



Instructions for designing auctions

1. Basic elements for creating all type of actions in a **Design** window:

Player type - to assign a subject a role in the experiment

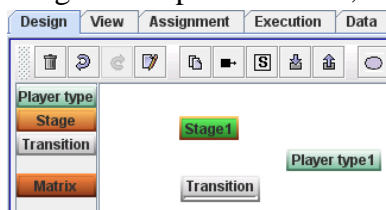
Stage - for instructions, bidding window, summaries.

Transition - for updating variables and deciding the stage transition

 (Toolbox) – for initializing variables, and choices. Click on  to open Toolbox and create **Variables**, and **Choices**.

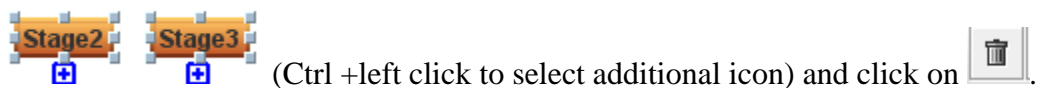
2. Basic manipulation of the icons

a. Drag and drop **Player type**, **Stage** and **Transition** on **Design** window:





The position of the icons on **Design** window can be arranged in any way. Note the first created **Stage** is always green (i.e. the start of the game).

b. To delete an icon from **Design** window select it/them by clicking on it

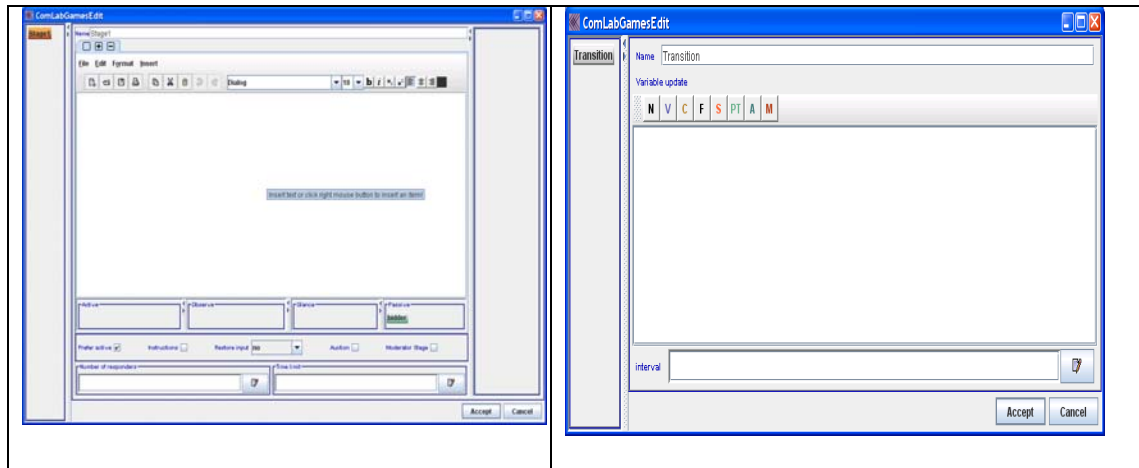




c. To rename **Player type**, **Stage** and **Transition**


Double click on any of the icons that were dropped on **Design** window to rename it. For example double click on **Player type** and rename it to **bidder** and then click “Enter” to see **bidder**. Similarly for **Stage** and **Transition**.

d. To open **Stage** and **Transition** by right double clicking on the **Stage** or **Transition** or select the icon  and click on .

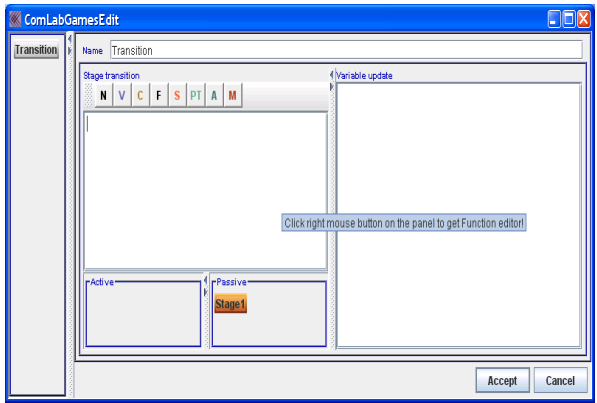




- e. To connect a **Stage** with the **Transition**, click on a **Stage** to select it: , point the mouse to , hold the left key and drag it to **Transition**.




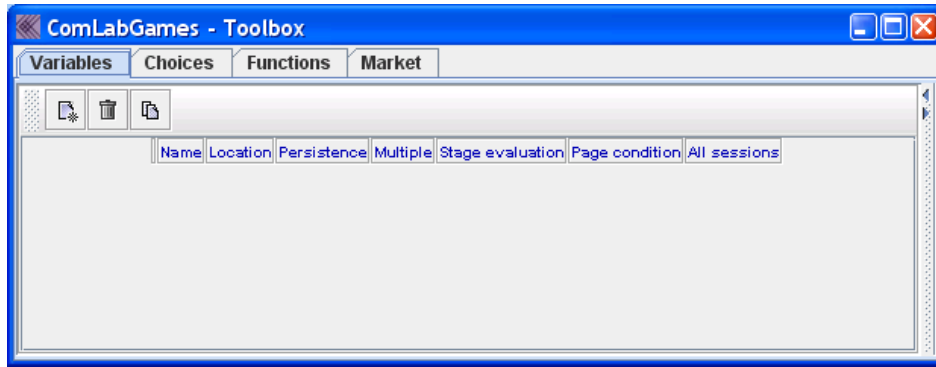
Stage transition is added to **Transition**






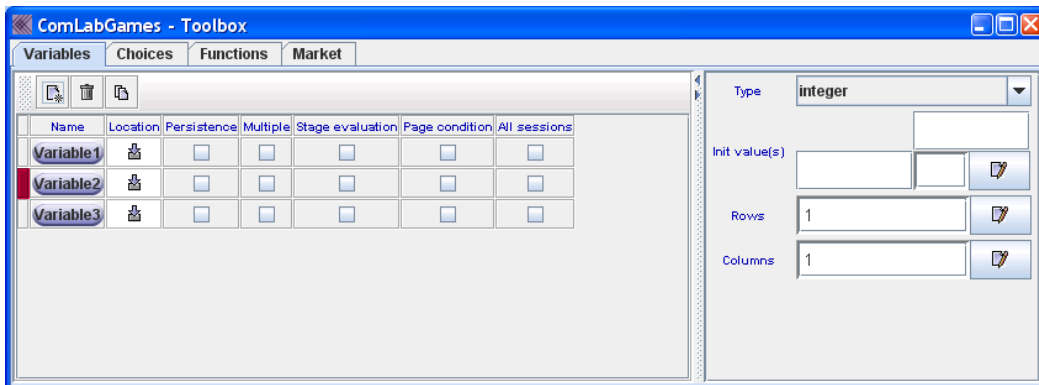
The left side where **Stage transition** is written allows a moderator to determine a rule from going from the current **Stage** to any other **Stage**. The rule is expressed as Boolean.



3. Create variables for the auction

- a. Click on  and the following window opens:



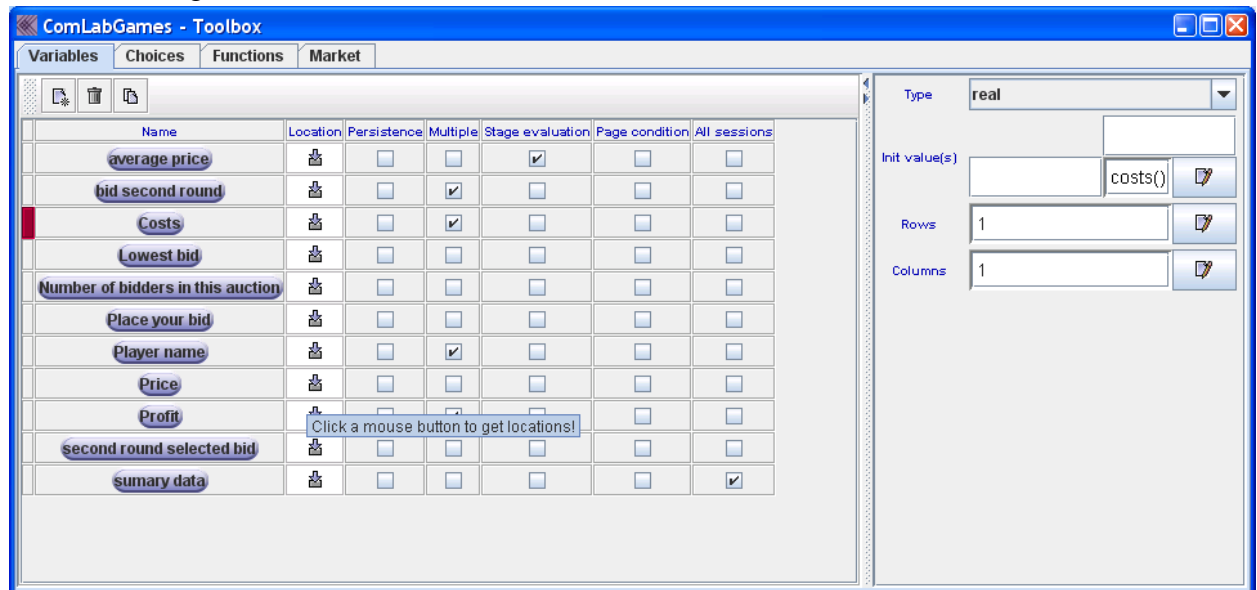
- b. There are **Variables** **Choices** **Functions** **Market** tabs in the toolbox. For auctions we will need Variables, Choices and sometimes Functions.
- Created variables are purple **Variable1**
 - Created choices are yellow **Choice1**. Choices are created in order for subjects to make a choice.
- c. Click on  to add a variable if you are in Variables tab.
- d. To delete a variable, select a variable by clicking on the variable, the red rectangle shows which variable was selected **Variable2**  and click on .



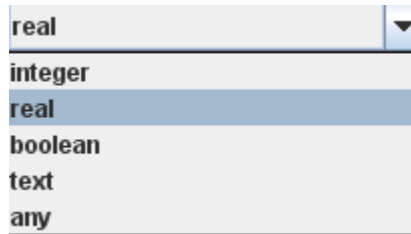
- e. To duplicate a variable, click on the variable that you want to duplicate **Variable2**  and click on .
- f. To rename a variable double click on the variable icon and rename it.



- g. For the First price sealed bid auction the following variables were created (we will discuss each variable in more detail under “4. Creating Bidding Window with **Stage** and 5. Creating Instructions with **Stage**)



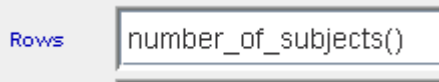
- h. For each variable a **Type** should be selected from the menu. The options are:




(any if you are not sure of the type).




By default the **Type** is integer.

- i. Each variable has the option for **Init value(s)** - initial value, which can be blank, a function or a constant.
- j. Each variable can be a scalar, a row vector, a column vector or a matrix . Number of row and columns can be a function (i.e. for example, the variable **summary data** has number of rows determined by the function

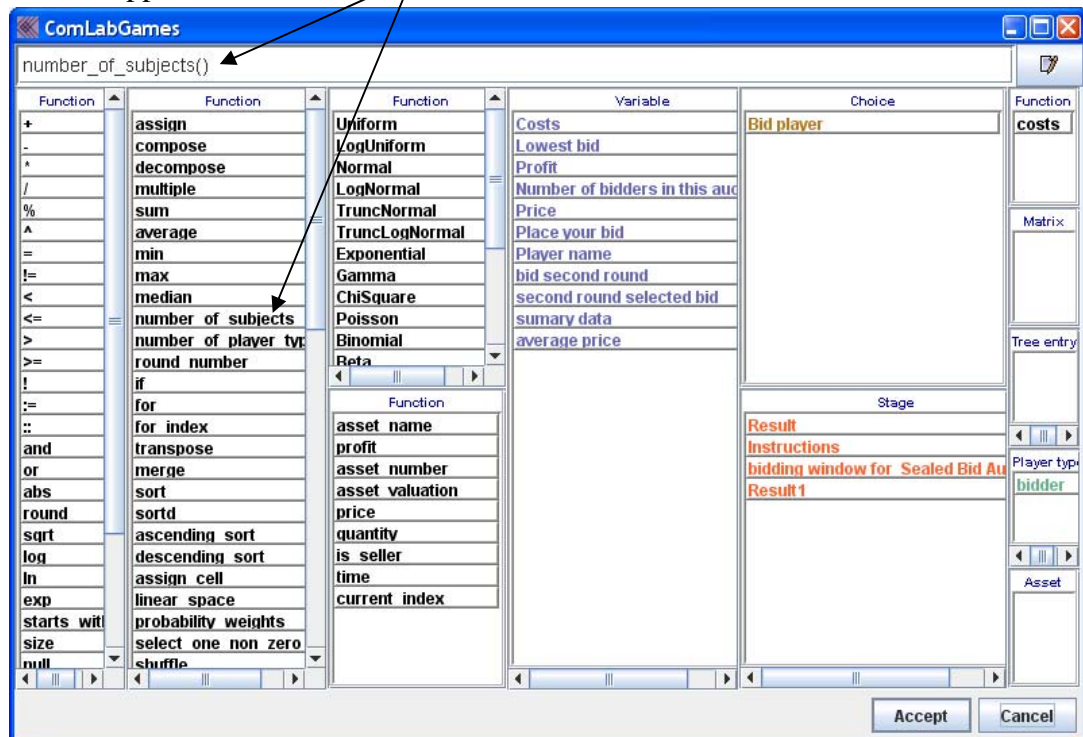


“number_of_subjects()”. If we change the number of subjects on the **Assignment** tab the number of rows will adjust to reflect that change.

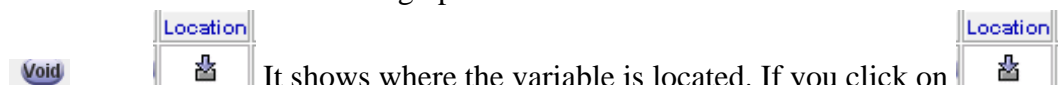
- k. To write an initial value as a function or a variable, you can click  next to initial value or Rows and Columns


Init value(s)		costs()	
Rows	1		
Columns	1		

And the function editor appears. Double click on function, variable, choice that you want to appear as a statement.



- l. Each variable has the following options:





It shows where the variable is located. If you click on  next to **Costs**, the following places are identified:






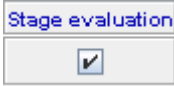
. If you click on


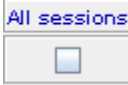




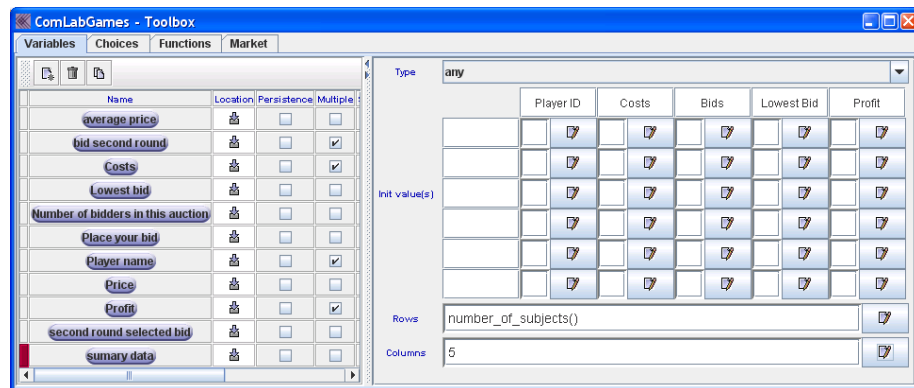
, the stage will open and you will be able to see where **Costs** are located.

  If selected the value of the variable will carry from one round to the next.




  If selected it means that a variable is individual specific.

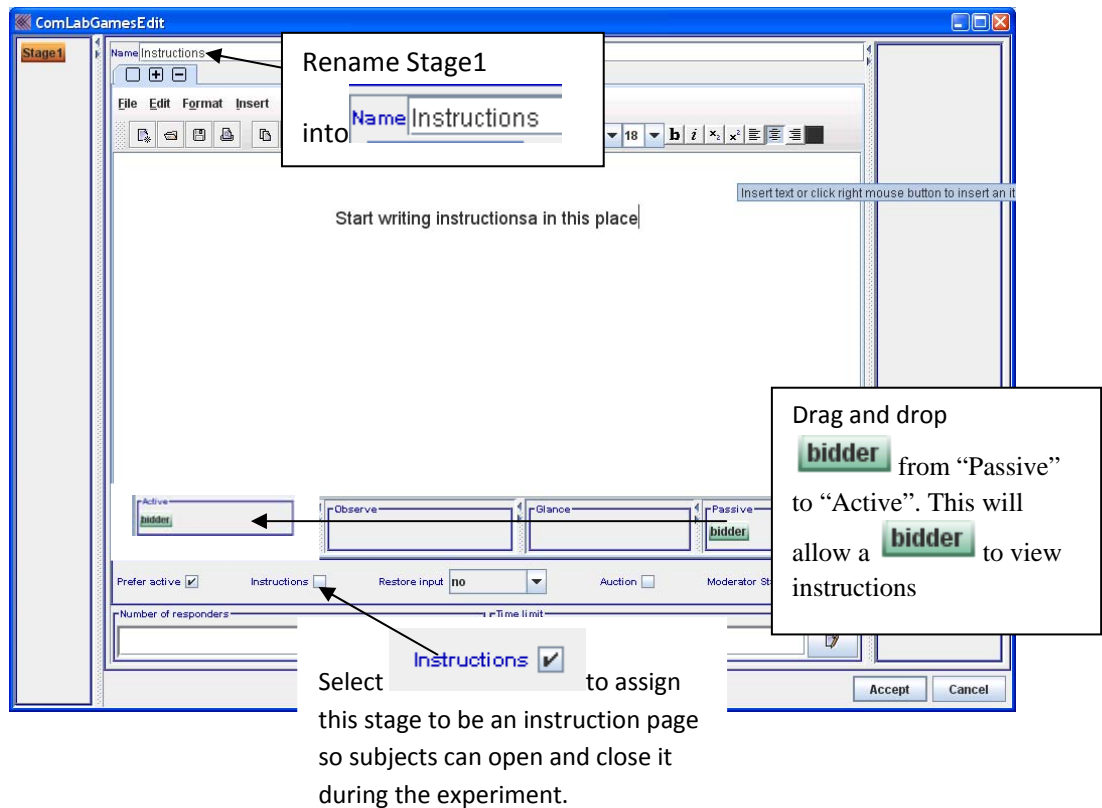
  If selected creates a variable that evaluates a variable for all sessions together (i.e. sum, average).


  If selected it reports the values of this variable for all sessions played in this experiment. For example  has All sessions  selected.



4. Create instructions with (example from First price sealed bid auction)


- Right click on a  that you droppend in the  window. The following  window should appear:

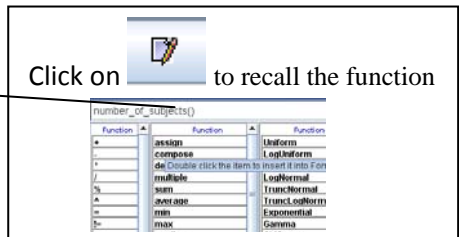
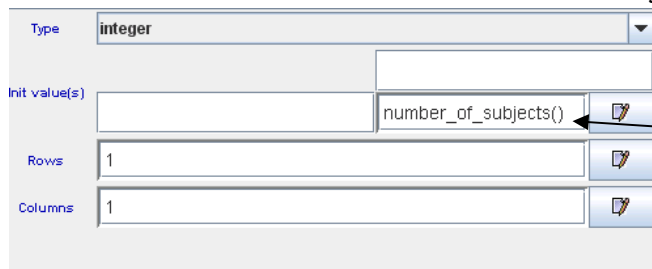




b. Instruction stage has the following color: 

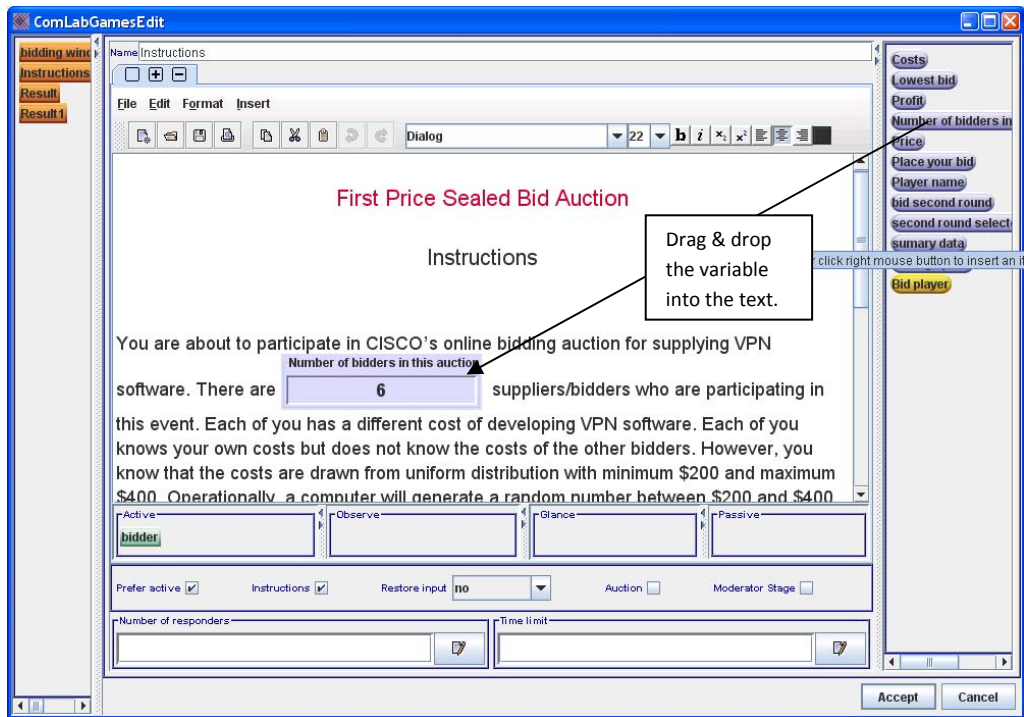
c. In the text you might want to insert a , for example a number of bidders.

To View this variable, click on  and select a variable:

. This variable is a scalar (1x1) and it is not multiple (i.e. not individual specific) because each subject sees the same value. It is an integer, and the initial value is a function "number_of_subjects()".

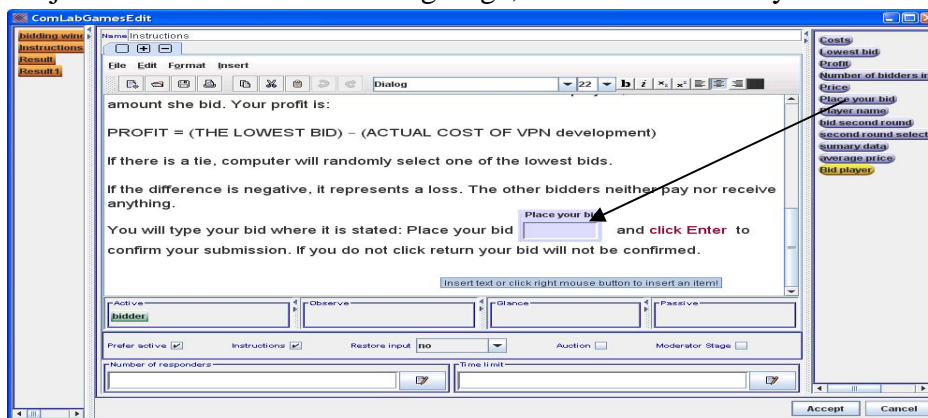


d. To insert a  into the text, drag&drop the variable from right to insert it into the text inside a 




Type	text
Init value(s)	
Rows	1
Columns	1



- e. Variable **Place your bid** was created to illustrate what subjects have to do in the bidding stage, and it was used only in the instructions :




5. Creating **Variables** needed for the Bidding **Stage** in the First Price Sealed Bid Auction


- a. Click on

- b. Go to **Functions** and create a function . We name a function **costs** . The costs are drawn from the uniform distribution with minimum 200 and maximum 400. Write the formula or use  to access the Function editor. Note “Evaluate once” is not selected. This means that each subject will get a different draw from the distribution.

Name	Location	Formula	Evaluate once
costs		Uniform(200, 400)	

- c. Go to **Variables** and create a new variable by clicking on , rename it to **Costs** . The variable is “Multiple” (i.e. individual specific).

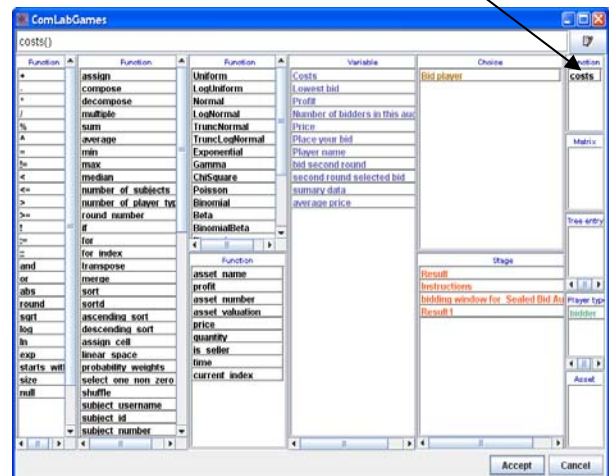
Type	Init value(s)	Rows	Columns
Costs		1	1


- d. Select “real” for Type, and type/insert the costs() function that you created under **Functions** into the initial value. If you click on  next to Init. Value(s) the costs() function will appear in Function editor

Type	Init value(s)	Rows	Columns
real	costs()	1	1

All the functions that you created are under Functions to the right.

Double click on it to bring it into the editor and then click “Accept”



- f. The variable **Costs** is a scalar, number of rows=1 and number of columns=1.
- g. Create a variable **Number of bidders in this auction**  which has the following function for the initial value: number_of_subjects()

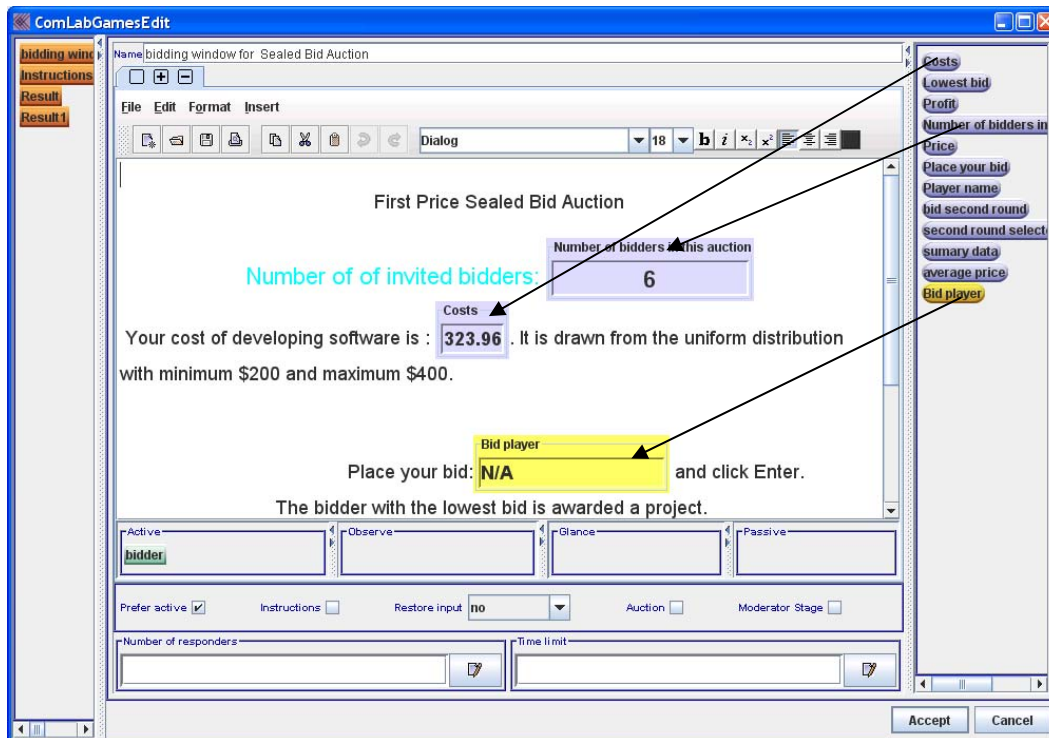
6. Created **Choices** needed for the Bidding **Stage** in the First Price Sealed Bid Auction

- Click on
- Go to **Choices** and create a new choice by clicking on . Rename the choice appropriately (**Bid player**). This choice should allow subjects to write a real number with the lower bound 0.

- Choice **Bid player** has ticked which means that a subject write a decision then clicks on “Enter” to confirm it. Type is real and the Minimum is set to zero.

7. Create Bidding Window with **Stage** for First Price Sealed Bid Auction

- Drag and drop a **Stage** into **Design** window
- Right click on a **Stage** to open it.
- Write the appropriate message in the stage.
- Drag and drop variables **Costs** and **Number of bidders in this auction** in the appropriate spot.
- Drag and drop the choice **Bid player**



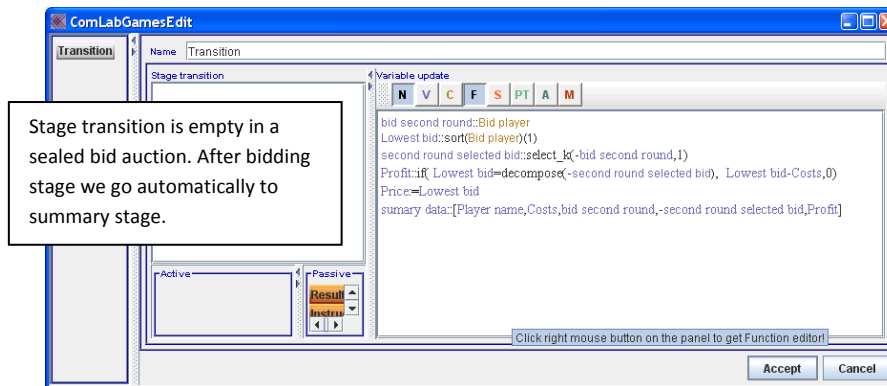
f. Make player type **bidder** "Active" by dragging from "Pasive" to "Active".

8. Create **Transition** by dragging and dropping it into the **Design** window

- Select the stage **bidding window for Sealed Bid Auction**, point the mouse on **+** and drag it to **Transition** to connect the stage with the transition. You should get the following:



- Right click on **Transition** to edit it.
- Write the update of the variables on the right side of **Transition**.



- d. General rule for updating variables in **Transition** for later use.

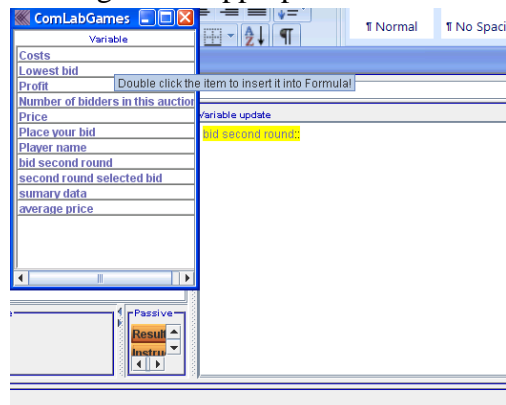
Variable update

N V C F S PT A M

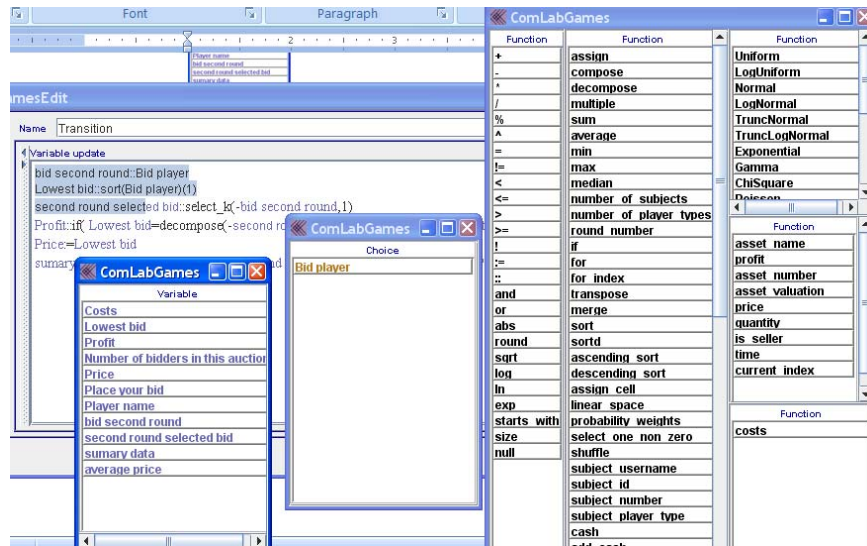
Void Click on the empty space below

Void The cursor has to be in the editor

Void Select the variable that you want to update by clicking on **V** and double clicking on the appropriate variable from the menu of variables that you created:



- Void** A double colon (:) has to follow the variable if the updated variable is used in the same transition and needs to be updated before the steps that follow.
- Void** A column and equal sign (:=) has to follow a variable if the updated value is not used in the next steps within the same transition but in the follow up stages.
- Void** The expression is highlighted in yellow until it is correctly written
- Void** **Do not click “Enter”** until you finish the expression for the updated variable.
- Void** **Click “Enter”** to start updating the next variable.
- Void** **To insert choices (C), functions (F), stages (S), player types (PT), assets into the expression, just click on C, F, S, PT, A.** Everything is listed in a separate windows. Just double click on the appropriate item, and it will be added where the cursor is in the editor.



Void

Alternative way is to click on:

- “Ctrl Shift and C” to get the choice color and write it,
- “Ctrl Shift and N” for function color
- “Ctrl Shift and V” for variable color
- “Ctrl Shift and A” for asset color

9. Creating a summary **Stage** for subjects to see after the experient

- a. Drag and drop another **Stage** into the **Design** window



- b. Rename it and Click on it to select it:

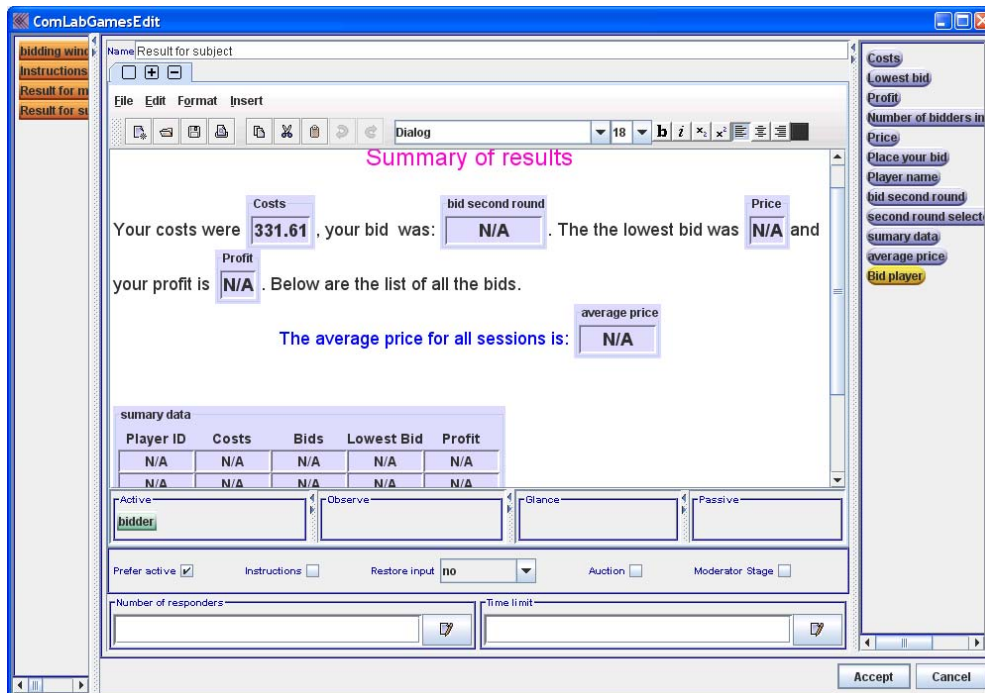


- c. Then click on  and changes the color to red:



which indicates that this is a summary stage for subjects.

- d. Open the stage and write the summary that you want to show the subjects. Drag and drop the appropriate variable:
- e. Make sure you have drag &dropped **bidder** to be active.



f. Note you do not need any transition to follow the summary stage.

10. Creating a summary Stage for moderator to see during and after the experient

a. You can duplicate subject's summary stage by selecting it



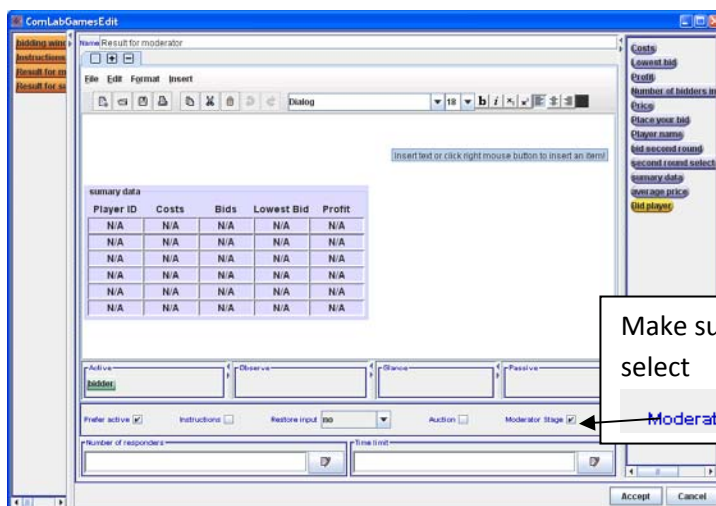
on



appears. Rename it to



b. Right click on to open the stage, and decide which variables a moderator would like to see:




Make sure to select

Moderator Stage ☒



- c. Make sure to select  at the bottom the stage editor.

11. Selecting a starting

- a. Select a  by clicking on it from which you want to experiment to start. In the sealed bid auction :






is the stage from which the experiment will start.


- b. Then click on  .  changes the color to green to indicate that this is a starting stage of the experiment:



. By default the first  that is dropped into design window is green.

- c.  that is selected as summary stage (red), instructions () or moderator stage () cannot be the starting stages. Note that instruction stage, summary stage or moderator stage do not need the a stage transition to point to them.

12. Specific updates in for the First Price Sealed Bid Auction

- a. Variable  is multiple and it will save the value of the choice that each subject made in the choice **Bid player** . The expression in the transition is:

```
bid second round::Bid player
Lowest bid::sort(Bid player)(1)
second round selected bid::select_k(-bid second round,1)
Profit::if( Lowest bid=decompose(-second round selected bid), Lowest bid-Costs,0)
Price::Lowest bid
summary data::[Player name,Costs,bid second round,-second round selected bid,Profit]
```

- b. To find the lowest bid, sort all subjects bids in descending order and select the highest bid: the function is sort():
 Lowest bid::sort(Bid player)(1)

- c. To select all the bids with the lowest bid use the function `select_k(variable, n)`. Function `select_k(-bid second round,1)` will assign a subject who submitted the highest bid (minus sign to assign it to the lowest value), the bid value (i.e. if there is a tie it will randomly select one of the subject). All other subjects will be assigned 0. The variable **second round selected bid** has to be a column vector with the subject number dimension, and it is NOT a multiple variable. All the subjects' bids are put in one of the rows.

Type	real	
		Second round selected bid
		<input type="text"/>
		<input type="text"/>
nit value(s)		<input type="text"/>
		<input type="text"/>
		<input type="text"/>
		<input type="text"/>
Rows	number_of_subjects()	<input type="text"/>
Columns	1	<input type="text"/>

Expression is: `second round selected bid::select_k(-bid second round,1)`

- d. **Profit** is a multiple variable, and it is a scalar. Each subject will be assigned a profit in the following way:

`Profit::if(Lowest bid=decompose(-second round selected bid), Lowest bid-Costs,0)`

With the `if(,,)` statement the lowest bid is compared to the lowest bid in the variable **second round selected bid**, and if this value is equal to the lowest bid, the profit is the difference between the lowest bid and the cost for that subject. Otherwise it is zero.

Note that we used the function: `decompose()` for the vector variable

second round selected bid to assign a value from the vector to each individual (i.e. from the vector to the scalar).

- e. **summary data** variable has all sessions selected ☒ to show results for all sessions to the moderator and subjects at the end of the game, and it is NOT a multiple variable. This variable is in **Result for subject** stage and in **Result for moderator** stage. Type is: any because it contains real, inter, and text variables. The dimensions are:

- Number of rows: number of subjects in a session defined with the function `number_of_subjects()`

- Number of columns: 5. There are labeled Player ID, Costs, Bids, Lowest Bid, Profit.

Type: **any**

	Player ID	Costs	Bids	Lowest Bid	Profit
Init value(s)	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Rows	number_of_subjects()				
Columns	5				

The expression that allows us to combine all these variables into variable is:

summary data

summary data:[Player name,Costs,bid second round,-second round selected bid,Profit]

Note that [Player name,Costs,bid second round,-second round selected bid,Profit] are all multiple variables so each of them represent a column with a dimension of the number of subjects in a session. Comma puts each variable in a separate column.

13. Specific **Choices**, **Variables** and updates in **Transition** for the English Auction

- a. **Choices** The characteristic of the choice in the English auction: The value (i.e. bid) is lowered automatically in designated time intervals. Subjects only click on the choice to accept the bid. If nobody clicks on the choice in allocated time interval the game is over.

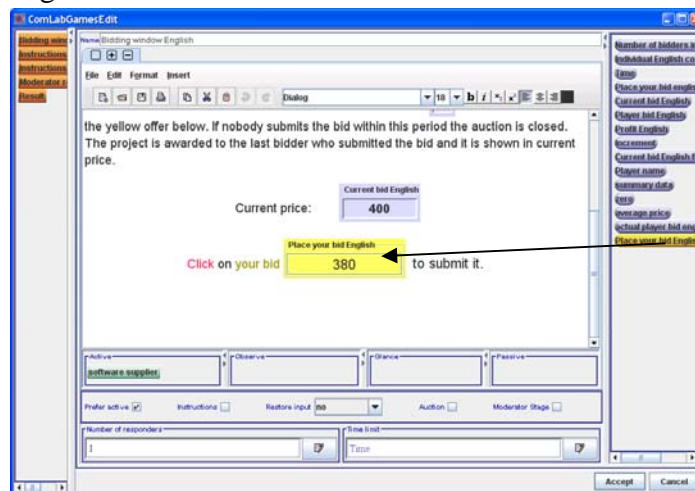
Name	Location	Show default value(s)	Row selection	Column selection	Drop-down selection	Immediate response
Place your bid English		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

- In order to only click on the choice; the default value has to be selected, row selection and column selection has to be selected as well as immediate response.
- The lower bound is set to 0. The upper bound is set the variable current bid English minus Increment.

- Initial value is set to **Current Bid English** – **Increment**. **Current bid English** has initial value at 400. **Increment** has initial value 20. Every 5 seconds (see variable **Time** and initial value set to 5) **Place your bid English** is reduced by 20. See the pictures below.

Type	real	
Min:Max	0	Current bid English-Incr
Default value(s)		Current bid English-Incr
Rows	1	
Columns	1	

Right click on **Bidding window English** to see the choice.




b. **Transition** - **Stage transition**

- We have to model the following: If nobody clicks on the choice in allocated time interval the game is over. If a subject clicks on the choice then the program should return to the **Bidding window English**. To do this:


- you should have a **Transition** connected to the **Bidding window English**

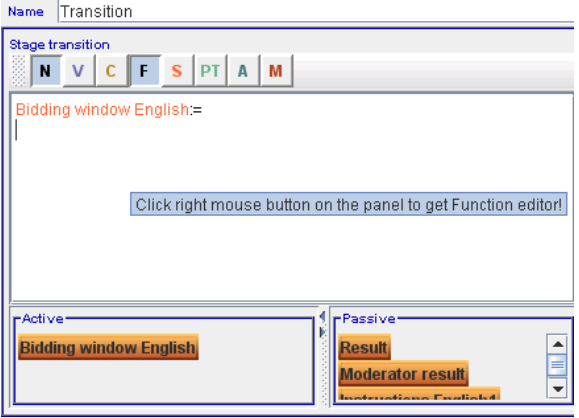


- right click on **Transition** to open it.

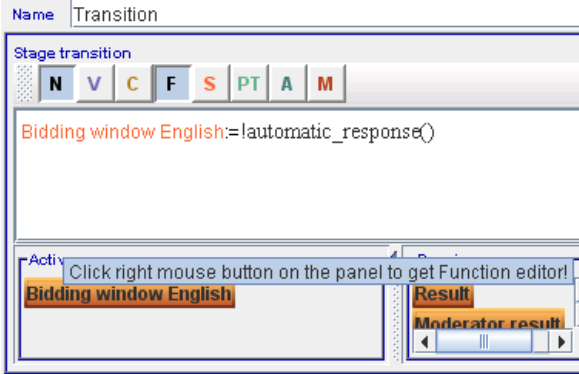


Drag **Bidding window English** from Passive to Active as in the picture below. If nothing is added to this statement, it means that with probability 1 the program will return to the **Bidding window English** after each 5 seconds.





To write a condition that the program returns to **Bidding window English** only if a subject clicks on the **Place your bid English** then we have to add the following: `!automatic_response()` which means no automatic response.



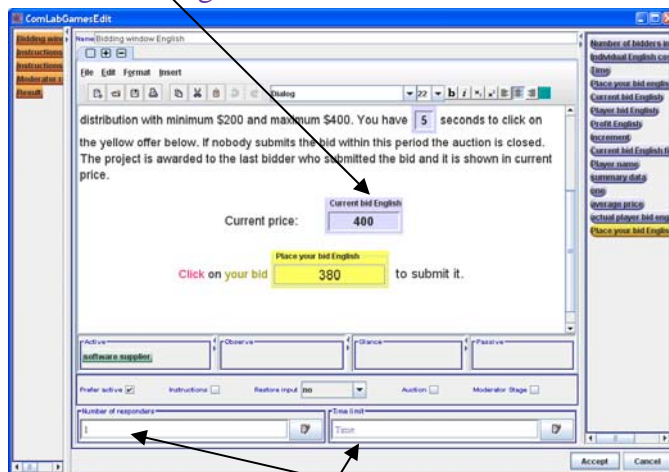
- a. **Transition** - Variable update
- The variables are updated in the following way:

```

Variable update
N V C F S PT A M
Player bid English:=Place your bid English
Current bid English final:=if(automatic_response(),Current bid English,Place your bid English)
actual player bid english:=if(automatic_response(),actual player bid english,Place your bid English)
Current bid English:=if(automatic_response(),Current bid English,Place your bid English)
Profit English:=if(Player bid English =Current bid English final ,Current bid English final -Individual English costs,0)
summary data:=[Player name,Individual English costs,Player bid English,one*Current bid English final,Profit English]

```

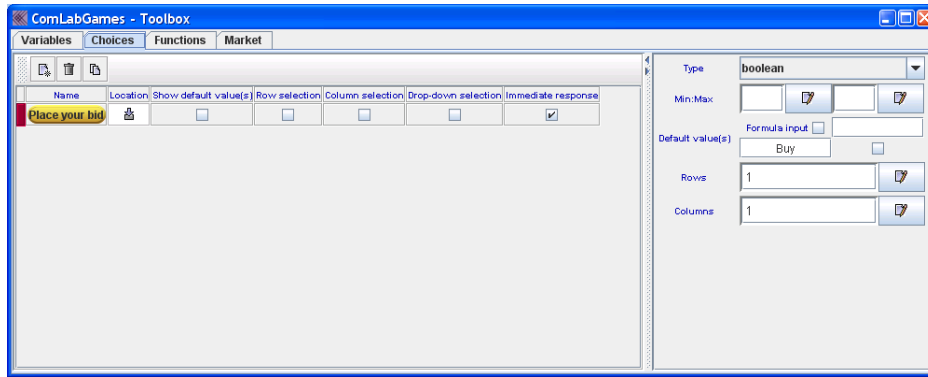
- **Player bid English:= Place your bid English**
Note that we have := which means that the actual bid is the last bid accepted before the **Increment** is subtracted from the choice **Place your bid English**. If we wrote :: instead of := then the value of **Player bid English** would include the value that is incorrect (i.e. smaller by **Increment**)
- **Current bid English** received the current value of **Current bid English** if there the time expires (automatic_response()) and is updated with the **Place your bid English** if a subject clicks on the choice.
- **Current bid English** is shown in the **Bidding window English** as follows:



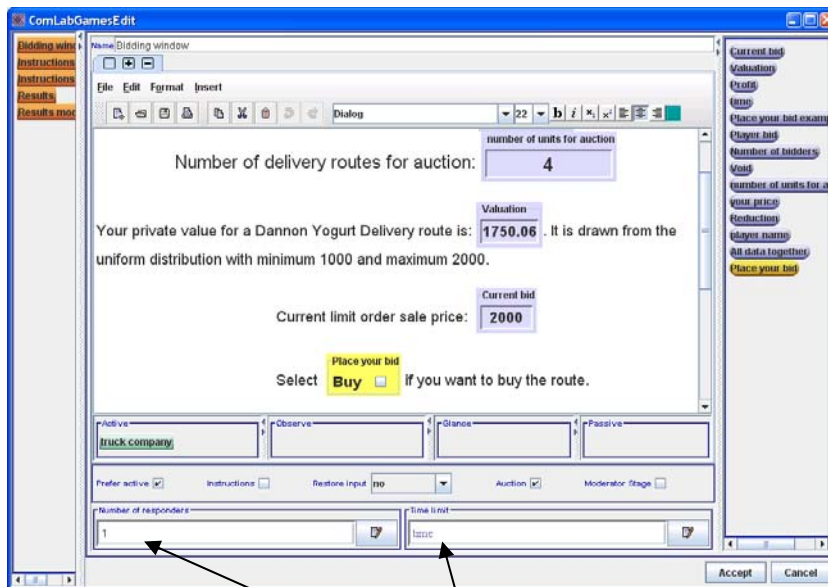
- Number of respondents is set to 1
- Time limit is set to **Time**

14. Specific **Choices**, **Variables** and updates in **Transition** for the Dutch Multiunit Auction

- Choices** Subjects have to click on Buy when they want to buy the object. In order to do this the choice **Place your bid** should have the following characteristics:



Type has to be “Boolean” and the immediate response has to be selected.



Time limit is set to **Time** in the

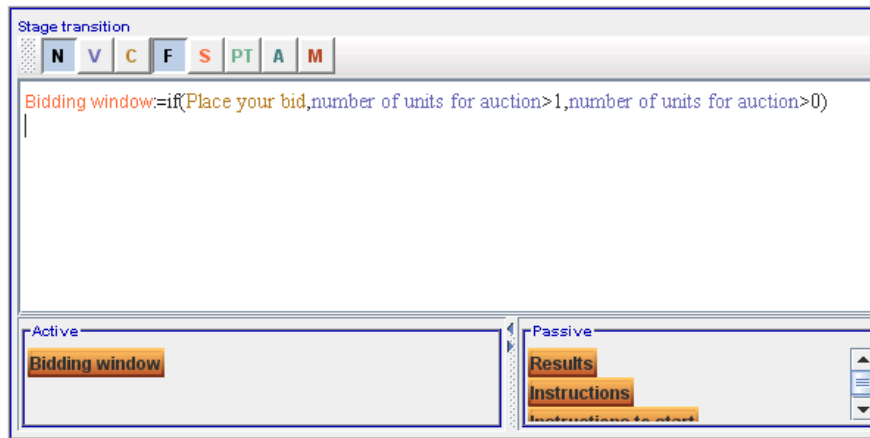
Bidding window

Number of respondents is set to 1 in the

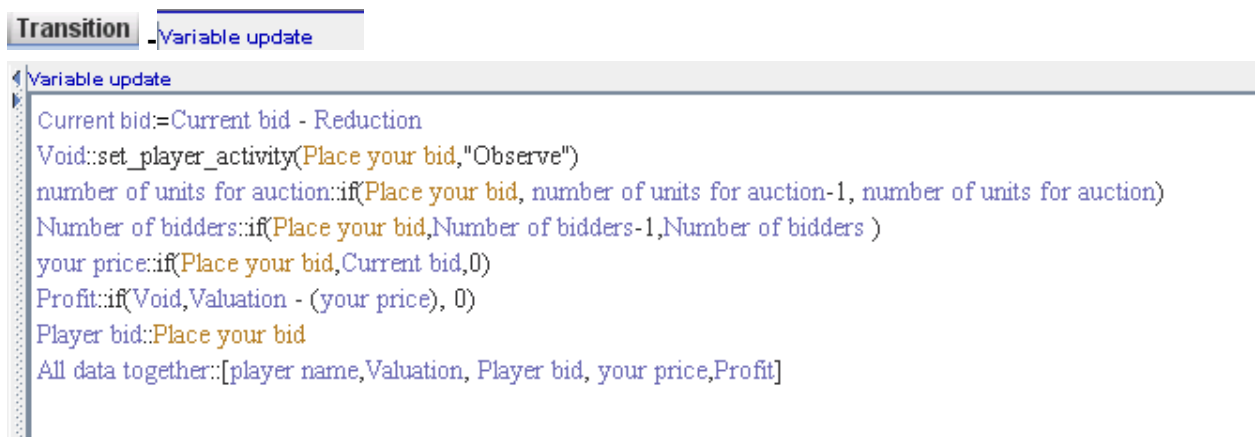
Bidding window

c. **Transition** - **Stage transition**

The program should return to **Bidding window** until all the units are sold out (i.e. **number of units for auction** > 0). The condition below fulfills this requirement.



d.



- **Void** Variable is a multiple variable

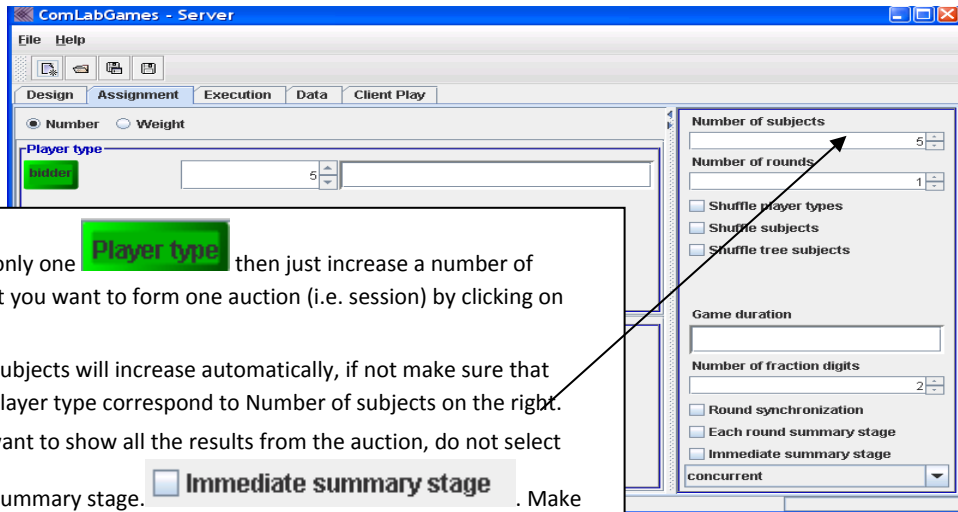


And it is Boolean.

Void:: set_player_acctivity(Place your bid, “Observe”)

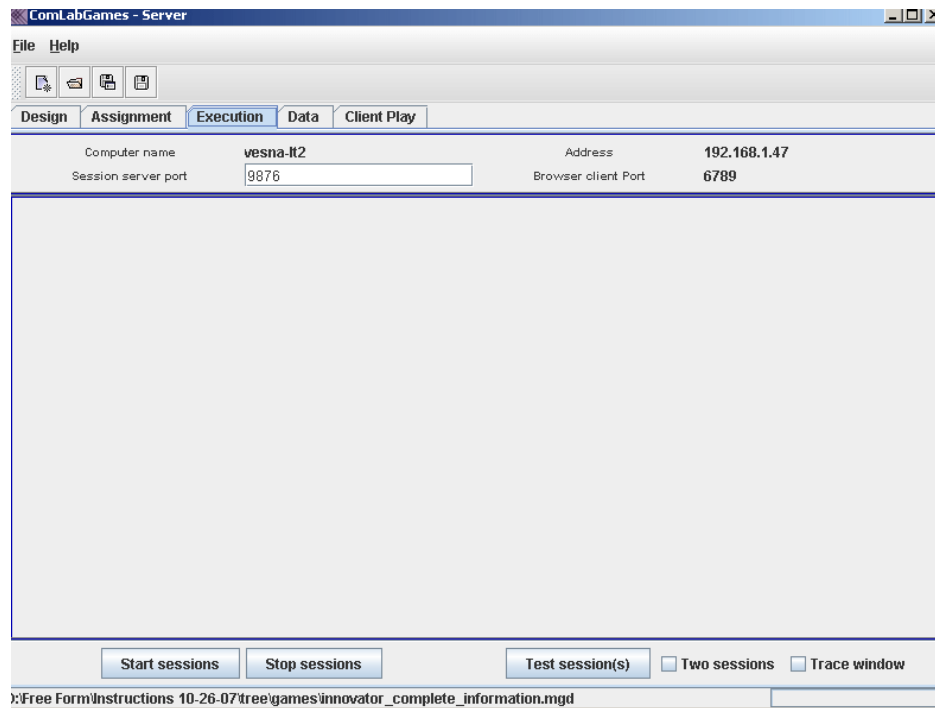
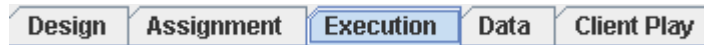
In the **Variable update** above means that when subjects click on buy (Place your bid), the variable **Void** is true and subjects become observers and cannot buy any more units).

15. Assigning subjects a role in **Assignment** window



- If you have only one **Player type** then just increase a number of subjects that you want to form one auction (i.e. session) by clicking on arrow.
- Number of subjects will increase automatically, if not make sure that number of player type correspond to Number of subjects on the right.
- When you want to show all the results from the auction, do not select Immediate summary stage. ☐ **Immediate summary stage**. Make sure to stop the game (**Stop sessions**) on **Execution** when you use this option in order to see the results for all sessions.
- In your experiment, you might want to select ☒ it so only subjects for their session see the data.

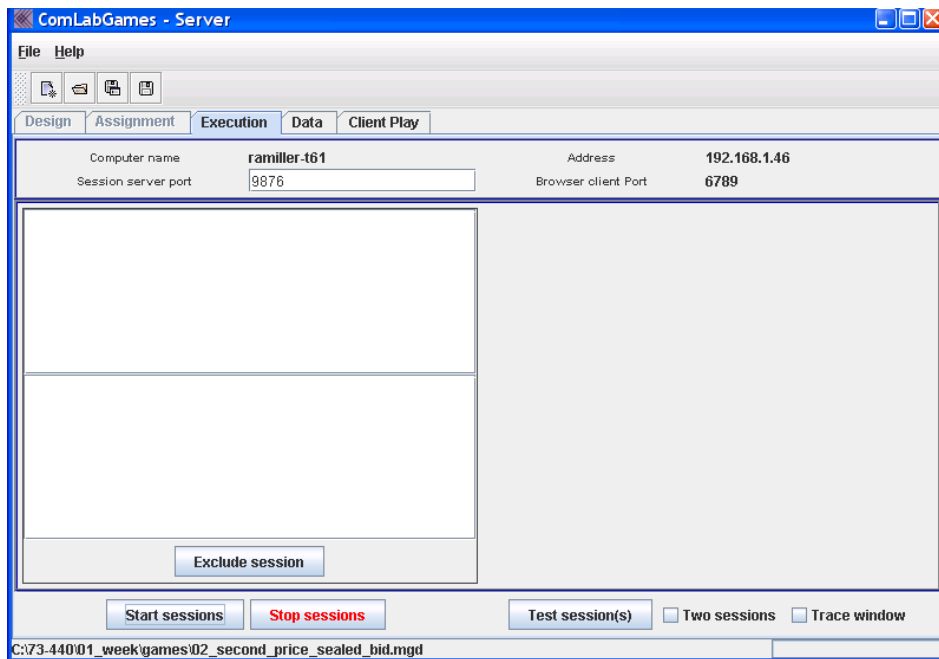
a. Select **Execution** on



- b. Make sure the game is loaded on the design window (the name of the game is shown at the bottom of the window or just click on **Design** to see if the game is loaded)
- c. To test the game before conducting an experiment, click on **Test session(s)** that can be found at the bottom right end of the window. This option allows the moderator to see how the client window looks like and to go thorough the experiment on the same screen as the program. Selecting ☒ **Two sessions** tests two sessions running at the same time. ☒ **Trace window** shows all the calculations during the test. Trace uses a lot of CPU time so it might be slow, use only one test session and limit the number of subjects in the **Assignment** window.

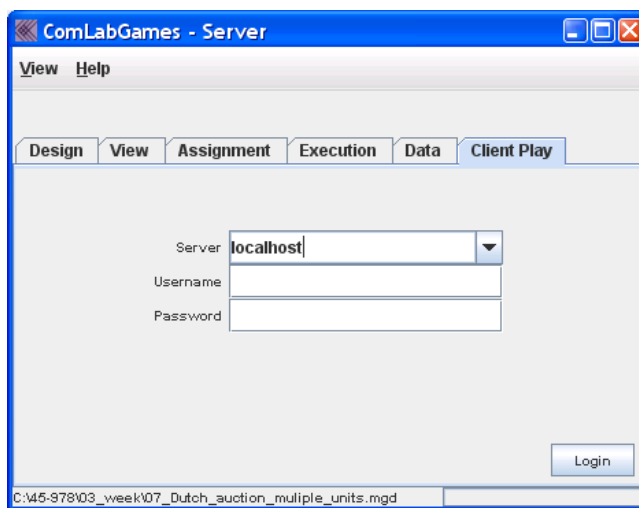
- d. To run an experiment, click on **Start sessions** and give the address of the server to the subjects. The address consists of two parts: the address and the port number. The address is located in the top right corner of the window. In our example the address is: 192.168.1.47. There are two port numbers: session server port: 9876 and Browser client port: 6789.

Computer name	vesna-ll2	Address	192.168.1.47
Session server port	9876	Browser client Port	6789



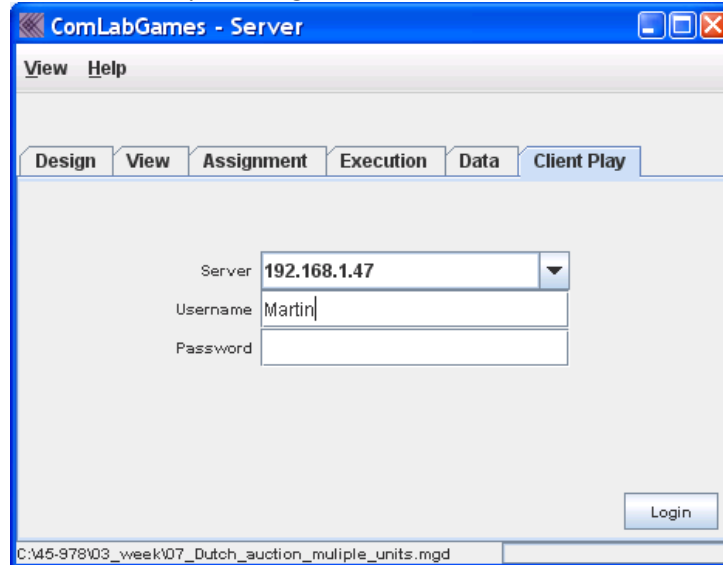
- Select **Client Play** on .

The following window will appear:



- Provide Url address, colon and session server port for users of **Client Play** : 192.168.1.47

- A subject writes a Url address under server, login name that can be any name. Password is not necessary. Clicking on **Login** connects the client to the game.

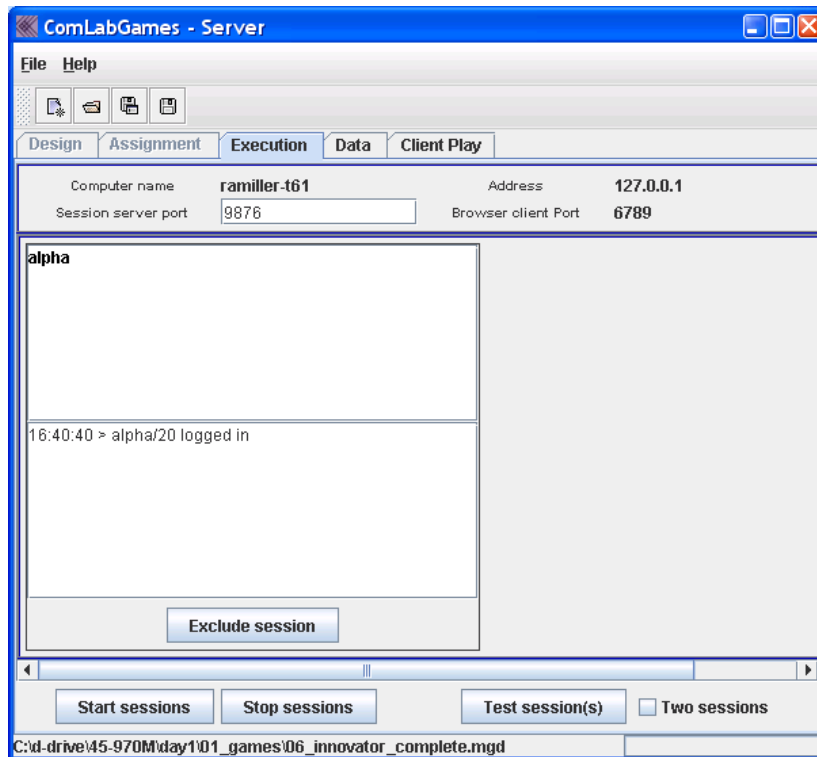


17. Moderator Viewing the Data

- On the execution window the moderator can follow the status of each subject:

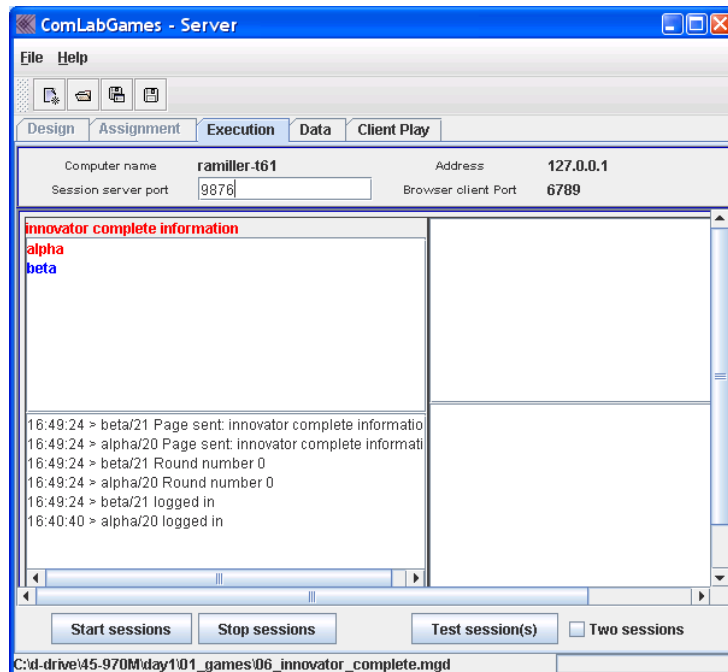
(1) A Subject connected to the game and not all the subjects for the session connected to the game (Login names are in black).

Subject "alpha" connected to the game.

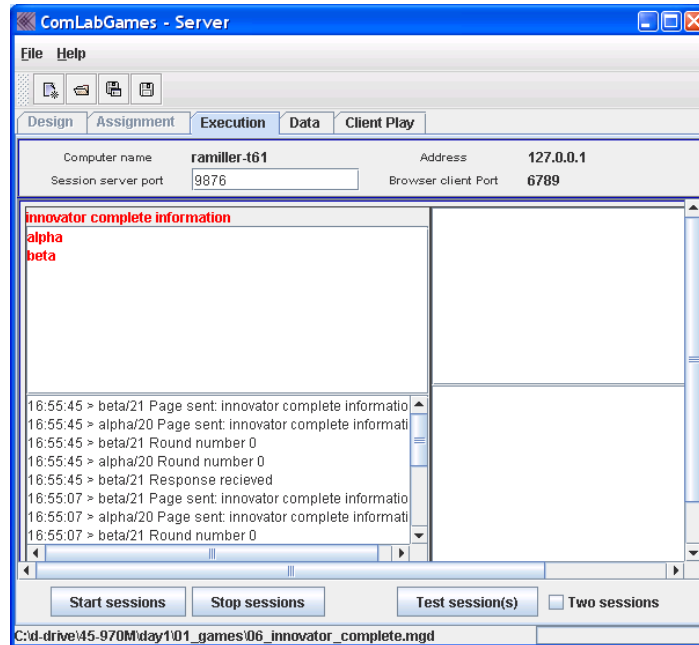


(2) All subjects are connected to the game but nobody made a decision yet

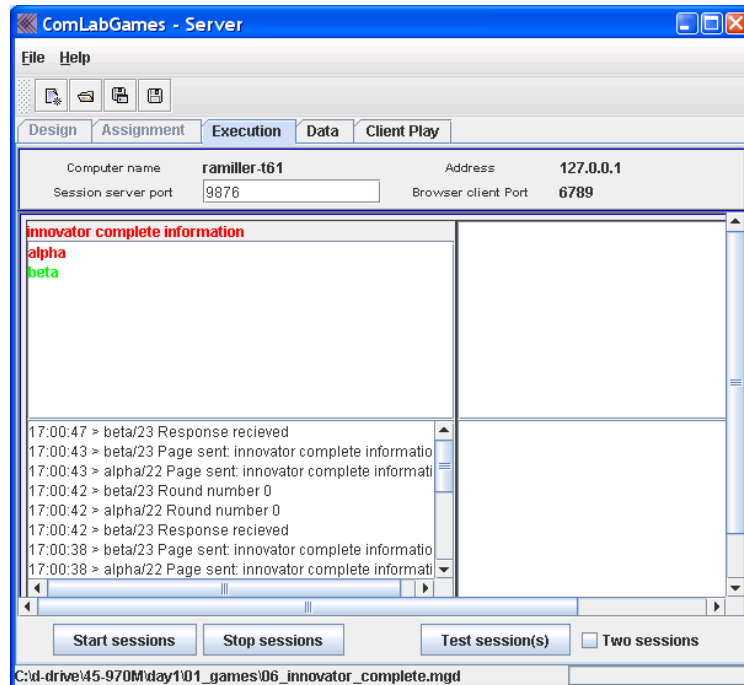
Subject in red is the one who has to make a decision and a subject in blue has to wait until the first subject selected a choice.



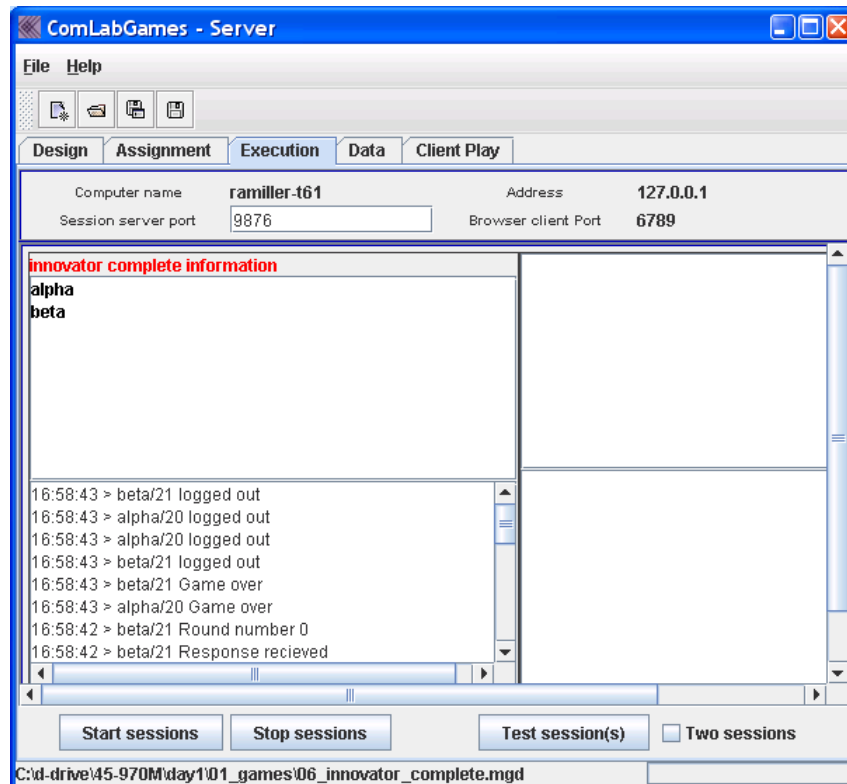
(3) All subjects have to make a decision (all login names for the session are in red)



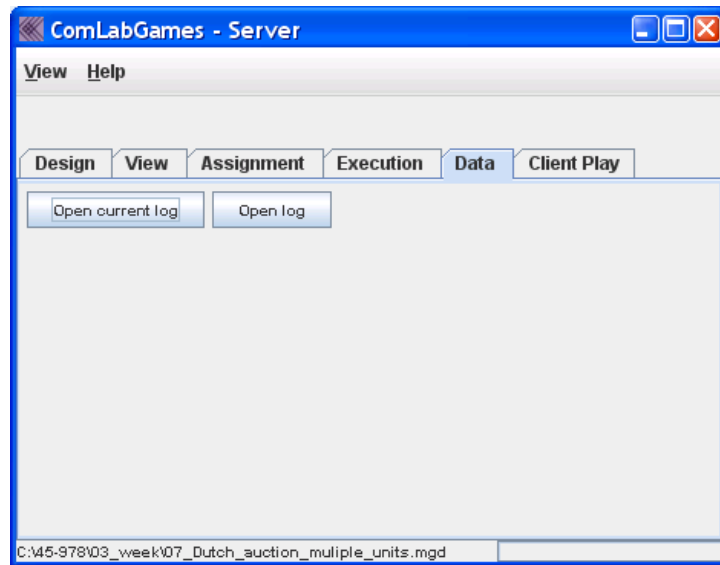
(4) Subjects who finished the tasks are in green. Subjects who did not finish the task are in red.




(5) The game is over. All subjects finished all the tasks.



- If any of the sessions have technical problems, moderator can disconnect them by clicking on **Exclude session** below the session that has a problem.
- a. **Viewing the Data after the experiment (i.e. after clicking on **Stop sessions**)**
 - Moderator can show the data immediately after the experiment
 - Click on **Data** and the following information appears in the window:



- Click on  and select the appropriate data file. Data file have start with "log-date-time-name-of-the-experiment".