Universidad Autónoma de Chihuahua

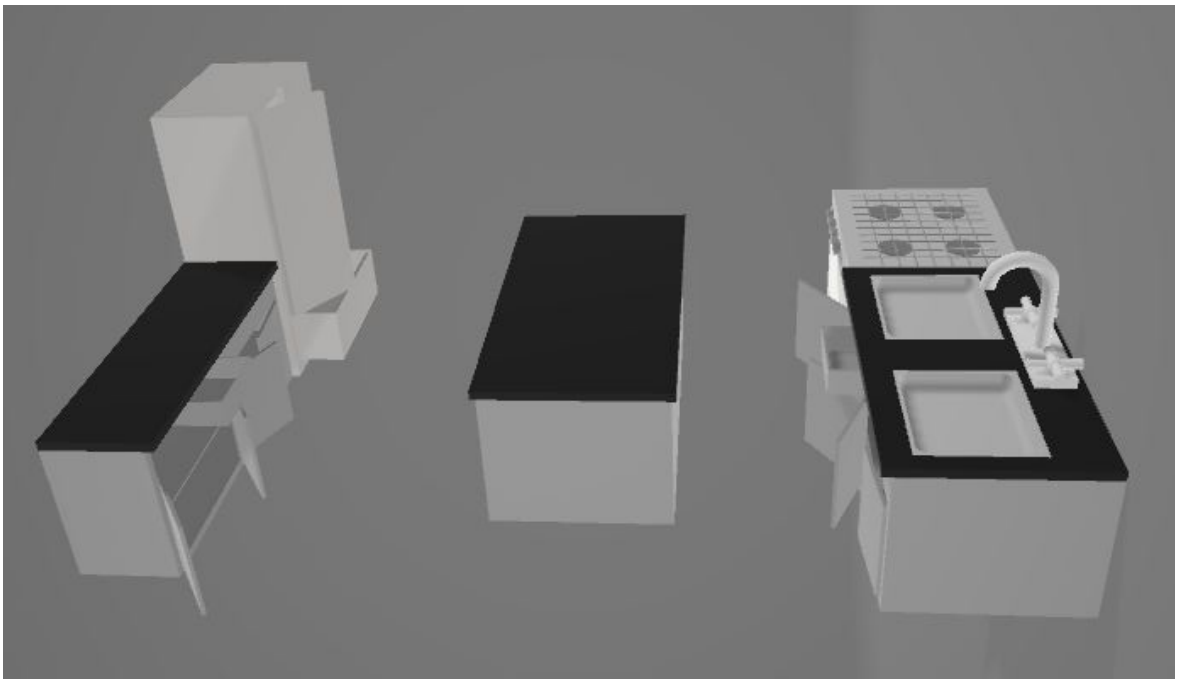
Desarrollo Basado en Plataformas

Javier Eduardo Camarillo Polanco - Ingeniería en Ciencias de la Computación

Reporte Técnico de "Prototipo de Comunicación Serial integrando tecnologías Arduino y Unity"

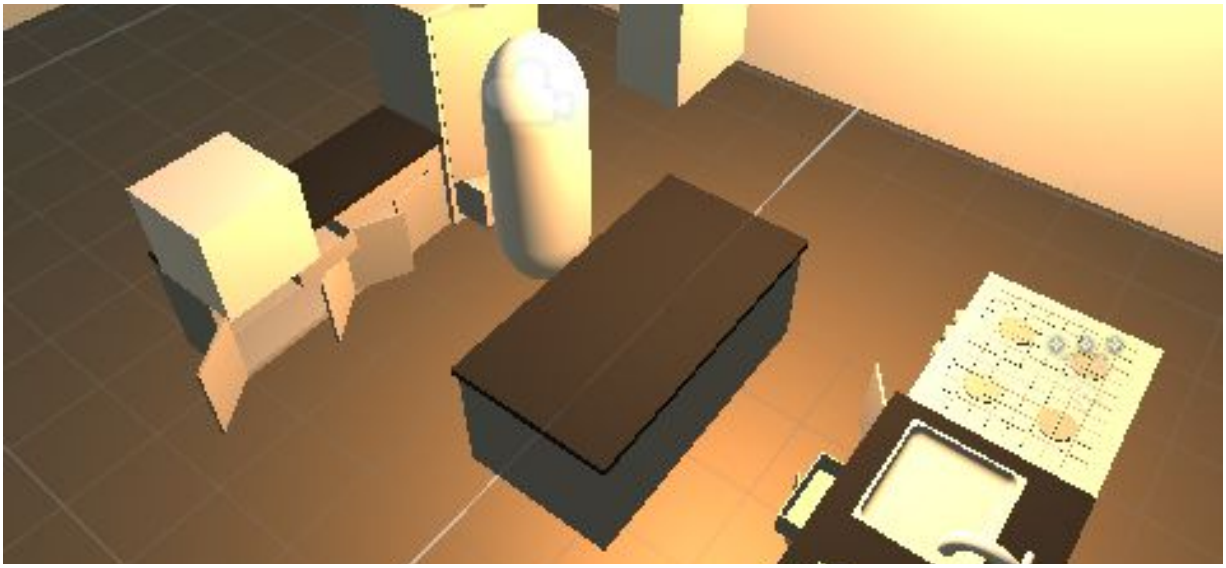
Antes de implementar la funcionalidad es útil obtener los objetos necesarios para generar la interfaz del prototipo.

Para esto se utilizó un programa gratuito de Autodesk llamado "Fusion 360" , el cual permite diseño tridimensional de forma sencilla, haciendo operaciones geométricas booleanas con figuras como Cubos, esferas, tuberías, etc. Como resultado obtenemos un diseño así.

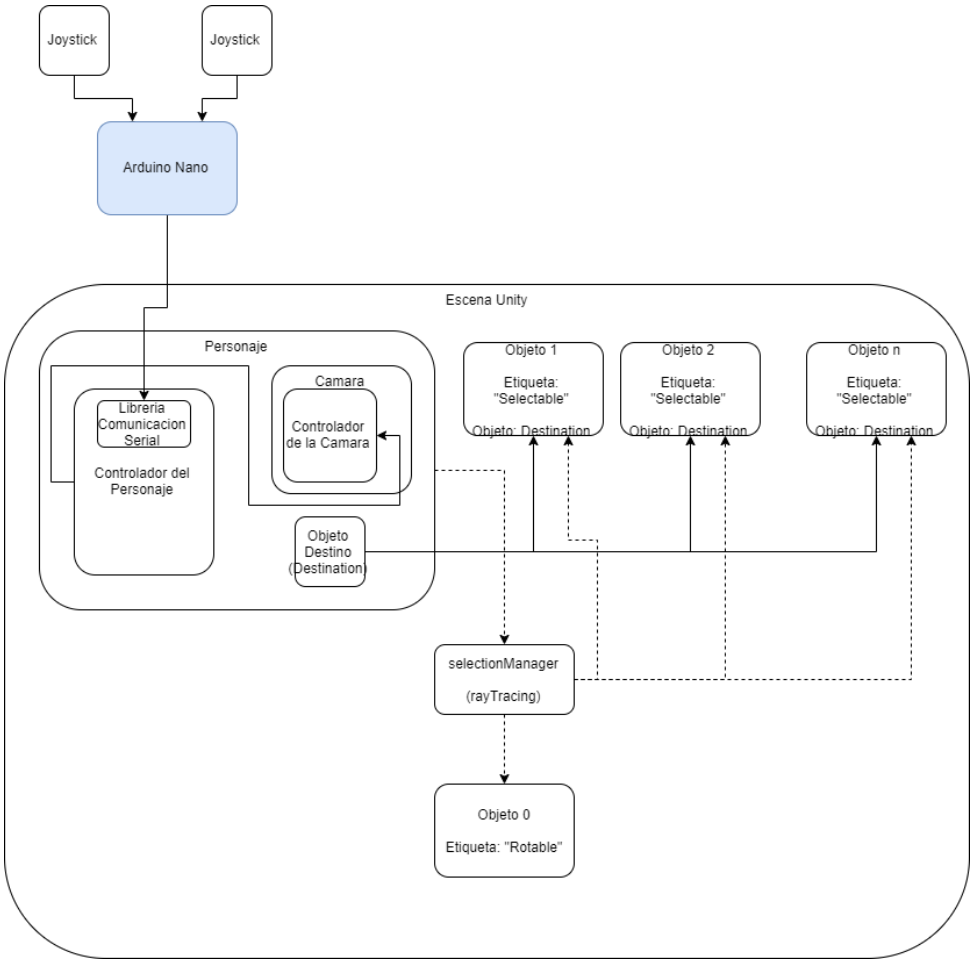


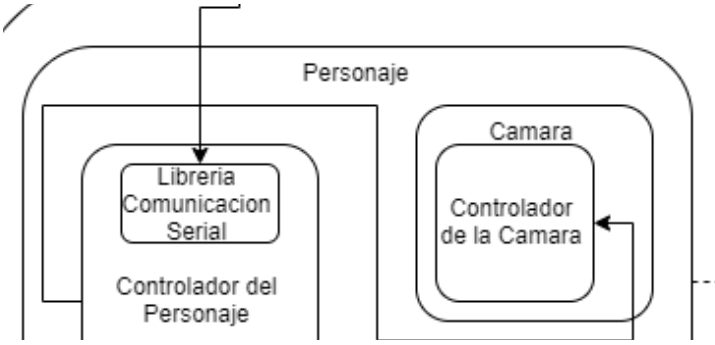
Este diseño corresponde a los artefactos visuales del prototipo, el programa Fusion 360 permite exportar los componentes separados en un sólo archivo con formato ".fbx".

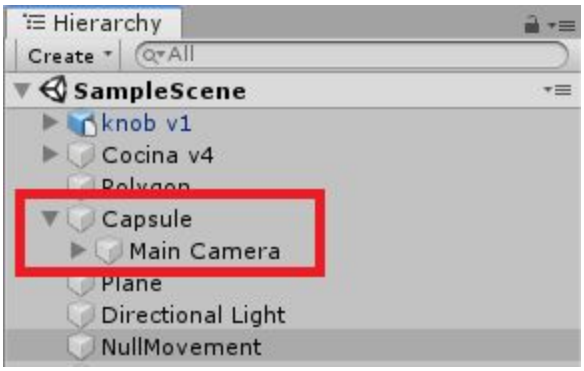
En la escena se procede a insertar una cápsula para representar al personaje, al ser una simulación en primera persona, la cámara se coloca por dentro de la cápsula, por defecto, los elementos físicos en Unity carecen de textura en su interior, por lo que se puede ver a través de los mismos.



Teniendo esto podemos visualizar la cámara y el personaje dentro del diagrama de componentes.





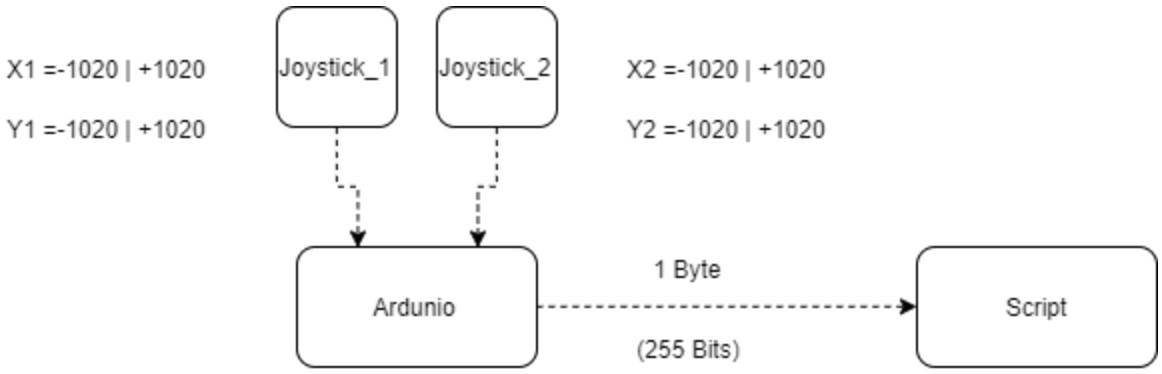


Esta anidación se genera al colocar a la cámara dentro de la cápsula en la pestaña de jerarquías de unity. A esto se le añade un Script de C# llamado "characterController.cs" para definir el movimiento y la comunicación Serial. Asimismo la Cámara cuenta con su propio script "camMouseLook.cs".

```
4   using System.IO.Ports;
5   using System;
6
7   public class characterController : MonoBehaviour
8   {
9       public float speed = 10.0f;
10      SerialPort sp = new SerialPort("COM5", 9600);
11
```

En el encabezado añadimos:

Ahora para comunicarnos con el arduino se podrían aplicar diferentes soluciones. Considerando que se pueden enviar 1 bye a la vez o 255 bits surgen los siguientes escenarios.



Una
Coneccion

Pos.
Joystick #1

1	2	3
4	0	5
6	7	8

Pos.
Joystick #2

11	12	13
14	10	15
16	17	18

n - Conecciones

-1	-.75	-.5	-.25	0	.25	.5	.75	1
1	2	3	4	5	6	7	8	9

		-100%	...	0%	...	100%
Joystick 1	X	10	...	50	...	90
	Y	1	...	5	...	9
Joystick 2	X	10	...	50	...	90
	Y	1	...	5	...	9

55 -> X:0%
Y:0%
92 -> X:100%

Con una sola coneccion es posible representar fácilmente valores discretos de la forma como se ilustra en el esquema del lado izquierdo. Un 3, por ejemplo, representa que el Joystick #1 se encuentre en los valores positivos del eje X y Y, definidos por un valor mayor a un rango arbitrario definido por una variable, por ejemplo si el joystick supera 100 unidades, se define por positivo en el eje que lo traspase y menor a 100 se considera negativo respectivamente.

Por otra parte contemplando n conecciones es fácil representar valores con mayor continuidad, por ejemplo, si se desarrolla un dispositivo que rastrea la cabeza y otro que rastrea el desplazamiento de la persona como una caminadora omnidireccional o un control simple, habría que utilizar 2 dispositivos seriales.

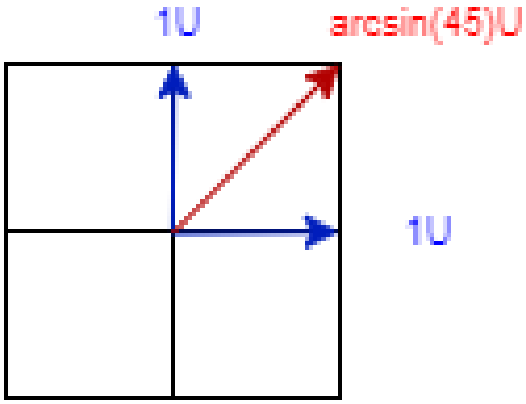
En este caso el mapa genera valores del 0 al 99, donde el 5 representaría un 0 en la escala del -1 al 1. Un 6 equivale a un 25%, un 7 un 50% y un 8 un 75% respectivamente.

Con esto se puede fácilmente separar el dígito izquierdo del derecho en un script, sin embargo para este programa solo se contempla la solución con un dispositivo arduino. Ya que al probarse, cumple con el propósito.

```
62      int moveSwitch(int key, float d, float sd, Vector3 md){
63
64          switch(key){
65              case 1: md.Set( d , 0 , 0); md = transform.TransformDirection(md); controller.Move(md);  break;
66              case 2: md.Set( -d , 0 , 0); md = transform.TransformDirection(md); controller.Move(md); break;
67              case 3: md.Set( 0 , 0 , d); md = transform.TransformDirection(md); controller.Move(md); break;
68              case 4: md.Set( 0 , 0 , -d); md = transform.TransformDirection(md); controller.Move(md); break;
69              case 5: md.Set(sd,0,sd); md = transform.TransformDirection(md); controller.Move(md); break;
70              case 6: md.Set(sd,0,sd*-1); md = transform.TransformDirection(md);controller.Move(md); break;
71              case 7: md.Set(sd*-1,0,sd*-1); md = transform.TransformDirection(md);controller.Move(md); break;
72              case 8: md.Set(sd*-1,0,sd); md = transform.TransformDirection(md);controller.Move(md); break;
73              }
74              return key;
75          }
76
77  }
```

Cada uno de estos valores discretos se interpretan con un switch como se muestra en la imagen anterior en donde al vector de 3 dimensiones se le añade la distancia a mover por iteración, el argumento d.

Sin embargo otro argumento se inserta en la función, el argumento sd, que corresponde a la distancia en diagonal. Ya que en muchos videojuegos se comete el error de asignar movimiento de 1 en X y 1 en Y para el desplazamiento diagonal.



Sin embargo este desplazamiento no es de una unidad sino del seno inverso de 45 grados (Aprox 1.4 unidades). Por lo que se genera una distancia alternativa de sin(45)*distancia de esta manera el personaje no se desplaza una distancia mayor o a una velocidad mayor que en un eje recto respecto a su posición actual.

```
33      void Update()
34      {
35              float dis = speed * Time.deltaTime;
36              float sin45 = (float) Math.Sin(45);
37              float sdis = dis*sin45;
38              //if( Input.GetKeyDown("escape") ) Cursor.lockState = CursorLockMode.None;
39              moveDir *= speed;
40              if(key<10) aux = moveSwitch(key, dis, sdis, moveDir);
41              else moveSwitch(aux, dis, sdis, moveDir);
42
```

Interaccion entre Scripts

```
 9      public float speed = 10.0f;
10      SerialPort sp = new SerialPort("COM5", 9600);
11
12      int key;
13      int aux;
14
15      GameObject cam;
16
17      private CharacterController controller;
18      private Vector3 moveDir = Vector3.zero;
19      Vector3 lastPosition;
20      Vector3 lastRotation;
```

Los scripts se pueden comunicar entre sí mediante la obtención del Script como un componente del objeto con el que se desea interactuar. En este caso se genera un objeto de nombre cam, para representar a la cámara principal.

```
43          if(sp.IsOpen){
44              try{
45                  int input = sp.ReadByte();
46                  if(input>=0 && input<255){
47                  key = input;
48                  }else key = 0;
49
50                  if(key>=10){
51                  cam = GameObject.FindGameObjectWithTag("MainCamera");
52                  camMouseLook cML = cam.GetComponent<camMouseLook>();
53                  cML.key=key;
54                  0
```

Despues el mismo script se obtiene del objeto del juego, el cual contiene una llave publica. Que es lo mismo en funcion al switch del personaje principal, con la diferencia de ser actualizado dados valores mayores a 10.

```
54              */
55      }
56          public void rightSwitch(int key, float d, float s){
57
58              switch(key){
59                  /*
60                  case 1: move(d,0); break;
61                  //case 1: transform.Translate(d,0,0); break;
62                  case 2: transform.Translate(d*-1,0,0); break;
63                  case 3: transform.Translate(0,0,d); break;
64                  case 4: transform.Translate(0,0,d*-1); break;
65                  case 5: transform.Translate(sd,0,sd); break;
66                  case 6: transform.Translate(sd,0,sd*-1); break;
67                  case 7: transform.Translate(sd*-1,0,sd*-1); break;
68                  case 8: transform.Translate(sd*-1,0,sd); break;*/
69                  case 11: move(0,d*-1); break;
70                  case 12: move(0,d); break;
71                  case 13: move(d,0); break;
72                  case 14: move(d*-1,0); break;
73                  case 15: move(s,s*-1); break;
74                  case 16: move(s*-1,s*-1); break;
75                  case 17: move(s*-1, s); break;
76                  case 18: move(s, s); break;
77              }
78          }
79
80          public void move(float x, float y){
81
82              var mouseDelta = new Vector2(x, y);
83
84              mouseDelta = Vector2.Scale(mouseDelta, new Vector2(sensitivity * smoothing, sensitivity * smoothing));
85              smoothV.x = Mathf.Lerp(smoothV.x, mouseDelta.x, 1f / smoothing);
86              smoothV.y = Mathf.Lerp(smoothV.y, mouseDelta.y, 1f / smoothing);
87              mouseLook += smoothV;
88              transform.localRotation = Quaternion.AngleAxis(-mouseLook.y, Vector3.right);
89              character.transform.localRotation = Quaternion.AngleAxis(mouseLook.x, character.transform.up);
90
91          }
92
93      }
```

Ray Tracing

Para atender el problema de interactuar con los objetos desde un crosshair del personaje existe un método llamado Raytracing, el cual consiste en disparar un rayo desde un punto dado y a partir del cual se pueden encontrar objetos dependiendo de si la representación del rayo coincide con los límites del objeto.

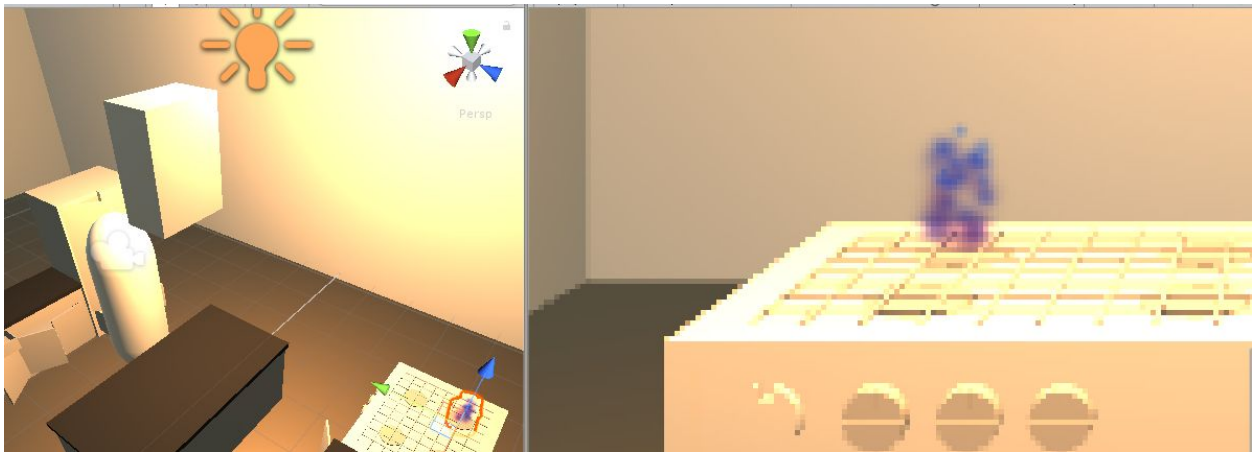
```
59          //var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
60          var ray = Camera.main.ScreenPointToRay( new Vector3(Screen.width/2,Screen.height/2,0) );
61          Debug.Log(Input.mousePosition);
```

Una vez encontrado el objeto se llama a un Script que cada objeto modificable contiene, el cual, de ser seleccionable (dada su etiqueta “Selectable”), se activa enviando el objeto a la posición de un objeto vacío enfrente del personaje llamado “Destination” el cual, mientras no se suelte la tecla presionada. Se queda suspendido enfrente del personaje.

```
64
65      RaycastHit hit;
66      if (Physics.Raycast(ray, out hit)){
67          var selection = hit.transform;
68          //Levantar seleccionable
69          if(selection.CompareTag(selectableTag)){
70
71              var selectionRenderer = selection.GetComponent<Renderer>();
72              if(selectionRenderer!=null)
73              {
74                  pickUp pu = selectionRenderer.GetComponent<pickUp>();
75
76
77
78                  if(key==1){
79                      pu.pick();
80
81                  }
82
83                  Debug.Log("Selected " + i);
84
85
86              }
87
88          _selection = selection;
89
90      }
```

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class pickUp : MonoBehaviour
6   {
7
8       public Transform dest;
9
10      public void pick(){
11          GetComponent<BoxCollider>().enabled = false;
12          GetComponent<Rigidbody>().useGravity = false;
13          this.transform.position = dest.position;
14          this.transform.parent = GameObject.Find("Destination").transform;
15      }
16
17      public void release(){
18          GetComponent<BoxCollider>().enabled = true;
19          this.transform.parent = null;
20          GetComponent<Rigidbody>().useGravity = true;
21
22      }
23
24      public void rotate(int dir){
25          int mult = 2;
26          this.transform.Rotate(0,0,mult*dir,Space.Self);
27      }
```

De igual manera podemos utilizar el mismo proceso para definir otro tipo de objeto, esta vez con la etiqueta “Rotable” el cual al ser seleccionado, se puede rotar en su misma posición, en lugar de ser trasladado al frente del personaje para ser desplazado.



Por último un ejemplo de como la variable de rotación se modifica con una funcion senoidal absoluta para ciclar la intensidad de la flama simulada por el sistema de partículas.

```
22      void Update()
23      {
24          Debug.Log("em: " + emissionVar);
25
26          ParticleSystem ps = GetComponent<ParticleSystem>();
27          //GameObject knob1 = GameObject.Find("knob1");
28          //rot knobRot = knob1.GetComponent<rot>;
29          //double sinx = knob1.rot;
30
31          //Debug.Log("knob: " + knob1);
32
33          //ps.emission.rate = 0.0f;
34          //particles.emission.enabled = true;
35          float sinEm = (float) Math.Abs(Math.Sin(emissionVar));
36          var em = ps.emission;
37          if(sinEm*10 < 1) sinEm = 0;
38
39          em.rateOverTime = sinEm*10;
40      }
41  }
```