Desarrollo de un entorno interactivo para asistir en el proceso de aprendizaje y enseñanza de circuitos de microondas.

Índice

Índice de imágenes.

Índice de tablas.

1. Introducción y Objetivos.

1. INTRODUCCIÓN

1. Introducción y Objetivos.

1.1Introducción.

El entorno educativo introducido por el tratado de Bolonia, donde se presta tanta importancia a los contenidos como a los procesos, obliga a las universidades a reinventar sus métodos de enseñanza. Se plantea un cambio estructural, práctico y de metas.

El cambio estructural se centra en el aprendizaje del alumno, es capital que la enseñanza esté centrada en el que la recibe, en lugar de ser la pieza clave el docente. Se trata de alcanzar una docencia amena, que consiga la participación constante del alumno, es clave facilitar el aprendizaje del alumnado.

El método docente pasa de expositivo a activo. Ganan importacia, los sistemas de autoaprendizaje frente a la lección magistral.

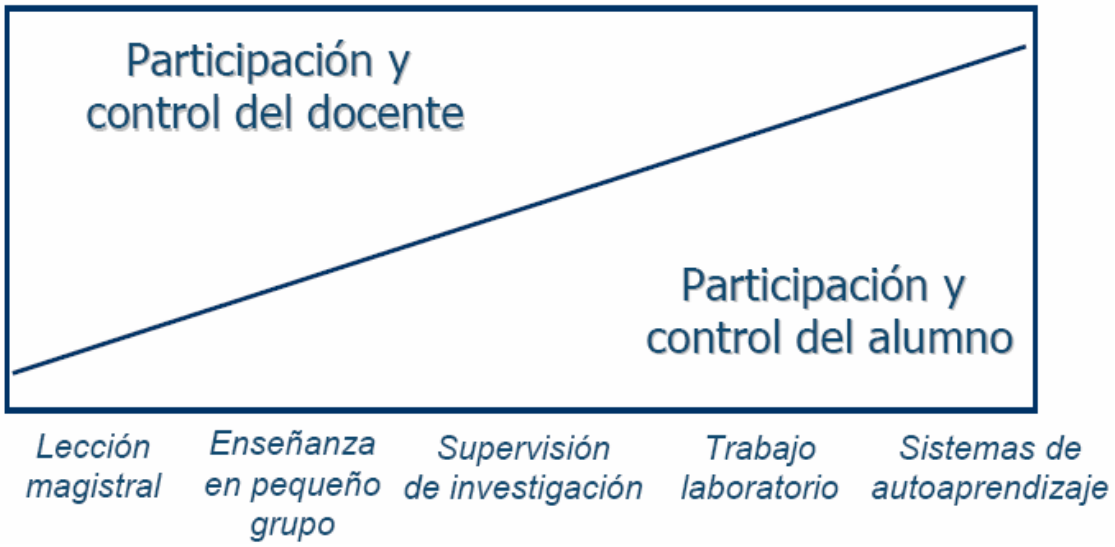

Ilustración 1. cuadro de metodologías

La dirección que toma la enseñanza crea un amplio campo para el desarrollo de herramientas didácticas. Dentro de este ámbito se encuadra la iniciativa MCSI, cuyo objetivo es crear herramientas interactivas de fácil acceso para todos los alumnos.

En concreto con este proyecto se ha diseñado una aplicación interactiva, que pone a prueba los conocimientos del alumno y almacena sus avances.

1.2 Objetivos

En este proyecto se marcó como objetivo desarrollar una aplicación interactiva basada en flash para motivar a los alumnos en el proceso de aprendizaje sobre circuitos de microondas. Para ello se ha realizado un juego de ordenador en el que el alumno asume la personalidad de un joven ingeniero de Telecomunicación a bordo de una nave espacial, en la que debe superar una serie de pruebas relacionadas con circuitos de microondas para pasar de nivel en nivel.

Este objetivo se enmarca dentro de los nuevos métodos docentes, que permitan al alumno aprender de forma amena y a distancia. Se ha creado una historia que contextualiza la resolución de circuitos de microondas, lo que permite introducir una gran cantidad de elementos, escenarios, personajes individualizados, enemigos, que hacen la aplicación atractiva. Estos aspectos son fundamentales, ya que un entorno gráfico cuidado redunda a favor del tiempo de juego, cuanto más atractiva sea la aplicación más uso se le dará. Al ligar el avance en la aplicación con la resolución de cuestiones teóricas se favorece la autoevaluación y la mejora continua.

El desarrollo de una aplicación que se utiliza a distancia elimina los problemas de accesibilidad que puedan existir y facilita el aprendizaje; el alumno tiene a su disposición toda la información para la resolución de los problemas planteados.

Podemos enumerar los siguientes objetivos concretos de este proyecto fin de carrera:

1- Desarrollar un entorno virtual en el que el personaje del juego pueda moverse e interactuar con objetos, entre ellos diversos circuitos de microondas.

2.- Desarrollar una base de datos donde el usuario pueda darse de alta de forma remota por Internet, para jugar en esta aplicación y poder acceder de manera continuada al nivel en el que se haya quedado.

3- Idear una serie de pruebas o fases para que el alumno pueda aprender de forma divertida diferentes conceptos sobre circuitos de microondas.

4- Programar en el entorno virtual estas pruebas, y proponer nuevos niveles.

La memoria de este trabajo se ha dividido en dos capítulos fundamentales. En el capítulo 2 se describe el desarrollo de la aplicación programada, explicando las acciones que debe realizar el jugador así como las pruebas que debe ir superando. En el capítulo 3 se describe de manera técnica cómo se ha desarrollado esta aplicación, distinguiendo entre el código programado en el cliente y en el servidor. El último capítulo se dedica a las conclusiones y las líneas futuras de este trabajo. También se puede encontrar un anexo en el que se detallan las instrucciones de instalación de la aplicación para que pueda funcionar en un ordenador personal que tenga instalado un servidor Apache

2. Guía de las tareas a realizar en el juego

2. Guía del juego.

2.1. Registro e identificación.

Cuando cualquier usuario inicia la aplicación lo primero que debe hacer es definir si ya es un usuario registrado o no.

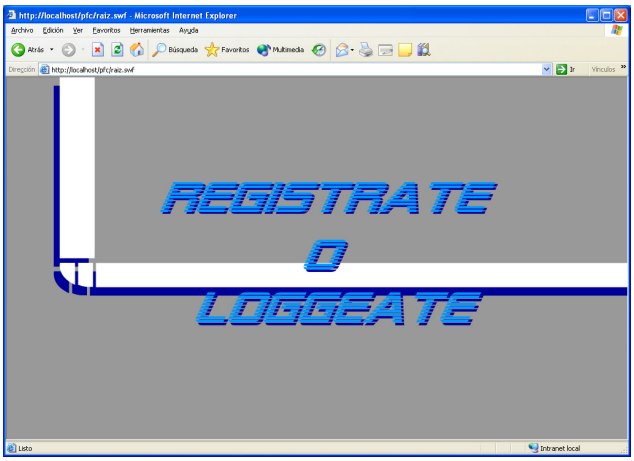

Ilustración 2. Pantalla inicial

En caso de serlo se le piden tanto su clave como la dirección de correo electrónico que utilizó para registrarse. En caso de que falle se le vuelven a pedir los datos, hasta que los introduzca correctamente.

Ilustración 3. Método de identificación

Se informará concretamente del fallo que se produce, tanto si el correo introducido no existe o si el correo no coincide con la clave asociada.

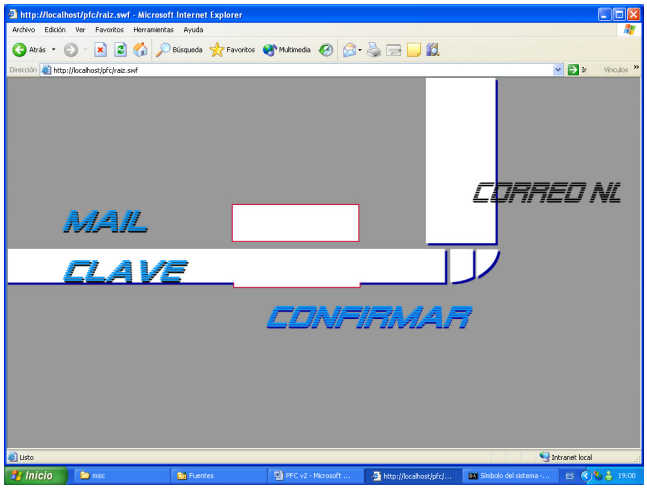

Ilustración 4. Error en la identificación

Por otro lado si es la primera vez que utiliza la aplicación deberá registrarse introduciendo algunos parámetros personales y su clave y dirección de correo electrónico que deben ser únicas.

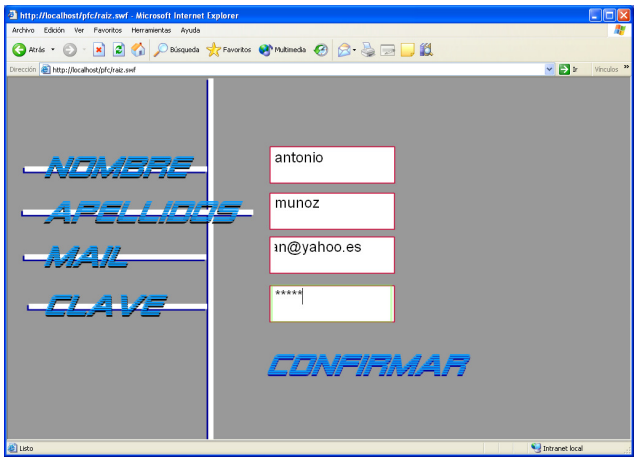

Ilustración 5. Método de registro

En caso de que ya existiera alguna de las dos se informa al usuario y se espera hasta que introduzca un par válido.

Si tiene éxito en el registro de sus datos personales pasará a registrar los datos relativos a su personaje.

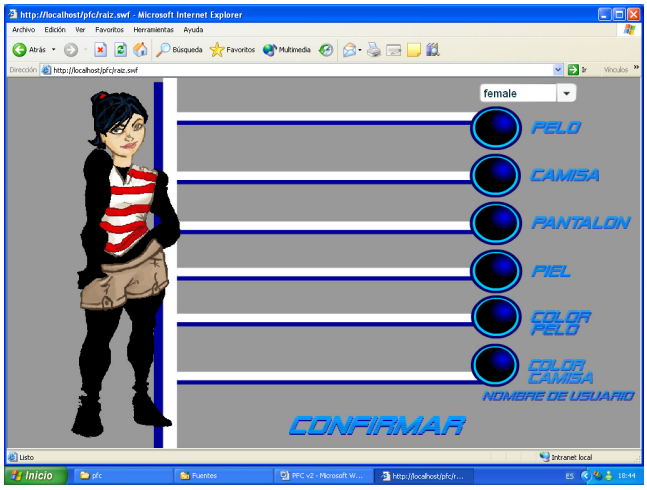

Ilustración 6. Registro de los datos del personaje

2.2. Primera pantalla

El objetivo es conseguir algo de material para resolver los circuitos q se presentarán a continuación.

En primer lugar se despliega un cuadro de texto en el que aparece el guía que el usuario tendrá en el desarrollo del juego. Su función será dar consejos sobre la resolución de los circuitos.

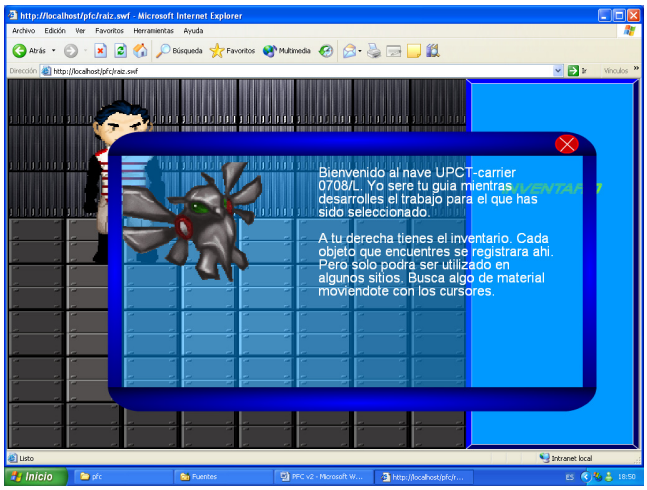

Ilustración 7. Presentación del guía

Si el usuario investiga un poco descubrirá una habitación contigua con un armario.

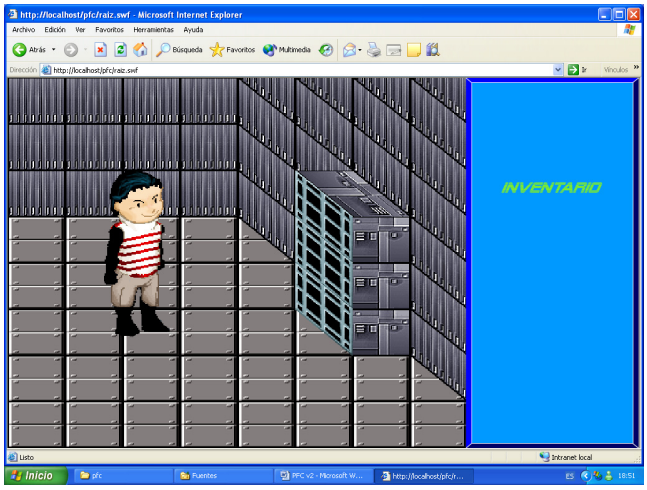

Ilustración 8. Habitación contigua y armario con material

Al aproximarse al armario éste se abre y el guía informa al usuario de los objetos que ha recogido.

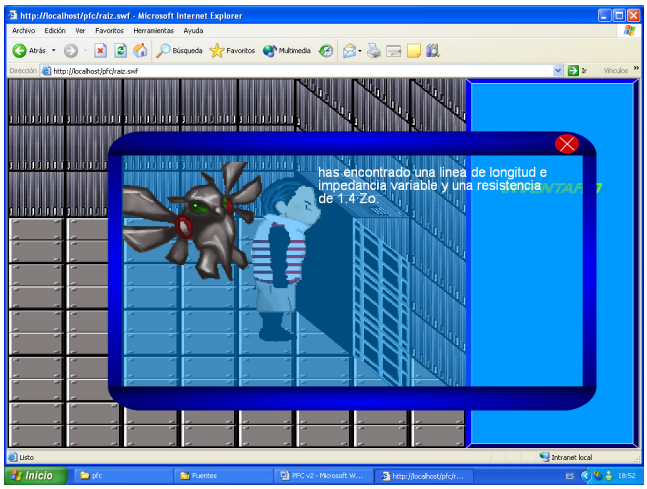

Ilustración 9. Apertura del armario e información de los objetos conseguidos

Pinchando sobre el botón inventario se despliega un cuadro en el que el usuario puede observar los objetos que ha recogido. El inventario se divide en varias columnas que representan los distintos tipos de objetos recogidos y las características de cada uno de ellos, como son la impedancia o longitud eléctrica.

3. Desarrollo de la aplicación.

Ilustración 10. Inventario

Existen otros personajes que también darán información al personaje mediante un diálogo predefinido.

Ilustración 11. Diálogo con otros personajes

Si se intenta salir de esta zona antes de obtener el equipo necesario el guía notificará al usuario que debe seguir buscando y no le permitirá continuar.

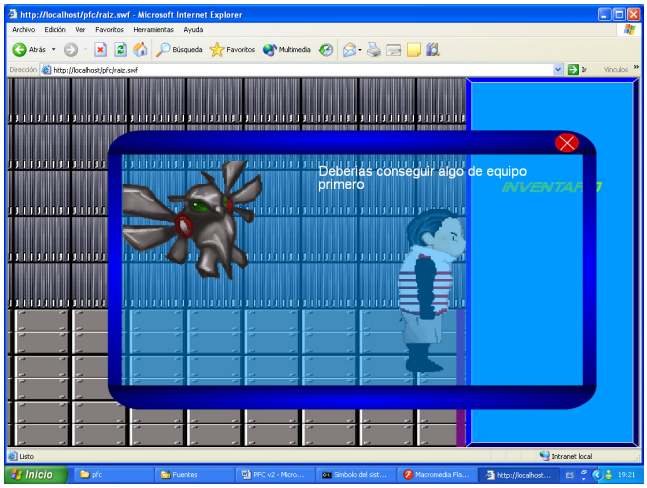

Ilustración 12. Consejos del guía

2.3. Puente con wilkinson

El guía informa al usuario en cada nuevo escenario de forma que sepa lo que tiene que hacer, siempre aporta información útil para el desarrollo del juego y la resolución de circuitos.

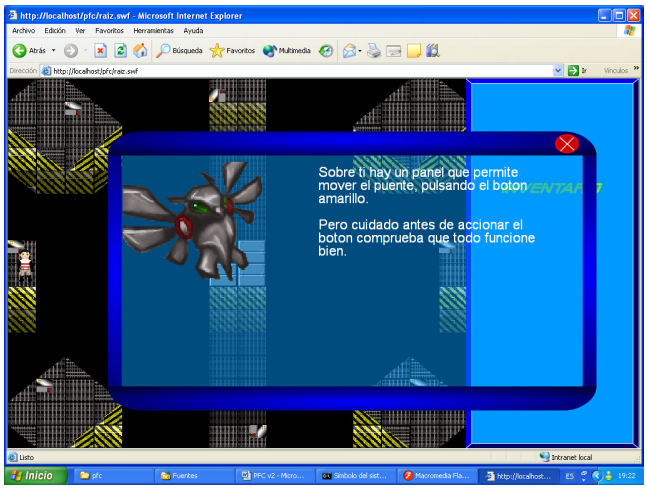

Ilustración 13. Información al empezar un nuevo escenario

Ciertas zonas del escenario provocan la carga de puzzles cuya resolución está ligada al avance en el juego. En este caso es necesario resolver un divisor wilkinson, mediante la inclusión de elementos, para conseguir girar el puente hasta la posición que ocupa el personaje. El divisor carece de una línea y de la resistencia necesaria para aislar los puertos y es necesario conseguir que la señal viaje en la dirección especificada.

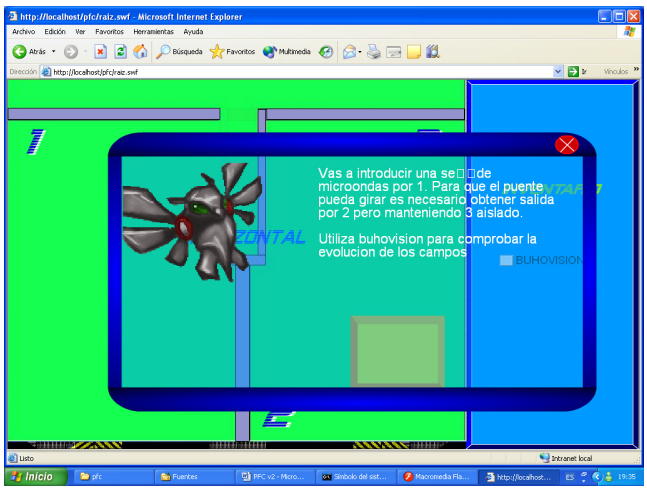

Ilustración 14. Información sobre la resolución del circuito

Haciendo clic sobre el botón "buhovision" es posible visualizar los campos y obtener información sobre lo que está sucediendo a medida que se resuelve el circuito. Antes de utilizar cualquiera de los elementos se observa que la señal no avanza, sin embargo la información aportada por el guía indica que para que el puente gire de forma correcta se necesita señal en el puerto dos, pero no en el tres. Por tanto después de utilizar la línea para conectar los puertos uno y dos se puede comprobar como la señal se reparte de forma equitativa hacia dos y tres. En este punto el alumno ha comprobado el funcionamiento de un divisor con líneas. Mediante la correcta distribución de impedancias equivalentes es posible dividir una señal. Sin embargo el objetivo de esta prueba no sólo implica una familiarización con el concepto de impedancia equivalente, también se quiere que el alumno conozca una forma efectiva de aislar dos ramas. Para cumplir este objetivo el puzzle no está resuelto hasta que se introduce la resistencia que convierte al divisor con líneas en un divisor wilkinson.

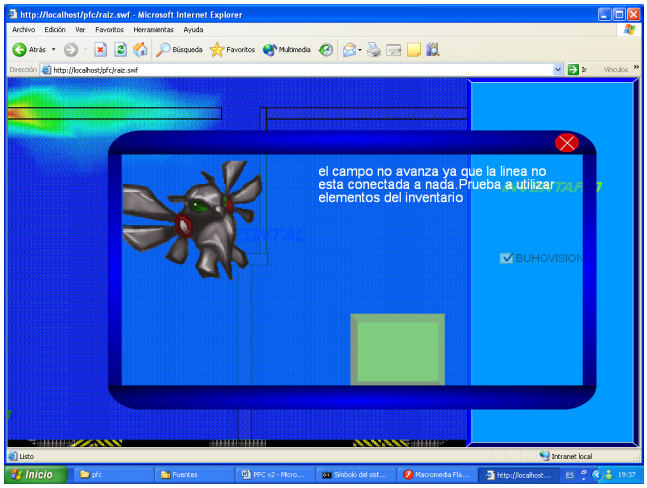

Ilustración 15. Vista de los campos e información sobre ellos

Para conseguir resolver el circuito es necesario utilizar correctamente los objetos del inventario. En la ilustración 16 se muestra la elección de características un objeto del inventario. Es decir seleccionar una línea de entre todas las posibles que permita que el conjunto funcione en primer lugar como un divisor con líneas y en segundo lugar utilizar la resistencia obtenida para aislar los puertos y conseguir que la señal vaya donde se ha especificado.

La línea recogida tiene longitud e impedancia variable y el usuario es capaz de seleccionar su valor, por tanto debe saber que valores elegir para solucionar el puzzle.

Como se quiere que la señal salga únicamente por el puerto dos el usuario debe saber que es necesaria una resistencia para conseguir aislar los puertos. Si no la utilizase se produce un error.

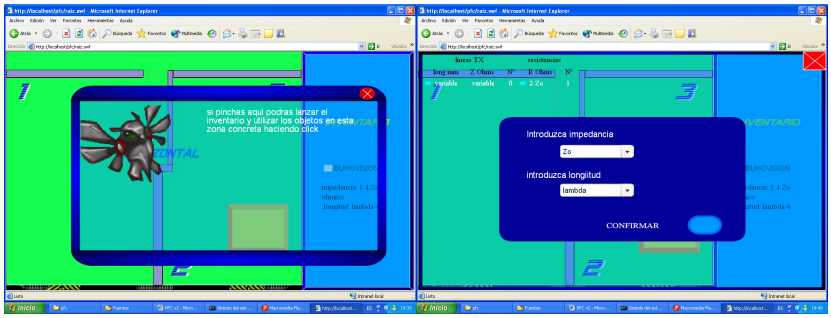

Ilustración 16. Elección de elemento del inventario para su uso

Tras la utilización de objetos pulsando buhovisión el usuario es capaz de ver el cambio en los campos que recorren el circuito y así tener una idea clara del objetivo de cada elemento.

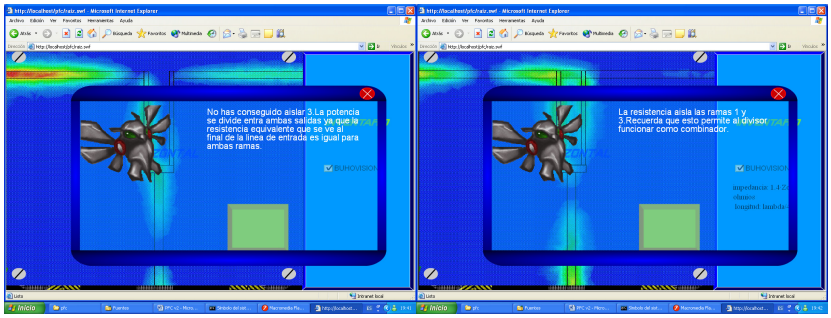

Ilustración 17. Distintas visualizaciones de los campos, con la resolución del circuito

Si se han seleccionado correctamente los objetos del inventario cuando se pulse sobre el botón amarillo el puente girará, como se ve en la ilustración 18, pero si no es así se muestra un mensaje informando del porqué no se ha solucionado correctamente el circuito. En este punto pueden darse dos circunstancias, o bien no se ha utilizado la resistencia que aísle los puertos uno y tres, o bien la línea elegida no tiene los parámetros adecuados. El guía informará de que suceso ha ocurrido, de esta forma el alumno sabe que fallo ha cometido y pueda subsanarlo.

Además se informa al usuario de que el juego ha terminado y tendrá que volver a empezar desde el último punto en el que salvó sus avances.

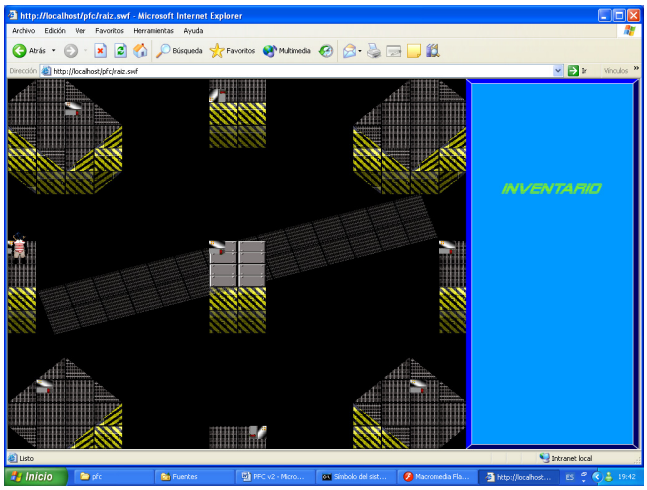

Ilustración 18. Giro del puente

En la siguiente imagen se puede observar el informe al usuario, que indica que la línea utilizada no es la adecuada.

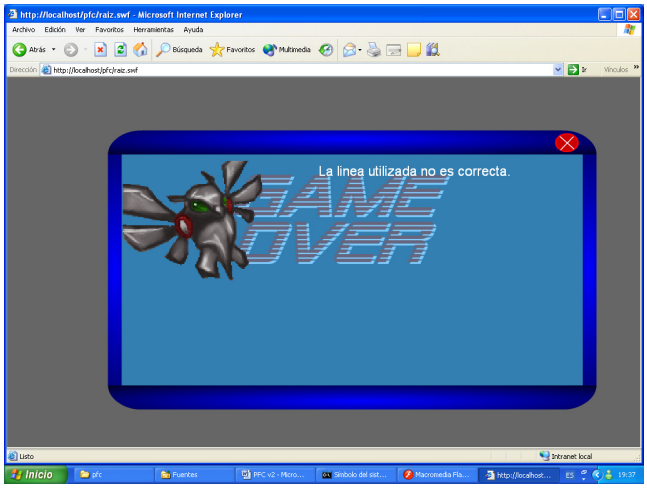

Ilustración 19. Pantalla de Game Over e información del error

2.4. Puente con híbrido y círculos de estabilidad.

El siguiente circuito controla el giro del puente para cualquiera de las posiciones que pueda tomar. En primer lugar se debe seleccionar la dirección en la que se desee encarar el puente. Se desea que el alumno conozca los efectos de la longitud de las líneas de transmisión. Para ello se utiliza un híbrido.

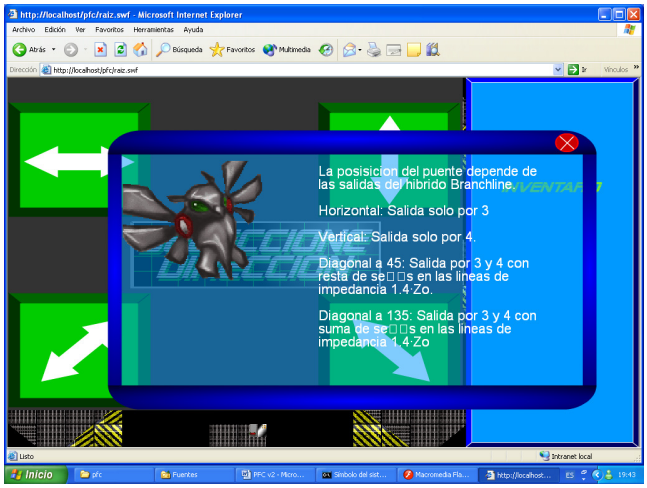

Ilustración 20. Selección de las direcciones del puente

A continuación se modifican las entradas de un híbrido branch-line para conseguir una salida concreta. Cada salida está asociada a una posición. En caso de no elegir correctamente las entradas se informará al usuario de que el juego ha terminado. Mediante la visualización de los campos el usuario observa en que puntos las señales se interfieren de forma destructiva o constructiva. La interferencia de las señales depende de la fase inicial con la que se introduzcan las señales en el híbrido, por esta razón las entradas del circuito son controladas por el alumno. Sin embargo la posición del puente depende de las salidas y que posición corresponde a cada salida queda a elección del docente.

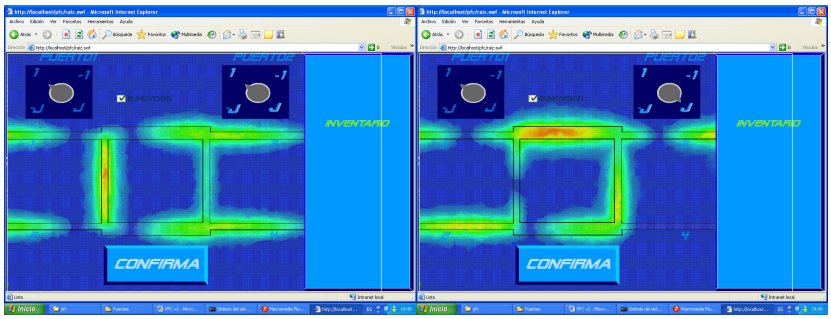

Ilustración 21. Campos en el híbirido

La última de las pruebas consiste en elegir la zona estable de un transistor, conocidos sus parámetros S y la localización de su círculo de estabilidad. Tradicionalmente ha sido complicado conseguir que los alumnos determinen que zonas son estables a partir de los parámetros S de un transistor y de la carta de Smith. Mediante este ejercicio se solventa esta circunstancia utilizando un método muy visual. Como se puede ver en la imagen esta vez el que informa al usuario no es su guía, sino uno de sus enemigos, un pirata. Es importante alternar personajes para poder desarrollar una historia que soporte la resolución de circuitos y despierte el interés del usuario.

Ilustración 22. Pirata informando como actuar ante el puzzle

Nuevamente si se ha cometido algún error se termina el juego cargando la pantalla de "Game Over" e informando al usuario de que ha hecho mal, especificando si el centro de la carta de Smith era estable o inestable y porqué, de esta forma el usuario puede relacionar su elección con los parámetros que se le dieron y en virtud de la experiencia descubrir qué zona seleccionar en cada caso.

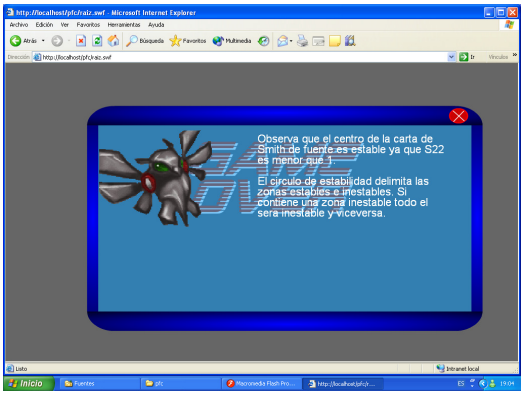

Ilustración 23. Pantalla de Game Over e información del error

3. DESARROLLO DE LA APLICACIÓN

3. Desarrollo de la aplicación.

Se ha creado una aplicación interactiva y a distancia, que permite al usuario conectarse remotamente a un servidor y hacer uso de ella. Por este motivo es necesario una estructura cliente servidor.

3.1. Cliente.

El cliente se ha desarrollado en su totalidad mediante la herramienta de Macromedia Flash 8, haciendo uso en la mayoría de los casos exclusivamente del lenguaje de programación ActionScript.

3.1.1. Estructura.

El cliente está compuesto de un moviClip o película llamado raíz. El clip raíz es el encargado de cargar a todas las demás películas y almacenar la información que deban conocer el resto de clips.

```
var raiz:MovieClip=_root.createEmptyMovieClip("raiz",0);
raiz.loadMovie("selecto.swf");
raiz._lockroot=true;
```

Tabla 1.Creación de contenedor y carga.

Es importante poner la propiedad lockroot a true en los contendores en los que se cargan las películas. Esta propiedad fija las rutas, de esta forma no hay confusiones entre películas o variables con el mismo nombre pero que pertenezcan a movieClips distintos.

El archivo raíz recibe todas las características del personaje tras la identificación, dependiendo del parámetro nivel se seleccionará un clip a cargar u otro. Además detecta que el clip cargado en la película raíz se ha descargado y actúa en consecuencia.

```
var raiz:MovieClip=_root.createEmptyMovieClip("raiz",0);
var clipLoader:MovieClipLoader=new MovieClipLoader;
clipLoader.loadClip("selecto.swf",raiz);
raiz._lockroot=true;
marcaFrame=1;
```

```
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(Rcorreo:String,RnomUs:String,Rgenero:Number,Rpiel:Number,Rpelo:Number,Rpeloc:Number,Rcamiseta:Number,Rcamic:Number,Rpantalon:Number,Rnivel:Number)
{
        correo=Rcorreo;
        usuario=RnomUs;
        genero=Rgenero;
        piel=Rpiel;
        cabello=Rpelo;
        peloc=Rpeloc;
        camisa=Rcamiseta;
        camic=Rcamic;
        pantal=Rpantalon;
        nivel=Rnivel;
        raiz.onUnload=gotoAndPlay(2);
};
receiving_lc.connect("raiz_name");
```

Tabla 2. Recepción de Parámetros y Método de Descarga

El segundo fotograma sirve para seleccionar el clip que se cargará en función del nivel especificado en los parámetros del usuario. En cada fotograma al que se salta se elimina el movieClip raíz, porque ya no se va a necesitar, y se crea uno nuevo sobre el que cargar la siguiente película. El motivo por el que se elimina raíz en lugar de reutilizarlo es que cuando esta segunda película se descargase, al tener acciones distintas asociadas al evento de descarga, flash no sabría que acciones llevar a cabo. Por otro lado tampoco es posible utilizar una función cuyos valores cambien conforme se van descargando películas debido a que flash detectaría más de una descarga, aunque realmente no se esten produciendo.

```
if(nivel==0){
        gotoAndPlay(3);}
if(nivel==1){
        gotoAndPlay(6);}
if(nivel==2){
        gotoAndPlay(9);}
```

Tabla 3. Selección de niveles.

```
raiz.removeMovieClip;
var raiz1:MovieClip=_root.createEmptyMovieClip("raiz1",1);
raiz1.loadMovie("moviendo vengo.swf");
raiz1._lockroot=true;
```

Tabla 4. Actualización de Contenedores.

3.1.2. Registro e identificación.

3.1.2.1 Registro

3.1.2.1.1 Información del usuario.

Cómo primer paso en la creación de la aplicación pareció lógico crear un sistema de registro e identificación que permita definir un usuario único con características diferenciadas. De esta forma es posible registrar los avances de los jugadores.

Por tanto, la primera acción que debe afrontar cualquier nuevo usuario del juego es el registro de sus datos.

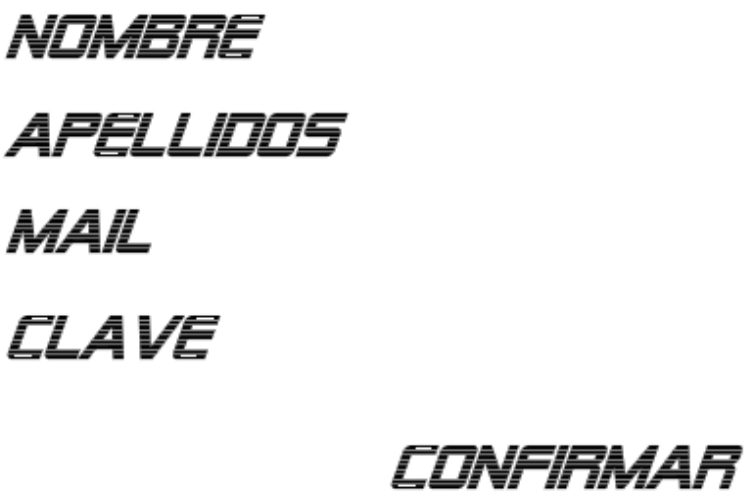

Ilustración 24. Registro de los datos del usuario

Cuando un usuario se registra, sus datos son archivados en una base de datos generada con MySQL. Se pide la dirección de correo eléctronico y una clave para tener un método de identificación seguro, ya que la aplicación no

permite la inserción en base de datos de direcciones de correo electronico ni de claves duplicadas.

La inserción de un nuevo usuario en la base de datos ha de hacerse utilizando PHP debido a que flash no permite la comunicación directa con bases de datos, aunque si dispone de métodos para enviar información a varios lenguajes de servidor como son PHP y XML.

```
var registro:LoadVars=new LoadVars;
        var recibo:LoadVars=new LoadVars;
        function enviaDatos(){
                registro.nombre=_root.nombre.text;
                registro.apellidos=_root.apellidos.text;
                registro.mail=_root.mail.text;
                registro.clave=_root.clave.text;
                regis-
tro.sendAndLoad("http://localhost/phpflash/datosAlumn1.php",recibo,"POST")
                gotoAndStop(3);
        }
```

Tabla 5. Envío de datos al servidor

Se utilizan objetos LoadVars que son los que proporcionan comunicación con PHP. Cómo se puede observar se han creado dos objetos: registro es el encargado de enviar al servidor las variables que desamos introducir en la base de datos y recibo recogerá la respuesta del servidor y nos informará de ella. Por esta razón el método elegido para el envio de datos es sendAndLoad, el cual permite la recuperación de la respuesta desde el servidor.

Una vez enviados los datos la aplicación salta al frame siguiente y queda a la espera de respuesta.

```
recibo.onLoad=function(exito){
        if (exito){
                var myNumber:Number = Number(this.mensaje);
                trace(myNumber);
                switch(myNumber){
                        case 0:
                        nombre.text="";
                        apellidos.text="";
                        mail.text="";
                        clave.text="";
                        _root.respuesta.text="correo ya utilizado";
```

```
                        break;
                        case 1:
                        nombre.text="";
                        apellidos.text="";
                        mail.text="";
                        clave.text="";
                        respuesta.text="clave ya utilizada";
                        break;
                        case 2:
                        gotoAndPlay(6);
                        break;
                }
        }
}
```

Tabla 6. Gestión de la respuesta del servidor.

El evento onLoad se invoca cuando se termina de cargar el archivo PHP especificado desde el método sendAndLoad. La razón por la que se ha incluido esta función en un frame distinto al que realizo el envio de datos es tener la certeza de que el archivo PHP se ha ejecutado integramente y los datos que se obtienen son fidedignos.

La función asociada a la carga del archivo realiza tres acciones que dependen de la respuesta del servidor. Si la dirección de correo o la clave han sido utilizadas ya, resetea los campos de texto para que estos parámetros se introduzcan de nuevo, en caso de que el registro sea correcto, se salta al frame en el que se definen las características del personaje.

3.1.2.1.2. Datos del personaje.

Si el registro de los datos relativos al usuario es correcto se pasa a registrar al personaje que se utilizará. Para una mayor interactividad el personaje puede tener distintas características, que pueden ser elegidas por el usuario.

En primer lugar se inicializan las variables que se utilizan para cargar el personaje. La inicialización tiene lugar en un fotograma anterior a la carga. Si no se hiciese de esta forma no sería posible mostrar los cambios ya que las variables quedarían fijadas a los valores de inicialización.

```
var sk=0;
var h=1;
var hc=0;
var s=1;
var sc=0;
var t=0;
```

Tabla 7. Inicialización.

Las variables corresponden al color de piel, tipo de pelo, color de pelo, tipo de camisa, color de camisa y tipo de pantalón respectivamente.

Una vez inicializado se puede cargar el personaje.

```
mixml=new XML();
mixml.ignoreWhite=true;
mixml.load("C:/AppServ/www/personaje/seleccion2.xml");
mixml.onLoad= function(){
        cuerpo=this.childNodes[0].attributes.src;//carga el cuerpo de la chica

        cabeza=this.childNodes[0].childNodes[0].childNodes[0].attributes.src;
        pelo=this.childNodes[0].childNodes[0].childNodes[h].attributes.src;

rayas=this.childNodes[0].childNodes[1].childNodes[0].childNodes[0].attributes.src;

camisa=this.childNodes[0].childNodes[1].childNodes[0].attributes.src;//camisa
        cuello=this.childNodes[0].childNodes[1].childNodes[0].childNodes[s].attrib
utes.src;          panta-
lon=this.childNodes[0].childNodes[2].childNodes[t].attributes.src
ojos=this.childNodes[0].childNodes[0].childNodes[14].attributes.src;//ojitos
                    _root.createEmptyMovieClip("body",100);
                    item=_root["body"];
                    item.loadMovie(cuerpo);
                    item._xscale=67;
                    item._yscale=67;
                    item._x=px;
                    item._y=py;//cuerpo
                    _root.createEmptyMovieClip("head",101);
                    item=_root["head"];
                    item.loadMovie(cabeza);
                    item._xscale=67;
                    item._yscale=67;
```

```
item._x=px;
item._y=py;//cabeza
_root.createEmptyMovieClip("eyes",102);
item=_root["eyes"];
item.loadMovie(ojos);
item._xscale=67;
item._yscale=67;
item._x=px;
item._y=py;//ojos
_root.createEmptyMovieClip("hair",103);
item=_root["hair"];
item.loadMovie(pelo);
item._xscale=67;
item._yscale=67;
item._x=px;
item._y=py;//pelo
_root.createEmptyMovieClip("stripes",106);
item=_root["stripes"];
item.loadMovie(rayas);
item._xscale=67;
item._yscale=67;
item._x=px;
item._y=py;//camiseta
_root.createEmptyMovieClip("shirt",105);
item=_root["shirt"];
item.loadMovie(camisa);
item._xscale=67;
item._yscale=67;
item._x=px;
item._y=py;//camiseta
_root.createEmptyMovieClip("neck",107);
item=_root["neck"];
item.loadMovie(cuello);
item._xscale=67;
item._yscale=67;
item._x=px;
item._y=py;//camiseta
_root.createEmptyMovieClip("trousers",104);
item=_root["trousers"];
item.loadMovie(pantalon);
item._xscale=67;
item._yscale=67;
```

```
                item._x=px;
                item._y=py;//pantalon
        }
```

Tabla 8. Creación de películas para la carga y extracción de rutas.

La información para la carga del personaje se obtiene de un archivo XML en el que están especificadas las rutas de los archivos que componen el personaje. Cuando el archivo XML ha terminado de cargarse se crea un objeto MovieClip para cada característica del personaje y se carga en él el archivo especificado. Cada MovieClip creado tiene una profundidad. Dos MovieClips no pueden tener la misma profundidad o es imposible visualizar ambos. Además los MovieClips con un valor mayor en su profundidad ocultarán a los que tengan un valor menor. Por tanto hay que saber que archivo se quiere ver.

El resultado de la carga es el siguiente:

Ilustración 25. Selección de las características del personaje.

Ahora es posible cambiar distintos parámetros del personaje cómo el género o el tipo de pelo utilizando los botones.

```
gender.addEventListener("change",mostrar);
piel.addEventListener("click",clicked);
pelo.addEventListener("click",pelos);
pelocolor.addEventListener("click",colorp);
cami.addEventListener("click",cami);
```

```
camicolor.addEventListener("click",camic);
pantalon.addEventListener("click",pant);
confirmacion.addEventListener("click",confi);
function clicked(){
        sk++
        if(sk>2)
                sk=0
        colorCara(head,sk);
}
function pelos(){
        h++;
        if(gender.selectedIndex==0){
                if (h>13)
                        h=1;
        }else{
                if(h>5)
                        h=1;
                }
        mostrar();}
function colorp(){
        hc++;
        if(hc>2)
                hc=0;
        colorPelo(hair,hc);
        }
function cami(){
        s++;
        if (s>2)
                s=1;
        mostrar();
        }
function camic(){
        sc++;
        if (sc>1)
                sc=0;
        colorCamisa(stripes);}
function pant(){
        t++;
        if(t>1)
                t=0;
        mostrar();}
```

Tabla 9. Acciones para el cambio de características.

Para que el cambio de parámetros sea posible es necesario añadir detectores de eventos que realizan la función especificada cuando los botones son pulsados. Cada una de las funciones cambia una característica del personaje seleccionando un nuevo archivo a cargar dentro del código XML o modificando los atributos de los archivos ya cargados.

```
function mostrar(){
        if(gender.selectedIndex==0){
                cuerpo=mixml.childNodes[0].attributes.src;//carga
el cuerpo de la chica
```

Tabla 10. Función mostrar

La función mostrar carga los archivos especificados por las funciones que la llaman, utilizando los mismos métodos que la función descrita en la tabla 7, con la salvedad de que antes de iniciar la carga comprueba que nodo XML debe cargarse; el que contiene los datos de la chica o el que contiene los del chico. Además de los parámetros que se cargan existen otros que se modifican sin tener que realizar ninguna carga. Estos son los que definen algún tipo de color.

Para realizar un cambio de color las funciones reciben un MovieClip, al que cambiaran el color, y un número que identifica el color al que se cambiará.

```
function colorCara (faz:MovieClip,numero:Number){
        var transformer:Transform = new Transform(faz);
        var colorTransformer:ColorTransform = transformer.colorTransform;
        if(sk==1){
                var numerR:Number=Number(0xf7)/Number(0xF2);
                var numerG:Number=Number(0xde)/Number(0xD9);
                var numerB:Number=Number(0xe4)/Number(0xB6);
                colorTransformer.redMultiplier = numerR;
                colorTransformer.greenMultiplier = numerG;
                colorTransformer.blueMultiplier = numerB}
        else if (sk==2){
                var numerR:Number=Number(0xda)/Number(0xF2);
                var numerG:Number=Number(0xa0)/Number(0xD9);
                var numerB:Number=Number(0x72)/Number(0xB6);
                colorTransformer.redMultiplier = numerR;
                colorTransformer.greenMultiplier = numerG;
                colorTransformer.blueMultiplier = numerB;}
                else{
```

```
                        colorTransformer.redMultiplier = 1;
                        colorTransformer.greenMultiplier = 1;
                        colorTransformer.blueMultiplier = 1;
                        }
        transformer.colorTransform = colorTransformer;
        return;};

function colorPelo(cabello:MovieClip,numero:Number){
        var transformer:Transform = new Transform(cabello);
        var colorTransformer:ColorTransform = transformer.colorTransform;
        if(hc==1){
                colorTransformer.redOffset = 162;
                colorTransformer.greenOffset = 73;
                colorTransformer.blueOffset = 32;}
                else if(hc==2){
                        colorTransformer.redOffset = 224;
                        colorTransformer.greenOffset = 187;
                        colorTransformer.blueOffset = 137;}
                        else{
                                colorTransformer.redOffset = 0;
                                colorTransformer.greenOffset = 0;
                                colorTransformer.blueOffset = 0;}
        transformer.colorTransform = colorTransformer;
        //transformarPelo.colorTransform = color;
        return;}

function colorCamisa(camiseta:MovieClip,numero:Number){
        var transformer:Transform = new Transform(camiseta);
        var colorTransformer:ColorTransform = transformer.colorTransform;
        if(sc==1){

                colorTransformer.redMultiplier=0.3;
                colorTransformer.greenMultiplier=1.6;
                colorTransformer.blueMultiplier=30;
                }
                else{
                        colorTransformer.redMultiplier=1;
                        colorTransformer.greenMultiplier=1;
                        colorTransformer.blueMultiplier=1;}
        transformer.colorTransform = colorTransformer;
        return;}
```

Tabla 11.Funciones para el cambio de color.

ActionScript define cualquier color mediante tres números hexadecimales en la terna RGB, esto implica que cualquier color en el que las componentes tengan valores muy próximos a 0, es decir colores próximos al negro, no pueden modificarse multiplicando, esta es la razón de que para modificar el color del pelo se haya utilido la función blueOffset.

Una vez que se han elegido las características que definen al personaje son enviadas al servidor para que las inserte en la base de datos. Las variables que definen al personaje deben estar asociadas a un usuario concreto, por lo que junto con ellas se envía también la dirección de correo electrónico.

Además también se preparan para enviarlas a la película raíz, de forma que estén disponibles para el resto de películas que se carguen. Para ello se almacenan los datos elegidos en variables que se envían a la película raíz. Además se eliminan los movieClips donde se cargaron las imágenes.

```
function confi(){
registro.genero=gender.selectedIndex;
registro.mail=correo;
registro.nombreU=nomUs.text;
registro.piel=sk;
registro.pelo=h;
registro.peloc=hc;
registro.camiseta=s;
registro.camisetac=sc;
registro.pantalon=t;/*asigna valores al objeto registro q se almacenaran en la base de datos*/
salir=true;
registro.sendAndLoad
("http://localhost/phpflash/registro.php",recibo,"post");
nomUs=nomUs.text;
genero=gender.selectedIndex;
trace("selectoGenero");
piel=sk;
pelo=h;
peloc=hc;
camiseta=s;
camic=sc;
pantalon=t;
```

```
removeMovieClip("body");
removeMovieClip("head");
removeMovieClip("eyes");
removeMovieClip("shirt");
removeMovieClip("trousers");
removeMovieClip("hair");
removeMovieClip("stripes");
removeMovieClip("neck");
gotoAndStop(5);}
```

Tabla 12.Envío de los datos del personaje al servidor y limpieza del escenario.

3.1.2.2. Identificación

La identificación se realiza mediante la comprobación de correo electrónico y clave. El usuario que desea identificarse introduce ambos parámetros, y flash los envía al servidor.

```
function confirma(){
    var log:LoadVars= new LoadVars;
    log.mail=_root.mail.text;
    log.clave=_root.clave.text;
    log.sendAndLoad("http://localhost/phpflash/log",recibo,"POST");}
```

Tabla 13.Identificación frente al servidor.

Una vez allí se intenta localizar en la base de datos algún registro, cuya dirección de correo coincida con la aportada por el usuario, en caso de éxito se comprueba que la clave asociada a esa cuenta de correo es la introducida por el usuario.

Si ambas coinciden se rescatan los datos relativos al personaje de la base de datos y se pasan a flash, que se encarga de almacenarlos de forma que cualquier pelicula que se cargue en la raíz pueda acceder a ellos.

```
recibo.onLoad=function(exito){
    if (exito){
        var myNumber:Number = Number(this.mensaje);
        switch(myNumber){
            case 0:
            nombre.text="";
```

```
                    apellidos.text="";
                    mail.text="";
                    clave.text="";
                    _root.respuesta.text="correo no existente";
                    break;
                    case 1:
                    nombre.text="";
                    apellidos.text="";
                    mail.text="";
                    clave.text="";
                    respuesta.text="correo y clave no coinciden";
                    break;
                    case 2:
                    correo=_root.mail.text;
                    nomUs=this.nomUs;
                    genero=this.genero;
                    piel=this.piel;
                    pelo=this.pelo;
                    peloc=this.peloc;
                    camiseta=this.camiseta;
                    camic=this.camic;
                    pantalon=this.pantalon;
                    gotoAndStop(5);
                    break;
                }
            }
        }
```

Tabla 14. Gestión de la respuesta del servidor.

3.1.2.3. Paso de datos

El registro y la identificación tienen un punto en común. Al final ambos procesos saltan al quinto fotograma, desde el que se mandan las variables relativas al personaje a la pelicula raíz que cargó los procesos de registro e identificación.

```
        var sending_lc:LocalConnection = new LocalConnection();
        trace("nombre"+nomUs);
        trace("genero selecto"+genero);
        sending_lc.send("lc_name", "methodToExecu-
te",correo,nomUs,genero,piel,pelo,peloc,camiseta,camic,pantalon);
        unloadMovie("selecto.swf");
```

Tabla 15. Paso de datos a raíz y descarga.

Además realizan la descarga de la pelicula donde se ejecutan y dejan libre a la pelicula raíz para que pueda cargar otros archivos.

3.1.3 Sprites.

Un sprite es una imagen sobre la que se define animación ante algún evento. En esta aplicación el personaje que se utiliza es un sprite. Las características del mismo vienen definidas por los parámetros que el usuario definió durante la fase de registro.

La animación correspondiente al sprite en esta aplicación es la de andar en cualquiera de las ocho direcciones principales del espacio.

2.1.3.1 Creación del Sprite.

2.1.3.1.1 Inserción de imagenes

En primer lugar se crea un MovieClip sobre el que se carga cada una de las direcciones en las que se desea que el personaje ande. Cada dirección dispone de una película para cargar las cinco imágenes que componen el movimiento, cada pelicula está separada de la anterior 200 pixeles en la dirección vertical; que es la altura del personaje.

```
var sprite:MovieClip=bicho.createEmptyMovieClip("sprite",0);
var izquierda:MovieClip=sprite.createEmptyMovieClip("izquierda",2);
var derecha:MovieClip=sprite.createEmptyMovieClip("derecha",3);
var posterior:MovieClip=sprite.createEmptyMovieClip("posterior",4);
var frontdcha:MovieClip=sprite.createEmptyMovieClip("frontdcha",5);
var frontizda:MovieClip=sprite.createEmptyMovieClip("frontizda",6);
var backdcha:MovieClip=sprite.createEmptyMovieClip("backdcha",7);
var backizda:MovieClip=sprite.createEmptyMovieClip("backizda",8);
izquierda._y=frontal._y+200;
derecha._y=izquierda._y+200;
posterior._y=derecha._y+200;
frontdcha._y=posterior._y+200;
frontizda._y=frontdcha._y+200;
backdcha._y=frontizda._y+200;
backizda._y=backdcha._y+200;
```

Tabla 16. Películas sobre las que se cargan las imágenes de cada dirección.

A continuación para cada una de las direcciones se debe componer la secuencia de imágenes que compone el movimiento. En el caso que ocupa a este proyecto han sido necesarias cinco imágenes.

Ilustración 26. Imágenes que componen el movimiento de los brazos en la dirección avance vertical.

```
url=dir+"frontal/";
        for(i=0;i<=4;i++){
                brazos[i]=frontal.createEmptyMovieClip("brazos"+i,46-i);
                brazos[i]._x=(200*i);
                clipLoader.loadClip(url+"brazos/"+i+".gif",brazos[i]);
```

Tabla 17. Carga de las imágenes para la dirección de movimiento avance vertical.

En este ejemplo se muestra únicamente la carga de los brazos. La acción se repetirá para cada una de las peliculas que se necesiten para generar una dirección completa del sprite. En nuestro caso han sido necesarias ocho en las que se cargan, la cara, el pelo, las piernas, los brazos, la camisa, el cuello, las rayas y el pantalón. El motivo por el que se utilizan ocho peliculas son las opciones que corresponden a cada una de ellas, así como la ocultación que por motivos de profundidad hacen unas de otras. Por ejemplo si las rayas y la camisa estuvieran en la misma pelicula el cambio de color de las primeras implicaria que la camisa tambien cambiase de color; de la misma forma, si el cuerpo incluyese los brazos estos quedarían ocultos bajo la camisa.

Ilustración 27. Dirección avance vertical completa.

```
for(i=0;i<=4;i++){
brazos[i]=frontal.createEmptyMovieClip("brazos"+i,46-i);
        brazos[i]._x=(200*i);
        clipLoader.loadClip(url+"brazos/"+i+".gif",brazos[i]);
        cuello[i]=frontal.createEmptyMovieClip("cuello"+i,41-i);
        cuello[i]._x=(200*i);
        clipLoader.loadClip(url+"cuello/"+camisa+".gif",cuello[i]);//cuello
        rayas[i]=frontal.createEmptyMovieClip("rayas"+i,36-i);
        rayas[i]._x=(200*i);
        clipLoader.loadClip(url+"rayas/0.gif",rayas[i]);
                if(camic!=0){
                        colorCamisa(rayas[i],camic);}
                                //y el camblo de color de las rayas
        camiseta[i]=frontal.createEmptyMovieClip("camiseta"+i,31-i);
        camiseta[i]._x=(200*i);
        clipLoader.loadClip(url+"camisa/0.gif",camiseta[i]);
        pantalon[i]=frontal.createEmptyMovieClip("pantalon"+i,26-i);
        pantalon[i]._x=(200*i);
        clipLoader.loadClip(url+"pantalon/"+pantal+""+i+".gif",pantalon[i]);
        piernas[i]=frontal.createEmptyMovieClip("piernas"+i,21-i);
        piernas[i]._x=(200*i);
         clipLoader.loadClip(url+"piernas/"+i+".gif",piernas[i]);
        cuerpo[i]=frontal.createEmptyMovieClip("cuerpo"+i,16-i);
        cuerpo[i]._x=(200*i);
        clipLoader.loadClip(url+"cuerpo/0.gif",cuerpo[i]);
        pelo[i]=frontal.createEmptyMovieClip("pelo"+i,11-i);
        pelo[i]._x=(200*i);
        clipLoader.loadClip(url+"pelo/"+cabello+".gif",pelo[i]);//pelo
                if(peloc!=0){
                        colorPelo(pelo[i],peloc);}
        cara[i]=frontal.createEmptyMovieClip("cara"+i,6-i);
        cara[i]._x=(200*i);
        clipLoader.loadClip(url+"cara/0.gif",cara[i]);//cara
                if(piel!=0){
                        colorCara(cara[i],piel);}
                        }
```

Tabla 18.carga completa de una dirección.

Esta porción de código se repite 7 veces, una para cada dirección.

La creación de las películas donde se cargan las imágenes debe ser dinámica, es decir, se crean cuando se van a utilizar, eso implica que se crean en tiempo de ejecución, es por esto que se generan arrays de películas. Es necesario un array para cada película y para cada dirección

```
var cuerpo:Array= new Array;
var piernas:Array=new Array;
var brazos:Array=new Array;
var camiseta:Array=new Array;
var pantalon:Array=new Array;
var rayas:Array=new Array;
var cuello:Array=new Array;
var cara:Array=new Array;
var pelo:Array=new Array;
```

Tabla 19. Arrays para cada imagen que compone el personaje.

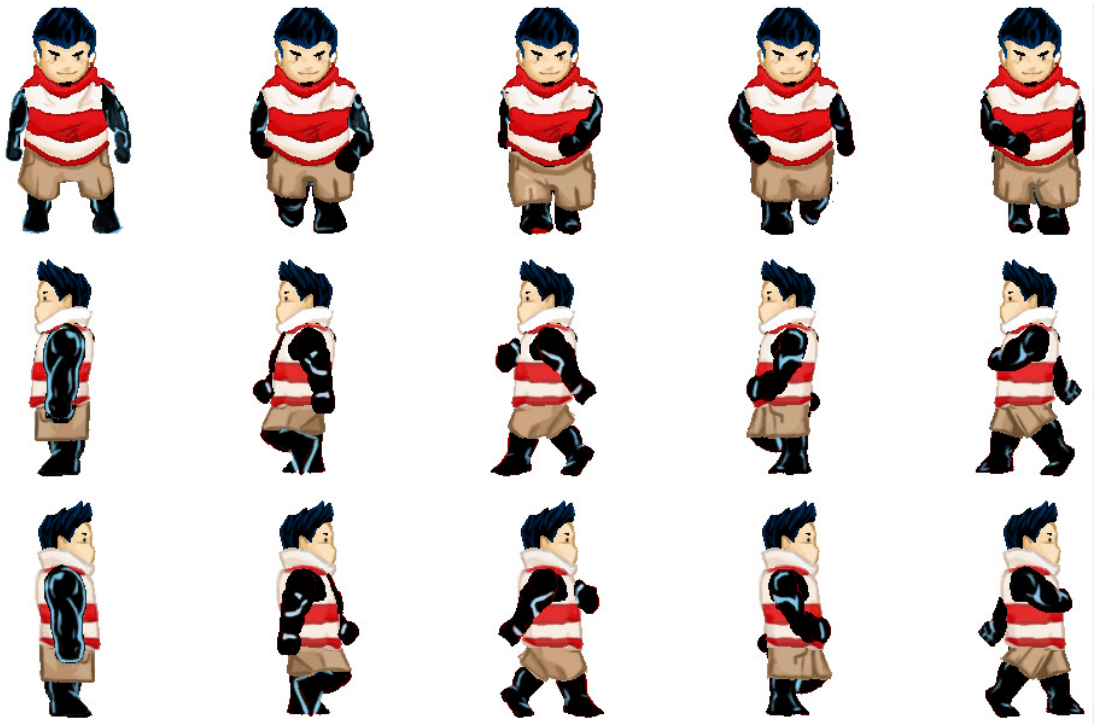

Ilustración 28.sprite sin enmascarar.

Una vez generado el sprite completo es necesario crear la ilusión de movimiento. Para hacerlo se utiliza una película que enmascara al sprite y solo permite la observación de una imagen.

.

Ilustración 29.personaje tras enmascarar el sprite.

```
var mascara:MovieClip=bicho.createEmptyMovieClip("mascara",2);
drawRectangle(mascara, 200, 200, 0x000000, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number, box-
Height:Number, fillColor:Number, fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(0, 0);
    lineTo(boxWidth, 0);
    lineTo(boxWidth, boxHeight);
    lineTo(0, boxHeight);
    lineTo(0, 0);
    endFill();
  }
}
sprite.setMask(mascara);
```

Tabla 20.método de enmascaramiento.

Ahora que el sprite está enmascarado es posible generar el movimiento. Para ello se detecta si hay alguna tecla pulsada y se mueve el sprite en consecuencia.

3.1.3.2 Interactividad

3.1.3.2.1 Movimiento

Se crea un objeto, teclado, que detecta si una tecla esta presionada y si lo está inicia el movimiento del sprite.

```
var teclado:Object = new Object();
teclado.onKeyDown=presionado;
Key.addListener(teclado);
function presionado(){
```

```
        if (Key.isDown(Key.LEFT)) {
                izi=true;
                correrrun();}
        if (Key.isDown(Key.UP)) {
                arriba=true;
                correrrun();}
        if (Key.isDown(Key.RIGHT)) {
                dere=true;
                correrrun();}
        if (Key.isDown(Key.DOWN)) {
                abajo=true;
                correrrun();}
        }
```

Tabla 21.funciones del objeto teclado para mover el sprite.

Antes de llamar a la función que realiza el movimiento, teclado modifica el valor de izi, arriba, dere o abajo, cuyo cometido es recordar si la tecla respectiva está presionada. Ya que flash no detecta la pulsación de dos teclas a la vez, es necesario el uso de estas banderas.

Correrrun comprueba si se ha dejado de presionar alguna tecla para modificar las banderas o detener el movimiento. Además si el sprite no se estaba moviendo llamará periodicamente cada cien milisegundos a la función ct.

```
var correrrun = function () {
        teclado.onKeyUp=comprueba();
   if (!isrunning) {
      mov = setInterval(ct, 100);
                teclado.onKeyUp=comprueba();
   }
};
```

Tabla 22. Función correrun que detecta las pulsaciones.

ct modifica la posición vertical del sprite lo que provoca que se vea una dirección de movimiento u otra.

Siempre que haya una tecla presionada llamará a la función correr.

```
var ct = function () {
   isrunning = true;
   teclad.onKeyUp=comprueba();
   //corriendo hacia abajo
        if(abajo==true&&arriba==false&&dere==false&&izi==false){
                sprite._y=0;
                if (Key.isDown(Key.DOWN)) {
        correr();
      } else {
        dejardecorrer();}
        }
        if(abajo==false&&arriba==false&&dere==true&&izi==false){
                sprite._y=-400;
                if (Key.isDown(Key.RIGHT)) {
        correr();
      } else {
        dejardecorrer();
      }
        }
```

Tabla 23. Ejemplo de las direcciones abajo y derecha.

Cuando se desea un movimiento en diagonal se hace uso de las banderas comentadas con anterioridad. En este caso para saltar a la línea que describe en movimiento hacia abajo y a la izquierda es necesario que abajo e izi sean verdaderas

```
if(abajo==true&&arriba==false&&dere==false&&izi==true){
                sprite._y=-1000;
                if (Key.isDown(Key.DOWN)) {
        correr();
      } else {
        dejardecorrer();
      }
        }
```

Tabla 24. Ejemplo de la dirección abajo y a la izquierda.

Dentro de la función ct también debe contemplarse el caso de que no haya ninguna tecla pulsada parando entonces el movimiento.

```
if(abajo==false&&arriba==false&&dere==false&&izi==false){
        dejardecorrer();
    }
```

Tabla 25. función a la que se llama cuando no hay teclas pulsadas

La función correr se encarga del desplazamiento horizontal del sprite, lo que provoca la sensación de movimiento. El sprite se desplaza en incrementos de doscientos píxeles, espacio que ocupa el personaje, de esta forma las imágenes no se solapan. Si la posición del sprite es tal que se está mostrando la última imagen de una fila se le obliga a volver a la posición inicial. La multiplicación que se hace es por cuatro quintos debido a que cada fila tiene cinco imágenes.

```
var correr = function () {
   if (sprite._x>0-sprite._width*(4/5)) {
      sprite._x -= 200;
   } else {
      sprite._x = -200; }
};
```

Tabla 26. función correr.

Cuando se sueltan todas las teclas se llama a la función dejardecorrer que devuelve el sprite a su posición inicial y hace que se deje de llamar a la función ct.

```
var dejardecorrer = function () {
   isrunning = false;
   sprite._x = 0;
   clearInterval(mov);
};
```

Tabla 27. función dejardecorrer.

2.1.3.2.2. Relación con otros elementos

La relación del sprite con otros elementos se consigue mediante una serie de ocho películas que rodean al personaje y detectan su colisión con otros elementos sensibles.

Los detectores de colisiones actúan en dos sentidos. En primer lugar se encargan de mantener al personaje dentro de los límites lógicos donde debería estar, jamás se permitirá que ande por las paredes o en el vacío. En segundo

lugar ofrece la posibilidad de relación con otros elementos de la película mediante el uso de cuadros de diálogo. Éstos se lanzarán cada vez que el personaje entre en el área activa de algún objeto como pueda ser un armario que deba abrirse, algún otro personaje con el que se hable o se quiera cambiar de habitación.

Existe un detector para cada una de las direcciones de movimiento ya que se debe obtener una respuesta distinta para cada dirección. No es lo mismo que el personaje alcance el extremo inferior de una habitación, suceso que obliga al personaje a que no pueda andar hacia abajo, que llegue a la parte superior de la misma en cuyo caso si podría bajar

La misma lógica se aplica a la relación con otros elementos del escenario, es distinto acceder a una habitación que se encuentra a la derecha que a una que se encuentra abajo.

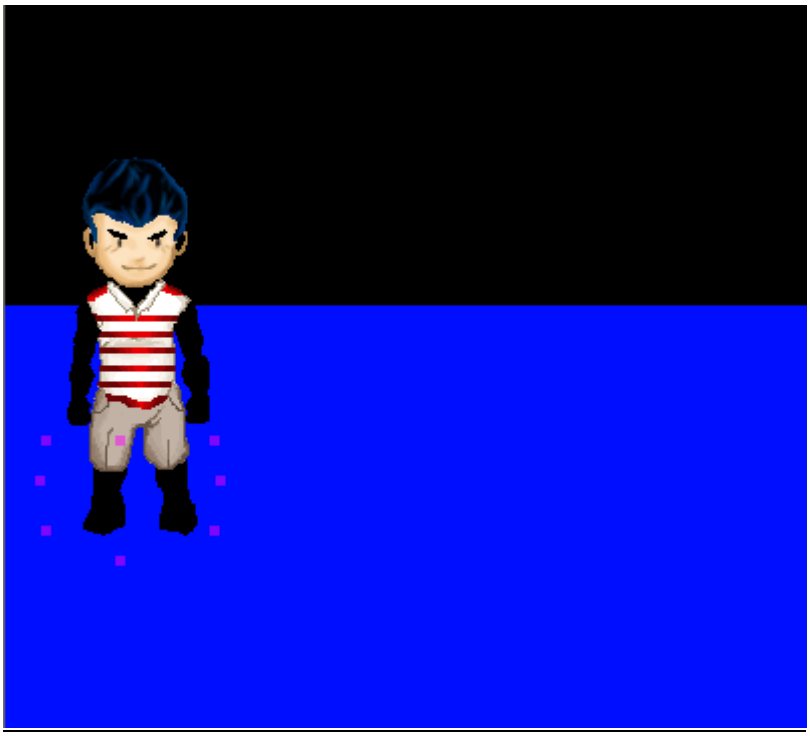

Ilustración 30. personaje con detectores de colisiones visibles.

Cada uno de los cuadros morados es un detector de colisiones y el área azul define el suelo. Si alguno de los detectores excede sus límites no se permite al personaje andar en esa dirección.

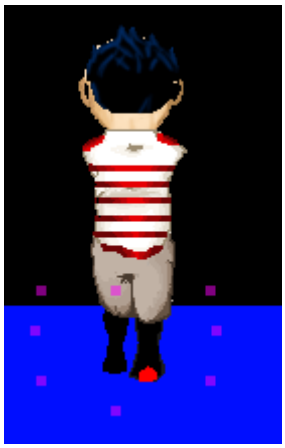

Ilustración 31. Exceso del límite superior del escenario

En cuanto a los detectores que accionan el cambio de habitación, en la siguiente figura se puede observar su mecanismo de funcionamiento. La salida de la habitación está situada a la izquierda, por tanto cuando el personaje sitúe sus detectores de colisiones de la izquierda sobre la franja roja se producirá el cambio de habitación. Sin embargo si el personaje entró a la habitación desde la izquierda podría tocar el detector para el cambio de habitación nada más entrar, lo que produce que jamás consiguiera entrar. Por esta razón se incluye la línea morada, que impide que se active el cambio de habitación hasta que se pase sobre ella. Así antes de salir de la habitación el personaje habrá podido explorarla.

Ilustración 32. Detectores para el cambio de habitación.

De forma análoga actúa la zona sensible de la siguiente imagen, cuando el sprite alcance el rectángulo verde se activará un cuadro de diálogo que informa de la obtención de un objeto.

Ilustración 33. otras zonas sensibles.

En la ilustración 34 se puede comprobar la activación de un cuadro de diálogo que ofrece información sobre los objetos encontrados.

Ilustración 34. Cuadro de diálogo

3.1.4. Escenarios.

La inclusión de las figuras geométricas vistas en el apartado anterior dotan al juego de funcionalidad, sin embargo no contribuyen a generar una estética atractiva.

Por esta razón se ha creado una plantilla para generar escenarios.

3.1.4.1. Tipos de escenario.

3.1.4.1.1. Escenarios sin scroll.

Se trata de aquellos cuyo tamaño es igual al de la pantalla que se visualiza. En ellos cuando el sprite alcanza el final de la pantalla se limitará su movimiento o se producirá un cambio de escenario.

El método utilizado es la división del escenario en movieClips de tamaño regular sobre los que se cargan las losas deseadas. Esta forma de trabajo aporta gran flexibilidad y permite la creación de cualquier escenario combinando un número limitado de losas.

```
contador=0;
for(i=0;i<=8;i++){
        tile[i]=new Array;
        for(j=0;j<=8;j++){
                contador+=1;
                tile[i][j]=stage.createEmptyMovieClip("tile"+i+j,257-contador);
                tile[i][j]._x=i*50;
                tile[i][j]._y=j*50;
                }
```

Tabla 28. División del escenario en movieClips regulares.

Cada clip se crea a una profundidad distinta de forma que no existan colisiones entre ellos.

El tamaño de la pantalla de visualización es de cuatrocientos por cuatrocientos pixeles y cada losa mide cincuenta por cincuenta pixeles, de ahí que sean necesarias 64 para cubrir la totalidad del área visible.

El área de visualización es cuadrada por lo que la forma más sencilla de rellenar el espacio es utilizar imágenes cuadradas, lo que implica que la

representación de un elemento en diagonal no puede realizarse con una única imagen, son necesarias dos, representando cada una de ellas la parte superior e inferior respectivamente de la forma que se quiera obtener.

Ilustración 35. Tiles y composición de un tile rectangular.

Sin embargo esto no es posible si se dispone únicamente de un clip donde cargar las imágenes, por esta razón cuando sea necesario se creará una matriz de peliculas igual a la matriz tiles.

```
	for(i=0;i<=8;i++){
		aux[i]=new Array;
		for(j=0;j<=8;j++){
			contador+=1;
			aux[i][j]=stage.createEmptyMovieClip("aux"+i+j,3*257-
contador);
			aux[i][j]._x=i*50;
			aux[i][j]._y=j*50;
			}
	}
```

Tabla 29. Creación de la matriz de tiles auxiliares.

3.1.4.1.2. Escenarios con scroll.

Son aquellos cuyo tamaño supera el área de visualización, al cargarse no pueden mostrarse en su totalidad pese a que el sprite se moverá con libertad a través de ellos.

A diferencia de los escenarios sin scroll el alcance del límite del área de visualización no provocará el salto a un nuevo escenario o la inmovilización del sprite. Por el contrario se produce un desplazamiento del escenario. Así se consigue el efecto de que el sprite avanza a través de una estancia larga.

En la siguiente porción de código se observa el funcionamiento del scroll. Si los detectores diestros del sprite alcanzan a la película límite mientras se tenga pulsado el cursor derecho, la posición del escenario se desplazará hacia la izquierda en incrementos finitos. Además se le impone una limitación que evita que a causa del desplazamiento puedan existir espacios en blanco en el área de visualización.

```
limite.onEnterFrame = function(){
        scroll1 = this.hitTest(sombra[1]);
        scroll2= this.hitTest(sombra[2]);
        scroll3=this.hitTest(sombra[3]);
        sortie1 = suelo.hitTest(sombra[1]);
        sortie2= suelo.hitTest(sombra[2]);
        sortie3=suelo.hitTest(sombra[3]);
if(stage._x>-795){
        if(sortie1==true||sortie2==true||sortie3==true){
                if(Key.isDown(Key.RIGHT)){
                        if(scroll1==false||scroll2==false||scroll3==false){
                                //trace(stage._x);
                                stage._x-=5;
                                bicho._x-=2.66;
                        }
                }
        }
}
if(stage._x<=0){
        if(Key.isDown(Key.LEFT)){
                if(scroll1==false||scroll2==false||scroll3==false){
                stage._x+=5;
                bicho._x+=2.54;
                }
        }
}
}
```

Tabla 30. Funciones para scroll horizontal.

3.1.4.1.3. Escenarios con algún tipo de animación

Flash es una potente herramienta de animación, sin embargo debido a la forma de generar los escenarios es imposible utilizar las utilidades que nos proporciona. Cualquier animación que se realice debe programarse, bien a través de bucles que muestren varias veces el mismo fotograma con cambios en las imágenes cargadas en él, bien a través del salto controlado entre fotogramas.

En el siguiente código se comprueba hacia donde debe girar el puente y se mueve en esa dirección, dejando que la cabeza lectora de la pelicula salte al siguiente fotograma y se ejecuten sus acciones. En el siguiente fotograma se compara la posición del puente con el giro que se quiere conseguir, en caso de no lograrlo la pelicula vuelve al fotograma anterior, donde se modifica la posición del puente. Este ciclo se repite hasta que el puente complete el giro deseado.

```
if (giro>0){
        puente._rotation+=0.5;
        cuadrado1._rotation+=0.5;

        } else{
                        puente._rotation-=0.5;
                        cuadrado1._rotation-=0.5;
                        }
```

Tabla 31. código para el giro del puente.

```
giro=final-original;
//trace(final);
if(puente._rotation==final){
        original=final;
        stop();}
        else
                gotoAndPlay(3);
```

Tabla 32. Código para comprobar si el puente debe o no seguir girando.

3.1.4.2. Escenario y Sprites.

Hasta ahora se había hablado de los escenarios y de los sprites por separado, pero la interacción entre ambos es la que dota a la aplicación de jugabilidad.

Como se comentó en la sección relativa a la interactividad de los sprites, se producen eventos cuando los detectores de colisiones del sprite entran en el área activa de los elementos sensibles del escenario. A continuación se describe la gestión de tales eventos.

Las zonas sensibles son movieClips dotados de superficie que están siempre preparadas para detectar una colisión. Esto se consigue mediante el controlador onEnterFrame(), que repite la función que se le asigna con una frecuencia igual a la velocidad de reproducción. Así se comprueba de forma constante si el sprite está sobre alguna zona sensible.

```
armario.onEnterFrame=function(){
        cerrado0=this.hitTest(sombra[0]);
        cerrado1=this.hitTest(sombra[1]);
        cerrado2=this.hitTest(sombra[2]);
        cerrado3=this.hitTest(sombra[3]);
        if(cerrado0==true||cerrado1==true||cerrado2==true||cerrado3==true)
                cerrado=false;
        if(!cerrado)
        gotoAndPlay(7);
        }
```

Tabla 33. Activación de un área sensible mediante el contacto de los detectores de colisiones.

Las variables cerrados son booleanos que recogen si los detectores de colisiones del sprite, sombra[i], están sobre el área activa de la película.

La misma lógica se aplica a la superficie que define el suelo del escenario.

```
suelo.onEnterFrame = function() {
        correU = this.hitTest(sombra[0]);
        correUR = this.hitTest(sombra[1]);
        correR= this.hitTest(sombra[2]);
        correDR=this.hitTest(sombra[3]);
```

```
correD=this.hitTest(sombra[4]);
correDL=this.hitTest(sombra[5]);
correL=this.hitTest(sombra[6]);
correUL=this.hitTest(sombra[7]);
}
```

Tabla 34. Limitación del desplazamiento del sprite debido a la finalización del suelo.

La función que hace que el sprite se desplace por el escenario es exactamente igual a la que se utilizó para el movimiento, descrita en las tablas veinte a veinticinco, con la excepción de que en lugar de mover el sprite, desplaza la parte visible del sprite en incrementos finitos.

```
if(abajo==true&&arriba==false&&dere==false&&izi==false){
                if (Key.isDown(Key.DOWN)) {
                        if(correD){
    bicho._y+=5;}
  } else {
    dejardecorre();
  }
```

Tabla 35. Función para el desplazamiento del personaje.

Se puede observar que el desplazamiento está gobernado además de por las teclas pulsadas por una variable llamada correD. Existe una de estas variables para cada dirección de movimiento y su valor viene determinado por la posición del sprite en el escenario. Si los detectores de colisiones del sprite dejan en algún momento la superficie del suelo, el valor de estas variables se hace false y el sprite no puede avanzar en esa dirección, aunque sí en las otras.

Flash otorga a todos los movieClips área rectangular independientemente de su forma. Esto supone un problema, ya que en caso de tener un escenario con cualquier otra forma no es posible limitar el movimiento con una sóla película.

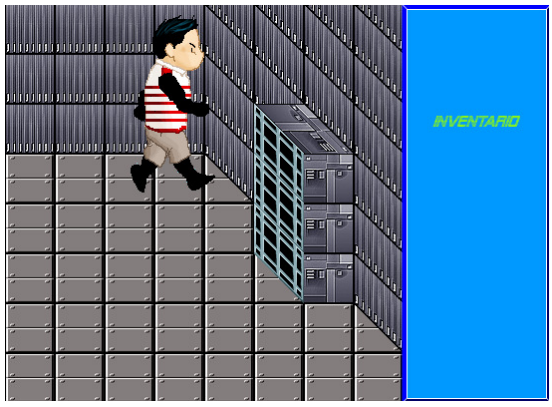

Ilustración 36. Habitación con límites no rectangulares

En la siguiente imagen se muestra el área sensible total sobre la que se implementa el suelo. Está compuesta por un cinco movieClips, el mayor llamado suelo y cuatro menores situados a la derecha y almacenados en un array llamados losa[i]. Para comprobar si el sprite deber detener su avance se debe comprobar si ha rebasado el límite en cada una de las películas.

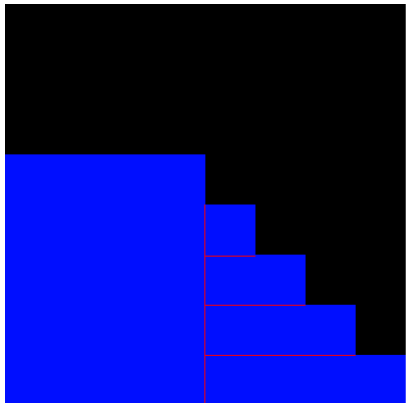

Ilustración 37. Estructura del suelo para límites no rectangulares

```
suelo.onEnterFrame = function() {
        correU = this.hitTest(sombra[0]);
        if(!correU){
                correU=losa[0].hitTest(sombra[0]);
                if(!correU){
                        correU=losa[1].hitTest(sombra[0]);
                        if(!correU){
                                correU=losa[2].hitTest(sombra[0]);
                                if(!correU){
                                        correU=losa[3].hitTest(sombra[0]);
                                        if(!correU){
                                        correU=losa[4].hitTest(sombra[0]);}}}}}
```

Tabla 36. limitación del movimiento combinando varios elementos sensibles.

3.1.4.3. Inventario.

Cada usuario debe tener la posibilidad de visualizar los elementos que componen su inventario en cualquier momento del juego. Para ello cada vez que se carga un escenario se realiza una conexión a la base de datos para extraer los objetos recolectados.

Por tanto el escenario hace una conexión a la base de datos especificando la dirección del correo eléctrónico del usuario que activo que está solicitando los datos de su inventario. A continuación hace uso de una función para ordenar los datos recibidos desde PHP.

```
envioDatos=function(){
        invEnvio.sendAndLoad("phpflash/invRec2.php",vacia,"POST");
        parseo();
}
```

Tabla 36. Conexión para extraer el inventario

Cada usuario dispone de un inventario distinto, por tanto el primer paso al invocarlo es rescatar de la base de datos los datos de los objetos que tiene. Los objetos son almacenados en la base de datos del servidor en tablas según su tipo, y cada uno de ellos tiene unas características distintas, cómo son su número o su valor. La base de datos devuelve estos valores en forma de matriz, pero flash no puede recibir matrices desde PHP, por esta razón se crea un objeto XML cuyas etiquetas permiten ordenar la información de forma similar a cómo se dispone en una matriz. El objeto XML, junto con el número de líneas que la consulta a la base de datos devolvió permite disponer los datos en pantalla una vez que se hayan generado las matrices donde se almacenan los datos. Por esta razón al llamar a la función creaArray se especifican el numero de líneas y resistencias que contendrán las nuevas matrices.

```
parseo=function(){
        var numTLin:Number;
        var longTLin:Number;
        var impTLin:Number;
        var numTOtro:Number;
        var valorTOtro:Number;
        vacia.onLoad=function(){
```

```
                numeroLineas=Number(this.numLineas);
                numeroResistencias=Number(this.numResistencias);
                myXML.ignoreWhite=true;
                myXML.parseXML(vacia.respuesta.toString());
                creaArray(numeroLineas,numeroResistencias,myXML);
        }
}
```

Tabla 37. preparación de los datos recbidos desde el servidor.

La función creaArray genera dos matrices. La primera de ellas contiene las líneas y la segunda las resistencias, por ser estos los únicos elementos que se utilizan en esta parte del juego. Para rellenar las matrices se recorre de forma ordenada el objeto XML extrayendo de él los valores respectivos a cada uno de los elementos. La razón de los índices es que el primer elemento del objeto XML no tiene valores legales y existe porque se utiliza para inicializar la base de datos. Las matrices se duplican y el duplicado sirve como elemento de comparación para detectar cuando se a producido un cambio y actualizar la base de datos en consecuencia.

```
creaArray=function(numeroLinea:Number,numeroResistencia:Number,param:XML){
        myXML=param;
                for(i=1;i<numeroLinea;i++){
                lineas[i]=new Array;
                segLineas[i]=new Array;
                for(j=0;j<=2;j++){

        lineas[i][j]=Number(myXML.childNodes[i].childNodes[j].firstChild.nodeValue);

        segLineas[i][j]=Number(myXML.childNodes[i].childNodes[j].firstChild.nodeValue);
                        }
        }
        for(i=1;i<numeroResistencia;i++){
                p=numeroLinea+1;
                resistencias[i]=new Array;
                segResistencia[i]=new Array;
                for(j=0;j<=1;j++){
                        resisten-
cias[i][j]=myXML.childNodes[p].childNodes[j].firstChild.nodeValue;
                        segResisten-
cia[i][j]=myXML.childNodes[p].childNodes[j].firstChild.nodeValue;
                }}}
```

Tabla 38. Creación de matrices a partir de la información del servidor

El siguiente cuadro se muestran en rojo los nodos a ignorar y en azul el segundo hijo.

```
<?xml version="1.0"?><linea><numero>0</numero><longitud /><valor
/></linea><linea><numero>9</numero><longitud>9999</longitud><valor>9999</valor></linea>
<linea><numero>3</numero><longitud>20</longitud><valor>9999</valor></linea>
<resistencia><numero>0</numero><valor
/></resistencia><resistencia><numero>6</numero><valor>50</valor></resistencia>
```

Tabla 37. XML respuesta desde el servidor a las consultas.

2.1.4.4. Registro automático.

Cada vez que el usuario consiga un hito importante en el juego, normalmente la resolución de un circuito especialmente complicado, se realiza un registro en la base de datos del estado de ese usuario. Se escribe el numero de objetos que tiene así cómo el nivel en el que debe aparecer la próxima vez que inicie sesión.

Esto implica que la base de datos sólo se actualiza cuando se supera una fase en lugar de escribir sobre ella cada vez que se utiliza un elemento del inventario. Por tanto es necesaria una forma de contabilizar los cambios que se producen en el inventario para, posteriormente, introducir los cambios en la base de datos.

```
	var receiving_lc2:LocalConnection = new LocalConnection();
	receiving_lc2.methodToExecute = function(param1:Number, param2:Number)
{
		for(i=0;i<numeroLineas;i++){
			if(lineas[i][1]==param1&&lineas[i][2]==param2){
				lineas[i][0]--;
				trace("hola recibo cosas"+lineas[i][0]);
				numeLinea=lineas[i][0];
				}
			}
	};
	receiving_lc2.connect("actLinea");
```

Tabla 38. Actualización de las matrices

En esta sección de código se observa la actualización de las matrices que componen el inventario. Cuando un objeto es utilizado el inventario notifica al escenario que lo invocó de este hecho, mandando un mensaje en el que se especifican las características del objeto usado. El escenario recoge estas características y recorre la matriz correspondiente hasta encontrar una coincidencia, en este momento decrementa en uno el número de unidades de este objeto.

Para realizar el registro se hace uso de una función recurrente.

```
var i:Number=0;
recorrer(i);
```

Tabla 39. Llamada a la función recurrente

La función se encarga de recorrer las matrices que almacenan un tipo de objeto en concreto. Tanto la matriz que se ha ido actualizando con las notificaciones del inventario como la matriz de seguridad que se comentó en el apartado del inventario. En cada paso se compara el número de elementos de cada objeto, si existe coincidencia se incrementa en uno el parámetro que se le pasa a la función y se vuelve a hacer uso de ella. Sin embargo si no existe coincidencia se utiliza una nueva función que permite la escritura en base de datos. En este caso el incremento del parámetro que permite el recorrido de las matrices sólo se realiza cuando hay una respuesta desde el servidor. Si no se hace de esta forma no se de tiempo para la escritura en la base de datos y sólo se actualizaría el ultimo elemento de la matriz. Una vez alcanzado el límite de la matriz se llaman a otras funciones cuyo cometido es el mismo pero recorren otras matrices.

El control sobre al avance en el recorrido es la razón de que se haya utilizado una función recurrente en lugar de un bucle, ya que el bucle es tan rápido que no es posible escribir cada elemento cambiado en la base de datos.

```
recorrer=function(j:Number){
        if(j<numeroLineas){
                if (lineas[j][0]!=segLineas[j][0]){
                        escribeLineas(lineas[j][0],lineas[j][1],lineas[j][2]);
                        inveRec.onLoad=function(exito){
                        if (exito){
```

```
                                        var myNumber = (this.id);
                                        var sos=this.control;
                                        j++;
                                        p=j;
                                        recorrer(p);
                                }
                        }
                }else{
                        j++;
                        p=j;
                        recorrer(p);}
                }else{
                        i=1;
                        recorrerR(i);}
        }
```

Tabla 40. Función para la escritura en base de datos.

En el momento que se termina de recorrer el último elemento de la última matriz se llama a una función que se encarga de eliminar de la base de datos todos aquellos registros cuyo número de unidades sea cero.

```
actualiza=function(){
        var inveLog:LoadVars=new LoadVars;
        inveLog.mail=correo;
        inveLog.sendAndLoad("phpflash/actualizar.php",inveRec,"POST")
                inveRec.onLoad=function(exito){
                        trace("hex");
                }
        }
```

Tabla 41. Notificación al escenario del una cambio en el inventario

3.1.5. Cuadros de diálogo.

Su cometido es informar al usuario de eventos relevantes, cómo la recolección de objetos o la comunicación con personajes del juego.

Siempre se cargan desde un escenario y de él reciben las variables para su generación.

El cuadro de diálogo muestra una imagen y un texto asociado a ella.

La selección del texto que se debe mostrar viene determinada por dos variables cuyo valor se define en la pelicula que invoca al cuaodro de diálogo.

La primera de esas variables define que personaje está hablando y la segunda que frase dirá.

Cada personaje dispone de un archivo de texto con distintas varibles que representan cada una de las frases que utiliza.

```
var cTexto:LoadVars=new LoadVars();
urltexto=_parent.urltexto;
cTexto.load(urltexto);
dialogo=_parent.dialogo;
cTexto.onLoad=function(){
if (dialogo==0)
                textoorl.text=cTexto.var0;}
```

Tabla 41. Obtención de variables para el texto a mostrar.

Por otro lado se muestra una imagen que corresponde al personaje que habla. La distinción entre personajes también viene determinada desde la película que invoca al cuadro de diálogo. Las posibles imágenes a cargar representan a personajes jugadores y a personajes no jugadores. Mediante la variable género se distingue entre unos y otros. Para los personajes jugadores deben especificarse los mismos parámetros que se definieron en la creación, por esta razón la película foto tiene bajo ella otras películas sobre las que se muestran el pelo, la camisa, u otros compenentes.

```
var genero:Number=_parent.genero;
var foto:MovieClip=_parent.contenedor1.createEmptyMovieClip("foto",0);
var pelo:MovieClip=foto.createEmptyMovieClip("pelo",1);
var cara:MovieClip=foto.createEmptyMovieClip("cara",0);
var cuello:MovieClip=foto.createEmptyMovieClip("cuello",4);
var rayas:MovieClip=foto.createEmptyMovieClip("rayas",3);
var camisa:MovieClip=foto.createEmptyMovieClip("camisa",2);
var clipLoader:MovieClipLoader = new MovieClipLoader();
if(genero==1)
switch (genero){
        case 0: url="flash/faces/chica/";
                piel=_parent.piel;
                cabello=_parent.cabello;
                peloc=_parent.peloc;
```

```
            camiseta=_parent.camisa;
            camic=_parent.camic;
        clipLoader.loadClip(/*url*+*/"flash/faces/chica/cara.gif",cara);
            clipLoader.loadClip(url+"rayas.gif",rayas);
            clipLoader.loadClip(url+"camisa.gif",camisa);.
            clipLoader.loadClip(url+"pelo/"+cabello+".gif",pelo);
            clipLoader.loadClip(url+"cuello/"+camiseta+".gif",cuello);
            colorCara(cara,piel);
            colorPelo(pelo,peloc);
            colorCamisa(rayas,camic);
            break;
        case 1: url="flash/faces/chico/";
            piel=_parent.piel;
            cabello=_parent.cabello;
            peloc=_parent.peloc;
            camiseta=_parent.camisa;
            camic=_parent.camic;
            clipLoader.loadClip(url+"cara.gif",cara);
            clipLoader.loadClip(url+"rayas.gif",rayas);
            clipLoader.loadClip(url+"camisa.gif",camisa);
            clipLoader.loadClip(url+"pelo/"+cabello+".gif",pelo);
            clipLoader.loadClip(url+"cuello/"+camiseta+".gif",cuello);
            colorCara(cara,piel);
            colorPelo(pelo,peloc);
            colorCamisa(rayas,camic);
            break;
        case 2: url="flash/faces/misc";
            indentificador=_parent.identidad;
            clipLoader.loadClip(url+""+indentificador+".gif",cara);
            break;
        case 3: url="flash/faces/piratas";
            url="flash/faces/misc";
            indentificador=_parent.identidad;
            clipLoader.loadClip(url+""+indentificador+".gif",cara);
            break;
        }
```

Tabla 42. Código para la selección y carga de la imagen a mostrar.

3.1.6. Inventario.

Durante el desarrollo del juego el personaje irá coleccionando objetos que le permitirán resolver circuitos o le ayudarán en su singladura. Para tener un registro de estos objetos es necesario un inventario.

El inventario desarrollado permite su proyección en pantalla, el acceso a ciertos objetos cuando esté permitido y la actualización de los arrays comentados con anterioridad cuando los objetos se hayan usado.

Los índices del bucle son de esta forma debido a que cuando un nuevo usuario se registra sus tablas de objetos se inicializan con un objeto vacío que debe ser ignorado.

```
for(i=1;i<numeroLineas;i++){
                j=i;
                miClip[j]=_root.createEmptyMovieClip("miClip"+i,500-j);
                numTLin=Number(_parent.lineas[i][0]);
                longTLin=Number(_parent.lineas[i][1]);
                impTLin=Number(_parent.lineas[i][2]);
                miClip[j]._y=10+12*i;
                miClip[j].soyBoton(numTLin,longTLin,impTLin);
}
```

Tabla 43. Creación dinámica de los botones para las líneas del inventario.

Este bucle se encargá de mostrar en pantalla los objetos de una tabla distinta a la del bucle anterior. Por ello las películas creadas tienen una posición horizontal distinta. Nuevamente la razón de los índices en el bucle es evitar el elemento vacío.

```
for(i=1;i<numeroResistencias;i++){
        j=numeroLineas+i;
        miClip[j]=_root.createEmptyMovieClip("miClip"+j,500-j);
        numTOtro=Number(_parent.resistencias[i][0]);
        valorTOtro=Number(_parent.resistencias[i][1]);
        trace("RESISTENCIA"+_parent.resistencias[i][0]);
        miClip[j].soyBoton2(numTOtro,valorTOtro);
        miClip[j]._y=20+12*(j-numeroLineas+1);
        miClip[j]._x=128;}
```

Tabla 44. Creación dinámica de los botones para las resistencias del inventario.

Cuando se invoca al inventario se muestran en pantalla varias columnas ordenadas con los objetos que el jugador ha ido obteniendo. La forma más sencilla de acceder a ellos es crearlos como botones, lo que permite que se haga click sobre ellos y tengan un área activa. Sin embargo esto plantea un problema, flash no permite la creación dinámica de botones y puesto que a priori se desconoce la cantidad y tipo de objetos que el usuario ha recogido es imposible crear los botones en otro momento que no sea la ejecución del script. No obstante la creación de películas sí es posible en tiempo de ejecución, aunque estas no disponen de las mismas propiedades que los botones, por esta razón se ha hecho uso del método prototype que permite añadir nuevas funciones a la clase MovieClip y que de esta forma se comporten como botones.

Cada botón representa uno de los objetos recogidos, por tanto debe especificarse tanto su valor o tipo cómo su número, estos son los argumentos que recibe la función que transforma las películas en botones.

Los botones deben reaccionar cuando se pulse sobre ellos, y poder cambiar cuando el puntero del ratón pase sobre su superficie activa. Estas funciones son las que se han añadido mediante prototype.

En primer lugar se crea la imagen del botón, compuesta por una pequeño rectángulo azul y campos de texto que especifican el valor del objeto y su cantidad.

A continuación se desarrollan las funciones. Las funciones onRollOver y onRollOut se activan cuando el ratón entra en el área activa del botón o cuando sale, cambiando su puntero. Las funciones onMouseUp y onMouseDown se ejecutan cuando se hace click sobre el botón siempre que el número de elementos de los que se disponga sea mayor que cero. Normalmente corresponden a un envío de información a la película que llamó al inventario. En este caso la información que se envía puede ser fija o varible. Si es variable se salta a un fotograma en el que se introducen los datos a enviar.

Como se comentó con anterioridad siempre que se utilice un objeto se informa al escenario desde el que se cargó en inventario.

```
MovieClip.prototype.soyBoton2 = function(info:Number,info2:Number){
        this.beginFill(0x000f0ff,100);
        this.moveTo(9,this._y+6);
        this.lineTo(9,this._y+12);
```

```
            this.lineTo(17,this._y+12);
            this.lineTo(17,this._y+6);
            this.lineTo(9,this._y+6);

            this.var1=info;
            this.var2=info2;
            texto1=this.createTextField("texto1",2,20,this._y,40,16)
            texto2=this.createTextField("texto2",1,70,this._y,40,16)
            texto1.textColor=0xFFFFFF;
            texto2.textColor=0xFFFFFF;
            if(this.var2==9999){
                  texto1.text="variable";}
            if(this.var2==0){
                  texto1.text="Zo";}
            if(this.var2==1){
                  texto1.text="2·Zo";}
            if(this.var2==2){
                  texto1.text="√Zo";}
            if(this.var2==3){
                  texto1.text="1/2·Zo";}
            if(this.var2==4){
                  texto1.text="3·Zo";}
            if(this.var2==5){
                  texto1.text="1/4·Zo";}
            texto2.text=this.var1;
      if(habilitado){
            this.onRollOver = function(){
            }
            this.onRollOut = function(){

            }
            this.onMouseDown = function(){
                  if(this.hitTest(_root._xmouse,_root._ymouse,true)){
                        this.oIsDown = true;
                        this._x += mover;
                        this._y += mover;
                        trace("Has hecho Click")
                  }
            }
            this.onMouseUp = function(){
                  if(this.oIsDown){
                        this.oIsDown = false;
```

```
                                this._x -= mover;
                                this._y -= mover;
                                if(this.var1>=0){
                                this.var1--;
                                this.texto2.text=this.var1;      }
                        var sending_lc2:LocalConnection = new LocalConnection();
                        send-
ing_lc2.send("actResistencia","methodToExecute",this.var2);


                                if(this.var2==9999){
                                        if(this.var1==0){
                                this.removeMovieClip;}
                                        gotoAndStop(6);
                                        }else{
                                                if(this.var1==0){
                                                this.removeMovieClip;}
                                                envioR(this.var2);}


                        }
                }
        }
        }
```

Tabla 45. Creación dinámica de un botón mediante la adición de nuevos eventos a traves de prototype.

Al pulsar en el botón azul se hace llamada a la siguiente función, que rellena los campos necesarios para hacer el envío de información.

```
        on(release){
                envioR(impedanciaResistencia.text);}
```

Tabla 46. Comunicación con la película que llamó al inventario.

Ilustración 38. Cuadro para la inserción de valores.

3.1.7. Circuitos.

Cómo se comentó en el apartado Objetivos la razón de este proyecto es disponer de una nueva herramienta didáctica, por tanto es fundamental que el desarrollo de la historia, que es el hilo conductor del juego, esté ligado a la evaluación de conocimientos de los usuarios.

Por este motivo para avanzar en el juego es necesaria la aplicación práctica de conocimientos de microondas por medio de la resolución de puzzles que interactúen con los escenarios.

Los clips de película de circuitos tienen una diferencia importante respecto al resto. Cómo se comentó al comienzo la sección tres se ha hecho uso casi exclusivo de ActionScript para la creación de cada clip, tanto del desarrollo gráfico, mediante cargas dinámicas, como de la funcionalidad de los elementos del juego. Sin embargo los circuitos no hacen uso de ActionScript para definir su aspecto gráfico, en lugar de eso se han importado a cada clip las distintas animaciones generadas por HFSS, sin realizar ningún tipo de carga dinámica.

3.1.7.1 Divisor.

El objetivo de este circuito es que el usuario se familiarice con el uso de elementos del inventario y revise sus conocimientos sobre las propiedades eléctricas de las líneas de transmisión.

Para ello se utiliza un divisor wilkinson cuyas entradas no están conectadas y que carece de la resistencia necesaria para aislar los puertos. El objetivo

del alumno es conseguir que la señal se propaga únicamente entre el puerto de entrada, determinado por el programador, y su adyacentes.

El archivo está construido mediante varias capas cuya visualización depende del grado de resolución que se haya alcanzado. En cada capa se aloja una animación distinta generada mediante HFSS.

```
buhovision.addEventListener("click", checkbox );
function checkbox(){
        if(buhovision.selected){
                _root.carcasa1._alpha=0;
                if(In._alpha==100){
         var contenedor1:MovieClip = _root.createEmptyMovieClip("contenedor1",4);
                                contenedor1._lockroot=true;
                                urltexto="flash/textos/nivel2/misc/in.txt";
                                dialogo=0;
                                clipLoader.loadClip("cuadro de dialogo2.swf", contenedor1);
                        }
                        if(In._alpha==0){
            var contenedor1:MovieClip = _root.createEmptyMovieClip("contenedor1",4);
                                contenedor1._lockroot=true;
                                urltexto="flash/textos/nivel2/misc/div.txt";
                                dialogo=0;
                                clipLoader.loadClip("cuadro de dialogo2.swf", contenedor1);
                        }
                        if(DivisorResuelto==true){
         var contenedor1:MovieClip = _root.createEmptyMovieClip("contenedor1",4);
                                contenedor1._lockroot=true;
                                urltexto="flash/textos/nivel2/misc/wilke.txt";
                                dialogo=0;
                   clipLoader.loadClip("cuadro de dialogo2.swf", contenedor1);
                        }
                }else{
                        _root.carcasa1._alpha=100;
                }
        }
```

Tabla 47. Funcionalidades de los elementos del divisor

A continuación se muestran las distintas imágenes cargadas en el clip de flash. La primera de ellas es la carcasa y su visualización es constante a menos que se active buhovisión. De esta imagen cuelgan los dos botones que llaman al inventario, pero son transparentes y no se muestran a menos que sean seleccionados correctamente en el inventario.

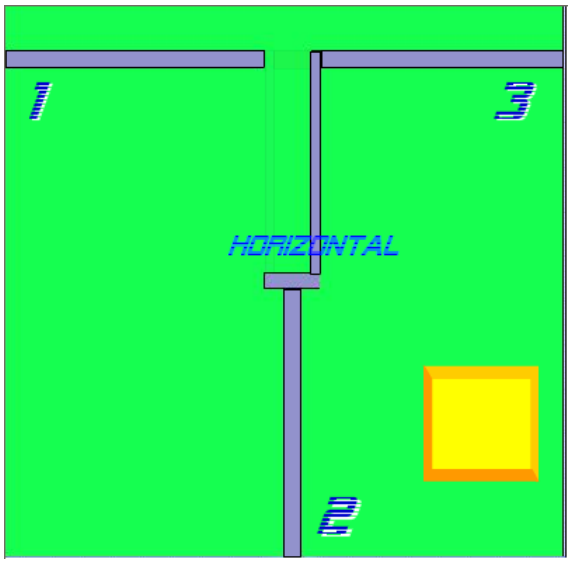

Ilustración 39. Carcasa

Si se selecciona buhovisión sin haber realizado ningún cambio la imagen que se muestra es al de una línea terminada sobre la que se produce una onda estacionaria.

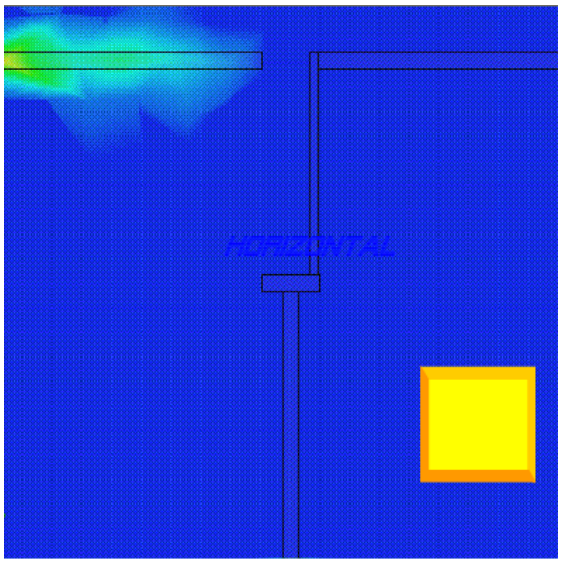

Ilustración 40. Campo que no avanza

Tras utilizar los objetos del inventario se produce un cambio en las animaciones que se muestran al utilizar la buhovisión. En caso de haber utilizado

la línea y de haber elegido acertadamente sus características el usuario visualiza los campos de un divisor con líneas.

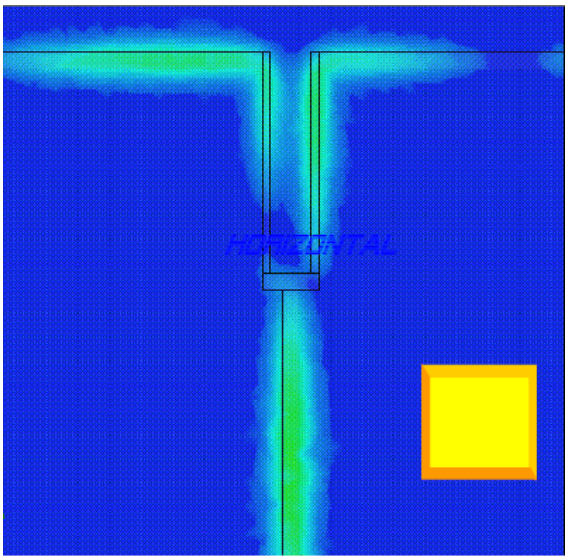

Ilustración 41. Campos del divisor con lineas

Sin embargo todavía queda una animación por mostrar, que es la que representa los campos de un divisor Wilkinson. Para que el usuario pueda ver esta animación es necesario que convierta el divisor con líneas en un divisor Wilkinson mediante la utilización de la resistencia recogida en la primera fasa.

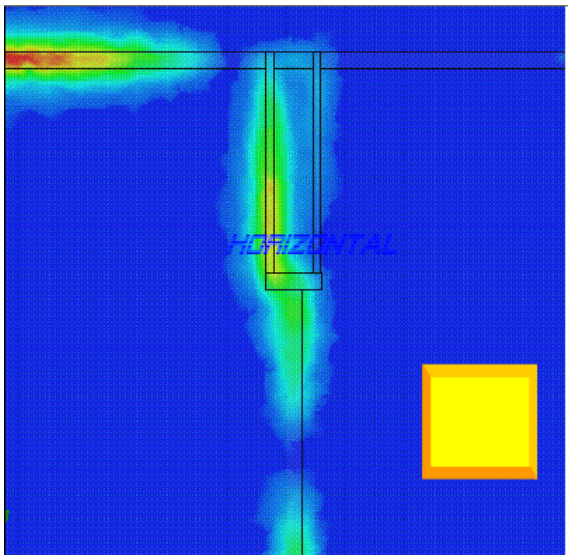

Ilustración 42. Divisor con los puertos aislados

Cada zona del circuito está preparada para realizar una acción. O bien se mostrará información relevante, como la longitud o impedancia de una línea o bien se realizará una llamada al inventario para utilizar elementos del mismo.

```
_root.carcasa1.resistencia.onRelease=function(){
        cargaInv=_root.createEmptyMovieClip("cargaInv",10);
        cargaInv.loadMovie("inventario4.swf");
        cargaInv._lockroot=true;}
    _root.carcasa1.resistencia.onRollOver=function(){
        if(inicio==true){
            var contenedor1:MovieClip = _root.createEmptyMovieClip("contenedor1",4);
            contenedor1._lockroot=true;
            urltexto="flash/textos/nivel2/misc/infboton.txt";
            dialogo=0;
            clipLoader.loadClip("cuadro de dialogo2.swf", contenedor1);
            inicio=false;
        }
    }
    _root.carcasa1.linea2.onRollOver=function(){
        buhotexto.text="impedancia: 1.4·Zo ohmios \r longitud: lambda/4";}
```

Tabla 48 . Código para cargar el inventario cuando se pulsa una linea

En este caso es necesario hacer uso de algunos elementos del inventario, así cuando se haga clic sobre el elemento que llama al inventario será posible usar los objetos en esa zona. Si han sido utilizados de forma incorrecta se envía al escenario que llamó a este circuito una señal de error.

```
on(Release){
    if(lineaResuelta==true&&DivisorResuelto==true){
        var sending_lc:LocalConnection = new LocalConnection();
        sending_lc.send("lc_name", "methodToExecute", 90);
        _parent.contenedor2.removeMovieClip();
        }else if(lineaResuelta==false){
            var sending_lc:LocalConnection = new LocalConnection();
            sending_lc.send("lc_name", "methodToExecute", 1000);
            _parent.contenedor2.removeMovieClip();}
            else{
```

```
                                var sending_lc:LocalConnection = new LocalConnec-
tion();
                                sending_lc.send("lc_name", "methodToExecute", 1001);
                                _parent.contenedor2.removeMovieClip();}
}
```

Tabla 49. Envío de datos al escenario

3.1.7.2 Hibrido.

La resolución de este circuito implica la elección correcta de las señales de entrada para obtener las salidas deseadas. Cada dirección seleccionada tiene asociada una salida en concreto.

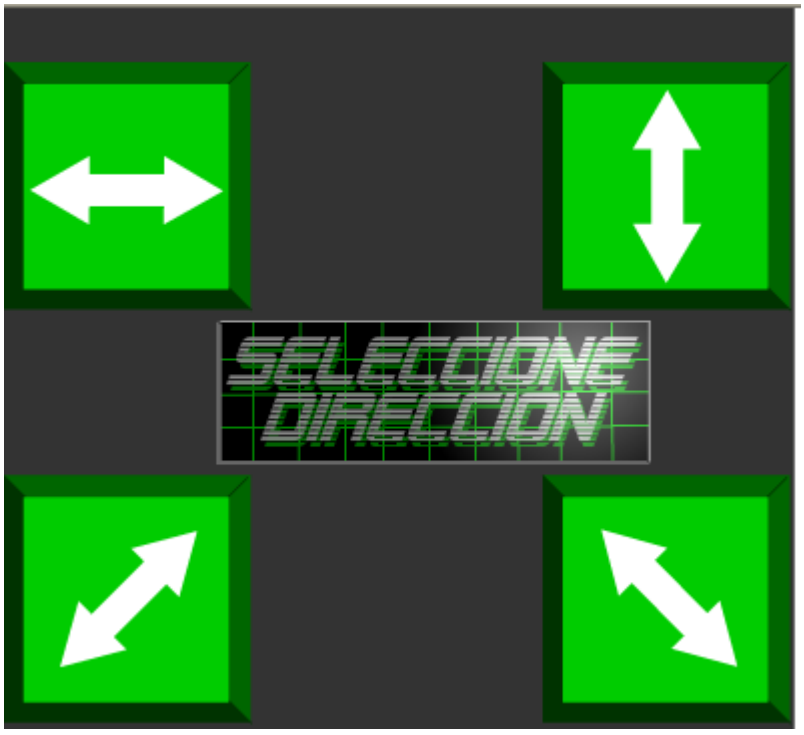

Ilustración 43. Selector de direcciones

El funcionamiento es similar al descrito para el divisor. Se dispone de varias animaciones que representan los campos para cada una de las combinaciones de entrada distribuidas en distintas capas. El cambio en las señales de entrada provoca que se muestre una u otra animación.

Cada vez que se selecciones una entrada se activa la variable que indica que señal se está introduciendo y se anulan el resto para a continuación llamar a la función comprueba.

```
botondoscuatro.addEventListener("click",funciondoscuatro)
function funciondoscuatro(){
        dosuno=false;
        dosmenosuno=false;
        dosjota=false;
        dosmenosjota=true;
        trace("doscuatro"+dosmenosjota);
        flecha21._alpha=0;
        flecha22._alpha=0;
        flecha23._alpha=0;
        flecha24._alpha=100;
        comprueba();
        }
```

Tabla 50. Modificación de variables para la visualización de los campos

En comprueba se confirma que señal se está introduciendo por los puertos de entrada y se determina en función de las entradas que animación se visualiza.

```
comprueba=function(){
        trace("esto es comprueba");
if(unouno==true){
        if(dosuno==true){
                branchx._alpha=100;
                branchy._alpha=0;
                branchz._alpha=0;
                branchp._alpha=0;
                trace("x");}
        if(dosmenosuno==true){
                branchx._alpha=0;
                branchy._alpha=0;
                branchz._alpha=100;
                branchp._alpha=0;
                trace("z");}
        if(dosjota==true){
                branchx._alpha=0;
                branchy._alpha=100;
```

```
                branchz._alpha=0;
                branchp._alpha=0;
                trace("y");}
        if(dosmenosjota==true){
                branchx._alpha=0;
                branchy._alpha=0;
                branchz._alpha=0;
                branchp._alpha=100;
                trace("p");}
        }
```

Tabla 51. Visualización de los campos

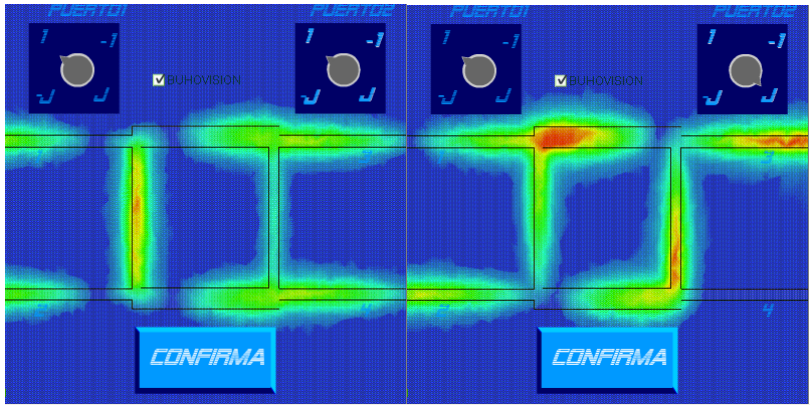

Ilustración 44. Campos en función de las distintas entradas

Una vez seleccionadas las entradas, si la salida no coincide con la que se esperaba se manda una señal de error al escenario que invocó a este circuito.

```
if(sP){
                if(branchp._alpha==100){
                        sending_lc.send("lc_name", "methodToExecute", 90);
                        trace("correcto");
                        }else{
                                sending_lc.send("lc_name", "methodToExecute", 1000);}
        }
}
```

Tabla 52. Envío de información

3.1.7.3 Círculos de estabilidad.

La resolución de los círculos de estabilidad depende de elegir que zona de la pantalla es estable, o bien el diagrama de Smith o bien la carta de estabilidad.

Para poder elegir una zona antes de forma aleatoria los parametros S de un transistor así como si la carta de Smith pertenece a la fuente o a la carga.

```
var ro:Number=Math.random();
if(ro<0.5){
        ros.text="reflexion en la fuente";}
        else{
                ros.text="reflexion en la carga";
                }
var centroDentro:Boolean;
var S11:Number=randRange(0,1.7);
_root.s11.text=S11;
var S12:Number=randRange(0,2);
_root.s12.text=S12;
var S21:Number=randRange(0,2);
_root.s21.text=S21;
var S22:Number=randRange(0,1.7);
_root.s22.text=S22;
```

Tabla 53. Parámetros del transistor

A continuación se genera también de forma aleatoria el círculo de estabilidad.

Una vez dibujado el círculo de estabilidad se determina si el centro de la carta de Smith está contenida dentro de él.

```
centroPantallaX=281-pantalla._x;
centroPantallaY=205-pantalla._y;
poserX=estabilidad._x+Radio/*-centroPantallaX*/;
poserY=estabilidad._y+Radio/*-centroPantallaY*/;
poserY=-poserY
poserX=Math.pow(centroPantallaX-(estabilidad._x+Radio),2);
poserY=Math.pow(centroPantallaY-(estabilidad._y+Radio),2);
dist=poserX+poserY;
dist=Math.sqrt(dist);
if(dist<Radio){
```

```
trace("centro dentro de circulo");
centroDentro=true;
}else{
        trace("centro fuera de estabilidad");
        centroDentro=false;}
```

Tabla 54. Localización del centro de la carta respecto al círculo de estabilidad

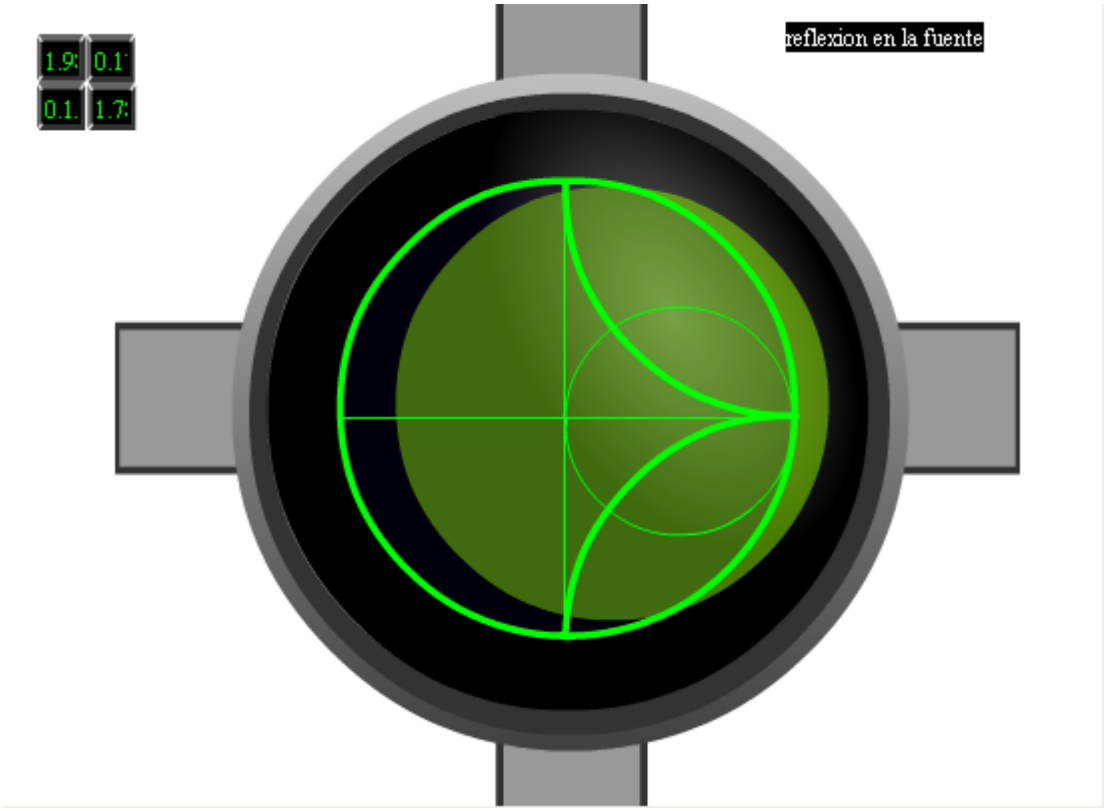

Ilustración 45. Carta de Smith y círculo de estabilidad

La película está preparada para determinar que sobre que circulo está el ratón. Cuando se pulse sobre uno de ellos se generará un mensaje en función de los parámetros descritos anteriormente para el escenario que invocó al circuito.

```
smiti.onMouseDown=function(){
if(estabilidad.hitTest(_root._xmouse,_root._ymouse,false)&&smiti.hitTest(_root._xmouse,_root._ymouse,true)){
        }else if(this.hitTest(_root._xmouse,_root._ymouse,true)){
                this.oIsDown = true;
                if(ro>0.5){
                        if(S11>1){
                                if(centroDentro){
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1000);
                                                }else{
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1001);}
                                                        }else{
                                                                if(centroDentro){
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1002);
                                                                }else{
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1000);}
                                                                }
                                }else{
                                        if(S22>1){
                                                if(centroDentro){
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1000);
                                        }else{
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1003);}
                                                        }else{
                                                                if(centroDentro){
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1004);
                                                                }else{
                                        var sendingBomb:LocalConnection=new LocalConnection();
                                        sendingBomb.send("bomb", "methodToExecute", 1000);}
                                                                }
                                                        }


                }
        }
```

Tabla 55. Determinación de éxito o fracaso y envío de datos

3.2. Servidor.

Debido a que la herramienta se ha planteado para utilizarse en línea se ha utilizado un servidor sobre el que se alojan las bases de datos y al que la aplicación hace acceso cada vez que debe registrar algo. Se ha elegido un servidor Apache instalado a través de appServ, una aplicación que configura au-

tomaticamente el servidor y lo deja preparado para utilizarse. Se ha elegido Apache porque ofrece bases de datos gratuitas mediante MySQL.

A través del símbolo del sistema se ha creado una base de datos llamada alumno con las tablas datos, personajes, lineas, resistencias.

3.2.1 Estructura de las tablas.

Se utiliza una base de datos llamada alumnos en la que se alojan las tablas necesarias para el registro, la identificación y la gestión del inventario.

3.2.1.1 Registro e identificación.

Los métodos de registro e identificación utilizan tablas comunes que son la tabla datos y la tabla personajes.

En la tabla datos se almacenan los datos personales del usuario y en la de personajes los relativos al personaje que el usuario utiliza durante el juego.

En la creación de cada tabla se debe especificar el tipo de cada uno de los registros que la componen.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(20)	NO			
apellidos	varchar(30)	NO			
mail	varchar(30)	NO			
clave	varchar(10)	NO			

Ilustración 46. tabla de datos del usuario

De los campos que se muestran en la imagen de la tabla datos el más importante es id por ser la clave primaria. Su valor es único y por lo tanto identifica a cada usuario. Además, con cada nuevo registro aumenta su valor automáticamente, lo que impide que pueda duplicarse, por cualquier motivo.

```
mysql> describe personajes;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int(11)     | NO   | PRI |         |       |
| nonUs    | varchar(20) | YES  |     | NULL    |       |
| genero   | int(11)     | YES  |     | NULL    |       |
| piel     | int(11)     | YES  |     | NULL    |       |
| pelo     | int(11)     | YES  |     | NULL    |       |
| peloc    | int(11)     | YES  |     | NULL    |       |
| cami     | int(11)     | YES  |     | NULL    |       |
| camic    | int(11)     | YES  |     | NULL    |       |
| pantalon | int(11)     | YES  |     | NULL    |       |
| nivel    | tinyint(4)  | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
10 rows in set (0.05 sec)
```

Ilustración 47. tabla de datos del personaje.

La tabla que guarda los datos del personaje también tiene como clave primaria la id, pero su valor no se incrementa automáticamente. En su lugar el valor es el mismo que el que se asignó al último usuario registrado. Es del tipo primario porque cada usuario sólo tiene un personaje y la relación entre ambos se define de forma unívoca.

3.2.1.2. Inventario.

Cómo puede observarse las tablas que se utilizan para gestionar el inventario son distintas de las vistas con anterioridad. La diferencia radica en que la id ya no es una clave primaria. Que un campo esté definido como clave primaria implica que su valor no puede repetirse, esta cualidad es de mucha utilidad para controlar el registro de nuevos usuarios, pero inutiliza el registro eficiente de nuevos objetos. Esto se debe a que un usuario tendrá distintos objetos de un mismo tipo. Es decir podrá poseer resistencias de cincuenta, setenta y cinco o cien ohmios, todas ellas registradas con la misma id porque pertenecen al mismo usuario.

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| numero   | int(11)     | YES  |     | NULL    |       |
| longitud | int(11)     | YES  |     | NULL    |       |
| valor    | int(11)     | YES  |     | NULL    |       |
| id       | smallint(6) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

Ilustración 48. tabla del inventario para las líneas.

3.2.1 Registro

Las acciones que realiza flash dependen de la respuesta que de el servidor. Que se generan a su vez en función de las consultas a la base de datos.

El archivo PHP recoge los datos enviados desde flash, conecta a la base de datos y realiza tres posibles consultas. La primera de ellas es localizar alguna dirección de correo igual a la que flash le envió, si lo consigue no seguirá consultando e informará a flash de que la dirección de correo ya ha sido utilizada. En caso de no encontrar ningún registro con la dirección de correo enviada realizará la búsqueda de una clave igual a la suministrada desde flash, actuando de la misma forma que en el caso del correo en el caso de encontrarla. Finalmente si no encontrase ninguno de los dos parámetros crea un nuevo registro en la base de datos e informa a flash. Es importante puntualizar que al insertar valores en la base de datos es necesario un dato por cada campo en la base datos.

La conexión a la base de datos requiere que se sepa el usuario que la creó y la contraseña, si estos parámetros son incorrectos o inexistentes la conexión falla. Por tanto PHP ofrece un entorno seguro y la certeza de que los datos no serán alterados sin autorización.

```
<?
$nombre=$_POST['nombre'];
$apellido=$_POST['apellido'];
$mail=$_POST['mail'];
$clave=$_POST['clave'];
$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
mysql_select_db("alumnos",$con) or die ("no se ha conectado");
$query="select mail from datos where mail='".$mail."'";
$result=mysql_query($query);
$consult=mysql_fetch_row($result);
if($consult){
        echo ?>&mensaje=0&<?;
        }
        else{
                $query1="select clave from datos where clave='".$clave."'";
                $result2=mysql_query($query1);
                $consult2=mysql_fetch_row($result2);
                if($consult2){
                        echo ?>&mensaje=1&<?;
```

```
                        }
                else{                   $que-
ry2="insertintodatosvalues('','$nombre','$apellido','$mail','$clave')";
                        $resultado=mysql_query($query2) or die ("q no actualizo
na de na");
                        ?>&mensaje=2&<?;}
        }
    mysql_close($con);
    ?>
```

Tabla 56. inserción en la base de datos de los datos proporcionados por el usuario.

Para finalizar el registro, es necesario dotar al personaje que se utiliza durante el juego de una serie de características que lo distingan del resto de usuarios. Características que son cargadas dinámicamente en flash para su elección. La carga responde a la selección de nodos en el siguiente archivo XML, en el que están especificados el tipo de pelo, de camisa y de pantalón.

Se ha elegido XML porque ofrece un sistema de etiquetas mediante el cual es posible encapsular la información y evitar algún tipo de confusión entre archivos que representan atributos del personaje masculino con atributos del personaje femenino.

```
    <cuerpom src="c:/AppServ/www/personaje/flash2/cuerpom.gif">
        <cabezasm>
            <archivos
src="c:/AppServ/www/personaje/flash2/cabezam.gif"></archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/pelom0.gif">
            </archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/pelom1.gif">
            </archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/pelom2.gif">
            </archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/pelom3.gif">
            </archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/pelom4.gif">
            </archivos>
            <archivos src="c:/AppServ/www/personaje/flash2/ojosm.gif">
            </archivos>
        </cabezasm>
        <camisetas>
```

```
                    <archivos
src="c:/AppServ/www/personaje/flash2/camisam.gif">
                        <color
src="c:/AppServ/www/personaje/flash2/rayam.gif"></color>
                        <cuello
src="c:/AppServ/www/personaje/flash2/cuellom0.gif"></cuello>
                        <cuello
src="c:/AppServ/www/personaje/flash2/cuellom1.gif"></cuello>
                    </archivos>
                </camisetas>
                <pantalones>
                    <archivos
src="c:/AppServ/www/personaje/flash2/pantalonm0.gif"></archivos>
                    <archivos
src="c:/AppServ/www/personaje/flash2/pantalonm1.gif"></archivos>
                </pantalones>
        </cuerpoh>
```

Tabla 57. XML que define las rutas para la carga del personaje en la pantalla de selección.

Una vez elegidas, las características del personaje son registradas. Cada usuario define un conjunto de parámetros únicos para su personaje, que son cargados durante el juego.

Es importante que cada conjunto de datos corresponda unívocamente con un usuario, ya que cada usuario define sólo un conjunto de parámetros para el personaje. Para asegurar este hecho, antes del registro de las características del personaje se hace una consulta a la tabla que contiene los datos personales del usuario. La consulta devuelve el identificador del usuario, que es único, y que se inserta como identificador en la tabla que registra los datos del personaje. De esta forma usuario y personaje quedan relacionados mediante un identificador único.

```
        <?
        $genero=$_POST['genero'];
        $mail=$_POST['mail'];
        $nombreU=$_POST['nombreU'];
        $piel=$_POST['piel'];
        $pelo=$_POST['pelo'];
        $peloc=$_POST['peloc'];
        $cami=$_POST['camiseta'];
        $camic=$_POST['camisetac'];
```

```
$pantalon=$_POST['pantalon'];
$nivel=$_POST['nivel'];
$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
mysql_select_db("alumnos",$con) or die ("no se ha podido conectar");
$sqlid="select id from datos where mail='".$mail."'";
$resultado1=mysql_query($sqlid);
$id=mysql_result($resultado1,0);
$sql="insert into personajes values('$id','$nombreU','$genero','$piel','$pelo','$peloc','$cami','$camic','$pantalon','$nivel')";
$resultado=mysql_query($sql);
$sql2="insert into lineas values(0,9990,9990,'$id')";
$sql3="insert into resistencias values('$id',0,9990)";
$resultado2=mysql_query($sql2);
$resultado3=mysql_query($sql3);
//mysql_free_result($resultado);
mysql_close($con);
?>
```

Tabla 58. inserción en la tabla personajes de los datos elegidos por el usuario.

3.2.2. Identificación.

La identificación requiere la coincidencia del par correo y clave enviado desde flash a la base de datos. Si alguno de los dos parámetros falla se informa al usuario de que se han introducido de forma incorrecta.

Una vez que el usuario se identifica de forma correcta se extrae su identificador único y por medio de él se accede al conjunto de características de su personaje. Que se envían a flash para ser utilizadas.

```
<?
$mail=$_POST['mail'];
$clave=$_POST['clave'];
$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
mysql_select_db("alumnos",$con) or die ("no se ha podido conectar");
$sql="select clave from datos where mail='".$mail."'";
$resultado=mysql_query($sql);
$log=mysql_fetch_row($resultado);
$response=mysql_result($resultado,0);
if(!$log){
        echo ?>&mensaje=0&<?; //clave incorrecta
}else{
```

```
                if ($clave!=$response){
                        echo ?>&mensaje=1&<?;
                        }else{
                                $sqlid="select id from datos where mail='".$mail."'";
                                $resultado2=mysql_query($sqlid);
                                $id=mysql_result($resultado2,0);
                                $sqlres="select nomUs, genero, piel, pelo, peloc, cami,
camic, pantalon from personajes where id='".$id."'";
                                $rescate= mysql_query($sqlres);
                                $response=mysql_fetch_row($rescate);
                                $nomUs=$response[0];
                                echo"&nomUs=".$nomUs;
                                $genero=$response[1];
                                echo"&genero=".$genero;
                                $piel=$response[2];
                                echo"&piel=".$piel;
                                $pelo=$response[3];
                                echo"&pelo=".$pelo;
                                $peloc=$response[4];
                                echo"&peloc=".$peloc;
                                $camiseta=$response[5];
                                echo"&camiseta=".$camiseta;
                                $camic=$response[6];
                                echo"&camic=".$camic;
                                $pantalon=$response[7];
                                echo"&pantalon=".$pantalon;
                                echo ?>&mensaje=2&<?;
                                }
        }
        mysql_close($con);
        ?>
```

Tabla 59. autenticación del usuario frente al servidor y rescate de sus datos si lo consigue.

3.2.3. Inventario.

3.2.3.1.Inserción de elementos.

Debe quedar constancia de la recolección de objetos que realiza el usuario para que puedan ser utilizados en los momentos requeridos.

La colección de objetos de cada usuario es única, sin embargo un usuario tiene varios objetos del mismo tipo. Por tanto la estructura de las tablas relacionadas con el inventario es distinta a la de las vistas con anterioridad y también el método de inserción.

En primer lugar PHP recibe el correo electrónico del usuario que realiza la consulta y el tipo de objeto que se desea insertar, Con el correo se localiza el identificador de ese usuario.

A continuación se selecciona la tabla correspondiente al tipo de objeto que se desea insertar. Se comparan las características del objeto con las de todos los que posee el usuario que realiza la consulta. Si se consigue coincidencia exacta se actualiza la base de datos incrementando el campo número en una unidad. En caso de que no exista coincidencia se inserta un nuevo elemento.

```
<?
	$tipo=$_POST['tipo'];
	$mail=$_POST['mail'];
	$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
	mysql_select_db("alumnos",$con) or die ("no se ha podido conectar");
	$sqlid="select id from datos where mail='".$mail."'";
	$resultado2=mysql_query($sqlid);
	$id=mysql_result($resultado2,0);//localiza el usuario activo
	$insertar=true;
	if($tipo =="linea"){
		$longitud=$_POST['longitud'];
		$impedacia=$_POST['impedancia'];
		$consulta="select longitud, valor from lineas where id='".$id."'";
		$resultado=mysql_query($consulta);
		while($inventario=mysql_fetch_row($resultado)){  .
			if($longitud==$inventario[0]){
				if($impedacia==$inventario[1]){
	$consultaLinea="select numero from lineas where longitud='".$longitud."'and valor='".$impedancia."'and id='".$id."'";
					$resultado1=mysql_query($consultaLinea);
					$numero=mysql_result($resultado1,0);
					$numero=$numero+1;
					echo"&id=".$numero;
	mysql_query("update lineas set numero='".$numero."'where longitud='".$longitud."'and valor='".$impedancia."'and id='".$id."'");
```

```
                                        $insertar=false;}
                                }
                }
                if($insertar==true){
        $insertio="insert into lineas values(1,'$longitud','$impedancia','$id')";
                        $insercion=mysql_query($insertio);
                        echo"&control=insertando nuevo";
                }
}?>
```

Tabla 60. Actualización de las tablas relativas al inventario

Como último paso para actualizar la base de datos se hace una búsqueda de todos los registros cuyo campo número sea cero y cuyo campo valor sea distinto de 9990, que es el valor utilizado para la inicialización de la base de datos.

```
<?PHP
$mail=$_POST['mail'];
$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
mysql_select_db("alumnos",$con) or die ("no se ha podido conectar");
$sqlid="select id from datos where mail='".$mail."'";
$resultado2=mysql_query($sqlid);
$numero=0;
$longo=9990;
$id=mysql_result($resultado2,0);//localiza el usuario activo
mysql_query("delete from lineas where id='".$id."' and numero='".$numero."'and valor <>'".$longo."'");
mysql_query("delete from resistencias where id='".$id."' and numero='".$numero."'and valor<>'".$longo."'");
?>
```

Tabla 61. Borrado de los elementos con el campo número igual a 0

3.3.2. Uso de elementos.

La utilización de los elementos recolectados se realiza a través de la película flash "inventario4.swf". La información obtenida mediante la consulta a la base de datos se envía a flash para ser utilizada por el usuario.

La consulta realizada a la base de datos extrae la totalidad de los elementos registrados en ella que pertenezcan al usuario activo. La propiedad se

comprueba mediante comparación de identificadores. Las consultas realizadas a MySQL son devueltas como un array, sin embargo no está permitido el paso de estas estructuras entre PHP y flash, por tanto como paso previo al envio de información el resultado de la consulta debe ser tratado.

Cada tipo de elemento distinto es registrado en una tabla diferente, la información que se obtiene es insertada en un archivo XML con un nodo distinto para cada elemento, que a su vez tiene varios nodos hijos conteniendo los valores que lo definen.

El archivo XML es enviado a flash junto con el número de elementos de cada tipo, de este forma combinando ambas informaciones es posible el acceso ordenado a los objetos registrados en la base de datos.

```
<?PHP
$mail=$_POST['mail'];
//echo"correo".$mail;
$con=mysql_connect("localhost","root","root") or die ("No se ha podido");
mysql_select_db("alumnos",$con) or die ("no se ha podido conectar");
$sqlid="select id from datos where mail='".$mail."'";
$resultado2=mysql_query($sqlid);
$id=mysql_result($resultado2,0);//localiza el usuario activo
$consultaLinea="select numero,longitud,valor from lineas where id='".$id."'";//pillo los resultados q necesito
$lineas=mysql_query($consultaLinea);
$consultaResistencia="select numero,valor from resistencias where id='".$id."'";//pillo los resultados q necesito
$resistencias=mysql_query($consultaResistencia);
$numLineas=mysql_num_rows($lineas);
echo"&numLineas=".$numLineas;
$numResistencias=mysql_num_rows($resistencias);
echo"&numResistencias=".$numResistencias;
echo"&respuesta=";
echo"<?xml version=\"1.0\"?>\n";
while($row=mysql_fetch_row($lineas)){
        echo"<linea>
                <numero>$row[0]</numero>
                <longitud>$row[1]</longitud>
                <valor>$row[2]</valor>
        </linea\n>";
}
```

```
while($rowR=mysql_fetch_row($resistencias)){
        echo"<resistencia>
                <numero>$rowR[0]</numero>
                <valor>$rowR[1]</valor>
        </resistencia\n>";
}
?>
```

Tabla 62. Rescate de los objetos del usuario para ser utilizados en el inventario.

4. POSIBLES LÍNEAS DE MEJORA Y CONCLUSIONES

4. Posibles líneas de mejora y conclusiones.

4.1. Posibles líneas de mejora

Existen dos principales vertientes a seguir para mejorar la aplicación.

La primera de ellas consiste en actuar sobre al jugabilidad de la aplicación. Es posible aumentar el número de capítulos así como la complejidad del argumento, factores que generarán un incremento del interés de los usuarios. Trabajar en este sentido es importante, ya que si se consigue que los alumnos jueguen con una frecuencia alta resulta sencillo la asunción de los conceptos de microondas que se quieran inculcar. En esta dirección es posible reutilizar los circuitos diseñados y relacionarlos con la resolución de nuevos puzzles.

No sólo es posible actuar sobre el desarrollo del juego en sí. La personalización del personaje es una baza importante que debe potenciarse. Hasta ahora no es posible conformar una seríe de características, puede aumentarse el número de parámetros a configurar, así como añadir nuevos elementos, tales como gafas, gorras y otros accesorios.

En esta fase del desarrollo se han introducido objetos que únicamente podían ser utilizados para la resolución de circuitos. Es posible trabajar en esta área creando elementos que se utilicen durante toda la apliacación para mejorar características del personaje o acceder a nuevos escenarios.

La segunda se centra en aumentar la complejidad de los circuitos a resolver. En lugar de cambiar una línea o alterar una entrada es posible diseñar un problema cuya solución final requiera de distintas soluciones parciales. Bajo esta perspectiva puede realizarse cualquier circuito imaginable, únicamente es necesario un análisis meticuloso antes de su implementación en flash. Si la dificultad de los circuitos a resolver aumenta se consiguen dos hitos Por un lado aumenta también la atractividad del juego y se pueden controlar las curvas de aprendizaje del alumno a voluntad.

Por otro lado si se tiene en cuenta la base de datos de la que se dispone, se abre la posibilidad de crear foros o salas para los usuarios donde puedan compartir experiencias y consejos en la resolución de los puzzles.

4.2. Conclusiones.

El objetivo del proyecto es adaptar la metodología docente a las exigencias propuesto por el tratado de Bolonia.

Bajo este escenario la enseñanza debe estar centrada en el alumno y se prima la autoevaluación y el aprendizaje.

Se ha pretendido desarrollar una aplicación que cumpla estas condiciones. Sin embargo el entorno de la aplicación es muy importante, no es posible que el alumno aprenda si no dedica el tiempo suficiente a esta herramienta, y no es fácil que lo haga si la aplicación no es atractiva.

Por tanto debe desarrollarse una herramienta que permita el aprendizaje, la autoevaluación y que sea atractiva al usuario.

Intentando cumplir la premisa de la apariencia atracitiva se ha la aplicación se ha envuelto dentro de un juego. Con esto se pretende que el usuario utilice la aplicación sin reparo y es más tenga ganas de usarla. Consiguiendo de esta forma un alto grado de uso.

El desarrollo de la historia es la consecuencia de la resolución de circuitos. Si un usuario no aplica correctamente los conocimientos sobre microondas que posee no será capaz de avanzar en el juego. Así se consigue la evaluación constante.

La forma de asegurar el aprendizaje es que no sea posible resolver los circuitos azarosamente. O bien tienen solución única o el número de pruebas que se realiza en su resolución es limitada.

La aplicación creada cumple con estas condiciones además de abrir un abanico amplisimo en su desarrollo futuro. Pudiendo desarrollarse con mucha profundidad, tanto creando nuevos circuitos como aplicaciones relacionadas.

Es por tanto una herramienta docente muy potente versatil y que abre un nuevo camino en la enseñanza universitaria.

ANEXO

Anexo.

Instalación para uso local de la aplicación.

Si se desea que la aplicación sea utilizada en modo local antes es necesario preparar la máquina del usuario.

El primer paso consiste en instalar y configurar un servidor Apache.

Es posible descargar un instalador desde la página web http://www.appservnetwork.com/. Una vez descargado basta con hacer doble clic y seguir las instrucciones del asistente.

La instalación se realizará en C:\AppServ y se instalarán todos los componentes que ofrece el asistente

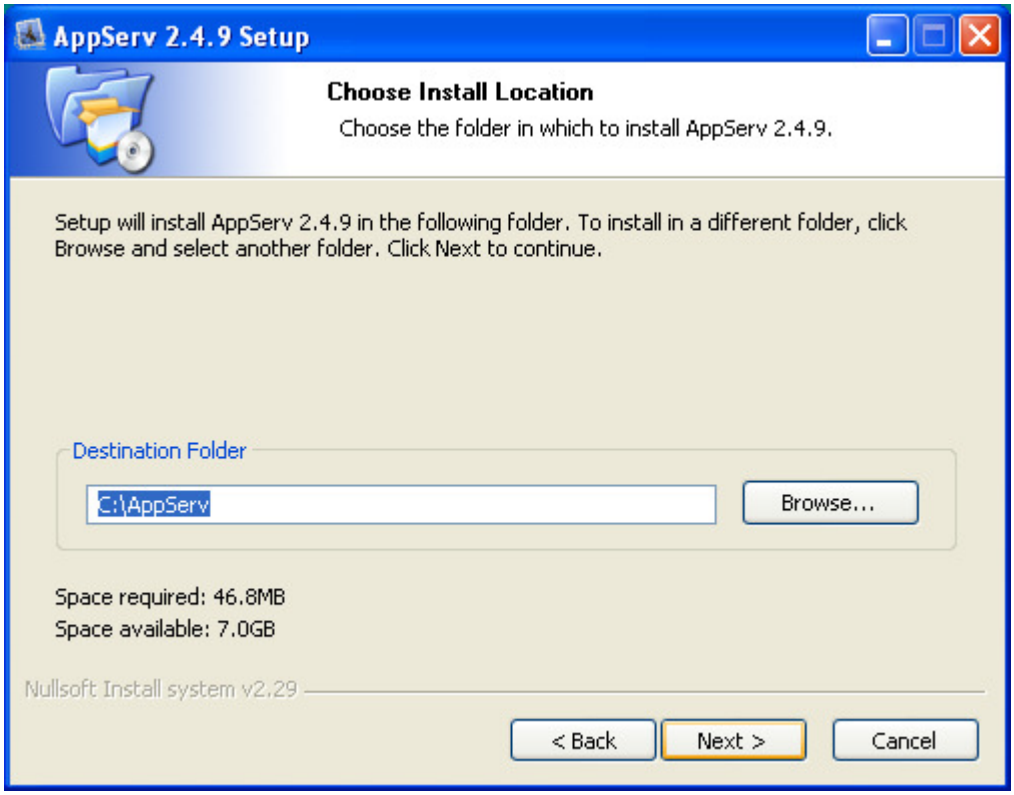

Ilustración 49. Ruta de instalación

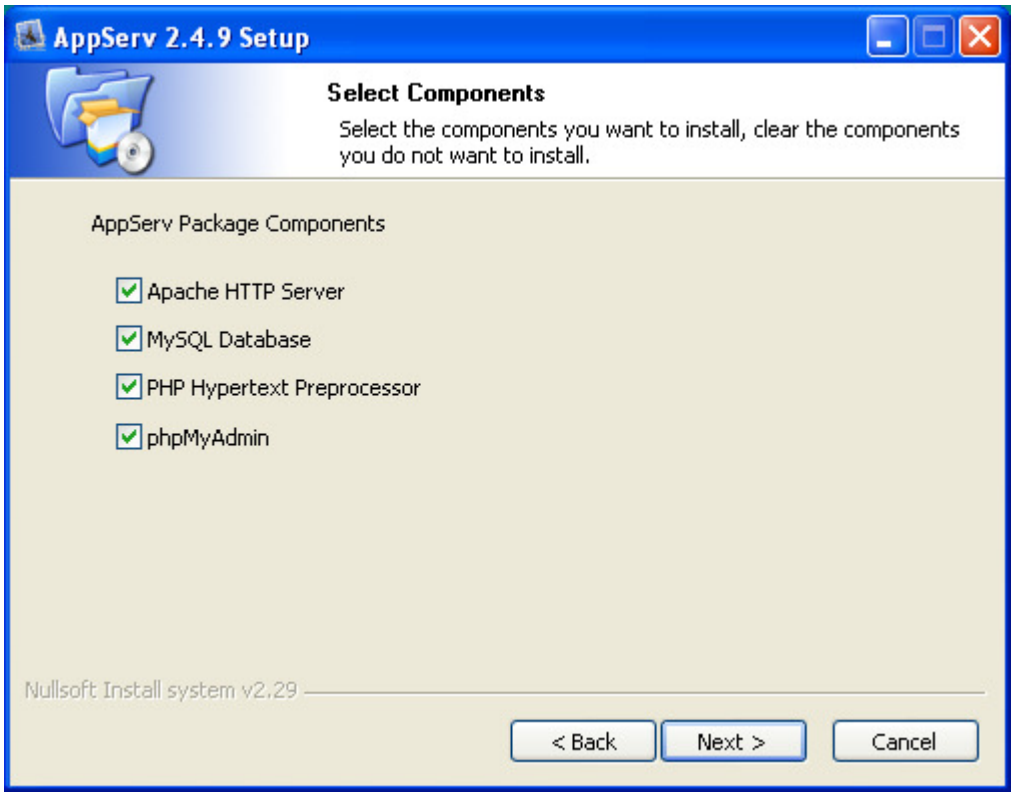

Ilustración 50. Componentes a instalar

El nombre del servidor será localhost y la dirección de correo electrónico es indifernte.

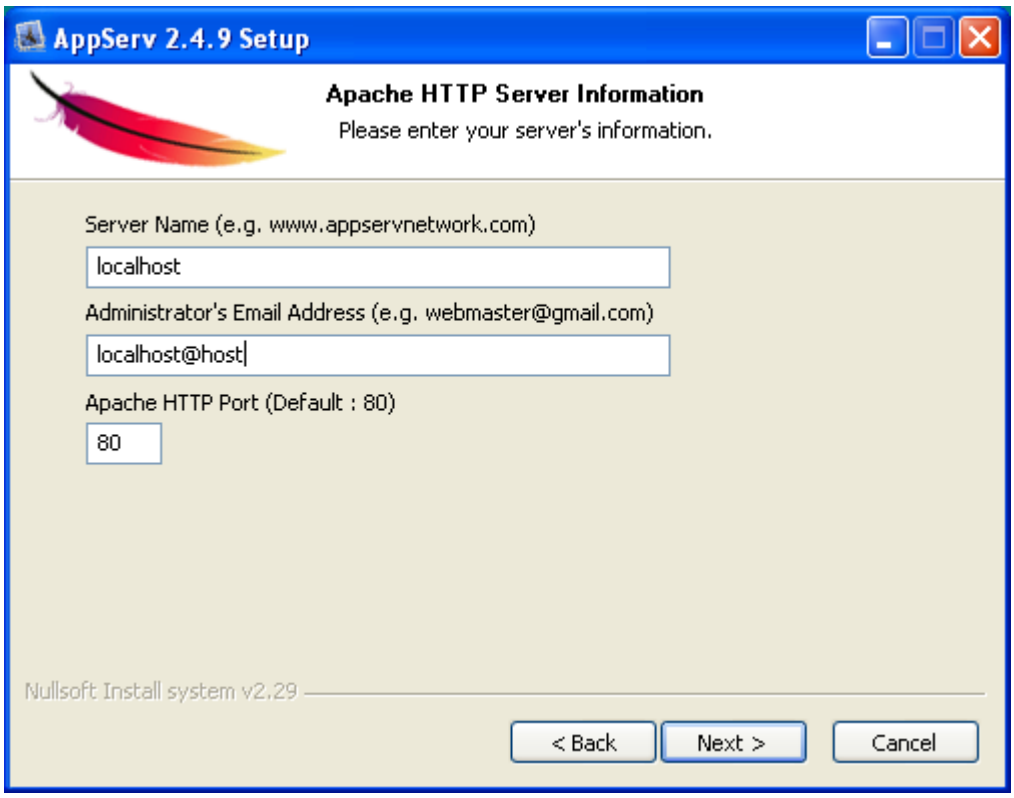

Ilustración 51. Nombre del servidor

Es importante que la clave que se utiliza para la base de datos sea root, si no es así es imposible conectar con la base de datos.

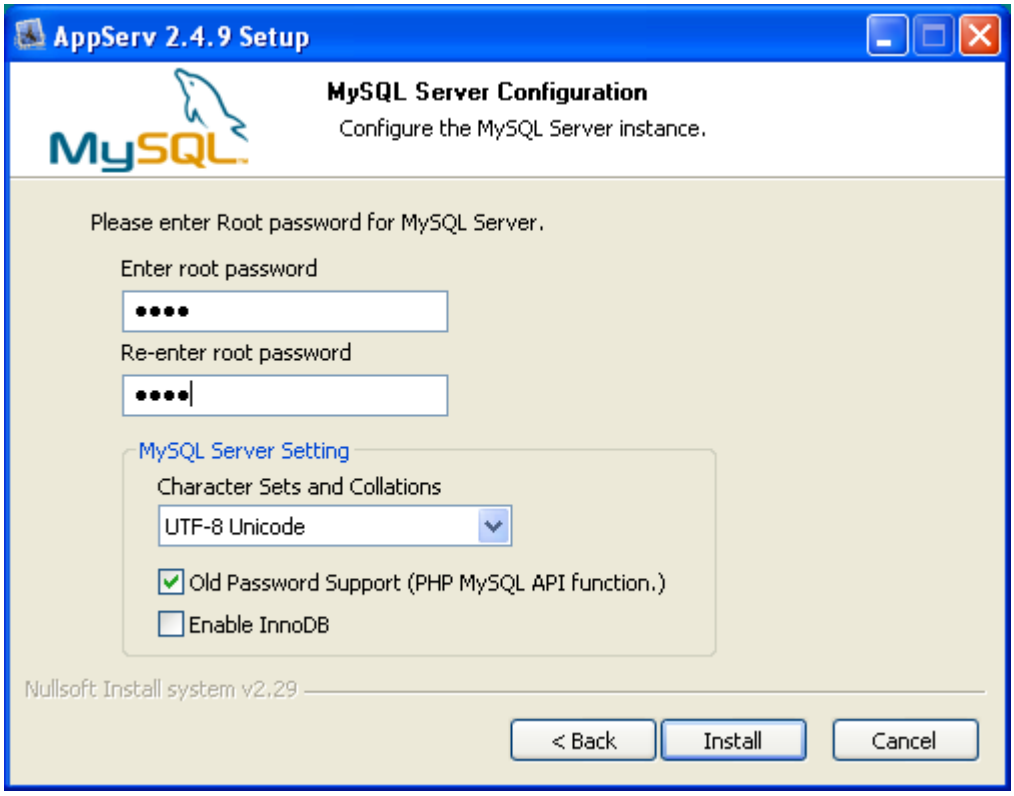

Ilustración 52. Clave de la base de datos

Una vez instalado el servidor solo es necesario copiar la carpeta MCSI en C:\AppServ\www.

Tras esto hay que crear las bases de datos. Se lanza MySQL desde el símbolo del sistema insertando el comando: *mysql –u root –p.* A continuación, cuando el promt cambia a *mysql>* se ejecuta el script para crear la base de datos y las tablas mediante el siguiente comando: source AppServ/www/MCSI/phpflash/crea.sql

Anexo.