UNIVERSIDAD NACIONAL DE COLOMBIA

TALLER	COODINACIÓN ENTRE AGENTES

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1. El problema de Subastas

NetLogo es un lenguaje de programación simple y adaptado al modelado/simulación de fenómenos en los que aparecen muchos individuos interactuando (como, por ejemplo, en los fenómenos habituales que se dan en la naturaleza, las sociedades, o muchas áreas de las ciencias).

El objetivo, A pesar de todas las ventajas, es analizar el modelo de coordinación entre agentes asociado a la actividad de una subasca como marco de referencia a los estándares planteados en los Sistemas MultiAgentes.

ACTIVIDAD 1: SIMULACION DE UNA SUBASTA

Objetivo de la simulación: El propósito es simular una Subasta.

Explicación general del problema:

En general, una subasta es una actividad donde se pueden vender objetos (es decir, automóviles, obras de arte, etc.) o bienes raíces, valores, derechos, o servicios según las reglas establecidas, que identifican:

- El tipo y número de mercancías.
- El tipo de acción
  - Privada o pública
  - Subasta de oferta ascendente o subasta de oferta descendente
  - Subasta de oferta sellada de primer precio o subasta de oferta sellada de segundo precio
  - A quién se le permite participar (todos o solo bajo invitación formal)
  - La forma de las ofertas (escritas, orales, en línea)
  - El castigo por no comportase con sencillez.

Los participantes de la subasta se dividen en:

- Los postores (que hacen las ofertas)
- Los vendedores
- El subastador.
- **Nota:** A veces, el vendedor y el subastador pueden ser la misma persona.

Las ofertas de los postores se denominan licitaciones. En general, las pujas tienen un valor mínimo y máximo y, dependiendo del tipo de subasta, la puja ganadora puede ser la puja más baja o más alta o la segunda puja más baja o más alta.

Las subastas pueden verse como *jugadas NO cooperativas*, porque los postores compiten unos contra otros por el mismo objeto y, al mismo tiempo, cada postor compite contra el vendedor por maximizar cada beneficio.

Se simulará particularmente el tipo de subasta inglesa (subasta privada de puja ascendente) que es el tipo de subasta más popular.

Hipótesis: En esta subasta inglesa:

- Solo se venden objetos; por lo tanto, las ofertas son en forma oral y quien hace la última oferta máxima es el ganador de la subasta.
- La última oferta máxima también se denomina "Precio del Martillo".
- Se establece el aumento mínimo de la oferta mediante un control deslizante en la interfaz, el cual se llama "OfertaMinima". Esto permite que el usuario pueda ejecutar el modelo con diferentes valores de este parámetro y comparar los resultados.
- En el modelo, el número de postores lo establecen los usuarios, utilizando el control deslizante "NumeroPostores" establecido en la interfaz.

Se puede establecer la duración máxima de una subasta mediante un control deslizante:

- El usuario puede decidir el tiempo máximo de aumento (número), si se alcanza este número, el modelo decide el ganador de la oferta, sin aumentar ni más.
- Este control deslizante se llama "Tiempo".

Otros supuestos del modelo:

- El vendedor y el subastador son la misma persona.
- Los postores son al menos dos.
- Solo hay un objeto a subastar (porque los objetivos son seguir la dinámica del precio máximo en la subasta y conocer la satisfacción del subastador).
- El precio inicial del objeto se define mediante el control deslizante "PrecioObjeto" en la interfaz.
- Cada postor tiene una cantidad de dinero establecida al azar. El dinero de los postores se establece aleatoriamente entre un valor mínimo (50 Euros) y un valor máximo, que puede establecer el usuario mediante el control deslizante "CantidadEuros" en la interfaz.

- Cada postor tiene un precio de reserva para cada objeto: es el máximo precio que cada postor está dispuesto a gastar por ese objeto. El resto de los participantes lo desconocen y su valor inicial es independiente del de otros postores.

    - Se crea una variable llamada "PrecioReservacion" y se establece cada valor inicial del precio de reserva de forma aleatoria en función del precio del objeto.
- En este modelo, la variable "PrecioReservacion" también es una característica del subastador, y es igual al precio inicial del objeto.
    - Esto ocurre porque uno de los objetivos del modelo es observar, como se dijo antes, la satisfacción del subastador.

- Se mostrará en la ventana de salida (llamada 'Satisfacción') y es la diferencia entre el precio de reserva del subastador y el precio de remate.

ACTIVIDAD 1. ANÁLISIS DE LA INTERFAZ

Descargar el archivo V0-SimulacionSubasta.nlogo y cargarlo en el ambiente NetLogo Web el cual contiene la interfaz inicial del sistema de subasta (Figura 1)

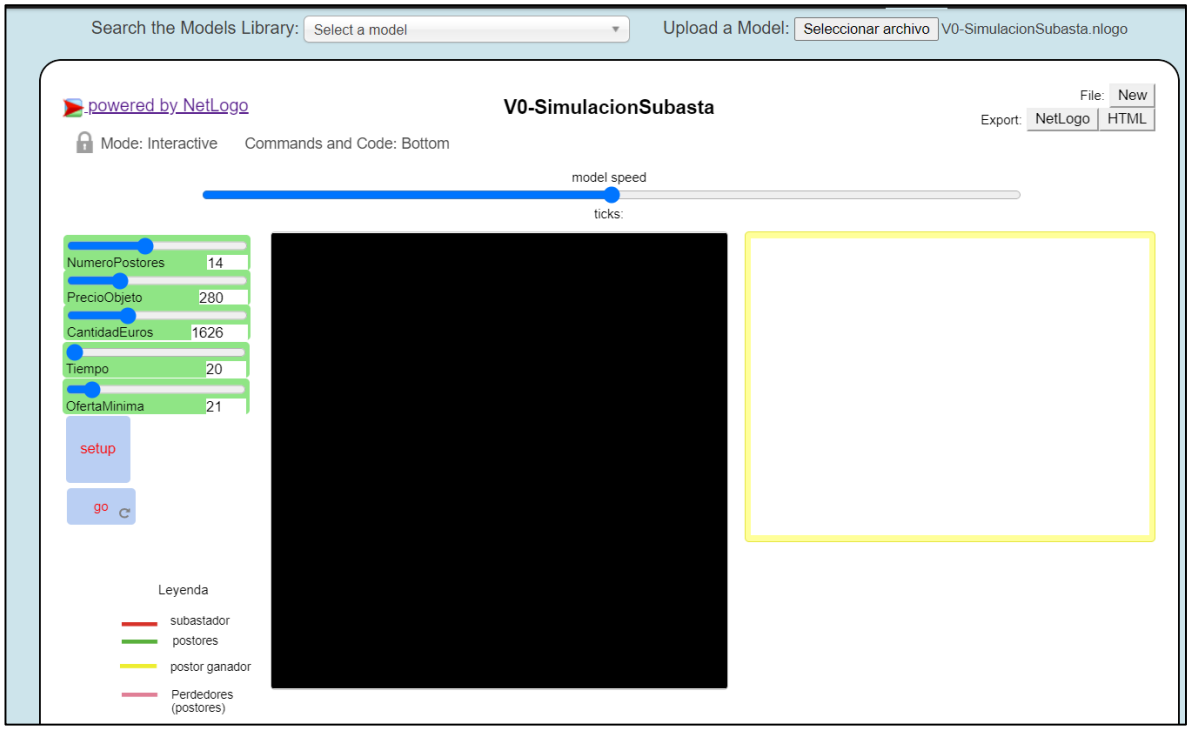

Figura 1. Interfaz inicial del sistema

Detalle los componentes principales del sistema y relaciónelos con la descripción inicial del modelo hecho al inicio del taller.
Detalle que debajo de las entradas hay una leyenda que explica todos los colores de los agentes en el mundo.

El espacio en blanco a la derecha en la Figura 1 es el espacio de salida. Cuando la simulación se realiza, los resultados se informan en esta parte.

Finalmente, se resalta que en el modelo, el flujo del tiempo es monitoreado por la oferta de un postor. El último **tick** es el período en el que no se realizan ofertas y en el que el modelo elige al postor ganador.

ACTIVIDAD 2. FUNCIONES Y CÓDIGO

En primer lugar, definamos todos los agentes del modelo como un todo y como un individuo de la siguiente manera:

```
breed [subastadores subastador]
breed [postores postor]
breed[objetos objeto]
```

Definimos tres conjuntos de agentes: los subastadores, los postores y los objetos.

Luego definimos propias de los agentes y las variables globales a utilizar en el sistema

```
subastadores-own[PrecioReservacionA Satisfaction]
postores-own[PrecioReservacionB  oferta euro]
objetos-own [PrecioOb]

globals [n K S OfertaPrecio]
```

La primer función del modelo es la función **setup.** En este se crean los agentes en el mundo (objetos, subastadores, postores) y define algunas características de los agentes.

```
to setup


 ca

 reset-ticks

 set K  false

 create-objetos 1 [set size 1.5 set shape "flag" setxy  16 15

                set PrecioOb PrecioObjeto set label PrecioOb]

 create-subastadores 1 [set size 1.5 set shape "person" set color red set size 1.5 setxy 16 8 set
PrecioReservacionA PrecioObjeto]

 create-postores NumeroPostores [set shape"person" set color green set size 1.5

                        set PrecioReservacionB (2 * PrecioObjeto + 80 + random (200))

                        set euro 50 + random(CantidadEuros)
                         ]
```

La primera línea inicializa el valor del indicador K como falso. K es un indicador que se utiliza para detener el modelo en dos situaciones particulares: cuando sólo hay uno o dos postores que quieren hacer ofertas por el objeto y cuando ningún postor hace una oferta (todos los postores tienen un precio de reserva menor que el precio del objeto).
Las líneas restantes de la función crean 1 objeto, 1 subastador y un número de postores igual al valor introducido por el usuario en la interfaz.
Cada objeto se crea con unas características establecidas en estas líneas comunes a todos los agentes. De hecho, los agentes tienen el tamaño (qué tan grande aparece el agente en el mundo de la interfaz), la forma (la forma del objeto en el mundo), el color y la posición en

la interfaz (a excepción de los postores, que están dispuestos en las siguientes filas en pocas líneas de códigos que siguen). Ver código siguiente asociado a los postores.

```
let an 2
let side sqrt NumeroPostores
let step 2.2
let x 0
let y 0
while [an < NumeroPostores + 2]
                    [ if x > (side - 1) * step
                                    [ set y y - step
                                      set x 0]
                      ask postor an [setxy x y]
                      set x x + step
                      set an an + 1]
```

Por último, se imprime el precio inicial y el tiempo máximo en el recuadro de texto al lado derecho del modelo gráfico

```
output-write "Precio Inicial  "
output-print  PrecioObjeto
output-write "Tiempo Maximo  "
output-print Tiempo

end
```

La siguiente función importante es **go.** Es una función que hace referencia a otras funciones, que se ejecutan en el orden dado por la función go. La función se detalla a continuación.

```
to go

if (all? postores [color != yellow]) [tick
                                hacer-oferta
                                perdedores
                                stop-model
                                ifelse(ticks >= Tiempo or K = true)
                                   [ganador stop]
                                   [output-write "Precio Actual del Objeto  "
                                       output-print OfertaPrecio
                                       output-write "Numero de ticks  "
                                       output-print ticks]]
end
```

Primero que nada, vemos el bucle “if”:

- Esto es necesario para mejorar la función **go** solo cuando la subasta no ha terminado
- Cuando la subasta termina, el ganador se colorea en amarillo: entonces, si hay un agente amarillo, hay es un ganador y la subasta es concluida.
- Finalmente, en cada interacción entre los agentes, se ejecuta la función go, y los 'ticks' del contador se aumenta en 1.

Hay cuatro acciones en la función **go**,

- La primera función es la función hacer-oferta que hace las ofertas y los aumentos.
- La segunda función es la de perdedores, que divide a
  - los postores que aún están en el juego (que tienen suficiente dinero para aumentar la oferta máxima y comprar el objeto)
  - de los postores fuera del juego (que no tienen suficiente dinero para aumentar la oferta máxima).
- La tercera función stop-model es un modelo de parada que establece el índice K a un valor de verdadero si se respetan las condiciones de un final anticipado de la subasta.
- La última función ganador ocurre solo cuando K es verdadero o cuando se alcanza tiempo máximo de la subasta.

Las respectivas funciones son las que se describen a continuación.

HACER OFERTA

```
to hacer-oferta


let firstpostor n-of 1 postores with [PrecioReservacionB > PrecioObjeto
                                and color != violet]

ask firstpostor [if (ticks = 1)
              [set oferta (PrecioObjeto + OfertaMinima)
               set OfertaPrecio max[oferta] of postores
               if (oferta <= euro)
                  [set label oferta
                   if (oferta > PrecioReservacionB)
                     [set oferta PrecioReservacionB set label oferta]]]]

let apostor one-of postores with [color != pink and
                            oferta != max[oferta] of postores and
                            color != violet and
                            PrecioReservacionB > PrecioObjeto]
ask apostor [if (ticks > 1)
           [set oferta (floor ((0.02) * PrecioReservacionB)+
              (max[oferta] of postores +  OfertaMinima))
            if (oferta > PrecioReservacionB)
              [set oferta 0
               set label oferta
               set K true]
            set label oferta
            set OfertaPrecio max[oferta] of postores]]


let bpostor one-of postores with [color != pink and color != violet]
ask bpostor [if (PrecioReservacionB > PrecioObjeto
              and PrecioReservacionB < euro)
              [if (ticks mod 3 = 0)
                 [set PrecioReservacionB
                   (PrecioReservacionB + OfertaPrecio )]]]

end
```

PARAR EL MODELO

```
to stop-model

 set n count postores with [color != pink
                       and color != violet ]
 if (n <= 2  or all? postores [oferta = PrecioReservacionB])
                        [ set K true ]

end
```

GANADOR

```
to ganador



 ask postores[if  (oferta = max[oferta] of postores)
               [set euro (euro - oferta)
                set color yellow
                set label oferta]]

 ask subastadores [set Satisfaction
                    (max [oferta] of postores - PrecioReservacionA)
                  set S Satisfaction
                  output-write "Precio del Martillo "
                  output-print OfertaPrecio
                  output-write "Satisfaction "
                  output-print Satisfaction
                  output-write "Tiempo Final "
                  output-print ticks
                  ]
 end
```

PERDEDORES

```
to perdedores

    ask postores[if (euro < max[oferta] of postores
                  and color != violet
                  and color != violet + 3)
                   [set color pink]]

end
```

EXPERIMENTOS

Realizar diferentes experimentos cambiando los valores de entrada del modelo. Esto permite observar las diferentes interacciones entre los agentes.