**ROTOPADS**

**Scripts desarrollados: PadGetPos, BeatDetector, Texter, UpDownFlag, StickTracker, FichaRotopad, DiscoRotopad, DiscoFeedback, AlgorithmNumber, RandomButtonBang y ResetButtonBang.**

**Materiales: tech\_pedestal\_COL, MatMesaChina, MatHandOKFuc, MatPaliEmi, LuzCyan, MatFichaNar(Nar,Ver,Roj,Azu), MatMonolitChino, LuzFucsia.**

**Estructura:**

**Objeto principal: PedestalRotopads**

Dentro tiene el script OSC principal que va a ser usado para mandar todos los mensajes.

In port 6970, IP 127.0.0.1, Out Port 6170.

**RotoPad(numerados 1 a 8, 8 en total)**

Ningún componente

**PadRotor**

Es el modelo de tech pedestal, con mesh renderer, rigidbody kinematic, interactable con circular drive. Circular drive en eje Z. No tiene asignado ni child collider ni linear mapping, por lo que se debe asignar solo cuando se hoverea, pero se le podría asignar lo que ya sabemos. Tiene tambien un box collider. Script **PadGetPos** que obtiene la posición del pad base y lo sigue cuando el base se mueve, variable publica asignada al PadBase. Script **BeatDetector** importantantisimo, es el que detecta colisiones con los palillos y envía la frecuencia de nota segun el angulo de rotacion del pad. Variables publicas asignadas al OSC del objeto principal y el circular drive al script CircularDrive de si mismo. Por ultimo un animator con la animacion de que el pad se hunde y se ilumina un poco. Material tech\_pedestal\_COL.

**PadBase**

Es el mismo modelo que el rotor, pero achatado. Sirve para ser el throwable del que se puede agarrar el pad entero y desplazarlo. Componentes tiene mesh renderer, rigidbody, box bollider, interactable, throwable (con restore original parent) y un velocity estimator al final. Mismo material que PadRotor.

**Nota**

Texto 3D que indica la nota que va a reproducir el pad segun el angulo en que haya sido rotado. Tiene un script **Texter** que es el que calcula qué mostrar segun la rotacion y va cambiando tanto el color como la opacidad del texto segun la frecuencia en la que se encuentra. Tiene variables publicas Texto nota con el texto 3D, Rotor con el objeto PadRotor y Circular Drive con el script CircularDrice del PadRotor.

**FondoBuda**

Un cilindro chato con material de un buda a simple modo cosmético para que, en caso de ver el pad desde abajo, no se vea hueco. No tiene otra funcion.

**MesaChina (una)**

Modelo prefabeado con meshrenderer y collider, material MatMesaChina y nada más, sirve de apoyo a los otros objetos.

**HandOK**

Modelo de mano haciendo OK que sirve para sostener los palillos. Tiene dos objetos que son los que conforman el modelo completo, los cuales tienen mesh renderer y collider y una animacion que destella. Material MatHandOkFuc.

**Espalda**

Cubo adaptado a las dimensiones de la espalda de la mesa para tapar y que no se vea el hueco.

**TeleportPoint(inactivo)**

Al momento inactivo, lo dejo por las dudas de que haga falta tenerlo para amenizar la version final.

**PalilloEmisor(2 iguales)**

Es importante que este objeto se llame así pq el script **BeatDetector** lo chequea. Es el palillo que va a colisionar con los pads y los va a hacer sonar. No hay otro objeto en el entorno que haga sonar los pads. Es una capsula simple 3D, con un box collider, rigidbody no kinematic con gravity, throwable con restore original parenting e interactable comun. Tiene un script **UpDownFlag** que chequea si el palillo está subiendo o bajando y solo activa el rotopad si el palillo esta bajando, con variables públicas previousY, currentY y Direction que no estan preasignadas y solo son publicas para que las acceda otro script. Por ultimo un velocity estimator y de material usa el MatPaliEmi. Están ubicados en el entorno dentro de la mano que hace OK, pero son objetos independientes entre si tanto la mano como los palillos.

**Colisionador**

Objeto hijo que sirve para que el beat detector lo detecte como colisión y haga lo que tiene que hacer. Tiene un rigidbody sin gravity ni kinematic, un capsule collider más grande que el mesh del palillo padre y un script **StickTracker** que lo hace seguir al palillo padre. Solo sirve para colisionar y chequear valores.

**Fichas**

Objeto padre que contiene a todas las fichas que se encuentran sobre la mesa.

**Moba(Naranja, Verde, Roja, Azul, cuatro en total)**

No tiene componentes, alberga toda la construcción de una ficha estilo Moba que controla el ratio del sinte.

**Start** Posicion de inicio para el linear mapping

**End** Posicion de final para el linear mapping

**LinearMapping**

GameObject que contiene el script linear mapping que alberga la posicion en que se encuentra la ficha momentaneamente, 0 es Start, 1 es End y un valor entre 0 y 1 para todo lo que está en el medio. También contiene el script **Ficha RotoPad** que envia por OSC el valor correspondiente al ratio del sintetizador, basandose en la posicion horizontal de la ficha en cuestión sobre la mesa. Variables públicas: Oscilator ID es el identificador de a qué oscilador debe enviarle el mensaje (Naranja = 1, Verde = 2, Roja = 3, Azul = 4), Parameter es el parámetro que envía, en este caso es 1 que es ratio (ratio = 1), Value Adjust es el ajuste de mapeo que debe hacerse para que un valor entre 0 y 1 sea util para controlar el parámetro deseado (para este caso es 64), Osc es el script OSC del objeto principal y LinearMapping que es el script LinearMapping del objeto LinearMapping nombrado anteriormente.

**Handle**

GameObject que nuclea los comportamientos de linearDrive y el modelo de laficha. Contiene un rigidboy kinematic sin gravity, un interactable comun y un linear drive (que al parecer agrega por si solo el script Interactable Hover Events que contiene más que nada listas de accions para eventos como el inicio del hover, el fin del hover y el agarre o soltado del objeto) con variables publicas de start, end y linear mapping. Esto ultimo es parecido al circular drive: start es el GameOBject start, end es el GameOBject end y linear mapping es el script linearMapping del GameObject LinearMapping del que ya hablamos. Start y End son GameObjects vacios que sirve de referencia al incio y fin del recorrido del objeto. Tiene tambien un script Haptic rack que no se bien que hace y un skeleton poser que tampoco se qué hace, pero creo que los agrega por si solo al meter el linear drive.

**Torre01**

Modelo de la torre Moba, tiene un mesh renderer y un mesh collider, usa material base MatFichaNar y LuzCyan para material de unas grietitas que tiene. LuzCyan es un material emisor que también está usado en los diamantes que iluminan el recinto.

**China(Naranja, Verde, Roja, Azul, cuatro en total)**

Es un objeto muy similar a Moba, habiendo de cuatro colores distintos. Controla el parametro level de cada oscilador y tiene los mismos objetos dentro de start, end, linearmapping, handle y el modelo. Lo unico que cambia es el modelo, que en vez de ser la torrecita Moba es una torre de la China medieval, y el parametro del script que envia los valores. Para el caso de la ficha china, el parámetro es el numero 2 que indica level y el ajuste de mapeo queda en 1.

**MonolitChino**

Monolito igual a las fichas chinas pero de gran tamaño que alberga los discos que controlan el ataque de cada oscilador y el feedback del sintetizador completo. Tiene un mesh renderer y una animacion que lo hace destellar, usa el material MatMonoliChino.

**DiscoAttack(Naranja, Verde, Rojo y Azul, cuatro en total)**

Disco hecho con un modelo de un escudo. Esta ubicado como parte del monolito chino y gira para modificar el ataque de cada oscilador, segun que color tenga. Al igual que las fichas, Naranja = 1, Verde = 2, Rojo = 3 y Azul = 4. El parámetro que modifica es el numero 3 que sería el ataque de cada oscilador y el ajuste de mapeo es el 500. Dentro tiene Mesh renderer, mesh collider, interactable comun, circular drive en eje Z y con la variable publica child collider su propio mesh collider, linear mapping el script linear mapping que está en si mismo y por ultimo un script **DiscoRotopad** que se encarga de enviar el valor por OSC. Usa material MatFichaNar. A diferencia del circular drive de los rotopads, este usa el linear mapping para enviar el valor.

**DiscoFeedback**

Objeto casi igual a los anteriores, con la salvedad de que usa otro script con menos variables de ajuste, que se llama **DiscoFeedback** y tambien envia su valor de rotacion por OSC.

**MonolitMoba**

Monolito Moba casi igual al chino, pero este contiene discos que controlan el release de cada oscilador y los ocho botones que indican el algoritmo a usar por el sintetizador.

**DiscoRelease(Naranja, Verde, Rojo, Azul y cuatro en total)**

Función y modelo idénticos a los del monolit chino, única diferencia es la del parámetro que modifican que es el 4, release y el ajuste de mapeo es 5000.

**AlgorithmButtons**

GameObject que contiene a los botones de algoritmo.

**AlgNButton(Donde N toma valores de 1 a 8, 8 objetos en total)**

Modelo sacado de un dado, es como un cubo con las aristas limadas, es el objeto boton que al ser presionado envia por OSC el algoritmo que debe usarse en el sinte. Tiene un Mesh renderer, mesh collider, interactable comun y los scripts Hover Button usando como MovingPart la transformada de si mismo (con arrastrar el objeto desde la jerarquia alcanza) el movimiento en Z -0.01 y eventos a realizar cuando está presionado y cuando se lo suelta. Luego un script ButtonEffect y por último el script **AlgorithmNumber**  con variables publicas Osc (el script OSC del pedestal rotopads) y Algorithm que se asigna manualmente entre 1 y 8 para que cada objeto dispare el numero correspondiente. EL script **AlgorithmNumber** contiene las funciones que se pondran en marcha segun la lista de eventos en el script Hover Button. Este ultimo tiene listas OnButtonUp(Hand) y OnButtonIsPressed que permite agregar comportamientos desde scripts. Esto se hace tocando el botoncito de +, seleccionando el script que contiene la funcion y eligiendo qué funcion queremos que realice. Para el caso de presionar por ejemplo, elegimos el script AlgorithmNumber del objeto Alg1Button y seleccionamos la función SendAlgNum.

**Alg**Plano con un material hecho a partir de un PNG que muestra el dibujo del algoritmo que se activará al presionarlo.

**RandomPatchButton**

Objeto que hace de botón para seleccionar una configuracion aleatoria del sintetizador. Contiene un mesh renderer desactivado, un mesh collider, un interactable, un hover button muy similar a los botones de algoritmo, con los comportamientos segun el boton es presionado o soltado, un button effect y por ultimo el script **RandomButtonBang** que contiene los metodos o funciones utilizados en las listas de hoverbutton. Envia un bang por OSC para que PD haga la randomizacion del patch.

**MovingPart**

Es la parte móvil del boton, asignada en HoverButton del objeto padre. Usa un modelo de boton comun y contiene un mesh renderer y un mesh collider.

**ResetPatchButton**

Objeto hermano del RandomPatchButton pero el comportamiento es enviar un basng que resetea el sintetizador a parámetros por defecto. El script que envía el valor se llama **ResetButtonBang**.

**GYROS**

**Scripts desarrollados: GiroscopoValues, FilterLever.**

**Materiales: sp\_texture\_mask3, Aluminums, light55x y un par más que usa el soporte.**

**Estructura**

**Objeto principal: PedestalGiroscopio**

Script OSC IN port 6969, Out Port 6169. Es solo un GameObject.

**EsferaGiroscopio**

Es un modelo de esfera decorativa que encontre en internet. Tiene mesh renderer, sphere collider, throwable con restore parent, rigidbody con gravity sin kinematic, velocity estimator, interactable y el script propio **GiroscopoValues** que posee las variables publicas Osc asignado al OSC del pedestalGiroscopio, HandleMapping asignado al script de linear mapping del objeto linear mapping del objeto que controla el volumen, y el parametro de ajuste de volumen.

**Soporte**

Tiene dentro los modelos de una pieza de unas armas espaciales. Tiene Rigidbody y box collider, solo para apoyar la esfera, no tiene otra funcion.

**Controles**

Objeto vacio para contener las palancas.

**VolLever**

Vacio para contener los objetos de la palanca

**Sphere**

Modelo de la esfera de la palanca. Tiene mesh renderer, sphere collider, interactable y linear drive.

**Cylinder**

Cilindro solo con fines estéticos.

**Start**

Objeto vacío usado para obtener su transformada y usarla como punto de partida del linear drive.

**End**

Analogo al anterior, pero para usarlo como punto de llegada del linear drive.

**LinearMapping**

Para albergar el script linearmapping y que se pueda usar para enviar valores.

**BPVolLever(BPFreqLever, LPVolLever y LPFreqLever, cuatro en total)**

En el contenido gráfico, de movimiento y lectura de valores, es igual al de volumen, pero dentro de este objeto, que en la palanca de volumen era vacío, tiene un script propio llamado **FilterLever** con las variables publicas Osc asignada al script OSC del pedestalgiroscopio, **HandleMapping** asignado al linear mapping de cada palanca que corresponda, Parameter que se rellena con el string que va a ser la direccion de llegada del OSC en PD y value adjust, para que el valor que envia sea coherente con lo que espera recibir PD.

**BataBall**

**Scripts desarrollados: BataBallToken, DrawRectangle, BataBallCubeFace, BataBallCube, BataBallSphere, BataBallHalo, BataBallDiscoRot, BataBallSpeed, BataBallSphereModifier.**

**Materiales: molten-604084\_1280, rare colors, MatRedRectangle, MatBlueRectangle, MatGreenRectangle, MatPurpleRectangle, MatBataBallSphere, MatCubeUp, MatCubeDown, HaloTextD1, HaloTexD2, free-futuristic-sci-fi-texture, sci fi rock, sci\_fi\_texture\_28, sci\_fi\_texture\_29, MatBataBallSphereMod, photos\_2017\_11\_10\_fst\_texture.**

**Estructura:**

**Objeto principal: PedestalBataBall**

Script OSC IN port 6970, Out Port 6170. Es solo un GameObject.

**Display**

Modelo y objeto con forma de tablero sci fi que sostiene la palanca que controla la velocidad de las esferas, y los rectángulos que controlan level y filtro de cada cuerpo. Material sci fi rock.

**Shaft**

Objeto vacío que contiene la palanca de velocidad de esferas.

**Handle**

Cilindro interactable que contiene dos barras, también cilindricas. Es el encargado de tomar el agarre de todo el objeto Shaft, usa los script de interactable y linear drive. Material photos\_2017\_11\_10\_fst\_texture.

**Bar (dos en total)**

Solo de estética, usan misma textura que handle.

**End** Punto para determinar final de linear drive.

**Start** Idem pero para el inicio.

**LinearMapping** Objeto contenedor del script LinearMapping.

**SphereMod**

Esfera que modifica el cuerpo de cada cubo, cuando uno de estos la atraviesa. Tiene renderer, collider y de vital importancia el RigidBody sin kinematic y sin gravity, pero con todos los ejes de movimiento restringidos, para que no se mueva al ser atravesada. Solo puede rotar. Tiene el script de comportamiento **BataBallSphereModifier.cs** y usa el material MatBataBallSphereMod.

**Balls**

GameObject que contiene el script **BataBallSpeed** que usa como variables públicas el script LinearMapping de Shaft y la variable tipo float speed, para que el resto de los objetos la tomen y usen. Contiene además los discos contra los que chocan las esferas y las esferas que rebotan.

**DiscoLU(y su análogo DiscoRU)**

Disco superior de rebote de la esfera que colisiona los cubos para generar sonido. Es casi estético, lo único que toma la esfera del mismo para rebotar es su posición, pero no colisiona ni es un rebote propio de la física de Unity, sino por script. Usa varios materiales y el del centro cambia su offset y tiling según la velocidad global, para generar un efecto dinámico. Usa el script **BataBallDiscoRot** que además usan todos los discos de este tipo, para modificar los valores del material. Materiales: Element 0: free-futuristic-sci-fi-texture, Element 1: sci fi rock, Element 2: sci\_fi\_texture\_28, Element 3: sci\_fi\_texture\_29.

**DiscoLD(y su análogo DiscoRD)**

Disco inferior de rebote de la esfera, casi igual al superior, con la diferencia de que este alberga al halo que se ve entre los dos discos. Mismo script y materiales que el superior.

**LeftHalo (dos en total, y el DiscoRD tambien tiene sus dos RightHalo)**

Cilindros que contienen una textura de partículas que se mueve y crea el efecto visual de halo. Cada uno tiene una variable pública de dirección, que está puesta en sentidos contrarios (uno tiene 1 y el otro -1) para que se vea como un halo que sale del disco superior o otro del inferior. Usa el script **BataBallHalo** y cada uno usa un material distinto, HaloTextD1 y HaloTexD2.

**SphereL(y su análogo SphereR)**

Las esferas que se mueven hacia arriba y hacia abajo, y que colisionan con los cubos. Tienen renderer, collider, usan el material MatBataBallSphere. Tiene Rigidbody kinematic y sin gravity. Tiene el script **BataBallSphere** que contiene al script OSC del pedestal, la variable pública GlobalSpeed que toma el script BataBallSpeed de Balls, el disco superior y el disco inferior.

**Cubes**

Objeto vacío que contiene los 16 cubos que pueden ponerse dentro del campo que atraviesan las esferas, para ser colisionados y producir sonidos.

**Cube (dieciséis en total)**

Son réplicas del mismo objeto. Tiene renderer y collider, y su RigidBody no tiene ni kinematic ni gravity, pero está restringido en movimiento y rotación, para que no lo muevan las esferas al colisionarlo. Tiene Interactable y Throwable y el script desarrollado **BataBallCube** con variables publicas Material Up, Material Down, orientacion, Cuerpo y lado. La teoría de este cubo es que al chocarlo las esferas, hace sonar un cuerpo del instrumento. Según la orientacion arriba o abajo del disco adosado a cada cubo, la esfera lo hará sonar cuando se esté moviendo hacia arriba o hacia abajo. La orientacion del cubo hace cambiar su material, por eso se pasan los dos materiales como variable publica, siendo estos MatCubeUp y MatCubeDown.

**Face**

Cada cubo tiene un hijo llamado Face que es un disco, idéntico a los discos entre los que se mueven las esferas salvo por su color, que es el objeto que representa qué cuerpo va a hacer sonar el cubo que lo porta. Usa los mismos materiales que los mencionados discos y el script desarrollado **BataBallCubeFace.**

**Tablero**

Objeto vacío con los rectángulos translúcidos que controlan los parámetros del instrumento.

**Rect1(y sus análogos Rect2, Rect3 y Rect4)**

Tiene el script desarrollado **DrawRectangle** el cual tiene como variables publicas Level Token, Filter Token, Material, OSC y cuerpo, siendo cuerpo la variable que identifica a qué cuerpo controla ese rectangulo.

**Chato**

Objeto vacío que alberga una de las fichas o tokens que controlan las dimensiones del rectangulo. Su nombre es porque es el más achatado.

**BotonChato1**

Tiene Interactable, renderer, collider, Linear Drive y el script desarrollado **BataBallToken.** Usa el material molten-604… y el script tiene como variables publicas el cuerpo que controla (1-4) y el tipo de parametro (filter-level) en este caso es 1 y Level (importante que se escriba con mayúscula)

**LinearMapping**

Para albergar al script de linear mapping.

**StartChato1**

Inicio del LinearMapping

**EndChato1**

Fin del linear Mapping.

**Alto**

Objeto vacío que alberga la otra ficha que controla las dimensiones del rectángulo. Su nombre es porque es el que sobresale.

**BotonAlto1**

Casi identico a BotonChato1, pero cambia su material a rare colors. Usa el mismo script, solo que se cambia el parametro con controla. En este caso es 1 y Filter (reocordar mayúscula).

**LinearMapping**

Para albergar al script de linear mapping.

**StartAlto1**

Inicio del LinearMapping

**EndAlto1**

Fin del linear Mapping.

**Wisp (BubbleDrones)**

**Scripts desarrollados: SphereTilingAnimation, SwordGrab, Filo Dibuja Circulo, FloatingScript.**

**Materiales: wall, MatWispBlueSphere, sp\_wood\_oak\_mask1, TileText Oreo Crunch, MatSwordLameBlue, Mat SwordLAmeGreen, Vidrio, AlienHand, Material\_Blueorb, Material\_Greenorb, Material\_Partiblue**

**Estructura:**

**Objeto principal: PedestalOrbes**

Objeto parent que tiene a todos los demas objetos y contiene el script OSC de este modulo.

**DarkAltar**

Modelo ·3D de un altar medio tenebroso que es donde se apoya la espada y flotan los orbes. Tiene el mesh filter del modelo3d, mesh renderer, rigidbody kinematic sin gravity, collision discrete y el mesh collider sin trigger ni convex. Usa el material wall.

**BlueSphere**

Esfera central que controla al orbe azul, el fat. Tiene mesh filter de esfera, mesh renderer, y el collider sin trigger. Luego viene un script importante **SphereTilingAnimation** comun a todas las esferas de este modulo, que tiene como variable serializada el nro de orbe que controla (para este caso el 1) y el script del filo de la espada, que va a aparecer más adelante. Lo que hace este script es que animar el material de la esfera para que se vean anillos de luz que se mueven dando forma al objeto. Después tiene un Interactable y el hover button para activar la accion. En la lista OnButtonDown está la funcion pertu() del script SphereTilingAnimation, que lo que hace es generar una perturbacion visual en los anillos de la esfera, cuando se la interactua, para mostrar que tuvo un efecto. Usa el material MatWispBlueSphere.

**GreenSphere1**

Esfera que está a la izquierda de la azul, que es practicamente igual, mismos componentes, incluso el script de animacion, pero usa el material MatWispGreenSphere y como nro de orbe tiene asignado el 3.

**GreenSphere2**

Casi igual a la otra esfera, pero está a la derecha y tiene asignado el orbe 2.

**OrbeAzul2.0**

Originalmente era un prefab en la carpeta project, pero lo agregué como objeto inactivo para que sea el objeto original del que luego se crean copias. Solo tiene un ParticleSystem que se encarga de formar la imagen, por decir algunos valores Duration 5, looping enabled, start size 1, más particles 50. Emission activada, shape activada en forma Sphere, radio 0.0001 y bueno, creo que con estos valores se puede ir jugando. También tiene activada la casilla lights. Lo importante es quetiene activada la casilla Renderer en modo Billboard y aca se selecciona como material el Material\_Blueorb, que es basicamente la forma misma del orbe. Tiene el script desarrollado **FloatingScript** que es el encargado del movimiento de flotación del orbe.

**ParticleSystem**

Otro sistema de particulas que además de lo mismo que tiene OrbeAzul, tiene activadas las casillas de Size over lifetime, que los hace encoger con el tiempo, y noise. Tambien es importante el renderer, que hace que las particulas tomen la forma del material dado, en este caso Material\_Partiblue.

**OrbeVerdeFloating2.0**

Es casi igual al azul, solo que usa el material en verde, se llama Material\_Greenorb.

**ParticleSystem**

Mismo particlesystem que el blueorb.

**StingSword**Objeto que contiene los elementos que componen al creador de orbes. Tiene un rigidbody y un box collider simulando la colision de la hoja de la espada.

**Manche**

Del modelo 3D (originalmente StingSword-Lowpoly) se sacaron los meshes Manche, Garde, Lame y Pommeau. En este caso Manche es el mango, tiene el mesh filter, mesh renderer, rigidbody con gravity sin kinematic, tiene un mesh collider con is trigger activado, el throwable, el velocity estimator y por ultimo el script desarrollado **SwordGrab** que es el que tiene el flag de si la espada está agarrada o no, el que usa el filo de la espada para saber si tiene que crear orbe o no. Usa el material sp\_wood\_oak\_mask1

**Garde**

Modelo 3D de lo que separa la empuñadura del filo, solo tiene renderer y usa el material TileTextOreoCrunch.

**Lame**

Filo de la espada. Es el modelo 3D de la hoja de la espada y además el que controla casi todo en este módulo: crea la preview, crea el orbe, lo destruye. Cambia de color segun el próximo orbe que va a crear. No tiene muchos componentes, casi todo está en el script. Tiene Mesh renderer y el script desarrollado **Filo Dibuja Circulo.** Tiene la variable serializadas Tecla crear (este campo y el script en general, tiene que tener una revisión en cuanto a cómo leer el input desde el mando VR, quizas podría implentar esa funcion antes de seguir con otro modulo, porque va a ser bastante de investigación creo) que es la tecla que se usa para activar la preview y creación de los orbes. Por ahora es una tecla, porque necesitaba hacer las pruebas y no tengo el equipo VR, luego va a ser un boton del mando, y seguramente este campo no exista. Seguimos con campos seralizados: Material preview es el material de que va a estar hecha la esfera preview (en este caso Vidrio), Material Sword blue el material que va a mostrar la hoja de la espada cuando el proximo orbe sea el azul, Material sword green lo mismo que el anterior pero cuando vaya a crear un orbe verde, Orbe azul prefab y orbe verde prefab son los prefabs de orbe que necesita tener pasados por variable para poder instanciar cuando lo requiera. Esfera activada es el flag de si se interactuó alguna de las esferas y qué numero de esfera se activó. Es en realidad una variable publica accedida por el script de las esferas, que aparece en el inspector, pero no debe ser modificada y por ultimo el campo para agregar el script OSC del objeto principal que es PedestalOrbes. Usa los materiales MatSwordLameBlue, Mat SwordLAmeGreen y Vidrio.

**Pommeau**

Es la parte más baja del mangod e la espada, será pomo en frandés calculo, es solo la mesh de vista. Usa el material AlienHand.

**EsferasDistancia**

**Scripts desarrollados: BotonesEsfera, SphereContButton, SphereDiscButton, OrangeSphereMovement, BlueSphereMovement, WaveformAnim .**

**Materiales: MatMonolitoEsferas, SphereContButton, MatBlueSphereButtonOn, MaterialSphereButtonOff, MatOrangeSphereOn y MatOrangeSphereButtonOff, MatOrangeSphere, MatBlueSphere, sinWave, triangleWave, squareWave y sawtoothWave, Material\_Orangelavastar, Material\_Partiblue.**

**Estructura:**

**Objeto principal: PedestalEsferas**

Contenedor de todo el instrumento y del script OSC para comunciarse con PD. IN 6960 OUT 6160.

**MonolitoEsferas**

Es el modelo del monolito antiguo estilo columna que alberga los botones del instrumento. Usa el material MatMonolitoEsferas.

**Botones**

Objeto contenedor de los botones que controlan los modos del instrumento. Tiene el script desarrollado **BotonesEsfera** que tiene variable publica Estado.

**EsferasBlueButton**

Boton que activa el modo continuo del instrumento. Tiene Renderer, collider y los script de comportamiento de botones de SteamVR. En la lista de acciones de OnButtonDown, tiene el metodo prenderContinuo de su script **SphereContButton**. Dicho script tiene variables publicas MaterialOn MaterialOff, OSC y EsferaState. Usa el material MatBlueSphereButtonOn y Off.

**EsferasOrangeButton01 (02, 03, …, son 12 en total)**

Similar al boton azul, pero estos son los botones naranjas, más pequeños, que controlan el modo de alturas justas del instrumento. Estan dispuestos como si estuvieran sobre las teclas de un teclado convencional, para representar las notas que están encendidas en el momento. Tiene comportamientos similares al boton azul, pero usa el script **SphereDiscButton** por discreto. Tiene las variables publicas MaterialOn, MaterialOff, Osc, EsferaState y Nota, para especificar qué nota controla (1 es C, 2 es C#, 3 es D, etc). Usa los materiales MatOrangeSphereOn y MatOrangeSphereButtonOff.

**SphereOrange**

Esfera naranja, al tomarla comienza a producir sonido variando altura segun la distancia a que se encuentre de la esfera azul. Contiene Renderer, Collider, Rigidbody con kinematic sin gravity, interactable y throwable y el script de comportamiento desarrollado **OrangeSphereMovement** con variables publicas Blue Sphere, Line (1-4) Material y OSC. Los materials de linea son las lineas estilo relampago que se formas cuando se toman las esferas, es un efecto visual. Usa el material MatOrangeSphere.

**ParticleSystem**

Sistema de partículas que se ve dentro de la esfera. Usa el material Material\_Orangelavastar.

**SphereBlue**

Casi igual a la esfera naranja, pero azul y sin el comportamiento de producir sonido. Solo flota cuando no está activa. Tiene el script desarrollado **BlueSphereMovement.** Usa el material MatBlueSphere.

**ParticleSystem**

Sistema de partículas que se ve dentro de la esfera. Usa el material Material\_Partiblue.

**CubeWaveform**

Cubo que permite cambiar la forma de onda del instrumento. Tiene Renderer, collider, rigidbody sin gravity ni kinematic, restringido en todas las posiciones y en las rotaciones X y Z para que solo gire en Y al ser interactuado. Usa el script desarrollado **WaveformAnim** que tiene como variables publicas OSC, Sin wave, triangle wave, square wave y saw wave, que son los materiales de cada forma de onda. Usa esos materiales, sinWave, triangleWave, squareWave y sawtoothWave.