText.GetComponent<Text>().enabled =
                        false;
549                             count += 1;
550                         }
551                     }
552                 }
553 
554                 else if (hit.transform.tag == "South12")
555                 {
556 
557                     if (canSumPoints121)
558                     {
559                         myBuildText.GetComponent<Text>().enabled = true;
560                         if (Input.GetMouseButtonDown(0))
561                         {
562                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
563                             script.Points(1);
564                             canSumPoints121 = false;
565                             myBuildText.GetComponent<Text>().enabled =
                        false;
566                             count += 1;
567                         }
568                     }
569                 }
570 
```

```
571                 else if (hit.transform.tag == "West1")
572                 {
573
574                     if (canSumPoints122)
575                     {
576                         myBuildText.GetComponent<Text>().enabled = true;
577                         if (Input.GetMouseButtonDown(0))
578                         {
579                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
580                             script.Points(1);
581                             canSumPoints122 = false;
582                             myBuildText.GetComponent<Text>().enabled =
                    false;
583                             count += 1;
584                         }
585                     }
586                 }
587
588                 else if (hit.transform.tag == "West2")
589                 {
590
591                     if (canSumPoints123)
592                     {
593                         myBuildText.GetComponent<Text>().enabled = true;
594                         if (Input.GetMouseButtonDown(0))
595                         {
596                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
597                             script.Points(1);
598                             canSumPoints123 = false;
599                             myBuildText.GetComponent<Text>().enabled =
                    false;
600                             count += 1;
601                         }
602                     }
603                 }
604
605                 else if (hit.transform.tag == "West3")
606                 {
607                     if (canSumPoints124)
608                     {
609                         myBuildText.GetComponent<Text>().enabled = true;
610                         if (Input.GetMouseButtonDown(0))
611                         {
612                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
613                             script.Points(1);
614                             canSumPoints124 = false;
615                             myBuildText.GetComponent<Text>().enabled =
                    false;
616                             count += 1;
617                         }
```

```
618                     }
619                 }
620 
621                 else if (hit.transform.tag == "West4")
622                 {
623                     if (canSumPoints125)
624                     {
625                         myBuildText.GetComponent<Text>().enabled = true;
626                         if (Input.GetMouseButtonDown(0))
627                         {
628                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
629                             script.Points(1);
630                             canSumPoints125 = false;
631                             myBuildText.GetComponent<Text>().enabled =
                    false;
632                             count += 1;
633                         }
634                     }
635                 }
636 
637                 else if (hit.transform.tag == "West6")
638                 {
639                     if (canSumPoints126)
640                     {
641                         myBuildText.GetComponent<Text>().enabled = true;
642                         if (Input.GetMouseButtonDown(0))
643                         {
644                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
645                             script.Points(1);
646                             canSumPoints126 = false;
647                             myBuildText.GetComponent<Text>().enabled =
                    false;
648                             count += 1;
649                         }
650                     }
651                 }
652 
653                 else if (hit.transform.tag == "West7")
654                 {
655                     if (canSumPoints127)
656                     {
657                         myBuildText.GetComponent<Text>().enabled = true;
658                         if (Input.GetMouseButtonDown(0))
659                         {
660                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
661                             script.Points(1);
662                             canSumPoints127 = false;
663                             myBuildText.GetComponent<Text>().enabled =
                    false;
664                             count += 1;
```

```
665                         }
666                     }
667                 }
668 
669                 else if (hit.transform.tag == "West8")
670                 {
671                     if (canSumPoints128)
672                     {
673                         myBuildText.GetComponent<Text>().enabled = true;
674                         if (Input.GetMouseButtonDown(0))
675                         {
676                             PointSystem script =
    gameObject.GetComponent<PointSystem>();
677                             script.Points(1);
678                             canSumPoints128 = false;
679                             myBuildText.GetComponent<Text>().enabled =
    false;
680                             count += 1;
681                         }
682                     }
683                 }
684 
685                 else if (hit.transform.tag == "West9")
686                 {
687                     if (canSumPoints129)
688                     {
689                         myBuildText.GetComponent<Text>().enabled = true;
690                         if (Input.GetMouseButtonDown(0))
691                         {
692                             PointSystem script =
    gameObject.GetComponent<PointSystem>();
693                             script.Points(1);
694                             canSumPoints129 = false;
695                             myBuildText.GetComponent<Text>().enabled =
    false;
696                             count += 1;
697                         }
698                     }
699                 }
700 
701                 else if (hit.transform.tag == "West10")
702                 {
703                     if (canSumPoints130)
704                     {
705                         myBuildText.GetComponent<Text>().enabled = true;
706                         if (Input.GetMouseButtonDown(0))
707                         {
708                             PointSystem script =
    gameObject.GetComponent<PointSystem>();
709                             script.Points(1);
710                             canSumPoints130 = false;
711                             myBuildText.GetComponent<Text>().enabled =
    false;
```

```
712                             count += 1;
713                         }
714                     }
715                 }
716 
717                 else if (hit.transform.tag == "West11")
718                 {
719                     if (canSumPoints131)
720                     {
721                         myBuildText.GetComponent<Text>().enabled = true;
722                         if (Input.GetMouseButtonDown(0))
723                         {
724                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
725                             script.Points(1);
726                             canSumPoints131 = false;
727                             myBuildText.GetComponent<Text>().enabled =
                    false;
728                             count += 1;
729                         }
730                     }
731                 }
732 
733                 else if (hit.transform.tag == "West13")
734                 {
735                     if (canSumPoints132)
736                     {
737                         myBuildText.GetComponent<Text>().enabled = true;
738                         if (Input.GetMouseButtonDown(0))
739                         {
740                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
741                             script.Points(1);
742                             canSumPoints132 = false;
743                             myBuildText.GetComponent<Text>().enabled =
                    false;
744                             count += 1;
745                         }
746                     }
747                 }
748 
749                 else if (hit.transform.tag == "West14")
750                 {
751                     if (canSumPoints133)
752                     {
753                         myBuildText.GetComponent<Text>().enabled = true;
754                         if (Input.GetMouseButtonDown(0))
755                         {
756                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
757                             script.Points(1);
758                             canSumPoints133 = false;
759                             myBuildText.GetComponent<Text>().enabled =
```

```
                    false;
760                         count += 1;
761                     }
762                 }
763             }
764
765             else if (hit.transform.tag == "West15")
766             {
767                 if (canSumPoints134)
768                 {
769                     myBuildText.GetComponent<Text>().enabled = true;
770                     if (Input.GetMouseButtonDown(0))
771                     {
772                         PointSystem script =
    gameObject.GetComponent<PointSystem>();
773                         script.Points(1);
774                         canSumPoints134 = false;
775                         myBuildText.GetComponent<Text>().enabled =
    false;
776                         count += 1;
777                     }
778                 }
779             }
780
781             else if (hit.transform.tag == "West16")
782             {
783                 if (canSumPoints135)
784                 {
785                     myBuildText.GetComponent<Text>().enabled = true;
786                     if (Input.GetMouseButtonDown(0))
787                     {
788                         PointSystem script =
    gameObject.GetComponent<PointSystem>();
789                         script.Points(1);
790                         canSumPoints135 = false;
791                         myBuildText.GetComponent<Text>().enabled =
    false;
792                         count += 1;
793                     }
794                 }
795             }
796
797             else if (hit.transform.tag == "West17")
798             {
799                 if (canSumPoints136)
800                 {
801                     myBuildText.GetComponent<Text>().enabled = true;
802                     if (Input.GetMouseButtonDown(0))
803                     {
804                         PointSystem script =
    gameObject.GetComponent<PointSystem>();
805                         script.Points(1);
806                         canSumPoints136 = false;
```

```
807                             myBuildText.GetComponent<Text>().enabled =
                    false;
808                             count += 1;
809                         }
810                     }
811                 }
812
813                 else if (hit.transform.tag == "West18")
814                 {
815                     if (canSumPoints137)
816                     {
817                         myBuildText.GetComponent<Text>().enabled = true;
818                         if (Input.GetMouseButtonDown(0))
819                         {
820                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
821                             script.Points(1);
822                             canSumPoints137= false;
823                             myBuildText.GetComponent<Text>().enabled =
                    false;
824                             count += 1;
825                         }
826                     }
827                 }
828
829                 else if (hit.transform.tag == "West19")
830                 {
831                     if (canSumPoints138)
832                     {
833                         myBuildText.GetComponent<Text>().enabled = true;
834                         if (Input.GetMouseButtonDown(0))
835                         {
836                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
837                             script.Points(1);
838                             canSumPoints138 = false;
839                             myBuildText.GetComponent<Text>().enabled =
                    false;
840                             count += 1;
841                         }
842                     }
843                 }
844
845                 else if (hit.transform.tag == "East1")
846                 {
847                     if (canSumPoints139)
848                     {
849                         myBuildText.GetComponent<Text>().enabled = true;
850                         if (Input.GetMouseButtonDown(0))
851                         {
852                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
853                             script.Points(1);
```

```
854                             canSumPoints139 = false;
855                             myBuildText.GetComponent<Text>().enabled =
                        false;
856                             count += 1;
857                         }
858                     }
859                 }
860 
861                 else if (hit.transform.tag == "East2")
862                 {
863                     if (canSumPoints140)
864                     {
865                         myBuildText.GetComponent<Text>().enabled = true;
866                         if (Input.GetMouseButtonDown(0))
867                         {
868                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
869                             script.Points(1);
870                             canSumPoints140 = false;
871                             myBuildText.GetComponent<Text>().enabled =
                        false;
872                             count += 1;
873                         }
874                     }
875                 }
876 
877                 else if (hit.transform.tag == "East3")
878                 {
879                     if (canSumPoints141)
880                     {
881                         myBuildText.GetComponent<Text>().enabled = true;
882                         if (Input.GetMouseButtonDown(0))
883                         {
884                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
885                             script.Points(1);
886                             canSumPoints141 = false;
887                             myBuildText.GetComponent<Text>().enabled =
                        false;
888                             count += 1;
889                         }
890                     }
891                 }
892 
893                 else if (hit.transform.tag == "East4")
894                 {
895                     if (canSumPoints142)
896                     {
897                         myBuildText.GetComponent<Text>().enabled = true;
898                         if (Input.GetMouseButtonDown(0))
899                         {
900                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
```

```
901                             script.Points(1);
902                             canSumPoints142 = false;
903                             myBuildText.GetComponent<Text>().enabled =
                    false;
904                             count += 1;
905                         }
906                     }
907                 }
908 
909                 else if (hit.transform.tag == "East6")
910                 {
911                     if (canSumPoints143)
912                     {
913                         myBuildText.GetComponent<Text>().enabled = true;
914                         if (Input.GetMouseButtonDown(0))
915                         {
916                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
917                             script.Points(1);
918                             canSumPoints143 = false;
919                             myBuildText.GetComponent<Text>().enabled =
                    false;
920                             count += 1;
921                         }
922                     }
923                 }
924 
925                 else if (hit.transform.tag == "East7")
926                 {
927                     if (canSumPoints144)
928                     {
929                         myBuildText.GetComponent<Text>().enabled = true;
930                         if (Input.GetMouseButtonDown(0))
931                         {
932                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
933                             script.Points(1);
934                             canSumPoints144 = false;
935                             myBuildText.GetComponent<Text>().enabled =
                    false;
936                             count += 1;
937                         }
938                     }
939                 }
940 
941                 else if (hit.transform.tag == "East8")
942                 {
943                     if (canSumPoints145)
944                     {
945                         myBuildText.GetComponent<Text>().enabled = true;
946                         if (Input.GetMouseButtonDown(0))
947                         {
948                             PointSystem script =
```

```
                    gameObject.GetComponent<PointSystem>();
949                         script.Points(1);
950                         myBuildText.GetComponent<Text>().enabled =
                    false;
951                         canSumPoints145 = false;
952                         count += 1;
953                     }
954                 }
955             }
956
957             else if (hit.transform.tag == "East9")
958             {
959                 if (canSumPoints146)
960                 {
961                     myBuildText.GetComponent<Text>().enabled = true;
962                     if (Input.GetMouseButtonDown(0))
963                     {
964                         PointSystem script =
                    gameObject.GetComponent<PointSystem>();
965                         script.Points(1);
966                         canSumPoints146 = false;
967                         myBuildText.GetComponent<Text>().enabled =
                    false;
968                         count += 1;
969                     }
970                 }
971             }
972
973             else if (hit.transform.tag == "East10")
974             {
975                 if (canSumPoints147)
976                 {
977                     myBuildText.GetComponent<Text>().enabled = true;
978                     if (Input.GetMouseButtonDown(0))
979                     {
980                         PointSystem script =
                    gameObject.GetComponent<PointSystem>();
981                         script.Points(1);
982                         canSumPoints147 = false;
983                         myBuildText.GetComponent<Text>().enabled =
                    false;
984                         count += 1;
985                     };
986                 }
987             }
988
989             else if (hit.transform.tag == "East11")
990             {
991                 if (canSumPoints148)
992                 {
993                     myBuildText.GetComponent<Text>().enabled = true;
994                     if (Input.GetMouseButtonDown(0))
995                     {
```

```
996                             PointSystem script =
                    gameObject.GetComponent<PointSystem>();
997                             script.Points(1);
998                             canSumPoints148 = false;
999                             myBuildText.GetComponent<Text>().enabled =
                    false;
1000                            count += 1;
1001                        }
1002                    }
1003                }
1004
1005                else if (hit.transform.tag == "East13")
1006                {
1007                    if (canSumPoints149)
1008                    {
1009                        myBuildText.GetComponent<Text>().enabled = true;
1010                        if (Input.GetMouseButtonDown(0))
1011                        {
1012                            PointSystem script =
                    gameObject.GetComponent<PointSystem>();
1013                            script.Points(1);
1014                            canSumPoints149 = false;
1015                            myBuildText.GetComponent<Text>().enabled =
                    false;
1016                            count += 1;
1017                        }
1018                    }
1019                }
1020
1021                else if (hit.transform.tag == "East14")
1022                {
1023                    if (canSumPoints150)
1024                    {
1025                        myBuildText.GetComponent<Text>().enabled = true;
1026                        if (Input.GetMouseButtonDown(0))
1027                        {
1028                            PointSystem script =
                    gameObject.GetComponent<PointSystem>();
1029                            script.Points(1);
1030                            canSumPoints150 = false;
1031                            myBuildText.GetComponent<Text>().enabled =
                    false;
1032                            count += 1;
1033                        }
1034                    }
1035                }
1036
1037                else if (hit.transform.tag == "East15")
1038                {
1039                    if (canSumPoints151)
1040                    {
1041                        myBuildText.GetComponent<Text>().enabled = true;
1042                        if (Input.GetMouseButtonDown(0))
```

```
1043                            {
1044                                PointSystem script =
                        gameObject.GetComponent<PointSystem>();
1045                                script.Points(1);
1046                                canSumPoints151 = false;
1047                                myBuildText.GetComponent<Text>().enabled =
                        false;
1048                                count += 1;
1049                            }
1050                        }
1051                    }
1052
1053                    else if (hit.transform.tag == "East16")
1054                    {
1055                        if (canSumPoints152)
1056                        {
1057                            myBuildText.GetComponent<Text>().enabled = true;
1058                            if (Input.GetMouseButtonDown(0))
1059                            {
1060                                PointSystem script =
                        gameObject.GetComponent<PointSystem>();
1061                                script.Points(1);
1062                                canSumPoints152 = false;
1063                                myBuildText.GetComponent<Text>().enabled =
                        false;
1064                                count += 1;
1065                            }
1066                        }
1067                    }
1068
1069                    else if (hit.transform.tag == "East17")
1070                    {
1071                        if (canSumPoints153)
1072                        {
1073                            myBuildText.GetComponent<Text>().enabled = true;
1074                            if (Input.GetMouseButtonDown(0))
1075                            {
1076                                PointSystem script =
                        gameObject.GetComponent<PointSystem>();
1077                                script.Points(1);
1078                                canSumPoints153 = false;
1079                                myBuildText.GetComponent<Text>().enabled =
                        false;
1080                                count += 1;
1081                            }
1082                        }
1083                    }
1084
1085                    else if (hit.transform.tag == "East18")
1086                    {
1087                        if (canSumPoints154)
1088                        {
1089                            myBuildText.GetComponent<Text>().enabled = true;
```

```
1090                        if (Input.GetMouseButtonDown(0))
1091                        {
1092                            PointSystem script =
                        gameObject.GetComponent<PointSystem>();
1093                            script.Points(1);
1094                            canSumPoints154 = false;
1095                            myBuildText.GetComponent<Text>().enabled =
                        false;
1096                            count += 1;
1097                        }
1098                     }
1099                 }
1100
1101                 else if (hit.transform.tag == "East19")
1102                 {
1103                     if (canSumPoints155)
1104                     {
1105                         myBuildText.GetComponent<Text>().enabled = true;
1106                         if (Input.GetMouseButtonDown(0))
1107                         {
1108                             PointSystem script =
                        gameObject.GetComponent<PointSystem>();
1109                             script.Points(1);
1110                             canSumPoints155 = false;
1111                             myBuildText.GetComponent<Text>().enabled =
                        false;
1112                             count += 1;
1113                         }
1114                     }
1115                 }
1116
1117                 else if (hit.transform == null) //If our raycast doesn't
                      hit anything don't show the text.
1118                     //That is because our raycast sometimes hits our
                        desired object, sets the text to active
1119                     //and then when we look away if the Raycast doesn't hit
                        another collider (so, it hits null),
1120                     //the text doesn't disappear because certainly it
                        hasn't hit something else. It hasn't hit anything.
1121                 {
1122                     myBuildText.GetComponent<Text>().enabled = false;
1123                 }
1124
1125                 else //If we hit something else, text to false.
1126                 {
1127                     myBuildText.GetComponent<Text>().enabled = false;
1128                 }
1129             }
1130
1131
1132             else //Again. Just in case the 2 first times it doesnt't work,
                    I guess...
1133             {
```

```
1134                    myBuildText.GetComponent<Text>().enabled = false;

1135                }
1136
1137            }
1138
1139            if (count >= 56) //If the number of blocks built is greater equal
                  or greater than 56 then it means the walls are built and the job
                  is done!
1140            {
1141                anim.animator.SetBool("isMovingBlock", false); //Stop the
                      animation where we hold the block.
1142                CmdBlocksFalse(); //Hey server! Do something!
1143            }
1144        }
1145
1146        [Command]
1147        void CmdBlocksFalse()
1148        {
1149            RpcBlocksFalse();
1150        }
1151
1152        [ClientRpc]
1153        void RpcBlocksFalse() //Oh hello clients. Set the blocks of the
              gameObject they're parented to to unactive please.
1154        {
1155            blocks.SetActive(false);
1156        }
1157    }
1158
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using UnityEngine.Networking;
6
7  //THIS SCRIPT WILL SHOW US THE TEXT TO ALLOW US TO LOAD THE BLOCKS ONTO
8  //THE CRANE'S PLATFORM. WE WON'T GET INTO IT AS IT'S BEEN DONE BEFORE.
9
10 public class TextBlocks : NetworkBehaviour
11 {
12     public Transform cameraTransform;
13
14
15     GameObject myLoadText;
16
17     // Use this for initialization
18     void Start()
19     {
20         myLoadText = GameObject.FindGameObjectWithTag("LoadBlocks");
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26
27         RaycastHit hit;
28         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
               1f;
29         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
30
31
32         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
33         {
34             if (hit.transform.tag == "Box")
35             {
36                 PickUpBlock script = gameObject.GetComponent<PickUpBlock>();
37                 if (script.hasBlocks)
38                 {
39                     myLoadText.GetComponent<Text>().enabled = true;
40                 }
41             }
42
43             else
44             {
45                 myLoadText.GetComponent<Text>().enabled = false;
46
47             }
48         }
49
50
51     }
```

```
52  }
53
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5  using UnityEngine.UI;
6
7  //THIS SCRIPT WILL CONTAIN THE FUNCTION THAT WILL ALLOW US TO DISABLE THE
     CONTROLLERS
8  //OF THE PLAYER AND SHOW THE TIME, POINTS AND LIFE ACHIEVED AT THE END OF THE
     GAME.
9
10 public class EndingInfo : NetworkBehaviour {
11
12     public Transform cameraTransform; //Camera Transform for the raycast.
         Some code lines never get old.
13     public GameObject fenceDoorText; //A public GameObject that will be the
         entrance door of the fence.
14     public GameObject cPRText; //Text allowing us to perform CPR if someone
         died.
15
16     public Text gamePointsFinal; //Text showing the final points.
17     public Text gameHealthFinal; //Text showing the final health.
18     public Text gameTimeFinal; //Text showing the final time.
19
20     public GameObject finalInfo; //GameObject finalInfo. It is the UI that
         will contain the info.
21     public UnityStandardAssets.Characters.FirstPerson.FirstPersonController
         fpsController; //Controllers of the fps.
22     public GameObject lifeUI; //Various UI text GameObjects.
23     public GameObject timeUI;
24     public GameObject pointsUI;
25     public GameObject reticule;
26     public GameObject liniaOn;
27     public GameObject cascOn;
28     public GameObject earmuuffsOn;
29     public GameObject arnesOn;
30     // Use this for initialization
31     void Start () {
32
33     }
34
35     // Update is called once per frame
36     void Update () {
37         RaycastHit hit;
38         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
             1f;
39         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
40
41         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
42         {
43             if (hit.transform.tag == "FenceDoor") //If our raycast hits a the
                 fence door tagged as "FenceDoor"...
```

```
44              {
45                  fenceDoorText.GetComponent<Text>().enabled = true; //Text
                      allowing us to exit the construction site.
46                  if (Input.GetMouseButtonDown(0)) //If we click run EndGame()
                      function.
47                  {
48                      EndGame();
49                  }
50              }
51
52              else //If we don't hit the door, set the text to unenabled.
53              {
54                  fenceDoorText.GetComponent<Text>().enabled = false;
55              }
56          }
57
58      }
59
60      public void EndGame() //Function EndGame().
61      {
62          Timer script6 = gameObject.GetComponent<Timer>(); //Get the script
              Timer in attatched to this gameObject.
63          string timeString = string.Format("Tiempo: {0:00}:{1:00}",
              script6.minutes, script6.seconds); //Create a string "timeString"
              and define it.
64          gameTimeFinal.text = timeString; //Make the timeString display the
              gameTimeFina gameObject text.
65          PointSystem script = gameObject.GetComponent<PointSystem>(); //Same
              with the points.
66          string pointString = string.Format("Puntuación: {0}pts",
              script.points);
67          gamePointsFinal.text = pointString;
68          Health2 script2 = gameObject.GetComponent<Health2>(); //Same with the
               time, although in the middle we round the currentHealth to an
              integer.
69          int finalPoints = Mathf.RoundToInt(script2.currentHealth);
70          string healthString = string.Format("Vida: {0}/100", finalPoints);
71          gameHealthFinal.text = healthString;
72          FallDamage2 script3 = gameObject.GetComponent<FallDamage2>(); //Get
              the FallDamage script.
73          CollisionExcavator script4 =
              gameObject.GetComponent<CollisionExcavator>(); //Get the Collision
              Excavator script.
74          CollisionPlatform script5 =
              gameObject.GetComponent<CollisionPlatform>(); //Get the Collision
              Platform script.
75          finalInfo.SetActive(true); //Set the UI displaying all the info to
              active.
76          script.enabled = false; //Disable the PointSystem script. We'll get
              no more point updates.
77          script2.enabled = false; //Disable the Health2 script. We'll get no
              more health updates.
78          script3.enabled = false; //Disable the FallDamage script. We'll get
```

```
            no more damage for falls (even though we wouldn't get none because
            Health2 is disabled).
79          script4.enabled = false; //Disable the CollisionExcavator script.
            Although our player will still be able to collide with it, we won't
             get damage.
80          script5.enabled = false; //Disable the Collision Platform script.
81          fpsController.enabled = false; //Disable the controllers!!! This way
            our player will stay still and we won't be able to move it.
82          lifeUI.SetActive(false); //Set some other UI's and text's to
            unactive.
83          timeUI.SetActive(false);
84          pointsUI.SetActive(false);
85          reticule.SetActive(false);
86          liniaOn.SetActive(false);
87          cascOn.SetActive(false);
88          earmuuffsOn.SetActive(false);
89          arnesOn.SetActive(false);
90          fenceDoorText.SetActive(false);
91          cPRText.SetActive(false);
92      }
93
94
95  }
96
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5
6  //THIS SCRIPT WILL ALLOW US TO BUILD THE WALLS. THIS SCRIPT IS
7  //REPEATED IN A VERY SIMILAR WAY FOR ALL THE BLOCKS TO BE BUILT.
8  //THE ONLY THING  THAT CHANGES BETWEEN SCRIPTS IS THE TAG OF
9  //THE WALL.
10
11 public class BuildWall : NetworkBehaviour {
12
13     public Transform cameraTransform; //Camera for the transform.
14
15
16     Renderer[] brickRender; //Array of Renderers called "brickRenderer".
17     GameObject[] wall; //Array of GameObjects called "wall".
18
19
20     // Use this for initialization
21     void Start () {
22
23     }
24
25     // Update is called once per frame
26     void Update () {
27         RaycastHit hit;
28         Vector3 rayPos = cameraTransform.position + cameraTransform.forward *
               1f;
29         Debug.DrawRay(rayPos, cameraTransform.forward * 5f, Color.green,
             0.5f);
30
31         if (Physics.Raycast(rayPos, cameraTransform.forward, out hit, 5f))
32         {
33             if(hit.transform.tag == "WallDef") //If our raycast hits a
                 gameObject tagged as "WallDef".
34             {
35                 Debug.Log("Build");
36                 PickUpBoxRoof script = gameObject.GetComponent<PickUpBoxRoof>
                     (); //Get the script PickUpBoxRoof attatched to this
                     gameObject.
37                 if (script.canBuild) //If the variable canBuild from
                     PickUpBoxRoof is true...
38                 {
39                     if (Input.GetMouseButtonDown(0)) //If we press the left
                       mouse button...
40                     {
41                         script.clicks += 1; //Add one to the clicks variable
                       in the PickUpBoxRoof script.
42                         CmdBuildWall(); //Hey Server! Do something!
43                     }
44                 }
45             }
```

```
46          }
47      }
48
49
50      [Command]
51      void CmdBuildWall()
52      {
53          RpcBuildWall();
54      }
55
56      [ClientRpc]
57      void RpcBuildWall()
58      {
59          wall = GameObject.FindGameObjectsWithTag("WallDef"); //Same stuff
              that we did several times already.
60          brickRender = new Renderer[wall.Length];
61          for (int i = 0; i < wall.Length; i++)
62          {
63              wall[i].layer = 0;
64              brickRender[i] = wall[i].GetComponent<Renderer>();
65              brickRender[i].enabled = true;
66          }
67      }
68  }
69
```

```
1  using UnityEngine;
2  using System.Collections;
3
4  //THIS SCRIPT WILL CONTROL THE UPWARD MOVEMENT OF THE CRANE, WHEN BEING
     LOADED WITH THE BLOCKS.
5
6  public class crane_animate1 : MonoBehaviour
7  {
8      public Animator animator; //Declare the animator.
9      public AnimatorControllerParameter animParam; //(this came with the
         prefab, I don't know what it does but I didn't want to remove it).
10
11     public float rotateYaw; //Floats that will act as variables.
12     public float dolly;
13     public float hook;
14
15
16     public bool canPickBlocksUp; //Bool that will allow our player to pick up
          the blocks from the platform.
17
18     public bool isMoving = false; //Bool that will declare that the platform
         is not moving
19
20     float randomYawIncrease; //Velocity at which the previous floats will
         increase.
21     float randomDollyIncrease;
22     float randomHookIncrease;
23
24     public float temps;
25
26     public void Start()
27     {
28         randomYawIncrease = 5f;
29         randomDollyIncrease = 10f;
30         randomHookIncrease = 10f;
31         temps = 0;
32         rotateYaw = 0;//Starting Position of the crane's yaw and hook.
33         dolly = 50;
34
35     }
36
37     void Update ()
38     {
39
40         MoveUp(); //Rune the function MoveUp().
41
42     }
43
44     public void MoveUp()
45     {
46
47
48             temps += Time.deltaTime; //Define the float temps. It will add
```

```
            Time.deltaTime every frame.
49
50
51          rotateYaw += Time.deltaTime * randomYawIncrease; //The rotation
              of the yaw will be equal to the time * velocity.
52          dolly += Time.deltaTime * randomDollyIncrease; //The rotation of
              the dolly will be equal to the time * velocity.
53          hook = (temps * (-randomHookIncrease) + 86); //The rotation of
              the hook will be equal to the time * (-velocity)
54          //because it's going down, and + 86 because this way it is at
              ground level
55
56          if (rotateYaw >= 60)
57          {
58              rotateYaw = 60; //If yaw is at 60 degrees stop.
59          }
60
61          if (dolly >= 90)
62          {
63              dolly = 90; //If dolly is at height 90 stop.
64          }
65
66          if (hook <= 10)
67          {
68              hook = 10; //If hook is at 10 then
69
70              if (rotateYaw == 60)
71              {
72                  hook = -((temps * (-randomHookIncrease)) + 100) /
                    2.0f; //Take down the hook.
73                  if (hook >= 43.8f) //When the platform is touching the
                    top floor...
74                  {
75                      canPickBlocksUp = true; //Our player will be able to
                    pick up the blocks.
76                      hook = 43.8f; //The hook will stay in position.
77                      temps = 0; //Temps will be reset.
78                      isMoving = false; //The platform won't be moving, so
                    our player won't get hurt by it.
79                      this.enabled = false; //Disable this script to let
                    the "Crane" script controll the movement.
80                  }
81              }
82
83          }
84
85
86          animator.SetFloat("Rotate_YAW", Mathf.Abs(rotateYaw) % 360);//
              animator values.
87          animator.SetFloat("dolly", dolly);
88          animator.SetFloat("hook", hook);
89
```

```
90      }
91
92
93
94
95  }
96
```

```
1  using UnityEngine;
2  using System.Collections;
3
4  //THIS SCRIPT WILL CONTROLL THE CRANE GOING DOWN.
5
6  public class Crane : MonoBehaviour
7  {
8      //Same paranmeters as in crane_animate1.
9      public Animator animator;
10     public AnimatorControllerParameter animParam;
11
12     public float rotateYaw;
13     public float dolly;
14     public float hook;
15
16     public bool isMoving = false;
17
18
19
20     float randomYawIncrease;
21     float randomDollyIncrease;
22     float randomHookIncrease;
23
24     //2 floats to control the time here.
25     public float temps1;
26     public float temps2;
27
28     public void Start()
29     {
30         randomYawIncrease = 6f;//Velocities
31         randomDollyIncrease = 10f;
32         randomHookIncrease = 10f;
33         rotateYaw = 60;//Initial Positions
34         dolly = 90;
35         temps1 = 0;
36     }
37
38     void Update()
39     {
40         GoDown();
41     }
42
43     void GoDown()
44     {
45         temps1 += Time.deltaTime; //Define temps1.
46
47
48
49         hook = (temps1 * (-randomHookIncrease) + 43.8f); //Move up the hook.
50
51
52
53         if (hook <= 10) //When the hook gets to 10...
```

```
54          {
55              hook = 10;
56              dolly -= Time.deltaTime * randomDollyIncrease; //Start moving
                  dolly.
57              rotateYaw -= Time.deltaTime * randomYawIncrease; //Start moving
                  yaw.
58
59              if (dolly <= 50) //When dolly hits 50...
60              {
61                  temps2 += Time.deltaTime; //Declare temps2...
62                  dolly = 50;
63                  hook += (temps2 * randomHookIncrease); //Get hook down.
64
65                  if (rotateYaw <= 0) //When yaw is at 0 degrees.
66                  {
67                      rotateYaw = 0;
68                      if (hook >= 86) //If hook is larger than 86...
69                      {
70                          hook = 86; //Hook equals 86.
71                          temps1 = 0; //Reset temps1 and temps2
72                          temps2 = 0;
73                          isMoving = false; //We cannot get hurt because it's
                        not moving.
74                          this.enabled = false; //Disable this script so that
                        "crane_animate1" can control it.
75
76                      }
77                  }
78              }
79          }
80
81          animator.SetFloat("Rotate_YAW", Mathf.Abs(rotateYaw) % 360); //
              Animator parameters.
82          animator.SetFloat("dolly", dolly);
83          animator.SetFloat("hook", hook);
84      }
85
86
87  }
88
89
90
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //THIS SCRIPT WILL ALLOW US TO DRAW PATHS TO BE FOLLOWED BY OUR AI'S.
6  //Check the following YouTube video for a better explanation:
7  //https://www.youtube.com/watch?v=1aBjTa3xQzE&t=1959s
8
9  public class EditorPathScript : MonoBehaviour {
10
11     public Color rayColor = Color.white;
12     public List<Transform> path_objs = new List<Transform>();
13     Transform[] theArray;
14
15     void OnDrawGizmos()
16     {
17         Gizmos.color = rayColor;
18         theArray = GetComponentsInChildren<Transform>();
19         path_objs.Clear();
20
21         foreach (Transform path_obj in theArray)
22         {
23             if (path_obj != this.transform)
24             {
25                 path_objs.Add(path_obj);
26             }
27         }
28
29         for (int i = 0; i < path_objs.Count; i++)
30         {
31             Vector3 position = path_objs[i].position;
32             if (i > 0)
33             {
34                 Vector3 previous = path_objs[i - 1].position;
35                 Gizmos.DrawLine(previous, position);
36                 Gizmos.DrawWireSphere(position, 0.3f);
37             }
38         }
39     }
40
41 }
42
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //THIS SCRIPT WILL CONTROL THE MOVEMENT OF THE EXCAVATOR
6
7  public class MoveOnPathScript : MonoBehaviour {
8
9      public EditorPathScript PathToFollow; //Public field called Editor Path
         Script where we'll drag the script in the inspector.
10
11     public int CurrentWayPointID = 0; //Current point in the path.
12     public float speed; //Speed of the excavator.
13     private float reachDistance = 2.0f; //Distance at which we will pass to
         the next point.
14     public float rotationSpeed = 2.0f; //Rotation speed.
15
16
17     Vector3 last_position;
18     Vector3 current_position;
19
20
21     // Use this for initialization
22     void Start () {
23
24         last_position = transform.position;
25     }
26
27     // Update is called once per frame
28     void Update () {
29
30
31         float distance = Vector3.Distance(PathToFollow.path_objs
             [CurrentWayPointID].position, transform.position); //We'll call
             distance to the
32         //distance between the point the excavator is going to and the
             positionof the excavator.
33         transform.position = Vector3.MoveTowards(transform.position,
             PathToFollow.path_objs[CurrentWayPointID].position, Time.deltaTime
             * speed);
34         //Move the excavator from its position to the next point position at
             a certain velocity.
35         var lookPos = PathToFollow.path_objs[CurrentWayPointID].position -
             transform.position; //Look towards the next point.
36         lookPos.y = 0;
37         var rotation = Quaternion.LookRotation(lookPos); //Rotate with
             lookPos.
38         transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
             Time.deltaTime * rotationSpeed); //Make the rotation smooth.
39
40         if (distance <= reachDistance)
41         {
42             CurrentWayPointID++; //Change point when reached the minimum
```

```
                distance.
43          }
44
45          if (CurrentWayPointID >= PathToFollow.path_objs.Count)
46          {
47              CurrentWayPointID = 0; //Go back to point 0 if the path is over.
48          }
49
50
51      }
52  }
53
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //THIS SCRIPT WILL CONTROL THE BEHAVIOUR OF THE A.I. BOT PERFORMING SOME WORK
6  //ON THE FIRST FLOOR. ITS MOVEMENT IS BASED ON THE SAME PRINCIPLE AS THE
7  //EXCAVATOR, SO ONLY THE NEW PARTS WILL BE COMENTED.
8
9  public class MoveOnPathScript2 : MonoBehaviour
10 {
11
12     public EditorPathScript PathToFollow;
13
14     public int CurrentWayPointID = 0;
15     public float speed;
16     private float reachDistance = 1.0f;
17     public float rotationSpeed = 5.0f;
18
19     public GameObject particles; //Public GameObject called "particles".
20
21
22     CapsuleCollider capsule; //A capsule coolider called "capsule"
23
24     Vector3 last_position;
25     Vector3 current_position;
26
27     public bool crouch = false; //Public bool to tell if it is crouching or
         not.
28
29     Animator anim; //Animator called "anim".
30     // Use this for initialization
31     void Start()
32     {
33         capsule = gameObject.GetComponent<CapsuleCollider>(); //Define the
             capsule as the CapsuleCollider component attached to the A.I.bot.
34         anim = GetComponent<Animator>(); //Same with anim.
35         last_position = transform.position;
36         particles.SetActive(false); //Make the particles invisible.
37     }
38
39     // Update is called once per frame
40     void Update()
41     {
42
43         anim.SetBool("IsWalking", true); //Make the AI bot walk.
44         float distance = Vector3.Distance(PathToFollow.path_objs
             [CurrentWayPointID].position, transform.position);
45         transform.position = Vector3.MoveTowards(transform.position,
             PathToFollow.path_objs[CurrentWayPointID].position, Time.deltaTime
             * speed);
46
47         var lookPos = PathToFollow.path_objs[CurrentWayPointID].position -
             transform.position;
```

```
48          lookPos.y = 0;
49          var rotation = Quaternion.LookRotation(lookPos);
50          transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
              Time.deltaTime * rotationSpeed);
51
52          if (distance <= reachDistance)
53          {
54              CurrentWayPointID++;
55          }
56
57          if (CurrentWayPointID >= PathToFollow.path_objs.Count)
58          {
59              CurrentWayPointID = 10; //When reaching the end of his path,
                  stop.
60              anim.SetBool("IsWalking", false); //Stop the walikng animation.
61
62              crouch = true; //Crouch is true.
63          }
64
65          anim.SetBool("IsCrouching", true); //Play the crouched animation.
66
67
68
69          if (crouch) //If it is crouching
70          {
71              particles.SetActive(true); //Make the particles visible (the bot
                  has started his work).
72              gameObject.GetComponent<AudioSource>().enabled = true; //Play the
                   AudioSource attached to the bot with the work sound.
73              capsule.height = 1.8f; //Change capsule height and center. (I'm
                  quite sure it's more or less the same).
74              capsule.center = new Vector3(0, 0.8f, 0);
75
76          }
77      }
78  }
79
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  //THIS SCRIPT CONTROLS THE MOVEMENT OF THE OTHER A.I.BOT.
6  //THE SCRIPT IS REALLY SIMILLAR TO THE TWO "MOVEONPATHSCRIPT"
7  //COMENTED BEFORE.
8
9  public class MoveOnPathScript3 : MonoBehaviour
10 {
11
12     public EditorPathScript PathToFollow;
13
14     public int CurrentWayPointID = 0;
15     public float speed;
16     private float reachDistance = 1.0f;
17     public float rotationSpeed = 5.0f;
18
19     CapsuleCollider capsule;
20
21     Vector3 last_position;
22     Vector3 current_position;
23
24
25
26     Animator anim;
27     // Use this for initialization
28     void Start()
29     {
30         capsule = gameObject.GetComponent<CapsuleCollider>();
31         anim = GetComponent<Animator>();
32         last_position = transform.position;
33     }
34
35     // Update is called once per frame
36     void Update()
37     {
38
39         anim.SetBool("IsWalkingWheel", true);
40         float distance = Vector3.Distance(PathToFollow.path_objs
             [CurrentWayPointID].position, transform.position);
41         transform.position = Vector3.MoveTowards(transform.position,
             PathToFollow.path_objs[CurrentWayPointID].position, Time.deltaTime
             * speed);
42
43         var lookPos = PathToFollow.path_objs[CurrentWayPointID].position -
             transform.position;
44         lookPos.y = 0;
45         var rotation = Quaternion.LookRotation(lookPos);
46         transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
             Time.deltaTime * rotationSpeed);
47
48         if (distance <= reachDistance)
```

```
49          {
50              CurrentWayPointID++;
51          }
52
53          if (CurrentWayPointID >= PathToFollow.path_objs.Count)
54          {
55              CurrentWayPointID = 0;
56
57          }
58      }
59  }
60
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Audio;
5
6  //THIS SCRIP WILL MAKE THE PLAYERS GET HURT IF THEY GET TOO CLOSE TO THE
     SPARKS.
7
8  public class TriggerEspurna : MonoBehaviour {
9
10     public AudioClip clips; //Sound of the work being performed.
11     // Use this for initialization
12     void Start () {
13
14     }
15
16     // Update is called once per frame
17     void Update () {
18
19     }
20
21     void OnTriggerStay(Collider other) //If a collider stays inside the
         trigger...
22     {
23         MoveOnPathScript2 path = gameObject.GetComponent<MoveOnPathScript2>
             (); //Get the MoveOnPathScript2 script attached to this gameObject
             (AI).
24         {
25             if (other.transform.tag == "Player" && path.crouch) //If the
                 collider inside the trigger is a player and the AI is crouching
26                 //(so it has already started to work)...
27             {
28                 Health2 player = other.GetComponent<Health2>(); //Get the
                     Health2 script attached to the object that has collided
                     with the trigger
29                 //(The Player)
30                 PlaySound(); //Play the sound of damage.
31                 if (player != null) //If health is not null
32                 {
33                     player.TakeDamage(0.5f); //Take a damage of 0.5 for every
                       frame the player is inside the trigger.
34                 }
35             }
36         }
37      }
38
39     void PlaySound()
40     {
41         AudioSource source = gameObject.AddComponent<AudioSource>(); //Add an
             audio source called source.
42         source.clip = clips; //The clip that the source will play is called
             clips.
43         source.Play(); //Play the source.
```

```
44          Destroy(source, clips.length); //Destroy the cource when "clips" has ↵
              stopped playing.
45      }
46  }
47
```