Actividades

1.- Modificación del contenido de un archivo ggb.

Abrimos GeoGebra. Dibujamos dos puntos y el segmento determinado por estos dos puntos. Insertamos una imagen cualquiera (que sea pequeña). Accedemos a las propiedades de la imagen y en la pestaña *Posición* elegimos el punto *A* para la *esquina 1*. Guardamos la construcción con el nombre *prueba1.ggb* y cerramos GeoGebra.

Con el programa 7-zip, o con cualquier programa de compresión de archivos, abrimos el archivo *prueba1.ggb* que hemos creado. Observamos su contenido.

Nombre	Tamaño	Tamaño comprimido	Modificado
83a070e2cd454796344a9a53a756b0eb	9 853	9 858	
geogebra_thumbnail.png	1 631	1 531	2013-07-24 19:34
geogebra_javascript.js	23	25	2013-07-24 19:34
geogebra.xml	4 497	1 531	2013-07-24 19:34

Extraemos su contenido en una nueva carpeta que llamaremos *actividad1*.

Con el programa Notepad++, o con cualquier editor de código, abrimos el fichero geogebra.xml. Observemos su contenido, un centenar de líneas de código. Intentemos localizar donde se determina:

- La versión de GeoGebra
- Las herramientas que se visualizarán en la barra de herramientas
- El número de decimales que se mostrarán
- Cada uno de los elementos creados (puntos, segmentos, imagen)
- Las coordenadas del punto *A*
- El color del segmento
- El anclaje de la imagen en el punto *A*

Vamos a cambiar un poco el código.

a) Ocultaremos el segmento: donde pone *true*, pondremos *false.*

```
<element type="segment" label="a">
    <show object="false" label="false"/>
```

b) Crearemos un nuevo punto: copiamos el código correspondiente al punto *B* y lo pegamos inmediatamente debajo con los cambios siguientes:

```
<element type="point" label="C">
        <show object="true" label="true"/>
        <objColor r="255" g="0" b="0" alpha="0.0"/>
        <layer val="0"/>
        <labelMode val="0"/>
        <animation step="1" speed="1" type="1" playing="false"/>
        <coords x=”-1" y=”-1" z="1.0"/>
        <pointSize val="3"/>
        <pointStyle val="0"/>
</element>
```

c) Insertaremos una nueva imagen anclada en el punto *B*: copiamos el código correspondiente a la imagen que habíamos insertado y lo pegamos inmediatamente debajo con los cambios siguientes, donde *segunda.png* es el nombre del archivo con la nueva imagen (después insertaremos el archivo png):

```
<element type="image" label="imagen2">
        <file name="segunda.png"/>
```

```
        <inBackground val="false"/>
        <startPoint number="0" exp="B"/>
        <show object="true" label="false"/>
        <objColor r="0" g="0" b="0" alpha="1.0"/>
        <layer val="0"/>
        <labelMode val="0"/>
        <animation step="1" speed="1" type="0" playing="false"/>
    </element>
```

d) Guardamos los cambios efectuados en el archivo geogebra.xml. Cerramos el Notepad++.

Buscamos en internet una imagen con formato png y la guardamos en la carpeta *actividad1* con el nombre *segunda.png*

Vamos a actualizar el contenido de *prueba1.ggb* con la nueva imagen y el fichero xml modificado. Una forma de hacerlo es arrastrando el contenido de la carpeta *actividad1* hasta la ventana del 7-zip. Guardamos y cerramos el programa.

Abrimos el fichero *prueba1.ggb* con GeoGebra y observamos el resultado de los cambios.

2.- Construcción con un objeto aleatorio fijo.

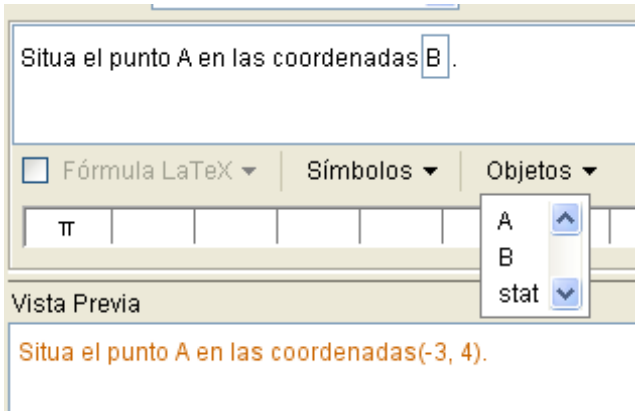
La idea es proponer al alumno una actividad aleatoria en Moodle. El alumno tiene que situar un punto *A* en unas coordenadas determinadas, que serán aleatorias. El alumno podrá repetir la actividad tantas veces como quiera y cada vez las coordenadas serán diferentes. Pero al mismo tiempo, profesor y alumno han de poder consultar la actividad realizada y, por tanto, una vez realizada la actividad, las coordenadas han de quedar fijadas. ¿Cómo se consigue figar la aleatoriedad?

Abrimos GeoGebra. Preparamos la zona gráfica tal y como se ve en la imagen: cuadrícula, ejes y graduación (de 1 en 1 entre -6 y 6), *Atracción de Punto a Cuadrícula* lo ponemos en automático.

Dibujamos un punto *A* en un lugar que no tenga coordenadas enteras. Será el punto que el alumno moverá. Definimos en la barra de entrada un punto *Baleatorio* cuyas coordenadas sean números enteros aleatorios entre -6 y 6.

```
Baleatorio = (AleatorioEntre[-6,6],AleatorioEntre[-6,6])
```

Definimos B = (-3,4), con estas u otras coordenadas. Ocultamos los puntos *B* y *Baleatorio*.



En la parte superior izquierda añadimos un texto con las instrucciones para el alumno y le cambiamos el color. Se trata de un texto dinámico que mostrará las coordenadas del punto *B*. Como se puede observar en la imagen de la derecha.

En la barra de entrada escribimos:

```
status =0
```

Accedemos a *Programa de guión-Scripting* | *JavaScript Global* ponemos[1]:

```
function ggbOnInit() {
  if(ggbApplet.getValue("status")==0){
    ggbApplet.evalCommand("SetValue(B,Baleatorio)");
    ggbApplet.setValue("status",1);
  }
}
```

Y hacemos clic en *OK*.

La primera vez que abramos el archivo, como *status=0*, el punto *B* cogerá los valores de *Baleatorio*, que se han generado aleatoriamente. Seguidamente *status* pasará a valer 1 y, si guardamos la construcción, *B* no volverá a cambiar de valor. Cada vez que abramos el archivo o actualicemos la construcción, *Baleatorio* cambiará, pero *B* no.

La autopuntuación se consigue mediante la definición de un objeto numérico *grade*, cuyo valor se pasa a Moodle cuando el alumno entrega la actividad. Este valor es la puntuación obtenida. En este caso, para conseguir 10 puntos en caso de acierto y 0 en caso de error, definiremos en la barra de entrada[2]:

```
grade=Si[A ≟ B, 10, 0]
```

Finalmente ocultamos la vista algebraica y guardamos la construcción. Y ya está todo listo para crear la actividad en Moodle.

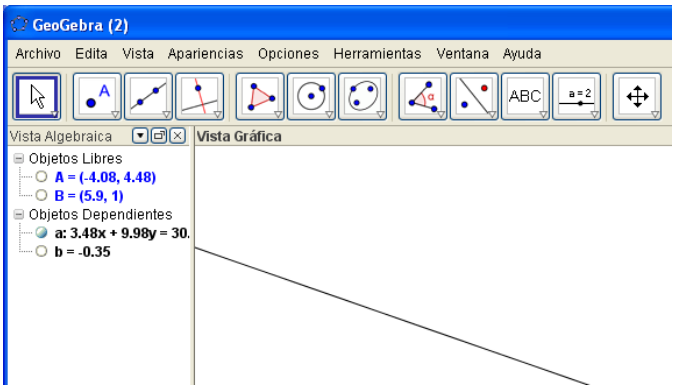
¡Muy importante! Si hacemos pruebas de funcionamiento de la actividad o modificamos la construcción, tenemos que acordarnos de reasignar a *status* el valor 0, *status=0*, antes de guardar, (ya que ha pasado a valer 1) de lo contrario habremos perdido la aleatoriedad.

3.- Detección de la construcción de una recta perpendicular a una dada

La actividad que proponemos hacer consiste en pedir al alumno que dibuje "con regla y compás" una recta perpendicular a una ya dibujada. La actividad tendrá que detectar que el alumno ha dibujado una recta y evaluar si es correcta. El método *registerAddListener* de GeoGebra nos permite saber en qué momento el alumno ha creado un nuevo objeto y cuál es el nombre de este objeto. Para determinar si la recta es correcta, si es perpendicular a la dibujada, compararemos las pendientes.

Abrimos GeoGebra y dibujamos una recta parecida a la de la imagen de la derecha (no tiene que formar ángulos de 45º o 90º con la horizontal o la vertical, sería demasiado fácil). Ocultamos los puntos *A* y *B*. Definimos *b* como la pendiente de la recta que hemos dibujado y lo ocultamos:

```
b=Pendiente[a]
```



[1] Para los scripts en JavaScript el signo de igualdad condicionada necesariamente debe ser el doble igual: == La S de SetValue tiene que ser mayúscula

[2] El símbolo ≟ equivale a poner dos iguales seguidos ==

Definimos un objeto numérico *p* que será el valor de la pendiente de la recta que dibuje el alumno. Ahora pondremos:

```
p=1
```

La recta que dibuje el alumno será correcta si *p = -1/b*. Como trabajamos con valores decimales pondremos la condición: *abs(b + 1 / p) < 0.001*. Por tanto definiremos *grade* de la manera siguiente:

```
grade:= Si[abs(b + 1 / p) < 0.001, 10, 0]
```

Imaginemos que en esta actividad, además de ser autopuntuable, queremos que el alumno reciba un mensaje cuando ha acabado correctamente la construcción. Para ello, crearíamos el texto "¡Conseguido!". Una vez creado, en *Propiedades* | *Avanzado* imponemos que sea visible cuando *grade* sea igual a 10.

Accedemos a *Propiedades de Objeto* | *JavaScript Global* y escribimos (después comentaremos qué hace):

```
function ggbOnInit(){
  ggbApplet.registerAddListener("onAdd")
}
function onAdd(nombre){
  if(ggbApplet.getValue("grade")==0){
    if(ggbApplet.getObjectType(nombre)=="line"){
    ggbApplet.evalCommand("pb: Slope["+nombre+"]");
    ggbApplet.setValue("p",ggbApplet.getValue("pb"));
    ggbApplet.deleteObject("pb");
    }
  }
  else{
    ggbApplet.deleteObject(nombre);
  }
}
```

Comentamos brevemente este código según la líneas de la imagen anterior:

1.- Activamos *registerAddListener*, así cada vez que el alumno cree un objeto nuevo se ejecutará la función *onAdd*.

2.- La función *onAdd* recibe el nombre del objeto creado y lo guarda en la variable *nombre*.

3.- Lo primero que hacemos es comprobar si ya se ha conseguido la recta perpendicular (si grade = 0, no se ha conseguido).

4.- Si no se ha conseguido, comprobamos si el objeto "nombre" es una recta ("line").

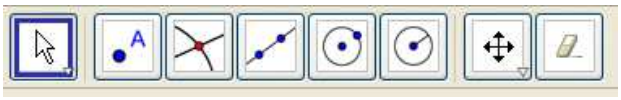
5.- En caso de ser una recta definimos *pb* como la pendiente ("slope") de la recta "nombre".

6.- Y este valor (*pb*) se lo asignamos a *p* (que de ser la recta perpendicular hará que *grade* valga 10).

7.- Y seguidamente borramos *pb*.

10.- En el caso que el alumno haya dibujado la recta, borramos el objeto construido con *ggbApplet.deleteObject(nombre)*.

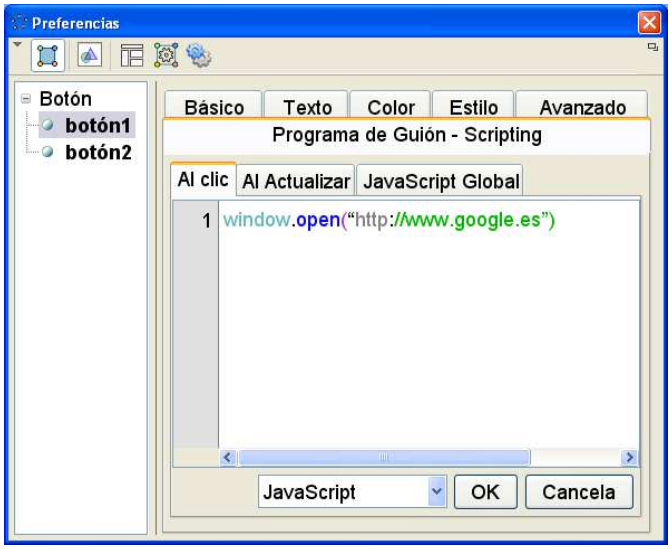
Sólo nos queda ocultar la *Vista Algebraica* y adecuar la barra de herramientas



4.- Multimedia: enlaces y capas.

Abrimos GeoGebra. Insertamos en la zona gráfica dos botones con nombres botón1 y boton2; dejamos en el código en blanco. Accedemos a *Al clic* del primer botón, seleccionamos JavaScript en el desplegable inferior y ponemos en la primera línea:

```
window.open("http://www.google.es")
```

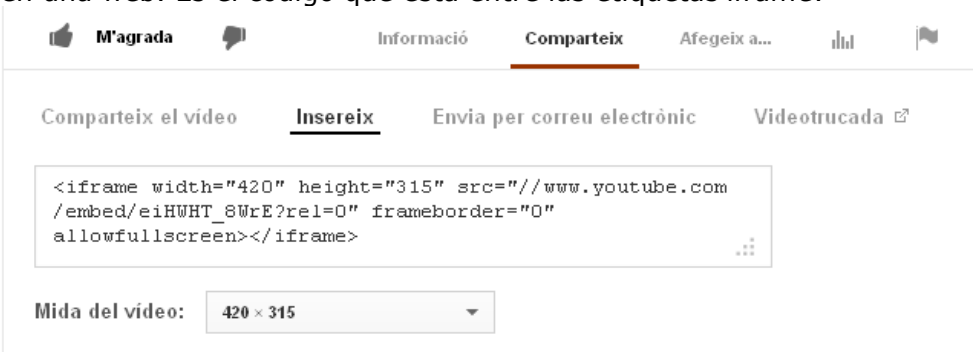


Accedemos a JavaScript Global y escribimos:

```
function ggbOnInit() {
  var
nodog=document.getElementsByTagNa
me("applet")[0];
  addDiv("divggb",nodog.offsetWidth,
nodog.offsetHeight, nodog.offsetLeft, nodog.offsetTop, "visible");
  addDiv("video",450,300,450,20,"hidden");
  video.innerHTML="<iframe width=\"450\" height=\"300\"
src=\"http://www.youtube.com/em bed/OEaxG1FbdQQ?list=PL79ACA33D989D888C\"
frameborder=\"0\" allowfullscreen></iframe>";
}
function addDiv(name,w,h,l,t,v) {
  var divNew = document.createElement("div");
  divNew.id=name; divNew.style.width =w+"px";  divNew.style.height =h+"px";
divNew.style.visibility =v;
  divNew.style.left =l+"px";  divNew.style.top = t+"px";  divNew.style.position
="absolute";
  if (name=="divggb"){
    var nodoggb=document.getElementsByTagName("applet")[0];
```

```
    nodoggb.parentNode.appendChild(divNew);
  }
  else {  divggb.appendChild(divNew); }
}
```

Nos vamos a Youtube, buscamos un vídeo que nos interese y copiamos el código para insertarlo en una web. Es el código que está entre las etiquetas *iframe*.



Volvemos a JavaScript Global y reemplazamos el código que hay en *video.innerHTML* por el que acabamos de copiar al portapapeles. Cambiamos " por \" y añadimos *http:* delante de *//www* [3]

```
  video.innerHTML="<iframe width=\"420\" height=\"315\"
src=\"http://www.youtube.com/embed/eiHWHT_8WrE?rel=0\" frameborder=\"0\"
allowfullscreen></iframe>";
```

Si nos interesa podemos cambiar el tamaño y la posición del vídeo en la llamada a la función:

```
addDiv("video",420,315,450,20,"visible");
```

Accedemos a *Al Clic* del segundo botón y escribimos el código javascript:

```
video.style.visibility="visible"
```

Guardamos y exportamos a hmtl con la opción avanzada *Habilita JavaScripts del navegador* marcada. Comprobamos que funciona correctamente.

5.- Elaboración de un cuestionario

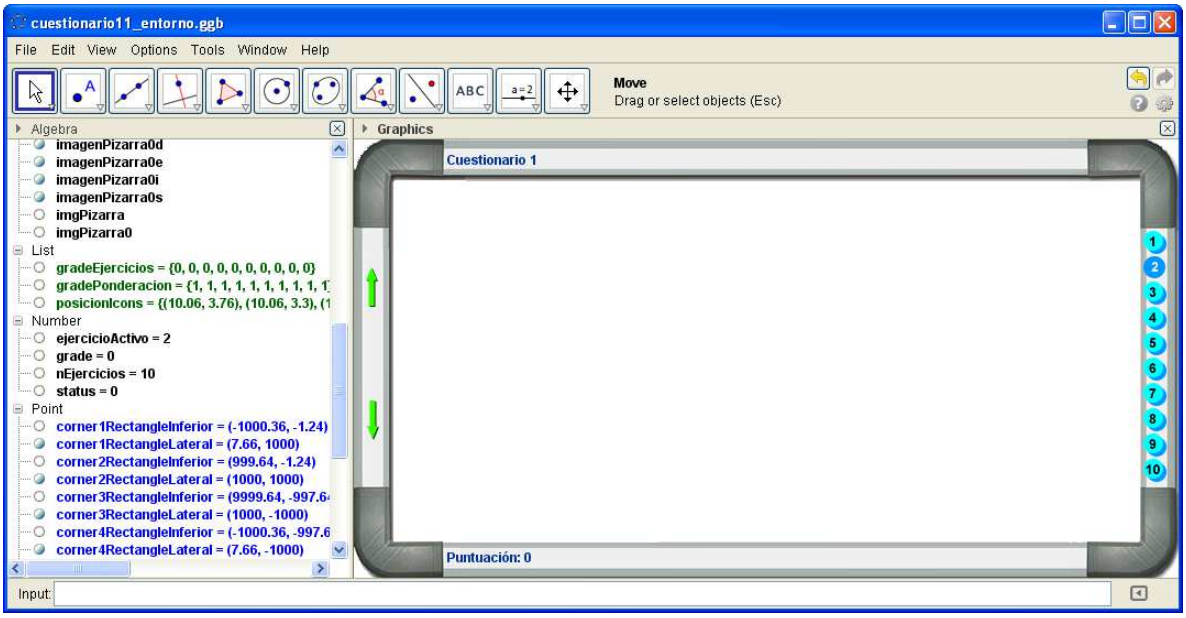
La actividad que proponemos hacer consiste en la elaboración de un cuestionario con cuatro ejercicios. Los cuatro ejercicios son autopuntuables.

Abrimos el fichero *cuestionario_entorno.ggb*. Observemos que aparece la imagen de una pizarra digital interactiva con unos botones a la derecha, unas flechas a la izquierda, un título en la parte superior y un título en la parte inferior de la pizarra con la puntuación.

Haciendo clic en los botones y las flechas podemos desplazarnos por el cuestionario. Cada ejercicio estará en una capa y sólo se visualiza la capa correspondiente al ejercicio activo.

Abrimos la vista algebraica. De todos los objetos nos interesan los siguientes:

[3] Alerta con las comillas. Las comillas del Word y las de GeoGebra son diferentes. Será necesario revisarlas.



Unos objetos que podemos definir al crear el cuestionario.

nEjercicios =10
gradePonderación ={1,1,1,1,1,1,1,1,1,1}
tituloPizarra =”Cuestionario 1”
tituloPie =”Puntuación: ”+grade
posiciónEnunciado = (-3.24, 5.04)

Unos objetos que irán cambiando su valor mientras el usuario realiza el cuestionario.

ejercicioActivo=2
gradeEjercicios={0,0,0,0,0,0,0,0,0,0}
grade=Sum[gradeEjercicios gradePonderacion, nEjercicios] / Sum[gradePonderacion, nEjercicios]*10

Guardamos la construcción con el nombre *triangulo.ggb*. Editamos y cambiamos el valor de los siguientes objetos:

nEjercicios=4
gradePonderación={1,2,3,4,0,0,0,0,0,0}
tituloPizarra = “El triángulo”

Comenzaremos introduciendo los enunciados de los ejercicios. Elegimos la herramienta *Insertar texto*. Hacemos clic dentro de la pizarra. Escribimos el texto del enunciado:

1.- ¿Cuántos lados tiene un triángulo?

Hacemos clic en OK y editamos el texto. Aumentamos el tamaño de la letra. Determinamos la posición en *posicionEnunciado*. Ponemos la condición *ejercicioActivo==1* para la visualización.

Repetimos el proceso con los otros tres enunciados:

2.- ¿Cuánto suman los ángulos de un triángulo?
3.- ¿Cuál es la fórmula del área de un triángulo de base b y altura h?
4.- Dibuja un triángulo de $6u^2$ de área.

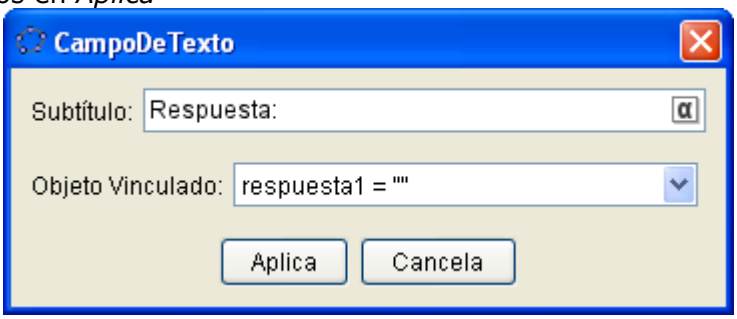
En la *Entrada* escribimos las soluciones a las preguntas y los objetos que contendrán las respuestas. Observemos que la primera solución es un número, la segunda un ángulo y la tercera una función de dos variables; en cambio, las respuestas siempre son texto, para evitar que el usuario haga uso de variables y comandos GeoGebra. El cuarto ejercicio no tiene solución ni respuesta.

solucion1 =3
solucion2 =180º

solucion3(b,h) = b*h/2
respuesta1=""
respuesta2=""
respuesta3=""

Ocultamos los objetos *respueta1*, *respuesta2* y *respuesta3*.

Seleccionamos la herramienta *Casilla de entrada*. Hacemos clic en la pizarra y rellenamos el formulario. Clicamos en *Aplica*



Ajustamos la posición de la *Casilla de entrada* a nuestra conveniencia. Accedemos a *Propiedades* y ponemos la condición *ejercicioActivo==1* para la visualización.

Repetimos el proceso para las otras dos casillas de entrada para *respuesta2* y *respuesta3*.

El siguiente paso es verificar que la respuesta introducida por el usuario tiene un formato correcto. En la primera pregunta la respuesta tiene que ser un número, por tanto los caracteres admitidos son las cifras del 0 al 9. En la *Entrada* escribimos:

listaCodigosAdmitidos1 = CódigoDeTexto["0123456789"]

Hacemos lo mismo con las otras peguntas:

listaCodigosAdmitidos2 = CódigoDeTexto["0123456789°"]
listaCodigosAdmitidos3 = CódigoDeTexto["0123456789bh*+-/() "]

Para realizar la verificación del formato codificamos también la respuesta (observemos que las respuestas no son texto, por ello tenemos que poner "”+respuesta1):

listaCodigosRespuesta1 = CódigoDeTexto["”+respuesta1]
listaCodigosRespuesta2 = CódigoDeTexto["”+respuesta2]
listaCodigosRespuesta3 = CódigoDeTexto["”+respuesta3]

El formato de la respuesta será válido si los caracteres utilizados están incluidos en el conjunto de caracteres admitidos. Por ello en la *Entrada* escribimos:

validacion1=listaCodigosRespuesta1 ⊆ listaCodigosAdmitidos1
validacion2=listaCodigosRespuesta2 ⊆ listaCodigosAdmitidos2
validacion3=listaCodigosRespuesta3 ⊆ listaCodigosAdmitidos3

Insertamos un texto grande de color rojo, que llamaremos *txtFormatoIncorrecto*:

Formato incorrecto. Revisa la respuesta.

Editamos el texto y en *Condición para exponer objeto* ponemos:

¬(validacion1 ∧ validacion2 ∧ validacion3)

Substituimos el código de *Al clic* en *icon1* por este otro código JavaScript (y de forma análoga para *icon2* y *icon3 poniendo en la segunda línea un 2 y un 3 en vez del 1*):

```
if (ggbApplet.getVisible("txtFormatoIncorrecto")==0){
  ggbApplet.setValue("ejercicioActivo",1);
  ggbApplet.evalCommand("ShowGrid[False]");
}
```

Para *icon4*, haremos una cosa parecida, pero tendremos que poner *True*, para que sólo en este caso se vea la cuadrícula.

```
if (ggbApplet.getVisible("txtFormatoIncorrecto")==0){
   ggbApplet.setValue("ejercicioActivo",4);
   ggbApplet.evalCommand("ShowGrid[True]");
}
```

Ara toca la corrección y la puntuación de las respuestas. Accedemos a las propiedades del objeto *respuesta1*. En la pestaña *Al actualizar* ponemos el siguiente código JavaScript:

```
ggbApplet.evalCommand("UpdateConstruction[ ]")
if(ggbApplet.getValueString("respuesta1")!="" && ggbApplet.getValue("validacion1")==1)
{
ggbApplet.evalCommand("SetValue[gradeEjercicios,1,If[solucion1=="+ggbApplet.getValue
String("respuesta1")+",1,0]]");
}
```

Y de forma análoga en *respuesta2*

```
ggbApplet.evalCommand("UpdateConstruction[ ]")
if(ggbApplet.getValueString("respuesta2")!="" && ggbApplet.getValue("validacion2")==1)
{
ggbApplet.evalCommand("SetValue[gradeEjercicios,2,If[solucion2=="+ggbApplet.getValue
String("respuesta2")+",1,0]]");
}
```

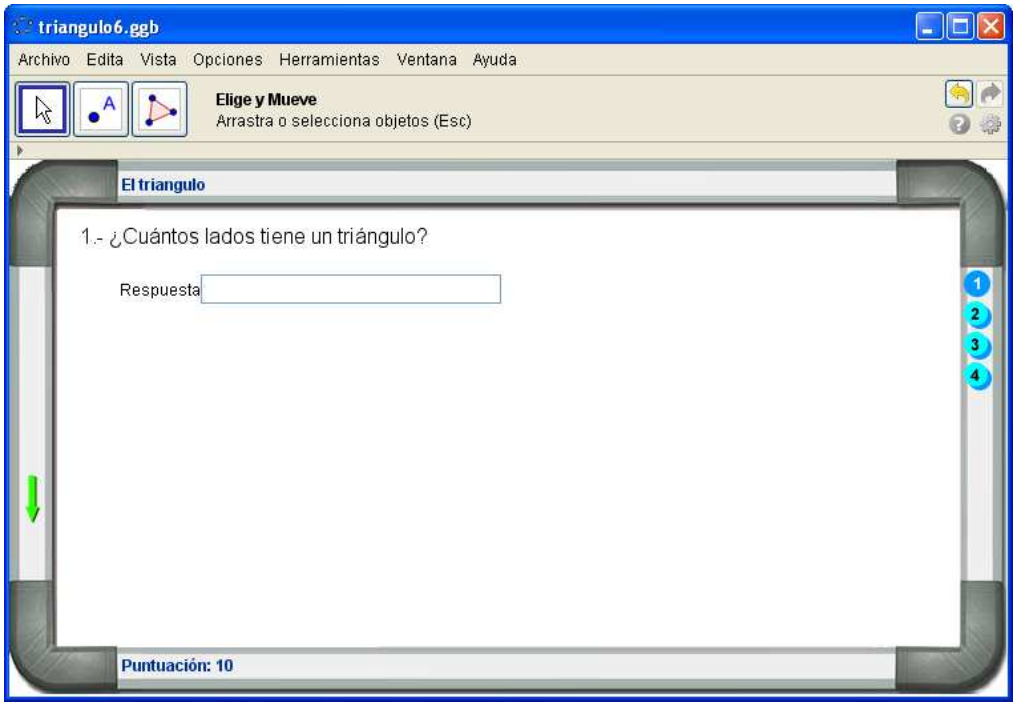
Pero en *respuesta3* podremos otro código debido a la tipología de objeto: una función multivariable

```
ggbApplet.evalCommand("UpdateConstruction[ ]")
if(ggbApplet.getValueString("respuesta3")!="" && ggbApplet.getValue("validacion3")==1){
   ggbApplet.evalCommand("g(b,h)="+ggbApplet.getValueString("respuesta3"));
   ggbApplet.evalCommand("SetValue[gradeEjercicios,3,If[solucion3==g,1,0]]");
   ggbApplet.evalCommand("UpdateConstruction[ ]")
   ggbApplet.deleteObject("g");
}
```

El último ejercicio será de construcción, en el que el usuario tendrá que construir con la herramienta triangulo un triángulo de área $6u^2$. Para la puntuación escribimos en Global JavaScript:

```
function ggbOnInit(){
   ggbApplet.registerAddListener("onAdd")
}
function onAdd(nombre){
   if(ggbApplet.getValue("ejercicioActivo")==4){
  ggbApplet.evalCommand("SetConditionToShowObject["+nombre+",ejercicioActivo==4]");
      if(ggbApplet.getObjectType(nombre)=="triangle"){
         ggbApplet.evalCommand("SetValue[gradeEjercicios,4,If[abs(6-
"+ggbApplet.getValue(nombre)+")<=0.001,1,0]]");
      }
   }
   else{
      ggbApplet.deleteObject(nombre);
   }
}
```

Antes de hacer la exportación a html: elegimos las herramientas de las que dispondrá el usuario para hacer el triángulo, ocultamos la ventana algebraica, ajustamos las dimensiones de la ventana gráfica a la pizarra y comprobamos que el contenido de los formularios está en blanco.



Cuando hagamos la exportación a html tendremos que activar la opción *Habilitat Javascripts del navegador* y desactivar la Barra de menús y la Barra de entrada.