Conducting Experiments with z-Tree

Mark Pigors

DUFE, December 2013

Behavioral Economics
University of Cologne

Part I

Goals of this course

- learn how to organize and conduct economic experiments in the lab.
- learn to program your own experiment

What is z-Tree?

- "Zurich Toolbox for Ready-made Economic Experiments", developed at the university of Zurich by Urs Fischbacher.
- designed to enable the conduction of economic experiments without much prior experience
- Two parts:
  - z-Tree: to define and conduct experiments (server program)
  - z-Leaf: program used by the subjects (client program)
- You get a z-Tree license under
  $\rightarrow$ http://www.iew.uzh.ch/ztree/howtoget.php

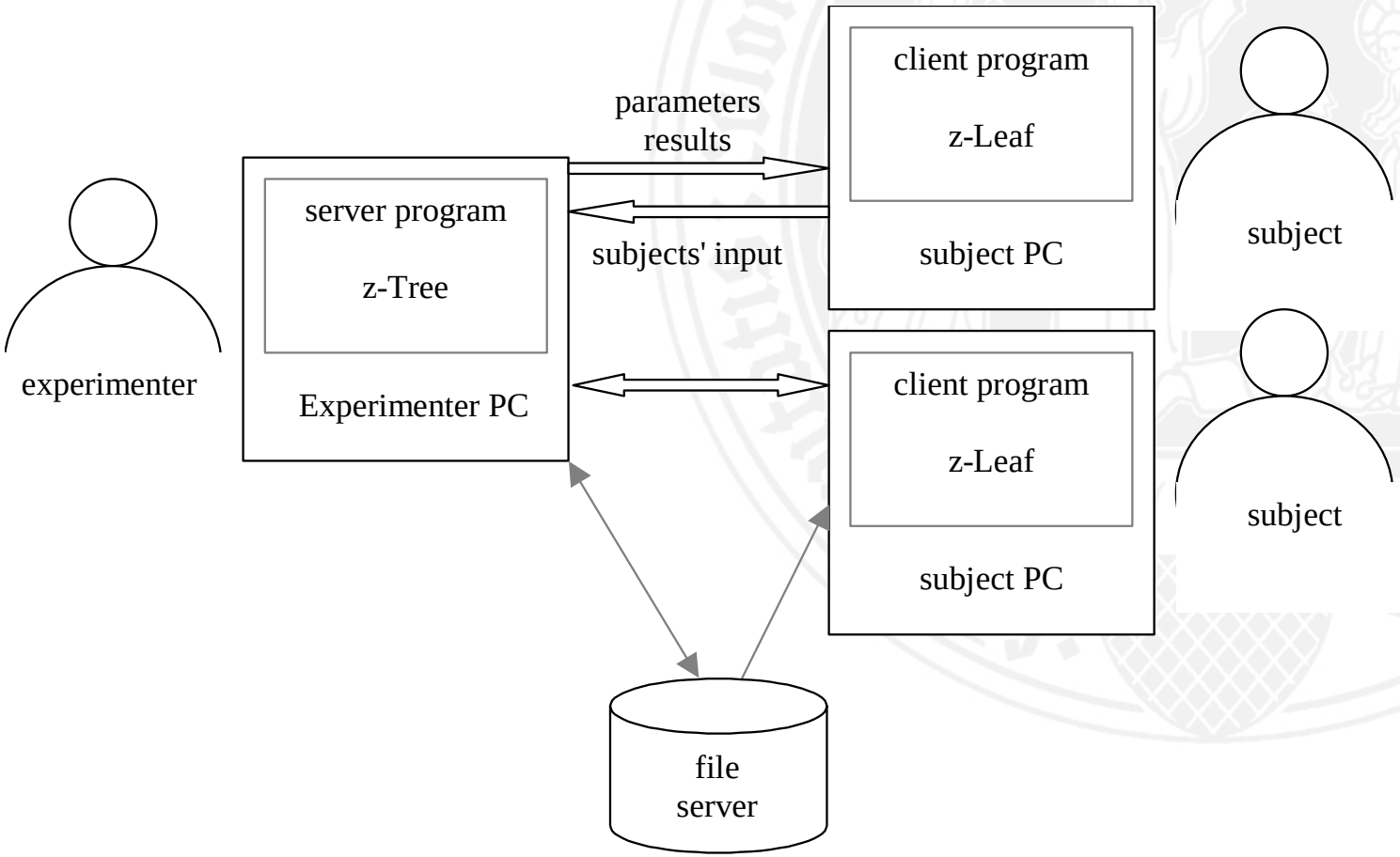
References

- Urs Fischbacher (2007): *z-Tree: Zurich Toolbox for Ready-made Economic Experiments*, Experimental Economics 10(2), 171-178.
- Urs Fischbacher (2002):
  - Experimenter's Manual (in German and English)
  - Reference Manual
- Thanks to:
  - Maria Bigoni: http://www2.dse.unibo.it/bigoni/courses/ztree/
  - Ernesto Reuben: http://www.ereuben.net/teach/
- Wiki and FAQ available online:
  - https://www.uzh.ch/iew/ztree/ssl-dir/wiki/
  - http://www.iew.uzh.ch/ztree/faq/

What does z-Tree do?

- (Instructions)
- Control questions:
  - record how many times a subject gives the wrong answer
  - measure how long it takes for a subjects to complete the control questions
- Treatment(s)
- Questionnaires
- Payment:
  - manage losses
  - add show-up fee
  - round earnings
  - prepare simple receipts

Client/Server architecture of z-Tree



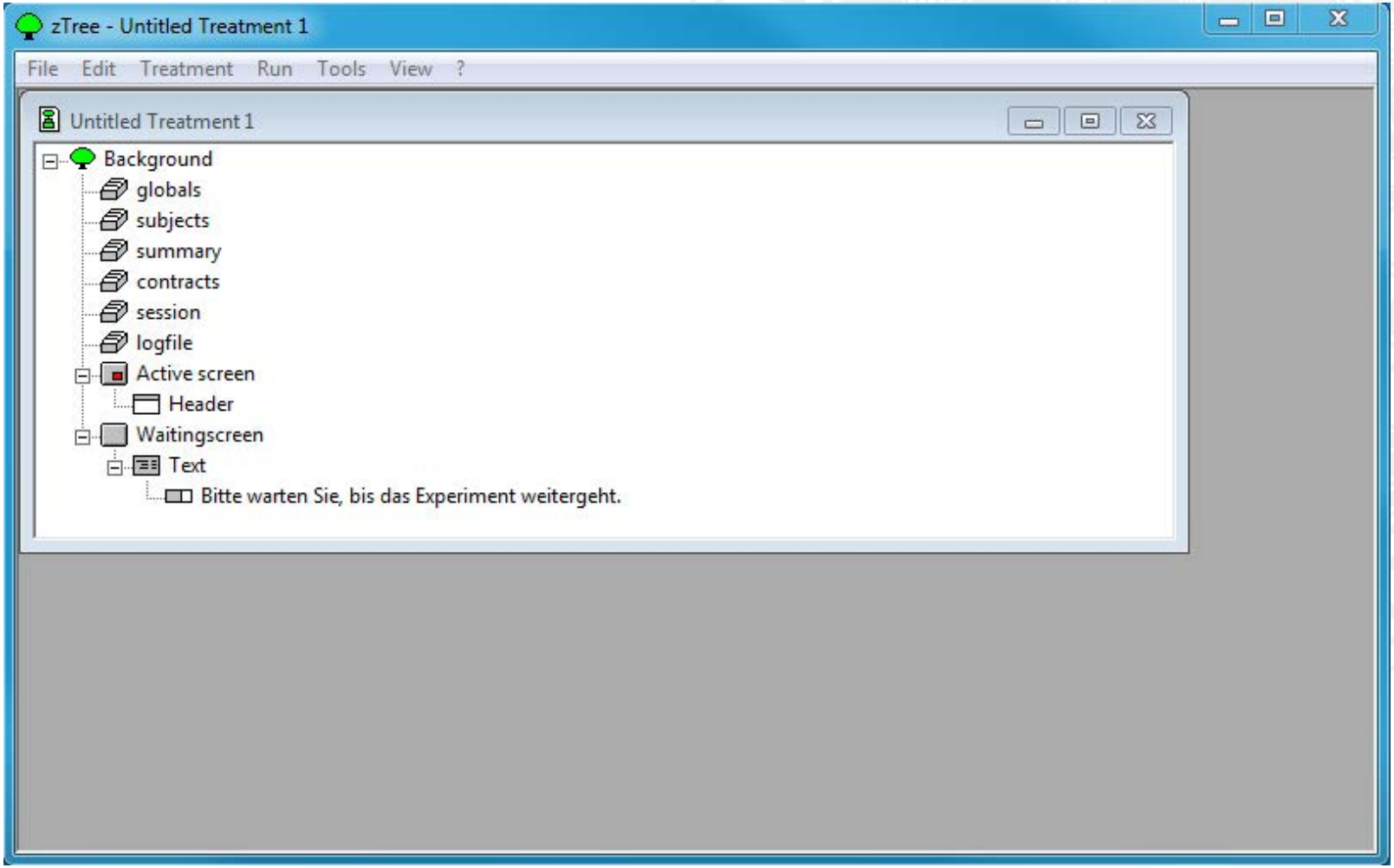
Mark Pigors

University of Cologne

Installing z-Tree

- With a file server
  1. Put the files z-Tree.exe and z-Leaf.exe in a single directory on the file server.
  2. Make sure you have read/write permissions on this directory and on the parent directory.
  3. Make sure clients have read permissions on the directory.
- Setting up a Test Environment
  1. No need for several computers to test programs.
  2. Start z-Tree and more than one z-Leaf on one computer.
  3. However, you have to give the z-Leaves different names (see page 21).

z-Tree screen



Mark Pigors

University of Cologne

z-Leaf screen

Example: A Lottery

- endowment: E
- the lottery yields: $(1 + a) \cdot x$ with probability p and 0 with probability $1 - p$
- task: choose x between 0 and E
- random draw: r between 0 and 1
- payoff: $E - x + I[r \leq p] \cdot (1 + a) \cdot x$

Mark Pigors

University of Cologne

Global variables

- are in the globals table, which contains one record
- freshly set up for each period (The globals table of the preceding period: `OLDglobals`.)
- are the same for every subject
- the following variables are default:
  - `Period`: Number of the period.
  - `NumPeriods`: ID of the last period.
  - `RepeatTreatment`: If this variable is greater than zero after the last period, the treatment is run again.
- here also: E, p, a

Subject variables

- are in the subjects table, one record for every subject
- The following variables are default:
  - `Period`: Number of the period.
  - `Subject`: Number of subject; starts at 1.
  - `Group`: Group number.
  - `Profit`: Profit made in this period (in experimental currency units).
  - `TotalProfit`: Total profit made in this treatment (should not be changed).
  - `Participate`: Indicates whether the subject is taking part in the current stage or not (1 or 0).
  - `LeaveStage`: If set to one, causes a subject to leave an active state and to move to waiting state.
- here also: x, r

Programming the example

→ `lottery.ztt`

- Initialize the variables
- Select “logfile” with the mouse, then from the menu choose “Treatment” → “New Program”
- Write a program in the globals table.

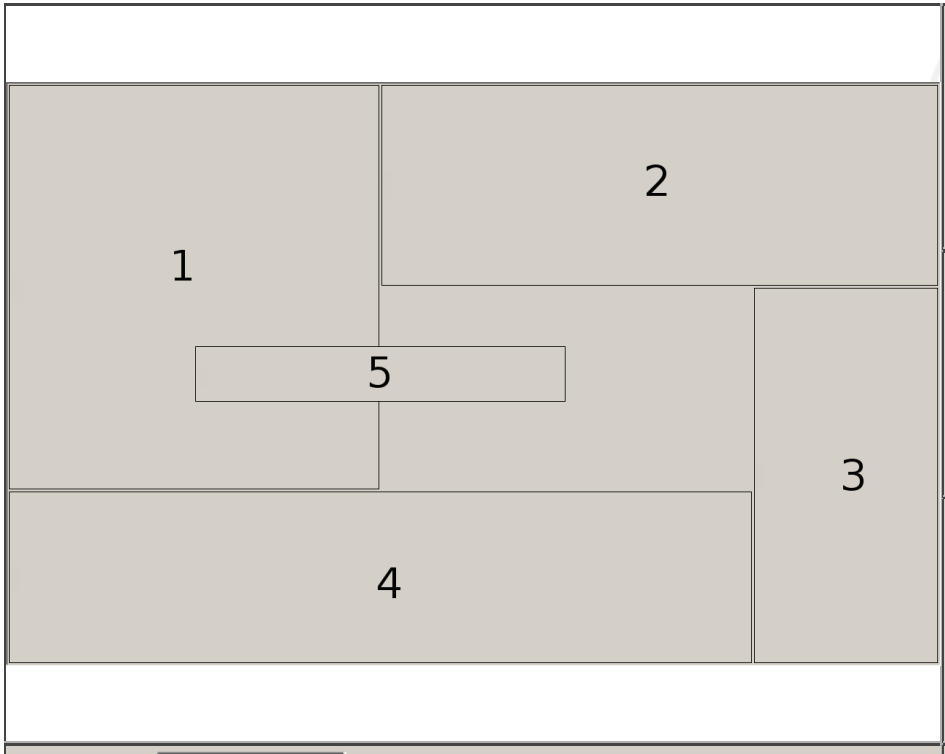
```
//GLOBAL VARIABLES
E=100; //endowment
p=0.5; //probability of winning the lottery
a=2; //factor which determines the prize of the lottery
```

- Write a program in the subjects table.

```
//SUBJECTS VARIABLES
x=-1; //money invested in the lottery
n=random(); //random number
```


Structure of the treatment

- Create a stage. Select “Background” then click: “Treatment” → “New Stage”
  - Choose a meaningful name for the stage (here: “Lottery”)
  - Set the timing (wait for all, start if possible, ...)
  - Set the timeout: choose “No” if you wish to record the time without forcing subjects to leave the stage
- Create a box. Select “Active Screen” in the “Lottery” stage. From the menu, select “Treatment” → “New Box” → “Standard Box”
  - Everything is displayed in boxes.
  - Many different types of boxes.
  - Different ways to align them → `screen_layout_example.ztt`



Name 2 with Frame
Width [p/%]
Height [p/%]
Distance to the margin [p/%]: 40% 65%
Adjustment of the remaining box: left top right bottom

Name 3 with Frame
Width [p/%]
Height [p/%]
Distance to the margin [p/%]: 35% 80%
Adjustment of the remaining box: left top right bottom

Name 4 with Frame
Width [p/%]
Height [p/%]
Distance to the margin [p/%]: 70% 20%
Adjustment of the remaining box: left top right bottom

Name 1 with Frame
Width [p/%]
Height [p/%]
Distance to the margin [p/%]: 60% 30%
Adjustment of the remaining box: left top right bottom

Name 5 with Frame
Width [p/%]
Height [p/%]
Distance to the margin [p/%]: 45% 20% 40% 45%
Adjustment of the remaining box: left top right bottom

Mark Pigors

University of Cologne

Structure of the Treatment

- Create Items. Select your box in the “Lottery” stage, then from the menu: “Treatment” → “New Item”
  1. Output:
     - Label: ‘‘`Your endowment`’’
     - Variable: `E`
     - Format: `1` (0.1 for 1 decimal digit...)
     - You can use “< >” to insert variable values into labels.
     - Label “`< > Your Endowment:  < E|1> coins.`”
  2. Input
     - Create a new Item
     - Label: ‘‘`Choose how much to invest in the lottery`’’
     - Variable: `x`
     - Format: `1` (0.1 for 1 decimal digit...)
     - select “input”
     - Minimum: 0
     - Maximum: `E`

Item Layouts

Layout	input variable	output variable
2		
!radio: 1 = "86.8"; 24 = "102.8";		
!radioline: 0="zero";5="five"; 6;		
!slider: 0 ="A"; 100= "B"; 101;		
!scrollbar: 0="L";100= "R";101;		
!checkbox:1="check me";		
!text: 1= "one"; 2 = "two"; 3 = "three"; 4 = "four"; 5 = "five"; 6 = "six"; 7 = "seven"; 8 = "eight"; 9 = "nine"; 10 = "ten";		
!button: 1 = "accept"; 0 = "reject";		

Mark Pigors

University of Cologne

Program continued

- Create a button. Click “Treatment” → “New Button”
  - You always need a button to leave a screen/stage
- Profit calculation: Write a program in the subjects table.

```
//PROFIT CALCULATION
Profit=E-x+if(r<=p,x*(1+a), 0);
```

- Show the profit in an item
- Result Screen
  - In the Active Screen, create a new Standard Box, and name it “Results”.
  - In this box, add an Item to display the random number r, with 2 decimal digits.
  - Add a second Item to display the subject’s profit.
  - Add a Button to end the treatment.

Run the Treatment

- In z-Tree, double-click on “Background” (the very first line of the program) to set the treatment’s parameters.
  - Exchange rate = $\frac{1}{n}$ if n points equal 1 Euro.
- Launch z-Leaf.exe and check the connection from z-Tree, in the Clients Table. Click “Treatments” $\rightarrow$ ”clients table”.
- In z-Tree, press F5 to start the treatment.
- Click “Treatments” $\rightarrow$ ”subjects table” to monitor the subjects activity.

How to launch z-leaf.exe

- Create a shortcut for z-Leaf.exe (see the manual).
- Use one of the batch-files provided:
  - `zmultiple_leafs.bat`: You can choose number of subjects, font size and language.
  - `zLeaf_xPlayers_cascade1024`: Starts a cascade of x zLeafs.
  - Write your own batch-files.
- Use `OpenZleafs.exe`: You can choose number of subjects, font size and screen size.

A Questionnaire

- Is needed for the payment file to be written.
- Click “File” $\rightarrow$ “New Questionnaire”
- Click “Questionnaire” $\rightarrow$ “Language” and select the language of your choice
- Click “Questionnaire” $\rightarrow$ “New Address Form”
- Click “Questionnaire” $\rightarrow$ “New Question Form” and name it “Empty”
- Address Form is needed for the questionnaire to run.
- Leave the fields “First Name” and “Last Name” empty and the Address form will not be displayed to subjects.
- When all the treatments of a session are over, select the questionnaire file from z-Tree and press F5 to run it.

Files saved by z-Tree

- Session data are saved in the directory containing z-Tree.exe
- `.pay`: the payment file, which lists the subjects' final profits including the show-up fee
- `.adr`: subjects' addresses (from the Questionnaire)
- `.sbj`: answers to questionnaire's questions, without subjects' names
- `.gsf`: backup file, in case a crash occurs
- `.xls`: contains all tables used in a session (subjects table, globals table, etc)

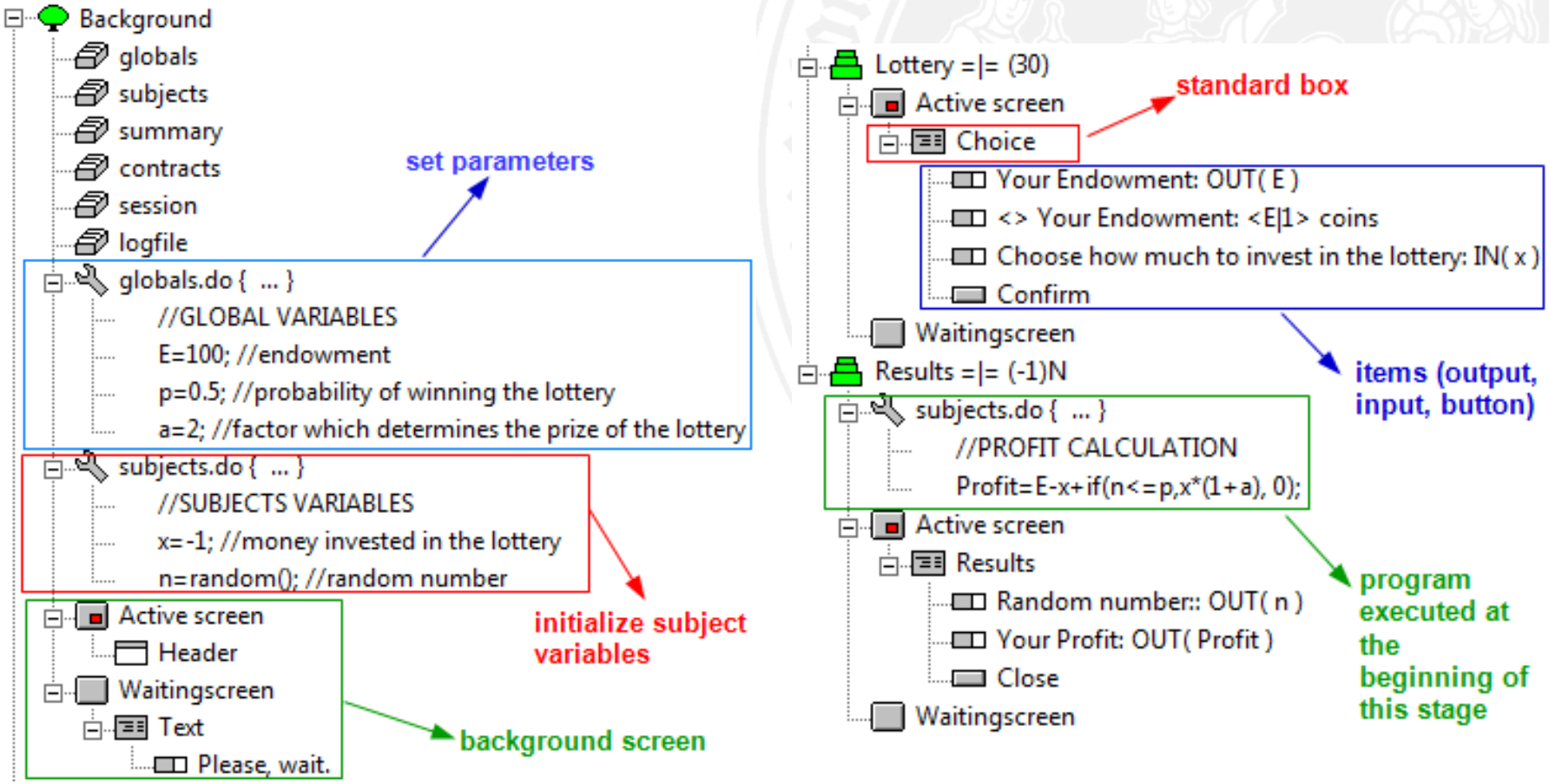
How to use the data

1. From z-Tree click “Tools” → “Separate Tables”, Select the `.xls` file generated by z-Tree and click “Open”. This generates one `.xls` file per each of the tables used in the session.
2. For those who use Stata, Kan Takeuchi developed an `.ado` file to import z-Tree data directly into Stata.
   → http://www.iew.uzh.ch/ztree/ztreelinks.php
3. For those who use R, Oliver Kirchkamp developed a utility to import data from z-Tree into R.
   → http://www.kirchkamp.de/lab/zTree.html#zTreeR

Again: Structure of a Treatment

1. background:
   - General parameters (number of subjects, number of periods...)
   - Tables used by the program (subjects table, globals table, ...)
   - Programs to be executed before the treatment starts
2. stages:
   - Each stage (roughly) corresponds to a screen.
   - Stages can contain several boxes.
   - Stages can also contain programs, to be executed at the beginning of the stage.

Again: Structure of a Treatment



Mark Pigors

University of Cologne

Again: Structure of a Session

1. Treatments:
   - Each .ztt program is a treatment
   - You can run many treatments in a session, but all the treatments must have the same number of subjects
2. Questionnaires:
   - At the end of the session, you ought to run a questionnaire (possibly empty) to generate the payment file
3. A session ends when z-Tree is closed.

Exercise: Program the following Treatment

- Subjects have to calculate the sine function for a randomly determined value: `sin(C*A/B)`.
- Subjects are paid according to the precision of their guess: Profit (in points): `100*(1-|sin(C*A/B)-guess|)`.
- A and B are random integer numbers between 1 and 10. They take different values for different subjects. C is equal to π

Exercise: Details

1. Subjects cannot make negative profits (set the profit function accordingly).
2. Do not include the header, nor the alert frame.
3. Layout of the input variable: slider.
4. Functions you need to use:
   - `abs()`: absolute value
   - `max(x,y)`: returns the maximum value between x and y
   - `pi()`: returns 3.1415...
   - `roundup(x,y)`: returns the smallest multiple of y that is greater or equal to x.
   - `sin(x)`: returns the sine of x.

Part II

5. **Interactive experiments**
   - Matching
   - Table Functions
   - The scope operator
   - IF as a Statement and as a Function
6. **Exercise: The Traveller's dilemma**
7. **Repeated Symmetric Games**
   - Matching
   - Different games in different periods
   - Some Types of Boxes
   - Indefinite number of Periods
   - do statements and loops
8. **Asymmetric games**
   - Assigning Types
   - The Participate Variable
   - Program execution
9. **Exercise: Battle of the Sexes**

Subjects interact

- Subject interact – subject's payoff also depends on the decisions of other subjects.
- Groups have to be created to define which subject interact

Example: Prisoner's Dilemma

- 2-player normal form game

		Player 2	
		cooperate	defect
Player 1	cooperate	3,3	0,5
	defect	5,0	1,1

- Hint: use parameters to make your program more flexible.

Steps

→ `prisoner_dilemma.ztt`

1. Create groups (pairs)
2. Set parameters: r, t, s, p
3. Initialize variables:
   - partner
   - choice
   - partnerchoice
4. Subjects' choice
5. Profit calculation
6. Results

Create pairs - a simple matching procedure

- Different Options:

1. In “Background” set:
   ```
   number of subjects=10
   number of groups=5
   ```
2. Matching in the “Parameter Table”
3. Do it manually in a program (advanced, but most control)

Matching in groups and types in a program

- Imagine you need a group size of 4 and within one group you have 3 types of players (1 A, 1 B, 2 C)
- In the background you define `number of groups=1`.
- Define your Types in a program in the globals table:

```
TYPEA=1; TYPEB=2; TYPEC=3;
```

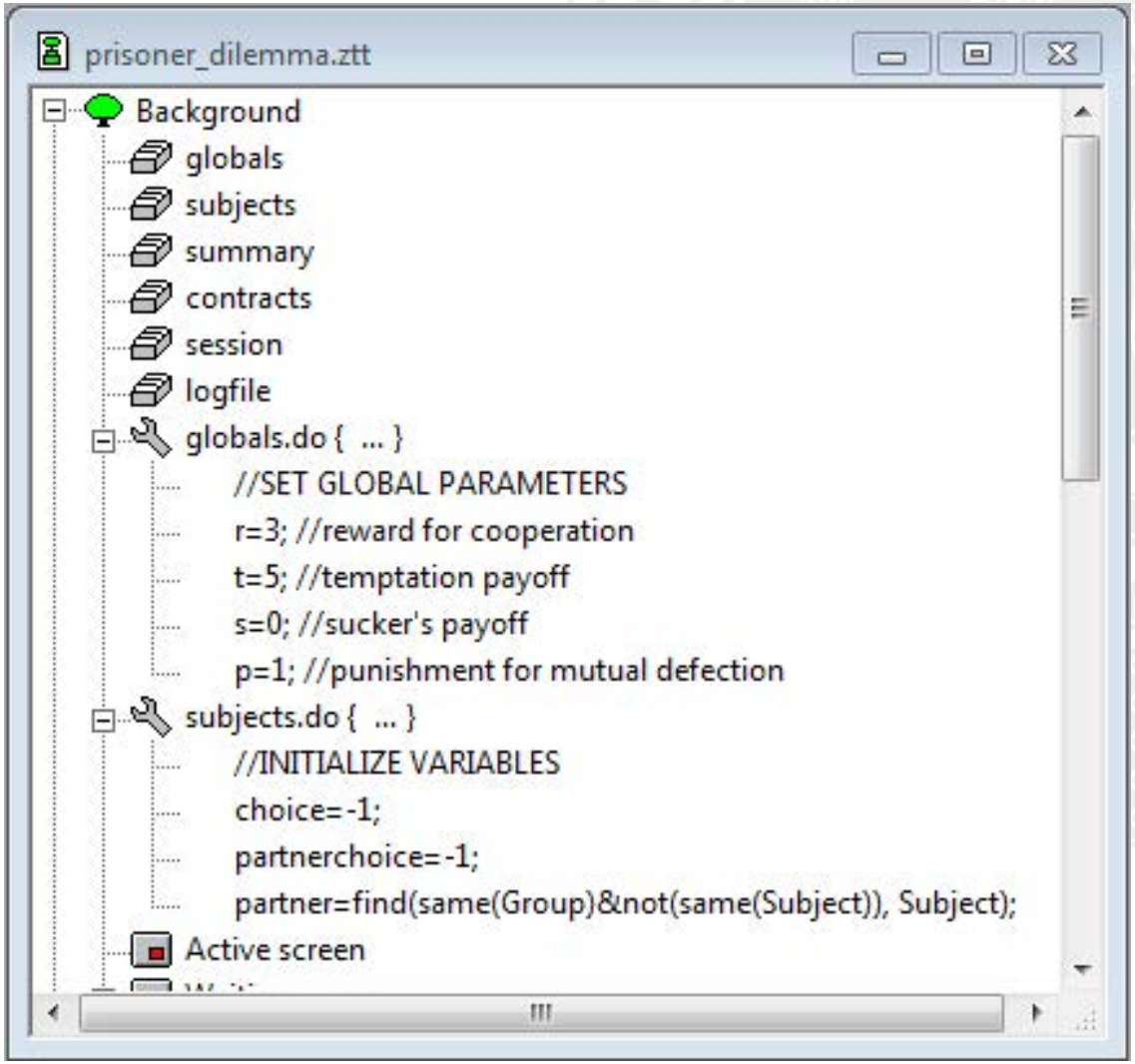
- Define a Type variable in a program in the subjects table:

```
Type = 0;
```

- In another program in the subjects table you write:

```
// Groups and Type
N = count();G = 4;
Group = roundup((Subject-.5)/G, 1);
Type = 1 + mod (Subject, G);
if (Type==4) {Type = 3;}
```


Find the Partner



Mark Pigors

University of Cologne

Table Functions I

- Function `count()`, `find()`,... are table functions, a function which does not only refer to a single record of a table, but runs over the whole table.
- A complete list of table functions can be found at page 46 of the Reference Manual.
- Find the partner
  `partner=find(same(Group)& not(same(Subject)),Subject)`
- `find(condition, x)` looks at all the records in a table, from top to bottom, and returns the value of variable x of the first record in which condition is TRUE.

Table Functions II

- `same(x)` is TRUE for all records in the table, where the variable (or expression) `x` takes the same value it has in the "reference record", `not(same(x))` is TRUE when `same(x)` is FALSE and vice versa.
- & is the logical AND; | is the logical OR.

The scope operator

- `partner=find(same(Group)& not(same(Subject)),Subject)`
  is the same as
  `partner=find(Group=:Group & Subject!=:Subject,Subject)`
- The scope operator refers to the next higher scope.
- `partnerchoice=find(Subject==:partner, choice)`
  The scope operator ":" indicates that the variable partner belongs to the record in which the cell partnerchoice lies, not to the record where Subject lies.
- A second function of the scope operator: Different Tables contain the same variable.
  → see page 27 of the z-Tree Tutorial
- "\" assigns highest possible scope. This is always the record of the globals table.

Mark Pigors

University of Cologne

IF as a Statement and as a Function

- `if(condition a)c1`
  `elsif(condition b1)c2`
  `elsif(condition b2)c3...`
  `elsec4`
  If condition a is TRUE, then command(s) c1 is (are) executed; if condition a is FALSE and condition b1 is TRUE, then command(s) c2 is (are) executed;(and so forth...)
  if conditions a, b1 and b2 are FALSE, then command(s) c4 is (are) executed;
- `if(condition a, x, y)`
  If condition a is TRUE, then the value of the function is x, otherwise y

Mark Pigors

University of Cologne

Exercise: The Traveller's dilemma

- The Traveller's dilemma is a simple two-players game.
- each player can choose a number between 2 and 100
- the choice is simultaneous
- the player's profit is:
  - equal to the number he chose, if this number is equal to the number chosen by his partner
  - equal to the number he chose +2, if this number is lower than the number chosen by his partner
  - equal to the number he chose -2, if this number is higher than the number chosen by his partner

Increase the number of periods

- Double-click on Background
- Set the number of periods and trial periods

→ `prisoner_dilemma_2.ztt`

Diffferent possibilities of matching in repeated games I

1. From the menu: “Treatment” $\rightarrow$ “Matching”
   - 1.1 partner - groups remain the same throughout the whole treatment
   - 1.2 stranger - groups are randomly formed at the beginning of each period
   - 1.3 absolute stranger - players never meet more than once in the treatment (not always possible)
2. Load the matching from an external table.
   - 2.1 Open the Parameter Table.
   - 2.2 Click on “Treatment” $\rightarrow$ “Import Variable Table”.
   - 2.3 `Name of variable = Group.`
   - 2.4 Select the matching table (tab separated .txt file).

Diffferent possibilities of matching in repeated games II

2.5 If you double-click on one of the cells of the parameter table, you'll see a program defining the Group variable, for a specific subject in a given period. Programs in the parameter table are executed after those in the background.

3. Define the matching in a program.

Changing of parameters of the game across periods I

1. Create a new table (with Excel, or similar programs)
2. The first column reports the name of the table you want to load.
3. Save the table as a tab delimited .txt file
4. In z-Tree, from the Background click on “Treatment” $\rightarrow$ “New Table”.
5. In z-Tree, from the Background click on “Treatment” $\rightarrow$ “New Table Loader”
6. Append/Replace filename: name of your .txt file
7. In the Background, after the “Table Loader” create a new Program, to load the new parameters:

Changing of parameters of the game across periods II

- An alternative possibility is setting parameters manually, adding programs in the Parameter Table.

Grid Boxes

- Now the payoff matrix of the game changes every period. You might want to show it on the subjects' screens.

YOUR PAYOFF		
your choice/your partner's choice	C	D
C	3	0
D	5	1

Your Choice: D C

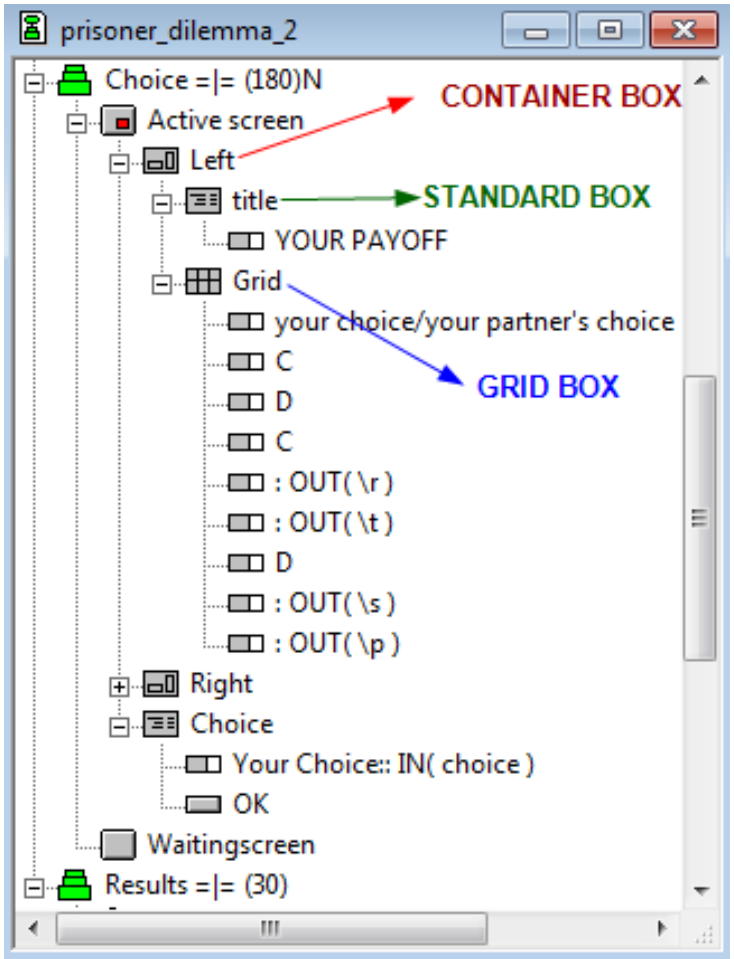
OK

YOUR PARTNER'S PAYOFF		
your choice/your partner's choice	C	D
C	3	5
D	0	1

Mark Pigors

University of Cologne

Container Boxes



- A **container box** is a box that can contain other boxes
- Note: the relative measures of the boxes it contains are defined with respect to the container box, not to the whole screen
- A grid box presents items in a tabular form.

History Box

- To remind subjects about what happened in past periods, you can insert a “history box”.
- Lists results from previous periods.
- Only takes variables form the subjects table.
- A label row contains the labels.
- If the table is too long, a scroll-bar appears.
- Option: showing/not showing the current period.

Mark Pigors

University of Cologne

Indefinite number of periods

→ `prisoner_dilemma_sequential.ztt`

- Define a treatment with a single period.
- In the Background, create a new program which runs on the globals table.
- In this program, you can set the variable “RepeatTreatment”, as follows:

```
//SET THE TERMINATION RULE
continuation_probability=0.30;
random_number=random();
RepeatTreatment=if(random_number<=
  continuation_probability,1,0);
```


Parameters with unknown length of a treatment

1. randomize:

```
//LOAD PARAMETERS FROM THE IMPORTED TABLE
n=max(1,roundup(3*random(),1)); //random number equal to
1, 2 or 3.
r=parameters.find(Period==\ n, r);
t=parameters.find(Period==\ n, t);
p=parameters.find(Period==\ n, p);
s=parameters.find(Period==\ n, s);
```

2. rotate:

```
//LOAD PARAMETERS FROM THE IMPORTED TABLE
n=if(mod(Period,3)==0,3,mod(Period,3)); //number taking
value 1, 2 or 3, in turn.
r=parameters.find(Period==\ n, r);
```

- Note: setting parameters in the Parameter Table is less flexible.

Random matching with unknown length of a treatment

- Hard to use the parameter table or to import an external matching table.
- Matching in a program in the Background, running on the globals table (it runs only once, at the beginning of each period).
  1. Generate a different random number (rand) for each subject;
  2. Generate the variable rank, which sorts subjects according to the variable rand;
  3. Group together subjects having consecutive ranks.

→ `random_matching_program.ztt`

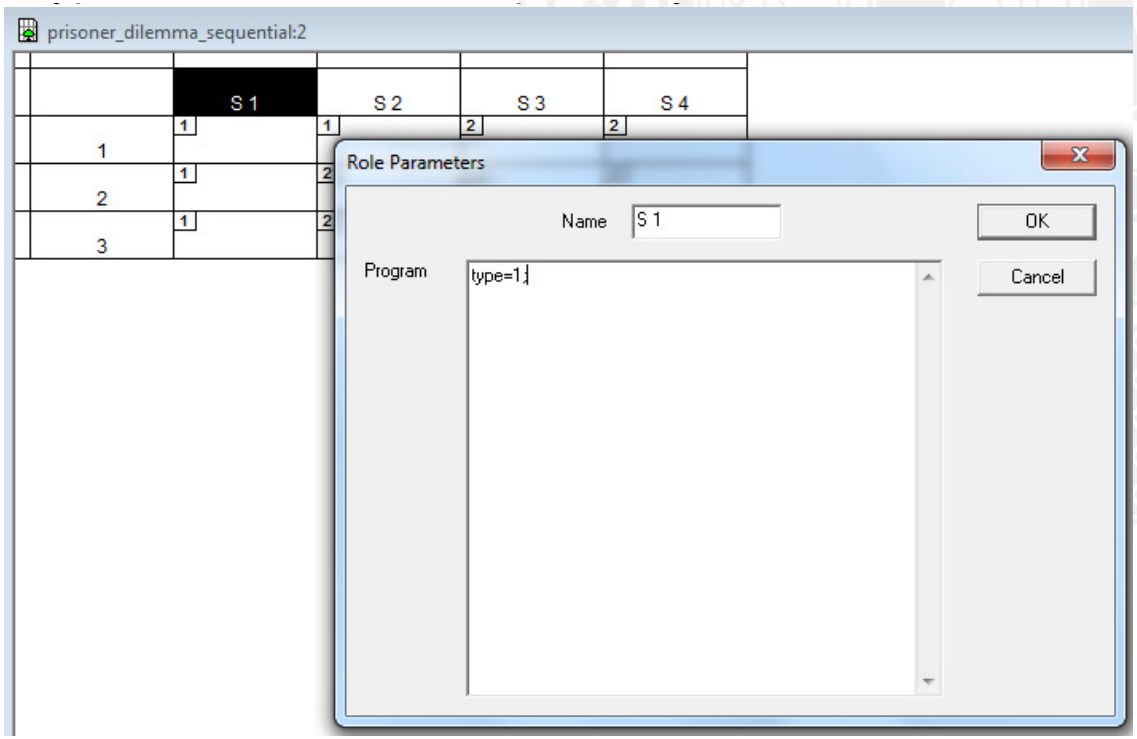
```
//CREATE GROUPS
i=1;
repeat{
  i=i+1;
  subjects.do{
    rand=random();
  }
  subjects.do{
    rank=subjects.count(rand<=:rand);
  }
}
while(subjects.sum(Subject)!=subjects.sum(rank)&i<10);
subjects.do{
  Group==roundup(rank/2,1);
}
```

do statements and loops

- do statement:
  With `do{commands}` the commands are executed for all records in the current table.
  With `subjects.do{commands}` we specify that the commands should be executed for all records in the subjects table.
- loops:
  `repeat{commands} while {condition} statement`
  The commands are executed. Then it is checked whether the condition is TRUE, and as long as it is, the commands are repeated.
- Loops can be left with the key combination **Ctrl+Alt+F5**

Assigning Types

- Suppose now you want to let subjects play the prisoner dilemma sequentially. 2 types of players: 1 (first mover) and 2 (second mover): type is a subject variable. If types remain fixed across periods, you can set them in the Parameter Table.



Mark Pigors

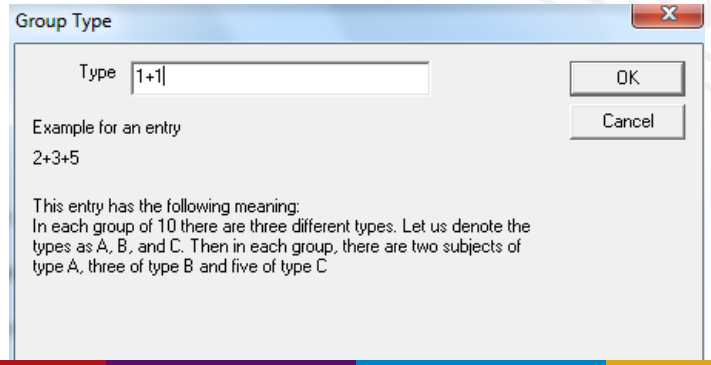
University of Cologne

Assigning Types

- If you don't want to set types manually, you can assign types with a program in the Background (on the globals table):

```
numsubjects=subjects.maximum(Subject);
subjects.do{
    type=if(Subject<=\numsubjects/2,1,2);}
```

- With types, you can use the "Absolute typed strangers" matching procedure (Reference Manual, page 32). Click on "Treatment" → "Matching"



Mark Pigors

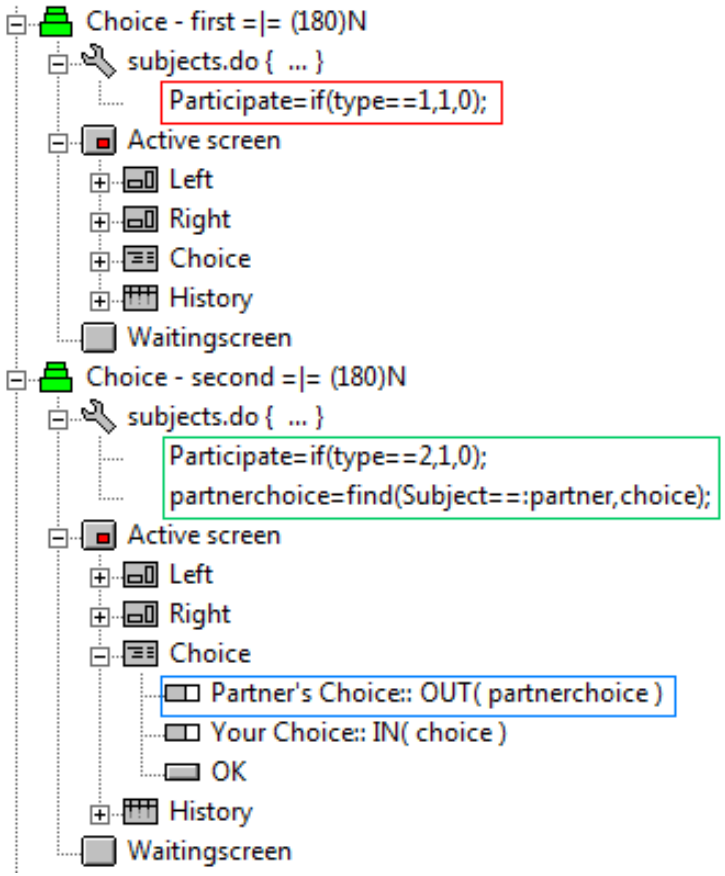
University of Cologne

Assign Types and Groups Randomly

- It is possible to set types randomly at the beginning of each period, with a program in the Background:
  1. Rank subjects according to some random variable
  2. Set type=1 for the first half of the subjects, type=2 for the others
  3. Form groups picking the first subject from the first half, and the second subject from the second half.

→ `prisoner_dilemma_sequential.ztt` → `assigning_types.ztt`

The `Participate` variable



- Once you have defined the types, you can modify the structure of the program.
- The Participate variable in the subjects table determines whether a subjects enters a stage or not.
- If this variable has value 1 then the subjects enters the stage, i.e., the corresponding Active Screen appears on the subjects computer screen.
- Programs in the stage are executed for all subjects.

Mark Pigors

University of Cologne

Program execution

- At the beginning of each period, the programs are executed in the following order:

1. Standard variables (Subject, Period, etc.) are set.
2. Programs in Background, in the order they appear in the .ztt file
3. Programs in the Parameter Table:
   - 3.1 cells: Subject programs (in current period) in the subjects table
   - 3.2 top row: Role program in subjects table
   - 3.3 first column: Period program in globals table
4. Programs of the first stage

Exercise: Battle of the Sexes

- The “Battle of the Sexes” is a two-player asymmetric game. Rob (row player) and Clara (column player) want to go out together tonight. She prefers boxing, he is a fan of ballet.

		Clara: Boxing	Clara: Ballett
Rob	Boxing	*m*,*hc*	*l*,*l*
	Ballett	*l*,*l*	*hr*,*m*

- They will go out together every Tuesday, as long as their engagement lasts.

Exercise: Details I

- Indefinitely repeated game: There is a 10% probability that Rob and Clara break up before next Tuesday.
- Partner matching (couples remain the same throughout the treatment). Types are also fixed across periods.
- $l = 0$, $m = 3$, while hc and hr parameters vary randomly across periods:
  sometimes the box match is particularly important: $hc = 8$, $hr = 5$;
  sometimes the ballet company is very good: $hc = 5$, $hr = 8$;
  otherwise $hc = 5$, $hr = 5$;
- Simultaneous moves.

Exercise: Details II

- Show a history box and two grid boxes to display the payoff matrices. You can use the sequential prisoner dilemma as a starting point.
- Create 3 stages:
  1. Rob's choice stage
  2. Clara's choice stage
  3. Results
- To let the two players play simultaneously, let Rob's choice stage and Clara's choice stage "start if possible".

Exercise: Details III

YOUR PAYOFF		
your choice/Clara's choice	box	ballet
box	3	0
ballet	0	5

Hi Rob, make your choice: ballet / box

OK

CLARA'S PAYOFF		
your choice/Claras's choice	box	D
box	8	0
ballet	0	3

Period	Partner	Your Choice	Clara's Choice	Profit
1	4	ballet	ballet	8
2	4	ballet	ballet	8
3	4	ballet	box	0
4	4	box	box	3

Mark Pigors

University of Cologne

Part III

10. Second price sealed-bid auctions
Checkers
The `while` statement

11. Dutch auctions
The `later` statement

12. Posted offer markets
Contracts

13. Double auctions

Mark Pigors

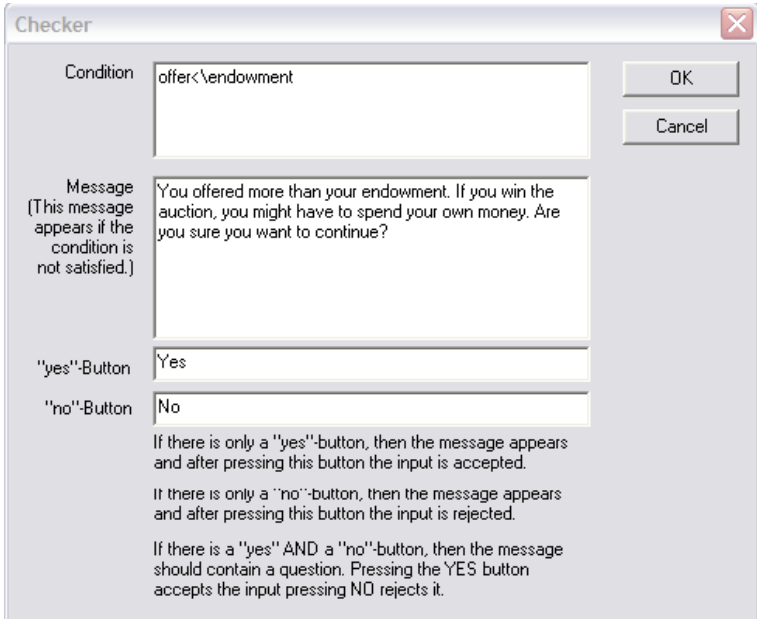
University of Cologne

Second price sealed-bid auctions

- Purpose: elicit the true value of a good (e.g. a mug) for each of the subjects.
- Each subject receives an initial endowment (20 euros)
- each subject makes an offer for the mug (possibly even above 20 euros). The offers are secret and simultaneous.
- The subject who makes the highest bid wins the mug and has to pay a price equal to the second highest bid.
- In case of a tie, the winner is drawn at random among the bidders who submitted the highest bid.

$\rightarrow$ `second_price_auction.ztt`

Checkers



- Checkers are used to verify the validity of an input.
- To create a checker, select the button you need to "check" and click on "Treatment" $\rightarrow$ "New Checker".

Mark Pigors

University of Cologne

The `while` statement

- while statement

```
while (condition) {
program;
}
```

- While the `condition` is TRUE, the `program` is executed.
- Reminder: loops can be left with the key combination `Ctrl+Alt+F5`.

Dutch auctions

- A Dutch auction is an auction in which the auctioneer begins with a very high asking price, which is progressively lowered until some participant accepts the auctioneer's price, or until a predetermined time is over.
- All subject are buyers.
- global parameters:
  - initial asking price
  - duration of the auction (in seconds)
  - step of decrease of the price
  - frequency of decrease of the price (in seconds)

→ `dutch_auction.ztt`

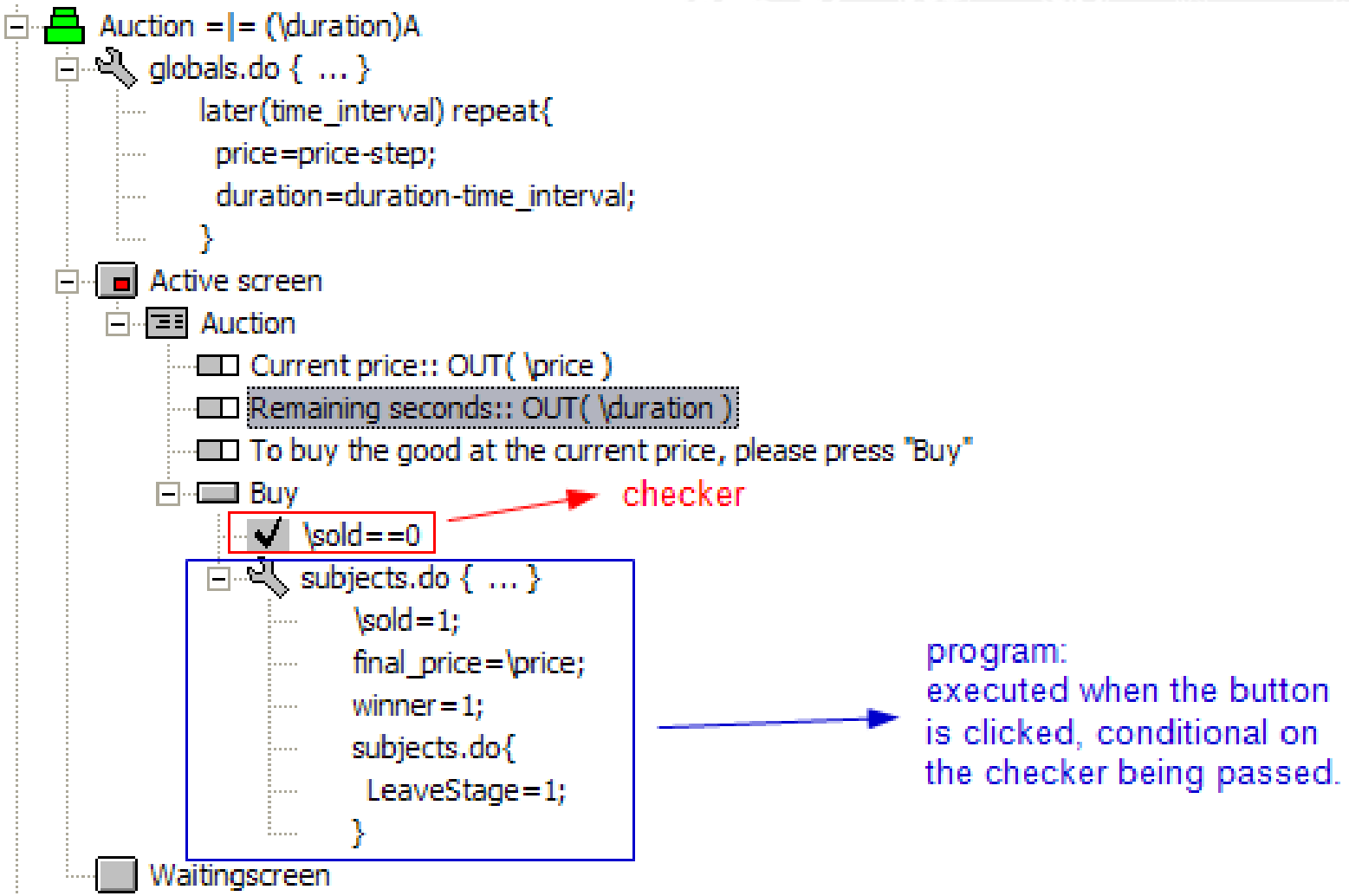
The `later` statement

- The later Statement
  ```
  later(a) repeat {
  program;
  }
  ```
- Expression `a` is calculated. The resulting number of seconds later, the `program` is executed.

Mark Pigors

University of Cologne

Programs into Buttons

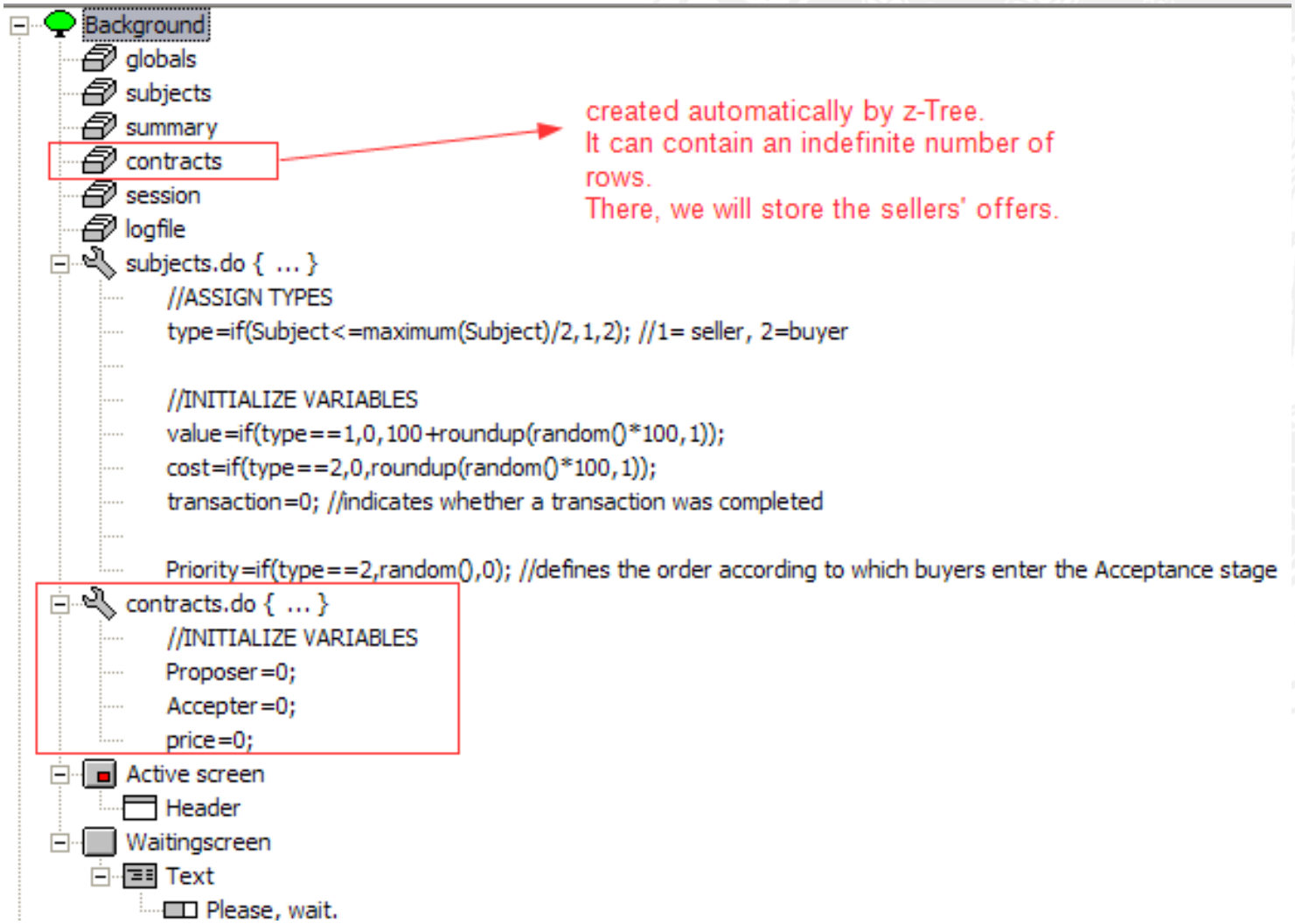


Posted offer markets

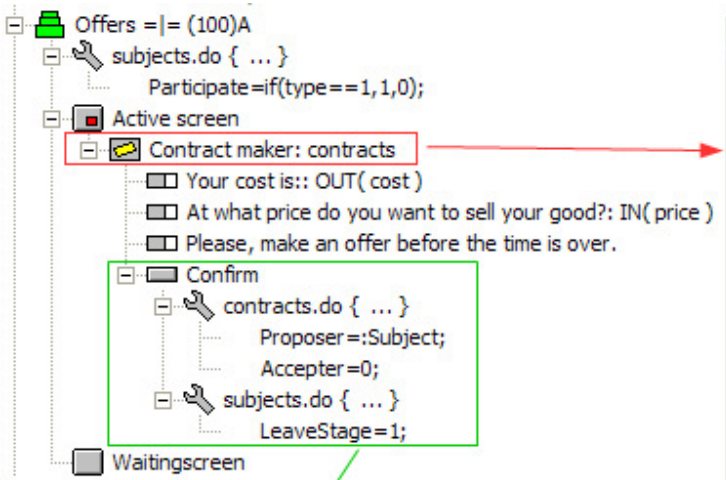
- Subjects in the role of buyers and sellers.
- Each seller makes an offer, without knowing the offers made by other buyers.
- Buyers act sequentially, in random order.
- They can see all the sellers' offers that are still open, and choose which one to accept, if any.
- Accepted offers are not visible anymore to subsequent buyers.

→ `posted_offers_markets.ztt`

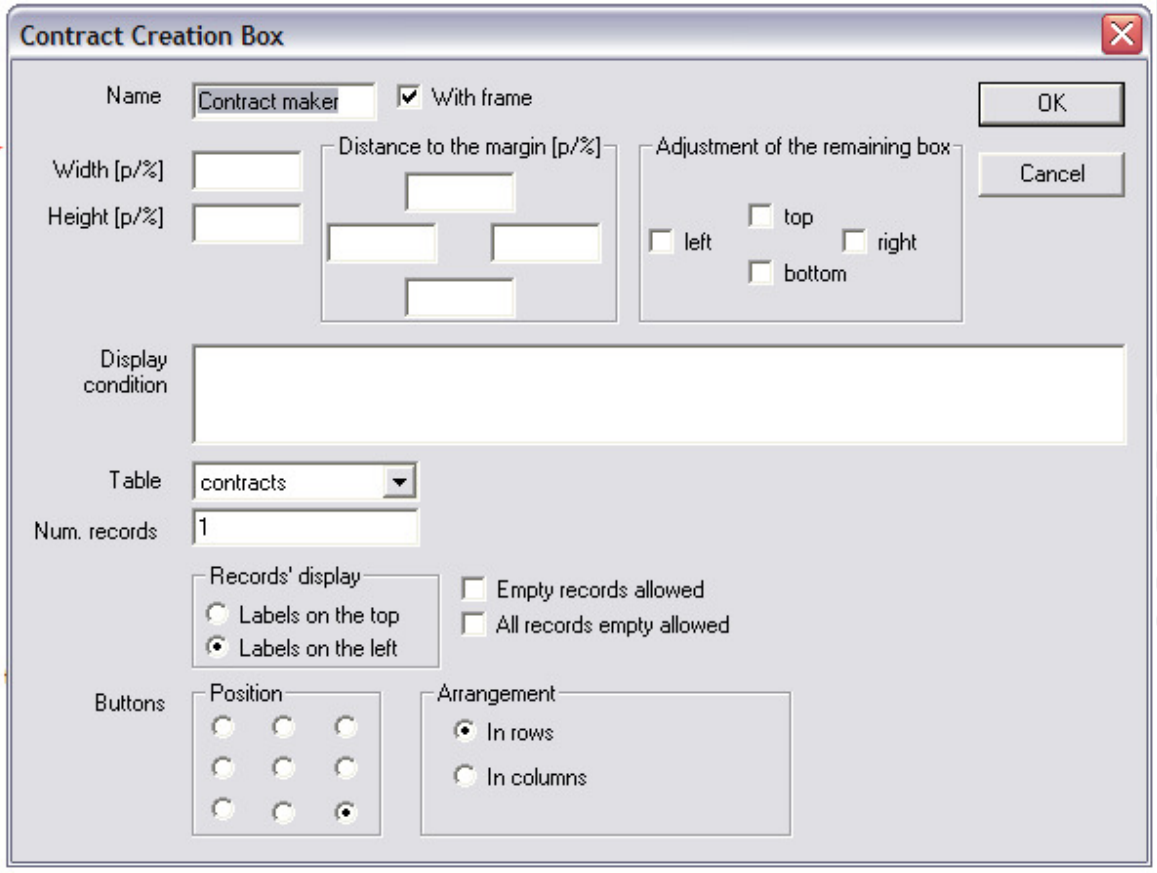
The Contracts Table



The Contracts Creation Box



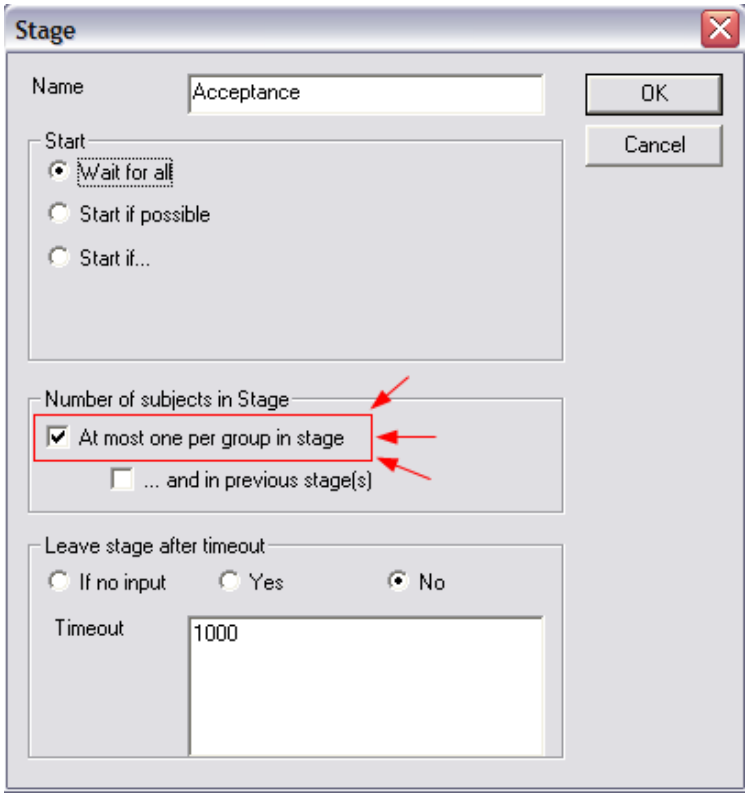
Adds one line in the contracts table.
In this line, price takes tha value of the input filed, Proposer is set equal to the subject who made the offer, and Accepter is set equal to 0.
After the seller has made his offer, he leaves the stage.



Mark Pigors

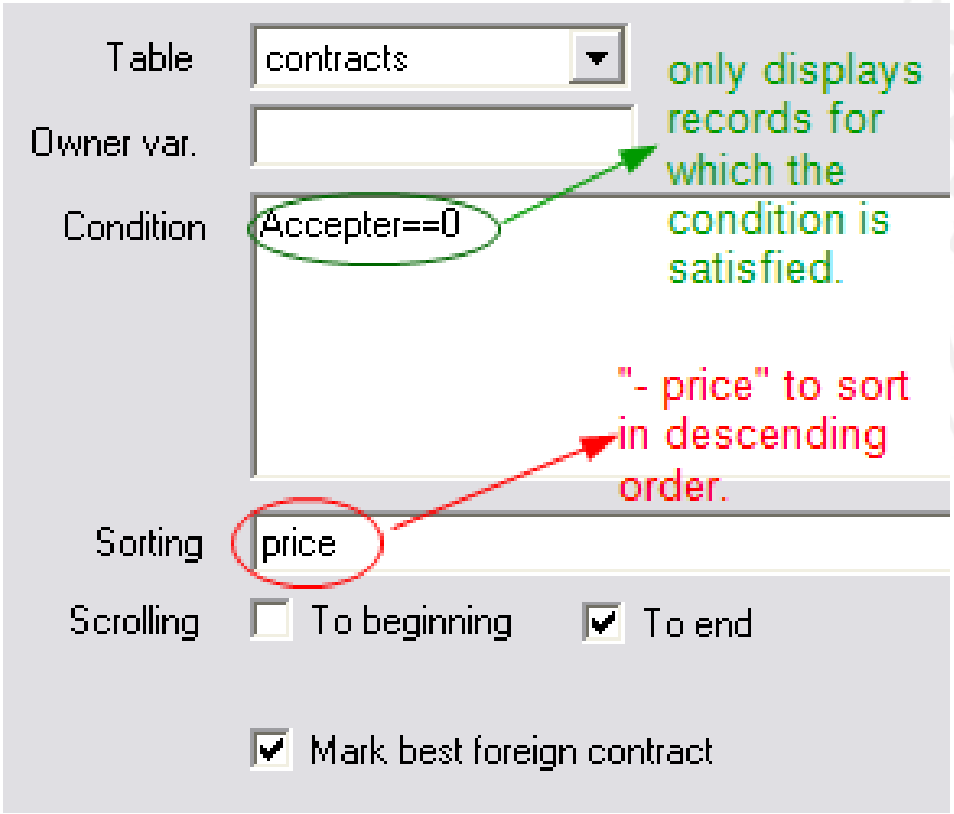
University of Cologne

Buyers enter the acceptance stage one by one



- The variable “Priority”, if set, defines the order according to which subjects enter the stage.
- If “Priority” is not set, subjects enter the stage in random order.

The Contract List Box




```
Contract list: contracts( Accepter==0 ), sorted by: price
    Proposer: OUT( Proposer )
    Price: OUT( price )
    Accept
        contracts.do { ... }
            Accepter=:Subject;
    Continue
```

Proposer	Price
3	100
2	106
1	142

Continue | Accept

Mark Pigors

University of Cologne

Double auctions

- See the z-Tree Tutorial, pages: 57-66
- Subjects in the role of buyers and sellers.
- Both sellers and buyers can make offers, at the same time.
- Each seller and each buyer can make more than one offer.
- Sellers and buyers can see all the open offers (made by buyers and by sellers)
- Each seller can accept only one of the buyers' offers.
- Each buyer can accept only one of the sellers' offers.

→ `double_auction.ztt`

Remaining Time [sec]: 52

YOU ARE A BUYER

The value of the good for you is: 185

Your offer 100

OK

BUYERS' OFFERS

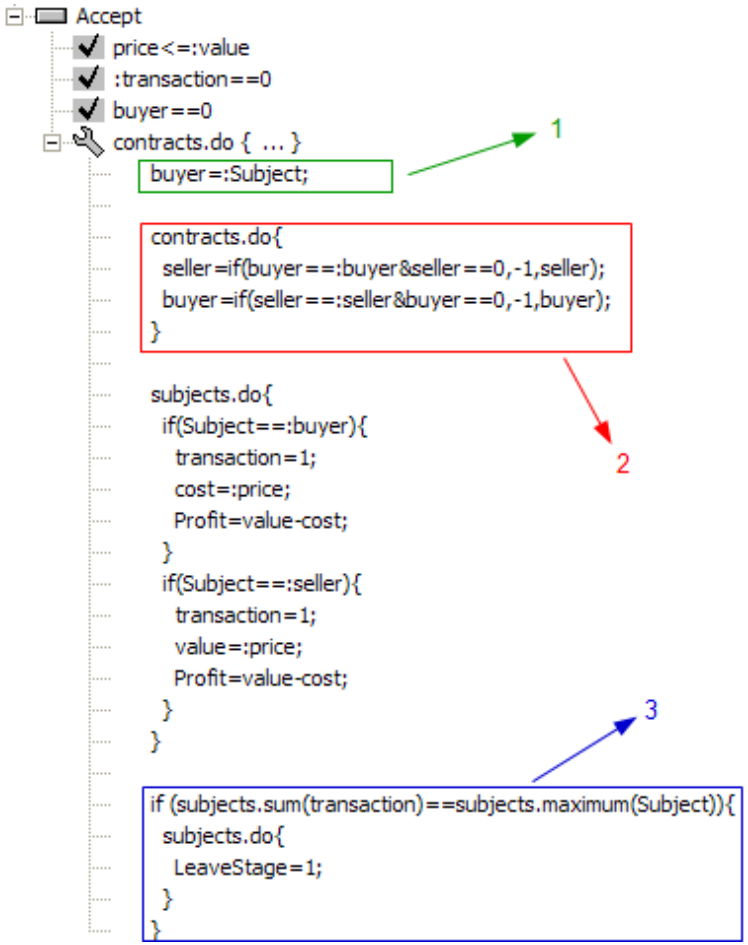
Buyer	Price
5	100
You	100
5	101
4	102
4	109
You	125

SELLERS' OFFERS

Seller	Price
3	106
2	117
3	120
1	138
1	168
2	177

Accept

Accepting an offer



1. Set the ID of the accepter.
2. Close all other offers by the same proposer and by the same accepter.
3. If all subjects have signed a contract, leave the stage.

- Note: the variable `LeaveStage` is preset in z-Tree.

Mark Pigors

University of Cologne

Part IV

14. Advanced concepts
- Arrays
- Iterators
- Retrieving data from previous periods
- Bankruptcy
- Chats

15. z-Tree tables

16. Questionnaires

17. Handling crashes
- Crash of the subject's PC
- Crash of z-Tree

Arrays

- Arrays are indexed variables. In an index set, every finite, equidistant subset of numbers is allowed (e.g 1,2,3,...; but also 5, 10, 15, ...).
- The index set needs to be defined before the first use of the array.
- This is done with one of the following instructions:
  - `array arrayname[]` defines an array with indices from 1 to the number of subjects
  - `array arrayname[n]` defines an array with indices from 1 to n
  - `array arrayname[x,y]` defines an array with indices from x to y
  - `array arrayname[x,y,z]` defines an array with indices x, x+z,x+2z,..., y.

Mark Pigors

University of Cologne

Example

```
//the index set of array p is {10, 15, 20}

array p[10, 20, 5];
p[10]=1;
p[15]=5;
p[20]=2;
p[30]=3;              //sets p[20] to 3
x = p[10] + p[20]; // x=4
```

- `arrayname[indexvalue]` reports the element of array arrayname, corresponding to the index `indexvalue`.
- The expression indexvalue is rounded to the nearest possible index.

Iterator

- Loops can be programmed with the statements repeat and while.
- Another way of programming loops is by using iterators.
- An iterator creates a small table that contains only one variable.
- When a table function or a do-statement is applied to an iterator, it corresponds to a loop over the values contained in the table.
- An iterator has the following syntax:
  - `iterator (varname)` variable varname runs from 1 to the number of subjects.
  - `iterator (varname, n)` variable varname runs from 1 to n.
  - `iterator (varname, x, y)` variable varname runs from x to y.
  - `iterator (varname, x, y, z)` variable varname runs from x to y with steps of z.

Mark Pigors

University of Cologne

Example

- This example provides two possible ways of calculating the sum of the square of the first five natural numbers:

```
SquareSum= 1 + 4 + 9 + 16 + 25 = 55
\\First way:
SquareSum = iterator(i, 1, 5).sum(power(i,2));
\\Second way:
SquareSum = 0;
iterator(i,5).do{
:SquareSum = :SquareSum + power(i,2);}
```

- Iterators are tables, so they have their own scope.

Retrieving data from previous periods

- In z-Tree, the subjects database is set up freshly in every period, so the information about earlier periods is not available directly.
- However, the tables of the previous periods can be accessed with the prefix `OLD`.
- So, if you want to copy some variable from the previous period, you can write:

```
if (Period >1) {
variable=OLDsubjects.find(same(subject), variable); }
```

- This way, you can only access data from the previous period. To access data from earlier periods, a different procedure should be followed.

Mark Pigors

University of Cologne

Example: random partner matching

- Example from Part II: Imagine you need a group size of 4 and within one group you have 3 types of players (1 A, 1 B, 2 C)
- Main part:

```
// Groups and Type
N = count();G = 4;
Group = roundup((Subject-.5)/G, 1);
Type = 1 + mod (Subject, G);
if (Type==4) {Type = 3;}
```

- Goal: partner matching where the first matching is random
- In period 1 a random number is drawn to define groups and types
- In later rounds variables are taken from `Old.subjects`

→ `random_partner_matching_program.ztt`

Example

- Suppose you want to run an experiment with the following structure:
  1. N periods
  2. In every period, each subject earns a certain amount of points
  3. The subject's profit is determined at the end of period N:
     - Three of the periods are drawn at random by the subject
     - The profit is set equal to the sum of the points earned by the subject in these three periods

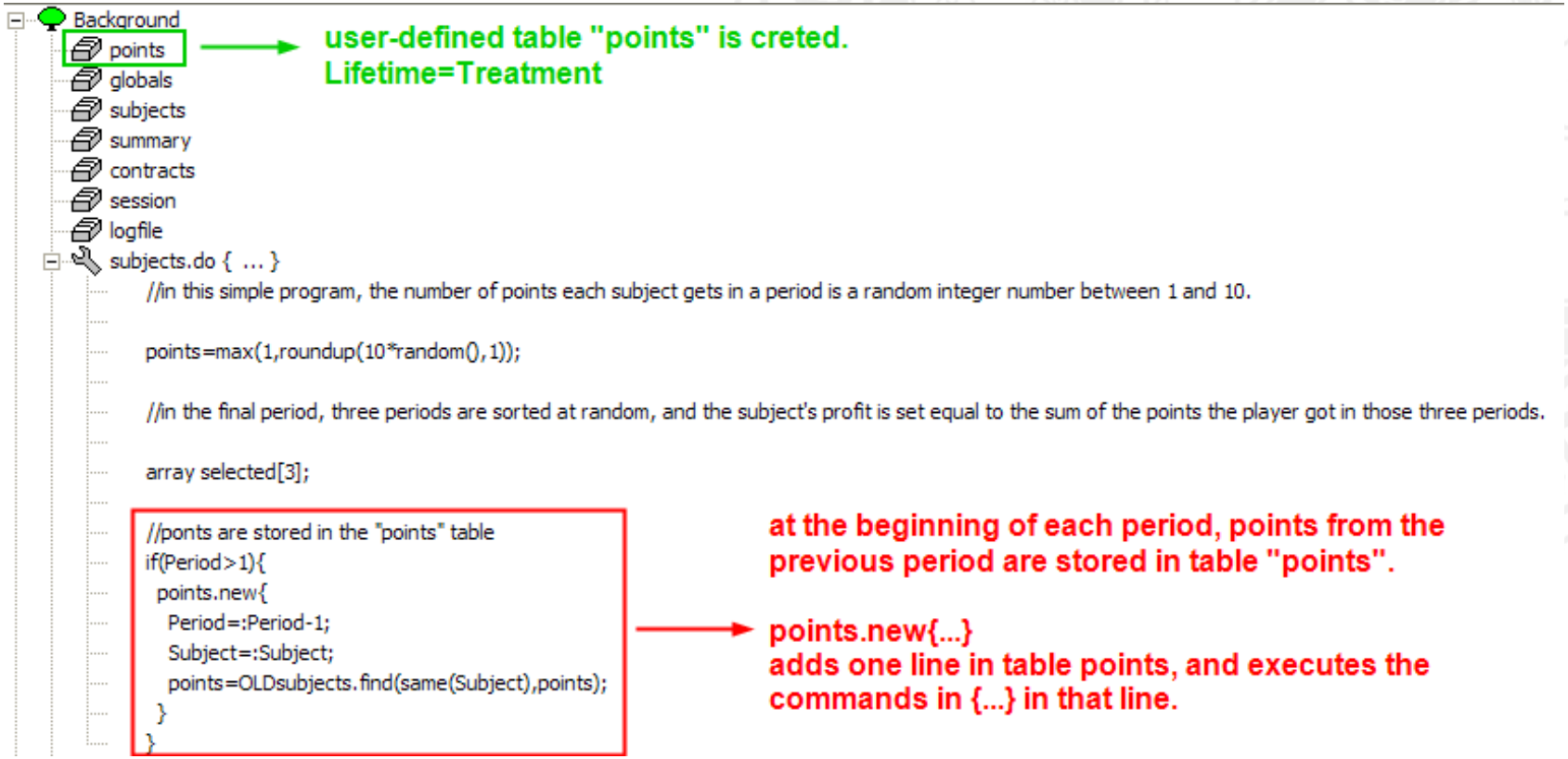
→ `values_from_previous_periods.ztt`

YOUR PONITS IN THE PAST 10 PERIODS

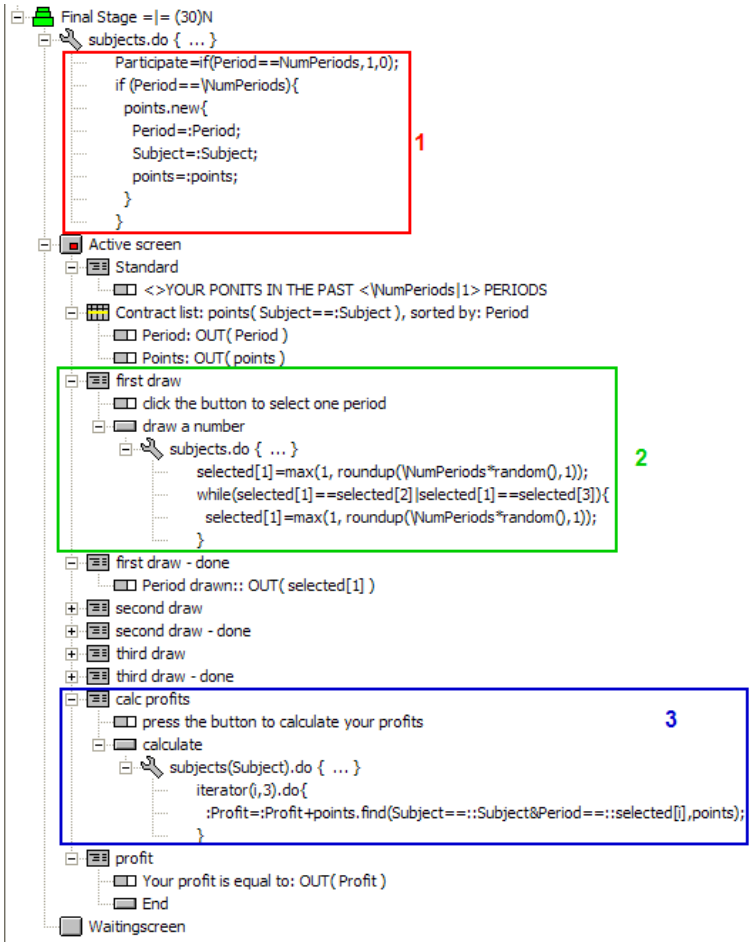
Period	Points
1	1
2	7
3	6
4	9
5	9
6	5
7	2
8	9
9	3
10	5

click the button to select one period	click the button to select one period	click the button to select one period
draw a number	draw a number	draw a number

- To retrieve data from all earlier periods, these data have to be stored in a user-defined table with “lifetime” set to “Treatment” or to “Session”.



Buyers enter the acceptance stage one by one



1. Store points form the last period.
2. Draw a random period. This box is shown only if the draw is yet to be done
   Display condition: `selected[1]==0`
3. Calculate the profit, using the iterator.

Mark Pigors

University of Cologne

Bankruptcy I

- Try to design the experiment in order to avoid possible losses!
- If losses occur, they can be covered by the following sources:
  1. Profits from previous periods and treatments
  2. Show-up fee
  3. Money injected by the experimenter during the session
- If previous profits are not enough to cover all losses, but the show-up fee is, a message appears on the subject's screen.
- This message informs the subject that he can choose either to use the show-up fee, or to drop out of the experiment.
- If the subject chooses to use the show-up fee, he may simply play on.

Bankruptcy II

- Otherwise, the subject reaches the state “BankruptShowupNo” (which you can read in the clients table), and you have to release him manually from the server.
- If the show-up fee cannot cover the losses (or has already been exhausted) another message appears on the subject’s screen, informing him that he has incurred a loss. This message can be concluded with a question.
- By answering “no”, the subject enters the state “BankruptMoreNo”. In this case, the experimenter has to release the subject from the server.

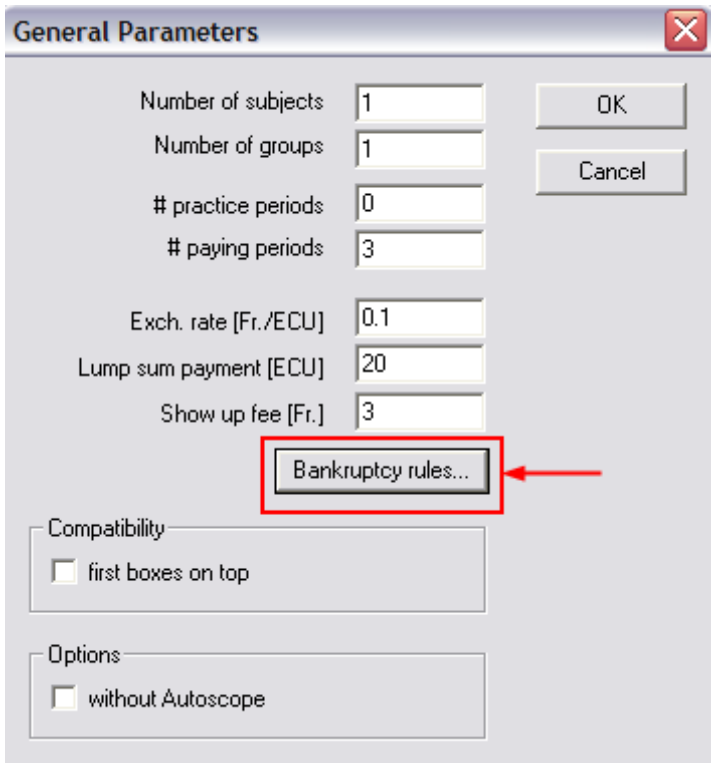
Bankruptcy III

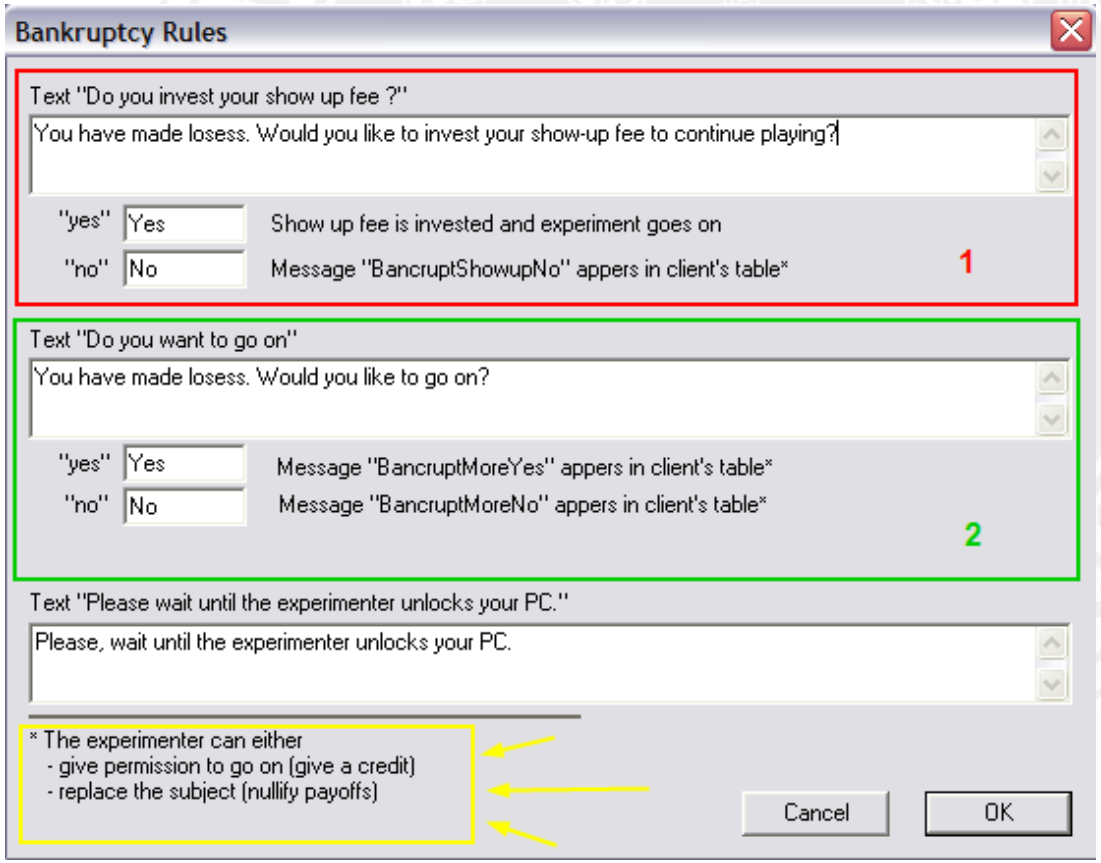
- If the subject answers this question with “yes”, he enters the state “BankruptMoreYes”. In this case, there are 3 possibilities:
  1. Allow the subject to continue, by injecting into his account an amount of money higher than his current losses.
  2. Another subject takes the role of the subject released.
  3. The subject drops out.

→ `bankruptcy.ztt`

Mark Pigors

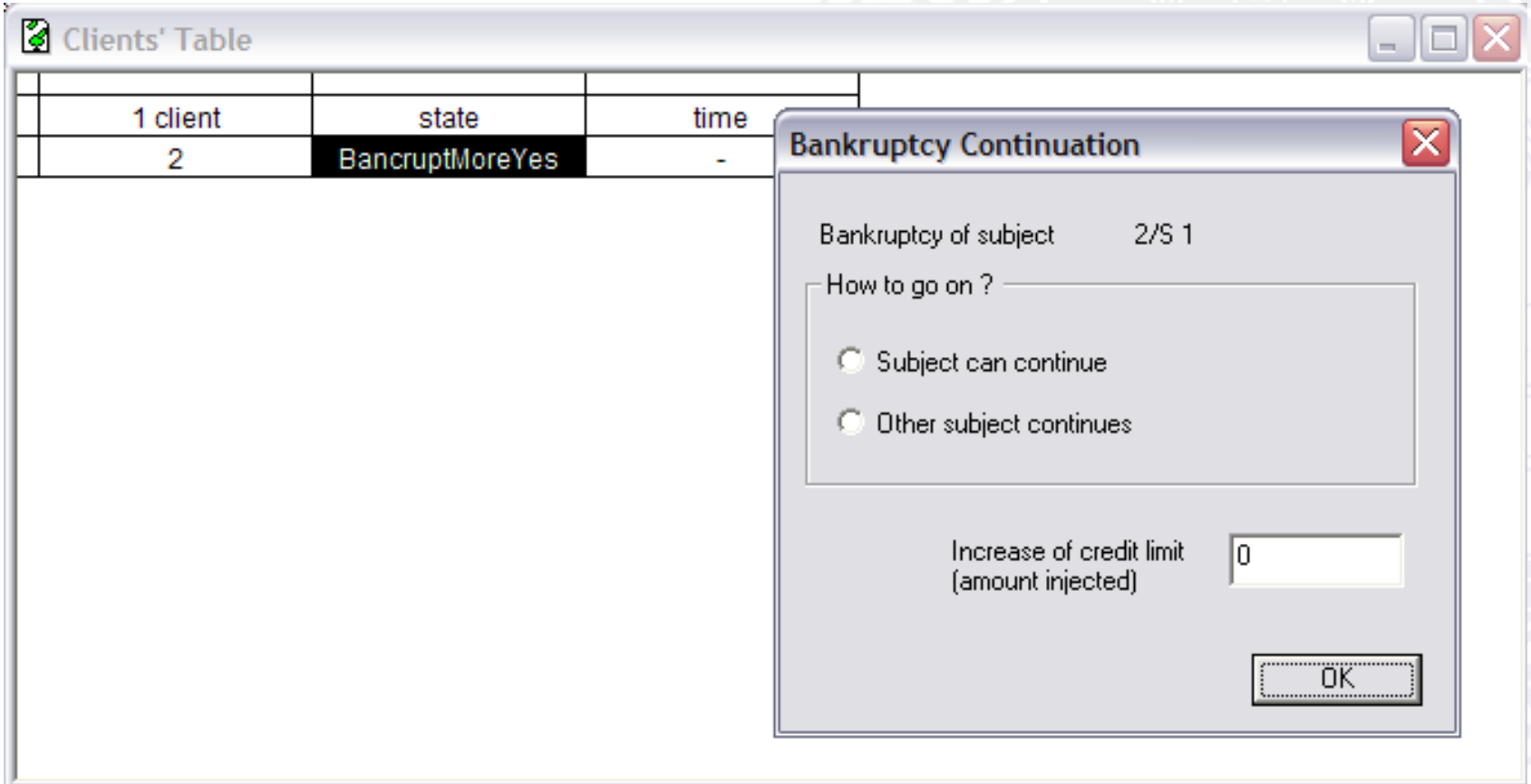
University of Cologne





Mark Pigors

University of Cologne



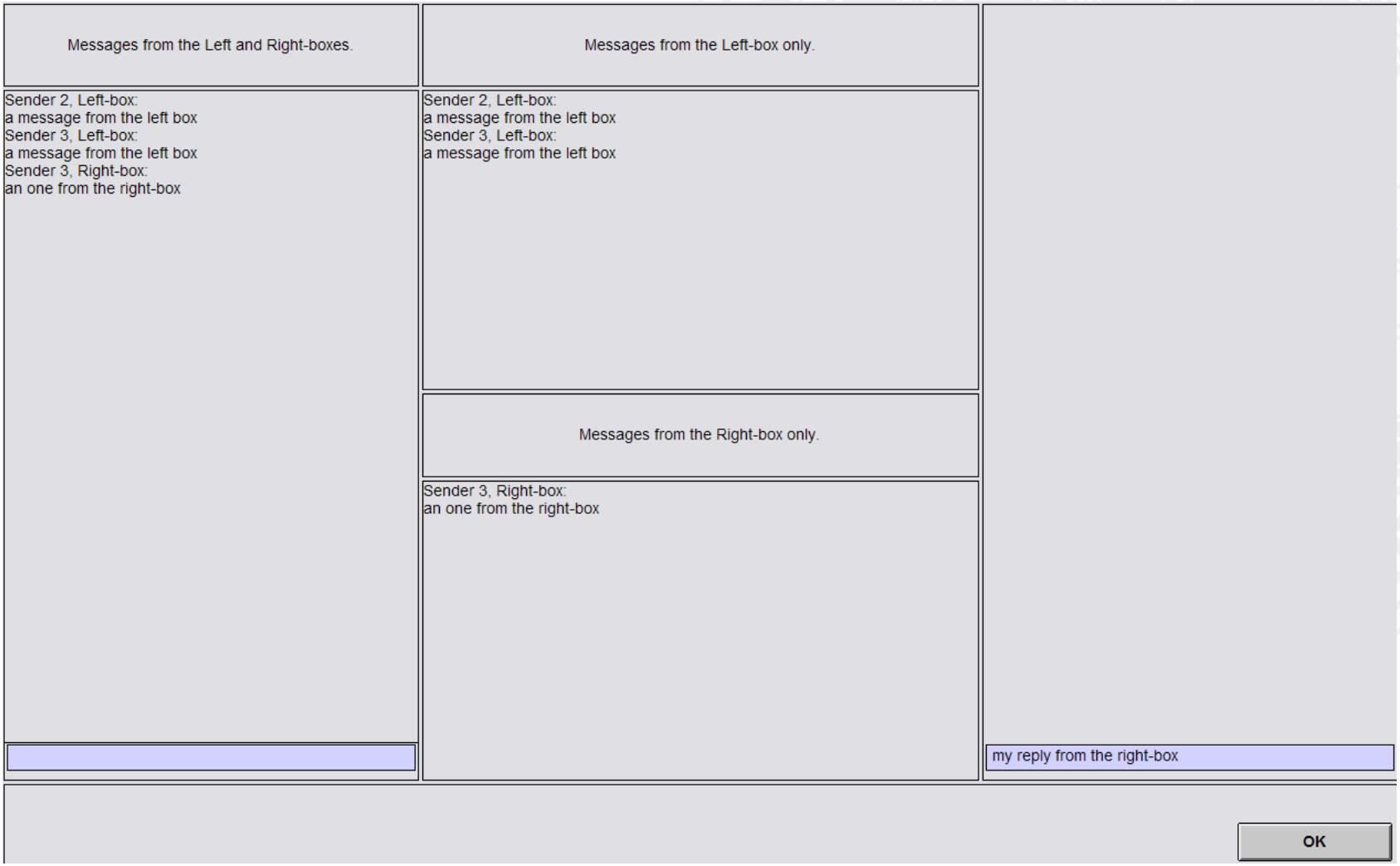
session table

Subject	FinalProfit	ShowUpFee	ShowUpFeeInvested	MoneyAdded	MoneyToPay	MoneyEarned
1	-4.7000000000	3	1	3	1.2999999999999	-1.7000000000000

Mark Pigors

University of Cologne

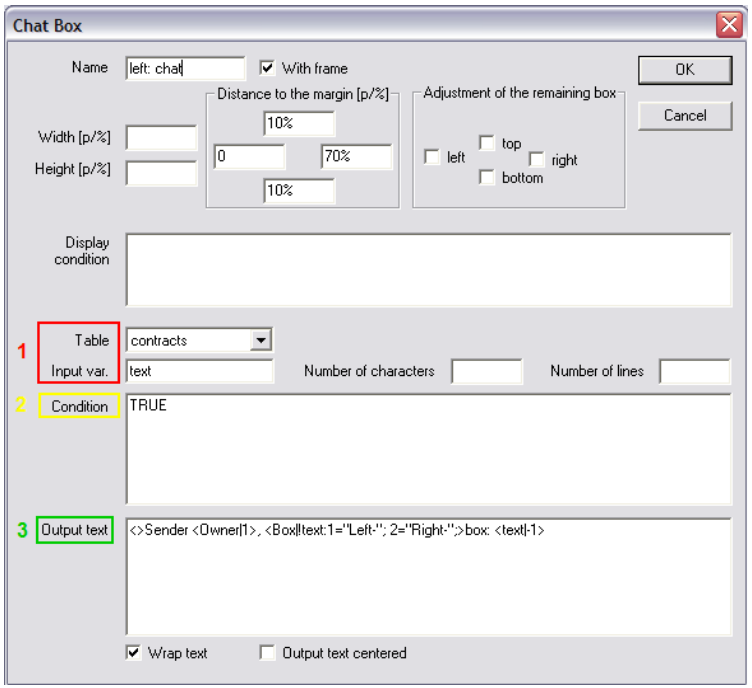
Chats



→ chat.ztt

Chat Box

- In a stage, you can add a chat box by clicking on: “Treatment” → “New Box” → “New Chat Box”



1. Specify the table where you want to store the chat data
   - contracts table
   - user-defined table
2. Specify what records you want to display on the subject’s screen.
3. Specify the output text.

Mark Pigors

University of Cologne

Chat Data

- Messages are saved in the contracts table, or in the user-defined table specified in the Chat-box dialog.
- The contracts table and the user-defined tables are the only tables in which text-variables can be saved.
- To display text variables on the screen, write “-1” in the layout field.

zTree - [contracts table]

File Edit Treatment Run Tools View ?

Period	Owner	Box	text		TimeStage
1	2	1	"message from the left-box"	-	99999
1	1	2	"message from the right-box"	-	99999
1	1	1	"another message from the left"	-	99999
1	2	2	"and another from the right"	-	99999

The subjects table

- Contains a record for each subject. It is freshly set up for each period.
- The subjects table of the previous period is available under the name "`OLDsubjects`".
- In the subjects table, the following variables are always defined by z-Tree:
  - `Period`
  - `Subject`
  - `Group`
  - `Profit`
  - `TotalProfit` – this is calculated automatically. It should not be changed manually.
  - Participate
  - LeaveStage

The globals table

- This table contains a single record, i.e. a single line. It stores values that are the same for all subjects. It is freshly set up for each period.
- The globals table of the previous period is available under the name "`OLDglobals`".
- In the globals table, the following variables are always defined by z-Tree:
  - `Period`
  - `NumPeriods`
  - `RepeatTreatment`

The summary table

- This table contains one record per period. It is not destroyed at the end of each period, as the subjects and globals table, but at the end of each treatment.
- This table is most useful for observing aggregate data of the treatment.
- In the summary table, the only variable defined by z-Tree is the variable `Period`.

Mark Pigors

University of Cologne

The contracts table

- This table is used mainly for market experiments.
- New records can be added to this table, and existing records can be changed.
- In the contracts table, the only variable defined by z-Tree is the variable `Period`.

The session table

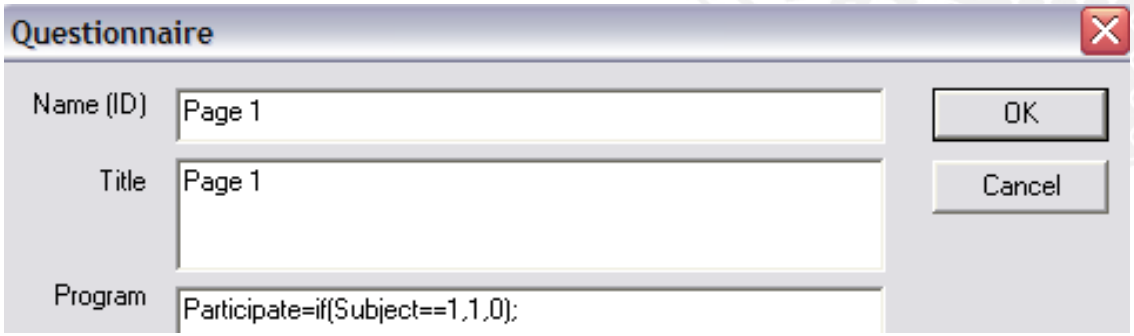
- This table contains one record per subject. This table is never destroyed during a session, as it stores the aggregate profits earned by subjects in earlier treatments.
- It can also be used to exchange information across treatments, or from treatments to questionnaires.
- In the session table, the following variables are always defined and automatically calculated by z-Tree:
  - `Subject`
  - `Final Profit`
  - `ShowUpFee`
  - `ShowUpFeeInvested`
  - `MoneyAdded`
  - `MoneyToPay`
  - `MoneyEarned`

Questionnaires – Address form

- Usually goes first.
- May/may not contain the name and the address of the subject.
- As soon as it is filled in by all subjects, the payment file is generated by z-Tree.

Questionnaires – Question forms

- Contain several questions with different possible layouts.
- Answers in questionnaires are of no consequences, no earnings can be made in questionnaires.
- Answers are saved as text and cannot be used in programs.
- The last question form of a questionnaire remains on the users' screen until you continue (with a new treatment, a new questionnaire, or simply shutting down z-leaf).
- Therefore, the final question form cannot contain a button.

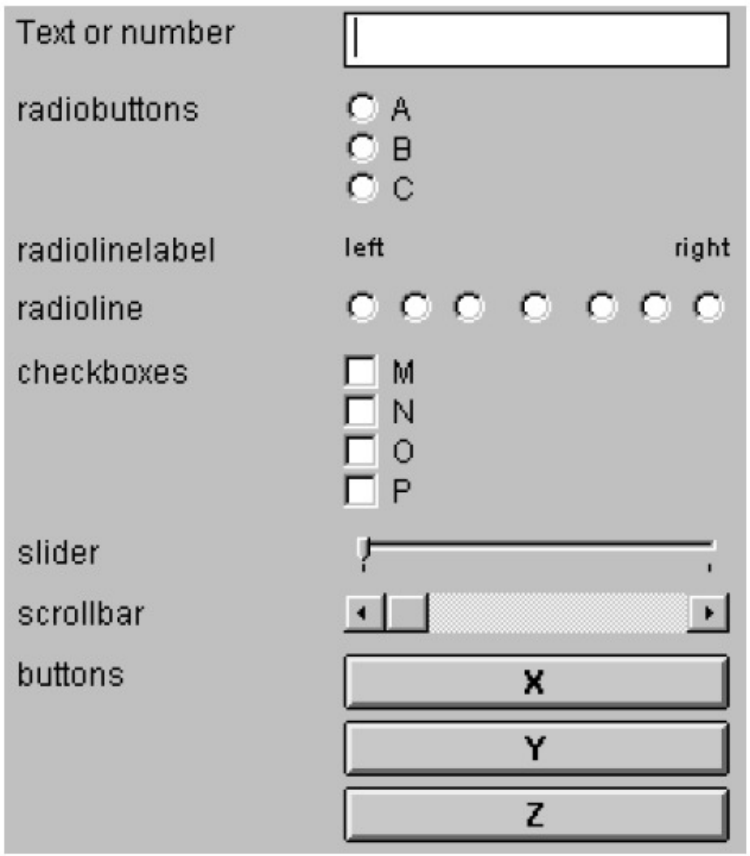


Mark Pigors

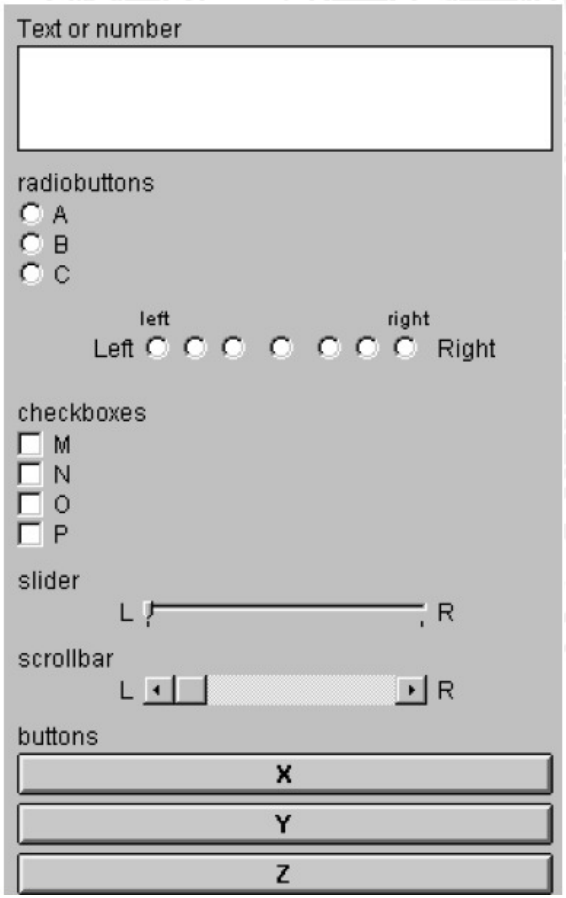
University of Cologne

Layout of the questions

- Example of normal layout:



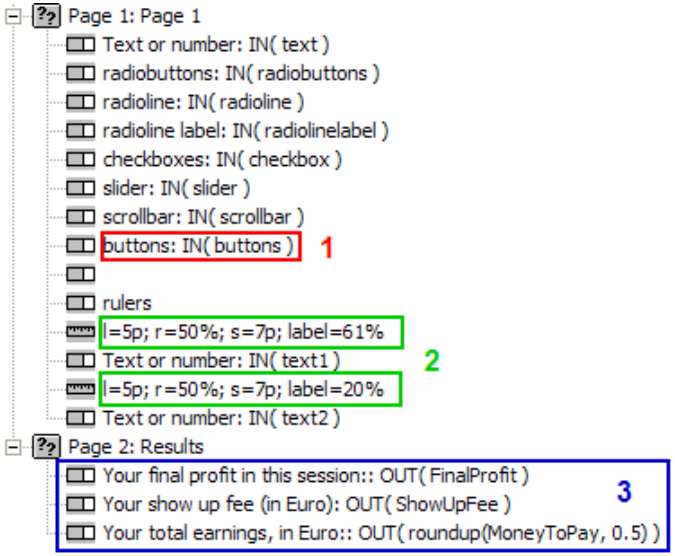
- Example of wide layout:



Mark Pigors

University of Cologne

Layout of the questions



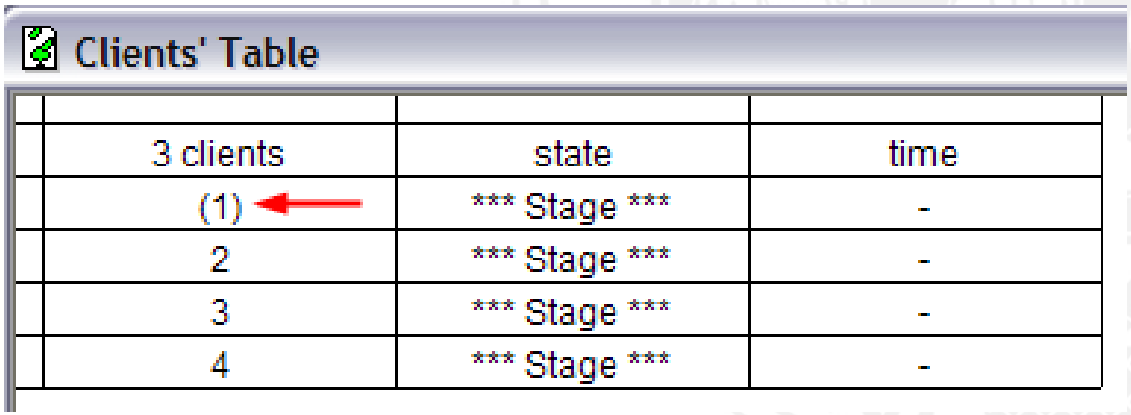
- Buttons: maximum one per question form. Clicking a button closes the question form, if possible.
- Rules: are used to set the regions where labels and questions are positioned

- In questionnaires, only variables from the session table can be retrieved.
- If you need to retrieve a specific variable from a treatment, you have to store it in the session table with a command in the treatment program (`.ztt`).

→ `questionnaire_complete.ztq`

Crash of the subject's PC I

- When a subject's PC disconnects from the server, the clients appears in parentheses in the clients table.



- First solution: Try to close the Leaf program on the subject's PC, and to restart it.

Mark Pigors

University of Cologne

Crash of the subject's PC II

- If the first solution does not work, you can try to move the subject to a different computer.
  1. Start a new PC and launch z-Leaf from there.
  2. From the client table, drag the name of the new client on the name of the client that does not work. This will release the old client, and replace the old with the new client.

Clients' Table

4 clients	state	time
(1)	*** Stage ***	-
2	*** Stage ***	-
3	*** Stage ***	-
4	*** Stage ***	-
new_client		

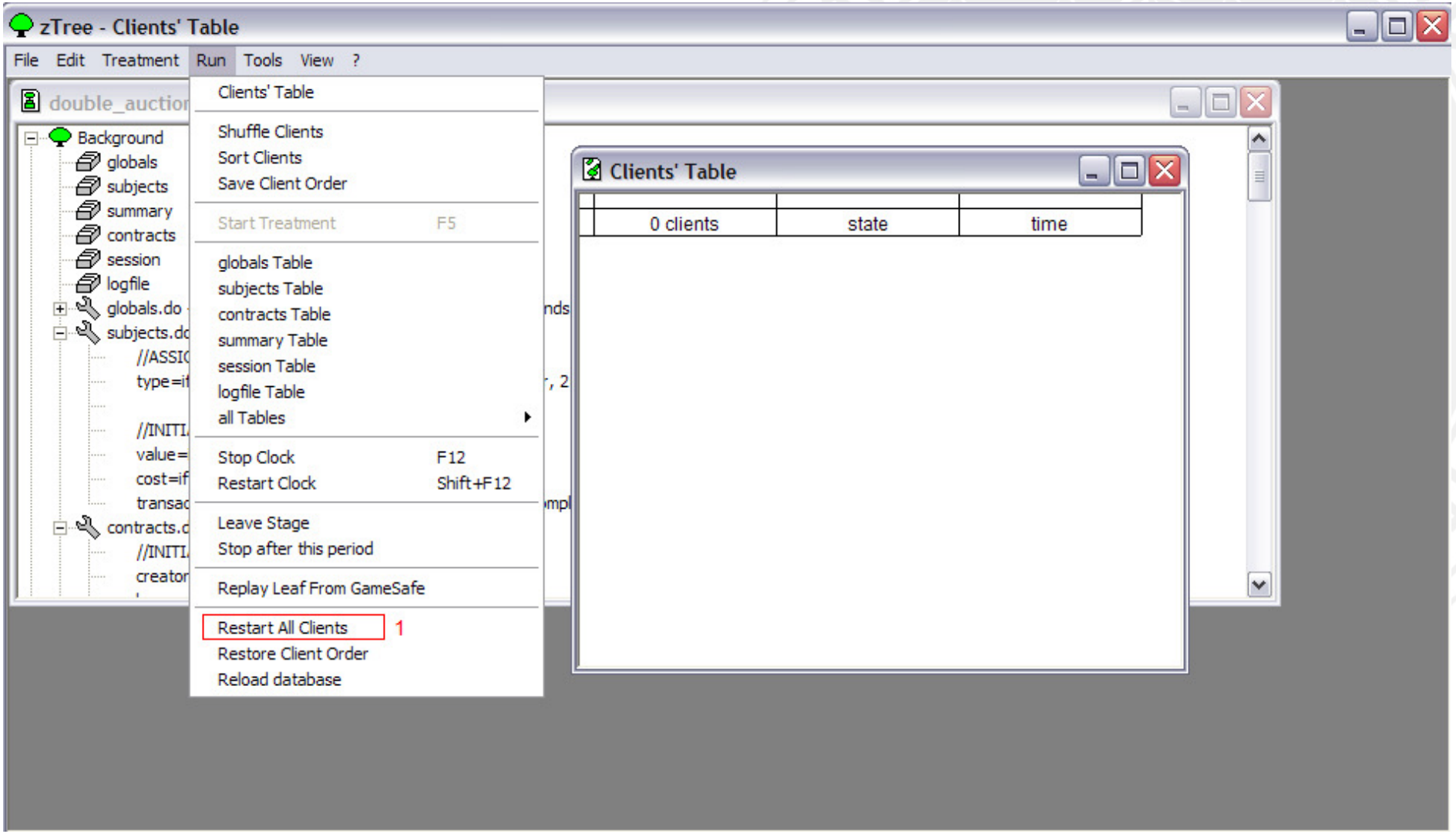
Mark Pigors

University of Cologne

Crash of z-Tree

- After a crash of z-Tree, you have to follow this procedure carefully:

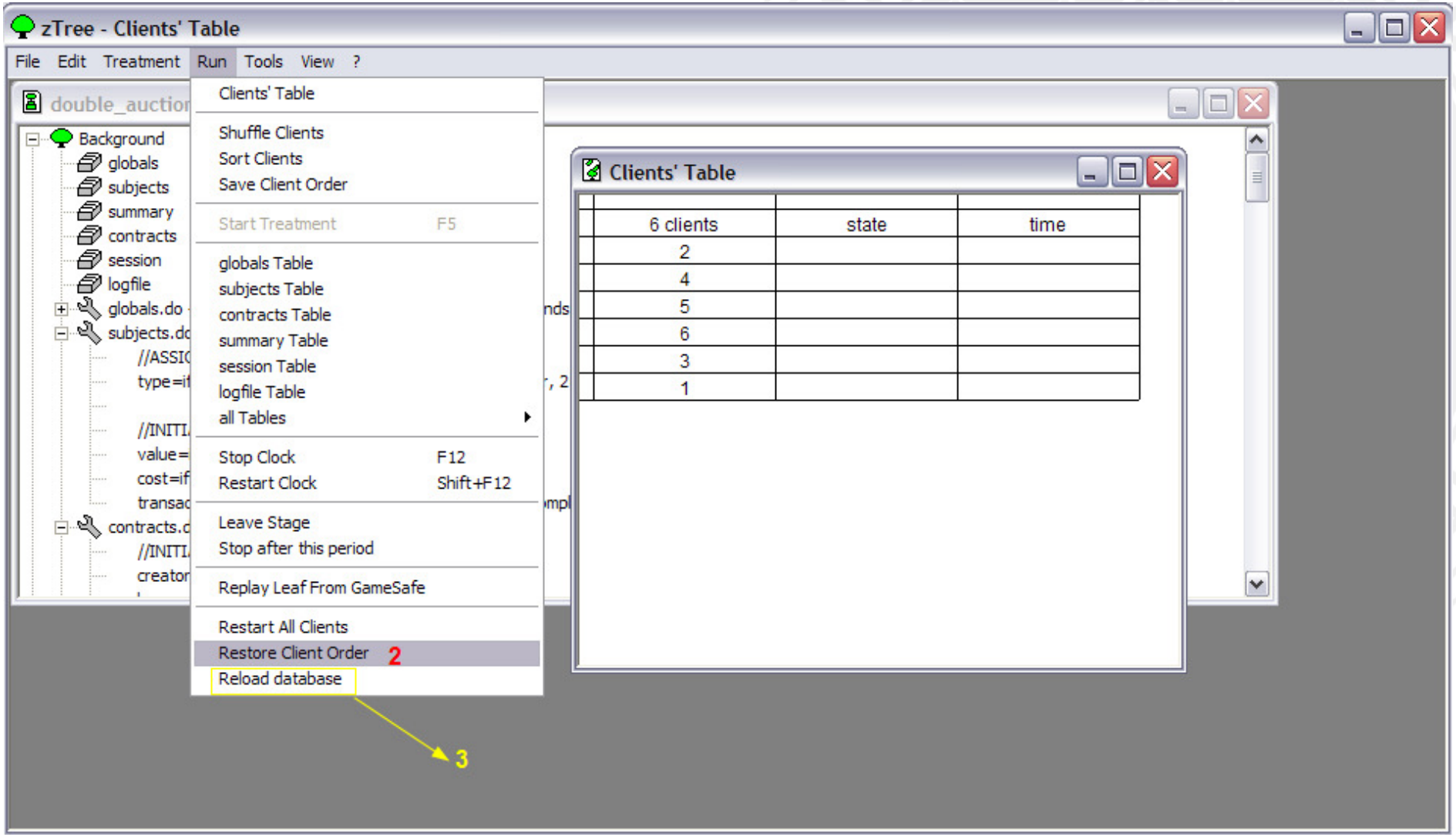
1. Restart z-tree.
2. Open the client's table.
3. Restart all clients with the menu Run , Restart all clients.
4. If no clients connect you have different options:
   - Try to restart the clients manually .
   - Wait (up to 4 minutes) and try again.
   - Shut down and restart the experimenter's PC, then follow the previous steps.
   - Start z-Tree on a different computer.



Mark Pigors

University of Cologne

- Click “Run” $\rightarrow$ “Restore Client Order” (2).
- Click “Run” $\rightarrow$ “Reload Database” (3).
- Check how many periods have been played (e.g., from the subjects table)
- Open the treatment that was running when the crash took place, and set the number of practice periods to -n (where n is the number of periods already played)
- “Click Run” $\rightarrow$ “Start treatment”.



Mark Pigors

University of Cologne

Part V

Some tips and tricks

- Standard input items need buttons to write input into the subjects table

→ `calculator.ztt`

- Other input items (slider, scrollbar, ...) can use a program to the store and show (!) the data
- Standard buttons are not needed

→ `slider.ztt`

Example: Slidertask

- Computerized real effort task based on moving sliders (Gill and Prowse)
- Task: set as many sliders (0 - 100) as possible in the “middle position” (50) in a given time

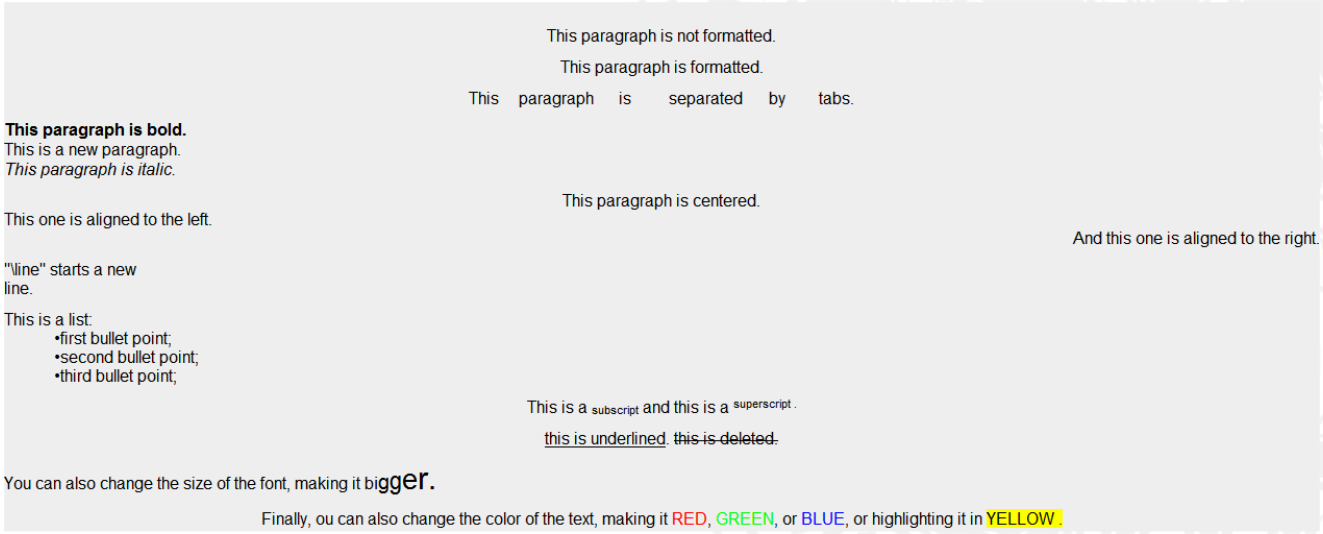
→ `GillProwseSliderExample.ztt`

Count the number of wrong answers

- Before the experiment starts, subjects have to answer control questions correctly
- You might want to know the number of mistakes

→ `control_q.ztt`

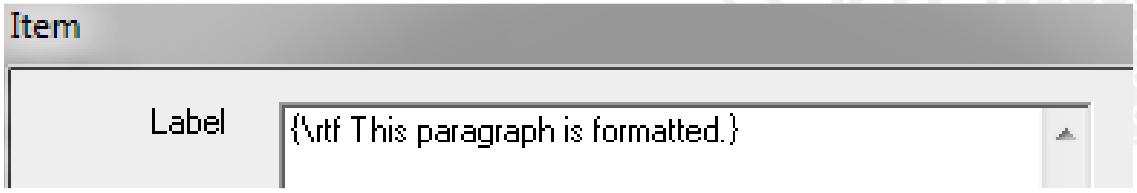
Text formatting I



- → `text_formatting.ztt`: source code for the above example,
- → `formatting_exercize.ztt`: colored text, dynamically changing size,
- → `timer.ztt`: a big, colorful timer.

Text formatting II

- See the Reference manual, at page 55.
- Text can be formatted in:
  - standard boxes
  - grid boxes
  - contract creation boxes
  - contract grid boxes
- The RTF format begins with `{\rtf` (with a blank space at the end), and ends with }.



Mark Pigors

University of Cologne

RTF formatting instructions

\tab	tabulator	\par	new paragraph
\line	new line	\bullet	thick dot (for lists)
\ql	aligned to the left	\fsn	font size in half dots
\b	bold	\b0	not bold anymore
\i	italics	\i0	not italics anymore
\qc	centered	\qr	aligned to the right
\sub	subscript	\super	superscript
\ul	underlined	\ul0	not underlined
\strike	crossed through		

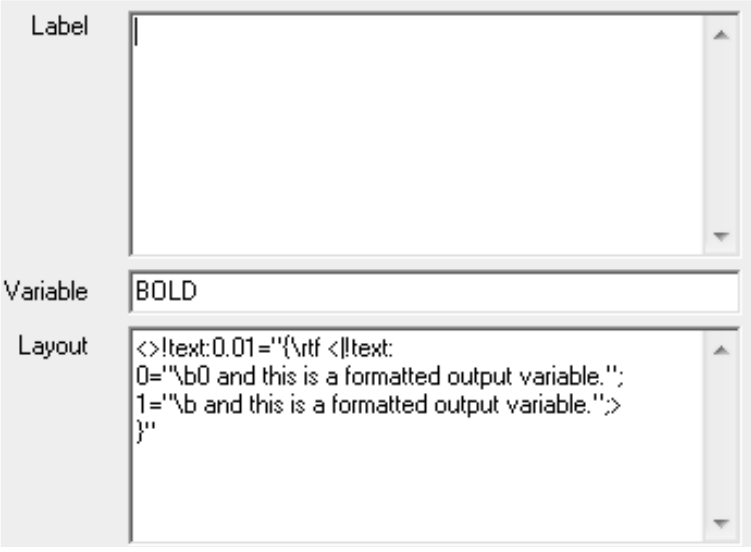
Mark Pigors

University of Cologne

Conditional formatting

- The insertion of variables is carried out before the interpretation of RTF instructions.
- This makes conditional formatting possible, as in the following example: when the `BOLD` variable is 1, “hallo” should be shown in boldface, otherwise it is shown in plain text.

```
<> {\rtf <BOLD|!text:  0="" ; 1="\b";> hallo}
```



- Labels do not change dynamically when the value of the variable changes.
- Put formatting in the layout field.

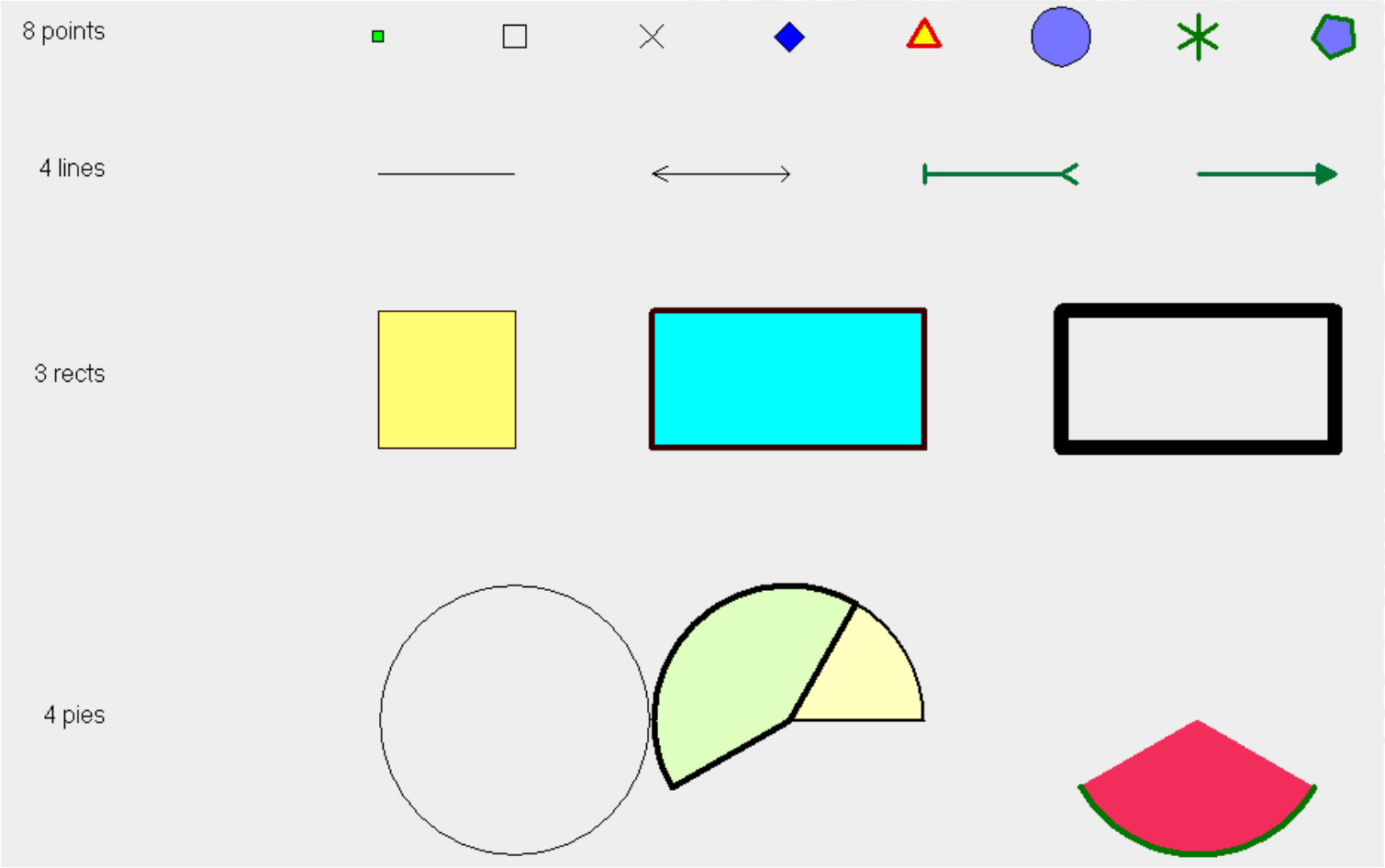
→ `conditional_formatting.ztt`

Text color

- With RTF formatting, you have to define first the colors you are going to use, with the instruction `\colortbl`.

```
{\rtf {\colortbl;
\red0\green0\blue255;
\red255\green0\blue0;
\red0\green255\blue0;
\red255\green255\blue255;
}
\cf1 this is BLUE
\cf2 this is RED
\cf3 this is GREEN
\cf4 this is WHITE
}
```


Plots and vectorial graphics



→ z-Tree wiki, → `plotitems.ztt`

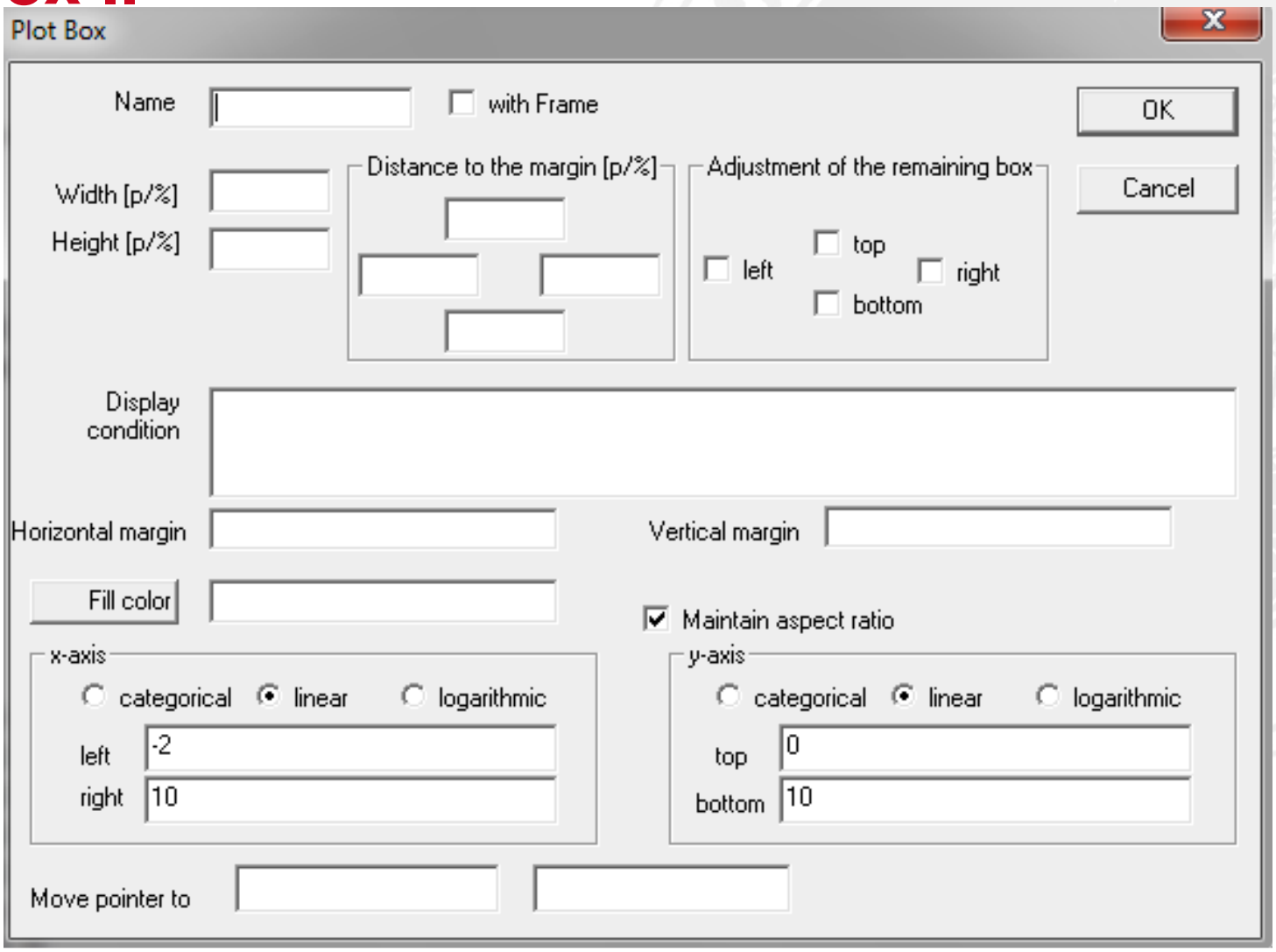
Mark Pigors

University of Cologne

Plot Box I

- The Plot Box sets up a coordinate system and allows to display Plot Items.
- Plot items are drawn sequentially. So items more down in the list of items can cover item more up in the list.
- The Plot box is opaque. It covers all boxes below.
- The plot box is a Box. It can be contained in Screens and Container Boxes. It can contain Plot Items.
- To create a new Plot Box, click on “Treatment” $\rightarrow$ “New Box” $\rightarrow$ “New Plot Box...”.

Plot Box II



Mark Pigors

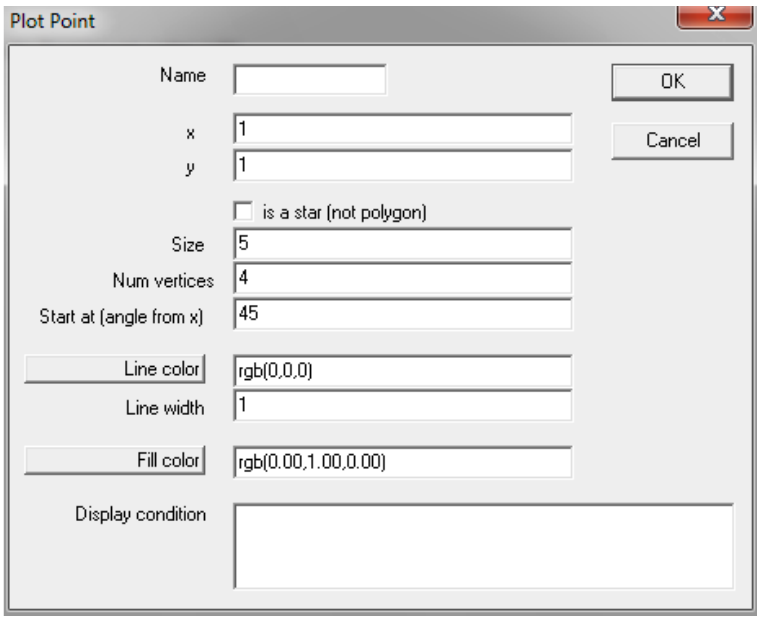
University of Cologne

Plot Items I

- In a plot box different types of graphical elements (points, lines, rectangles,...) can be displayed. These elements are called plot items.
- They can be placed into the coordinate system that is set up in the plot box.
- The position of the plot items is thus defined with reference to this coordinate system.
- Furthermore, these elements have features, for instance the lines can be thicker or thinner and drawn in different colours.
- The size used to describe these features is always in screen pixels.

Points

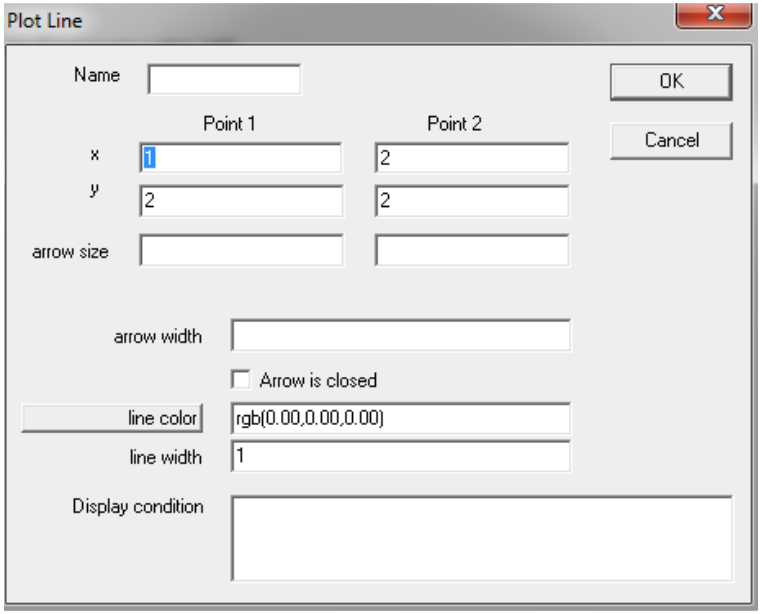
- To add a Point, click “Treatment” → “Graphics” → “New Point...”.
- This displays a point either as a regular polygon or as a star consisting of line elements.



- Plot points are plot items.
- Plot points can be contained in plot boxes and in plot graphs.
- `Start at (angle from x)`: Angle where the first point is drawn. The angle is measured from the x-axis in counter clockwise direction.

Lines

- To add a Line, click “Treatment” → “Graphics” → “New Line...”.
- This displays a connection between two points as straight line or an one-sided or two-sided arrow.
- Plot lines are plot items. Plot lines can be contained in plot boxes and in plot graphs.



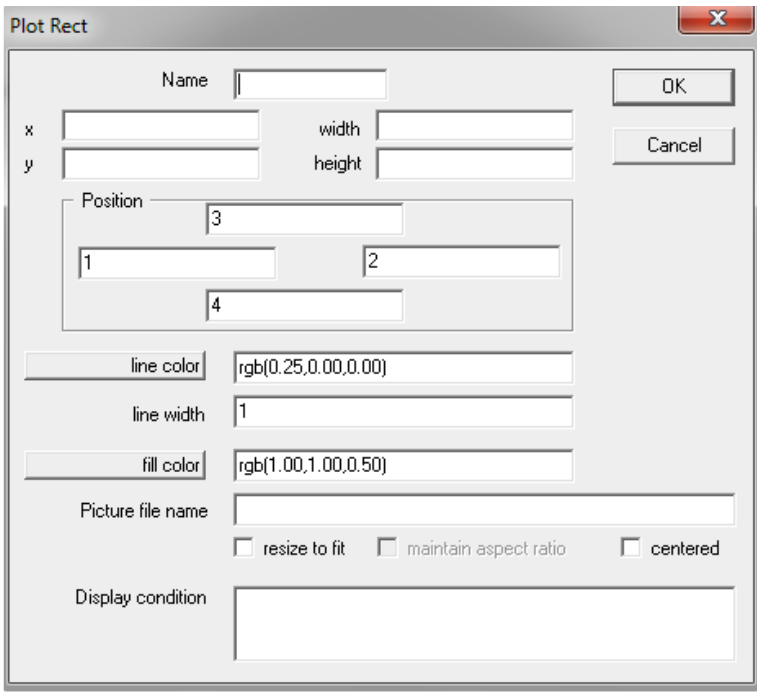
- `Arrow size`: Arrow size in the direction of the line. Negative: arrow points from the outside to the line.
- `Arrow width`:Total size of the arrow orthogonal to direction of the line.
- `Arrow is closed`: Not checked: arrow consists of two lines. Checked: the arrow is a filled triangle.

Mark Pigors

University of Cologne

Rectangles

- To add a Rectangle, click “Treatment” → “Graphics” → “New Rect...”. This displays a rectangle.
- Plot rectangles are plot items. Plot rectangles can be contained in plot boxes and in plot graphs.



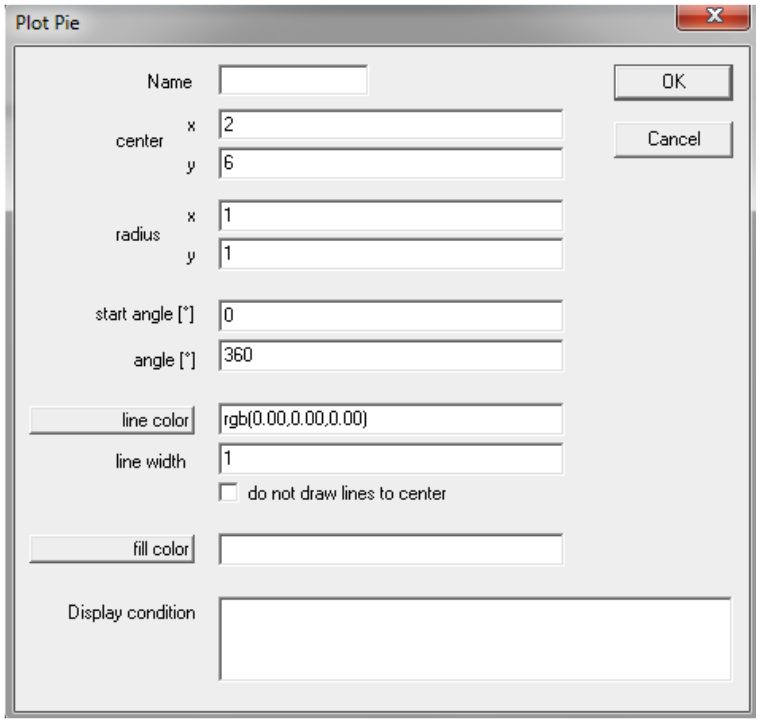
- `x/y`: coordinates of the center of the rectangle.
- `width/height` and `position`: defined with respect to the coordinate system that is set up in the plot box.

Mark Pigors

University of Cologne

Pies

- To add a Pie Plot, click “Treatment” → “Graphics” → “New Pie...”.
- This displays a ellipse or a piece of a ellipse.
- Plot pies are plot items. Plot pies can be contained in plot boxes and in plot graphs

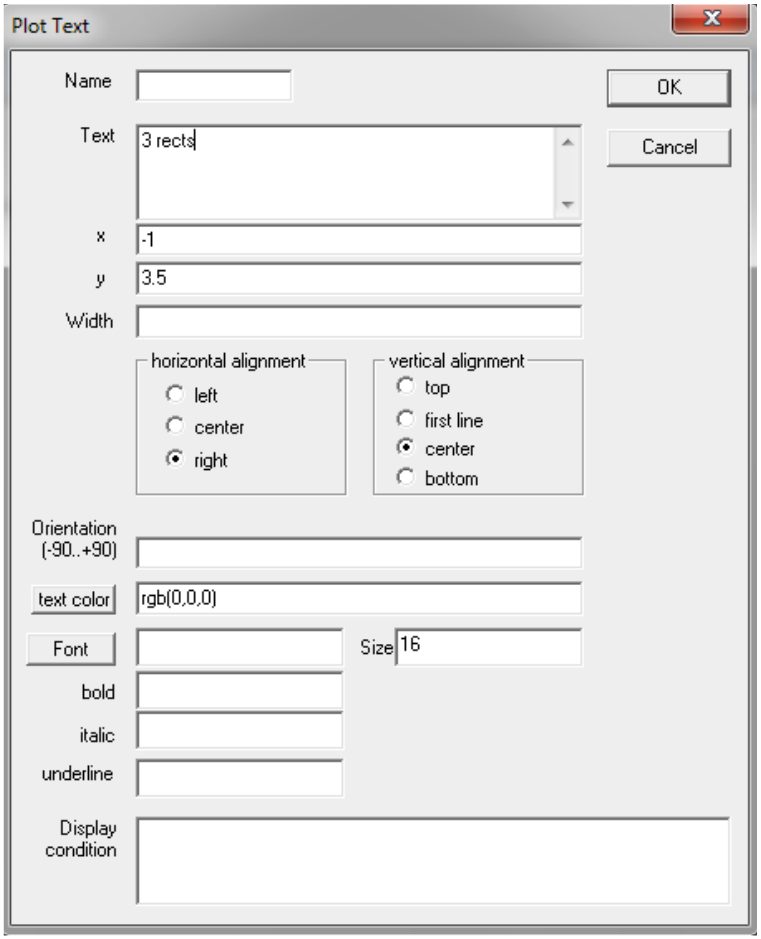


- `start angle`: Angle in degrees where the segment starts.
- Measured from the x-axis in counter clockwise direction.
- `angle`: Size of the segment in degrees.

→ `spin_the_wheel.ztt`: a rotating pie, representing the outcome of a lottery.

Points

- To add text to a plot, click “Treatment” → “Graphics” → “New Plot Text...”.



- Plot Texts are plot items.
- Plot Texts can be contained in plot boxes and in plot graphs.
- `x/y`: position of the text.
- `Horizontal/vertical alignment`: Defines which border of the text is determined by the text position.
- `Orientation`: Does not work yet.

Mark Pigors

University of Cologne

Graphs

- To add graph to a plot, click “Treatment” → “Graphics” → “New Plot Graph...”.
- The plot graph is used to display polygons or series of data.
- The plot graph displays series of records in a table.
- Plot items that are placed into the plot graph are drawn for every record and the variables used in these items are evaluated in this record.
- Plot graphs can be nested, i.e., placed into each other. If they are nested the data from different records can be accessed with the scope operator.
- Plot graphs are plot items. Plot graphs can be contained in plot boxes and in plot graphs.
- They can contain plot items.

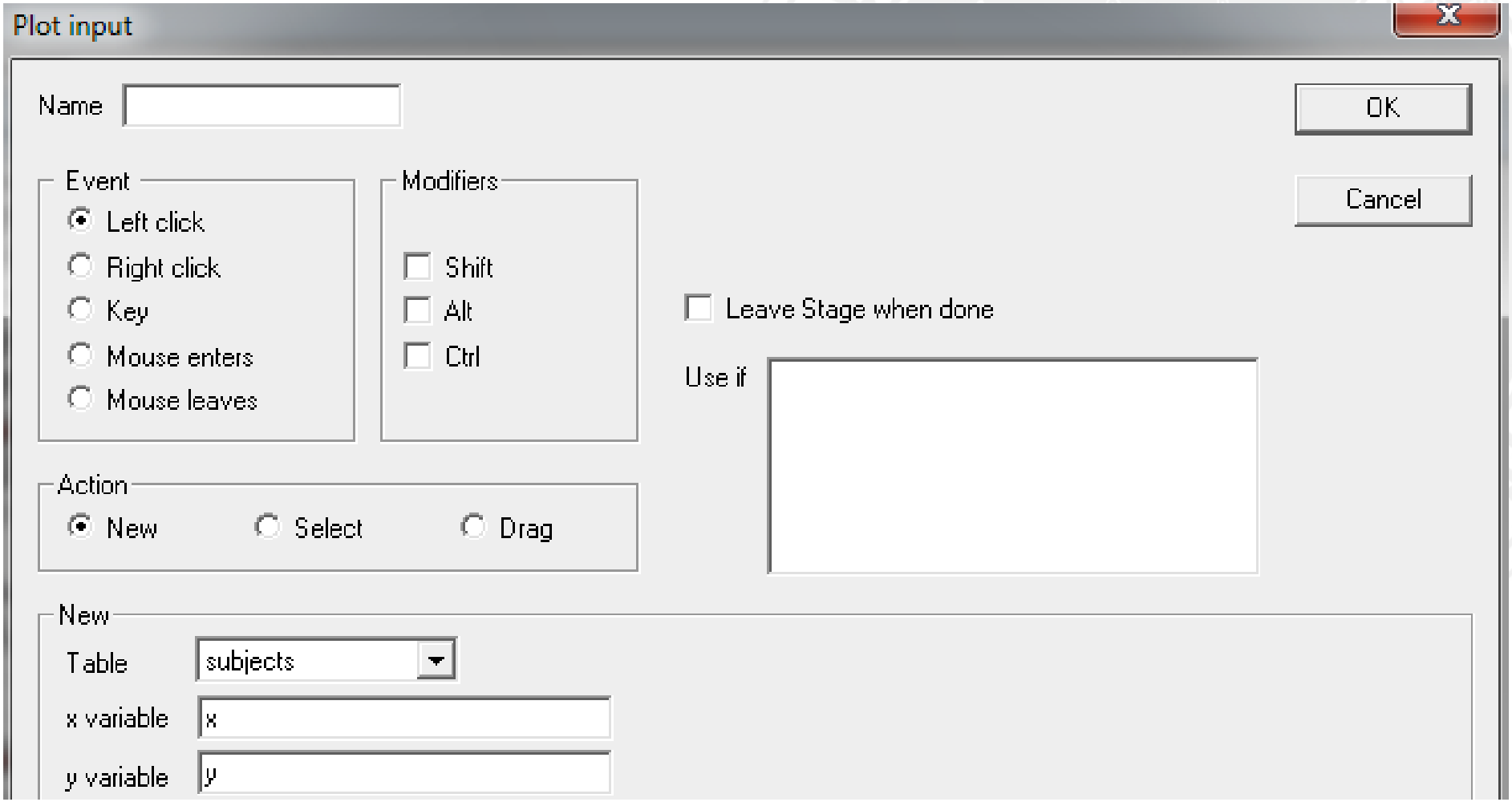
→ `boxplotdemo.ztt`

Plot input I

- To add plot input to a plot, click “Treatment” → “Graphics” → “New Plot Input...”.
- Plot inputs can be placed into plot boxes and plot items.
- They can contain checkers and programs.
- With plot input item, you define how subjects interact in a plot box;
- Subjects can click at a new position, they can select one of several objects on the screen, and they can even drag objects around.

→ `movepointdemo.ztt`

Plot input II



Mark Pigors

University of Cologne

Multimedia box I

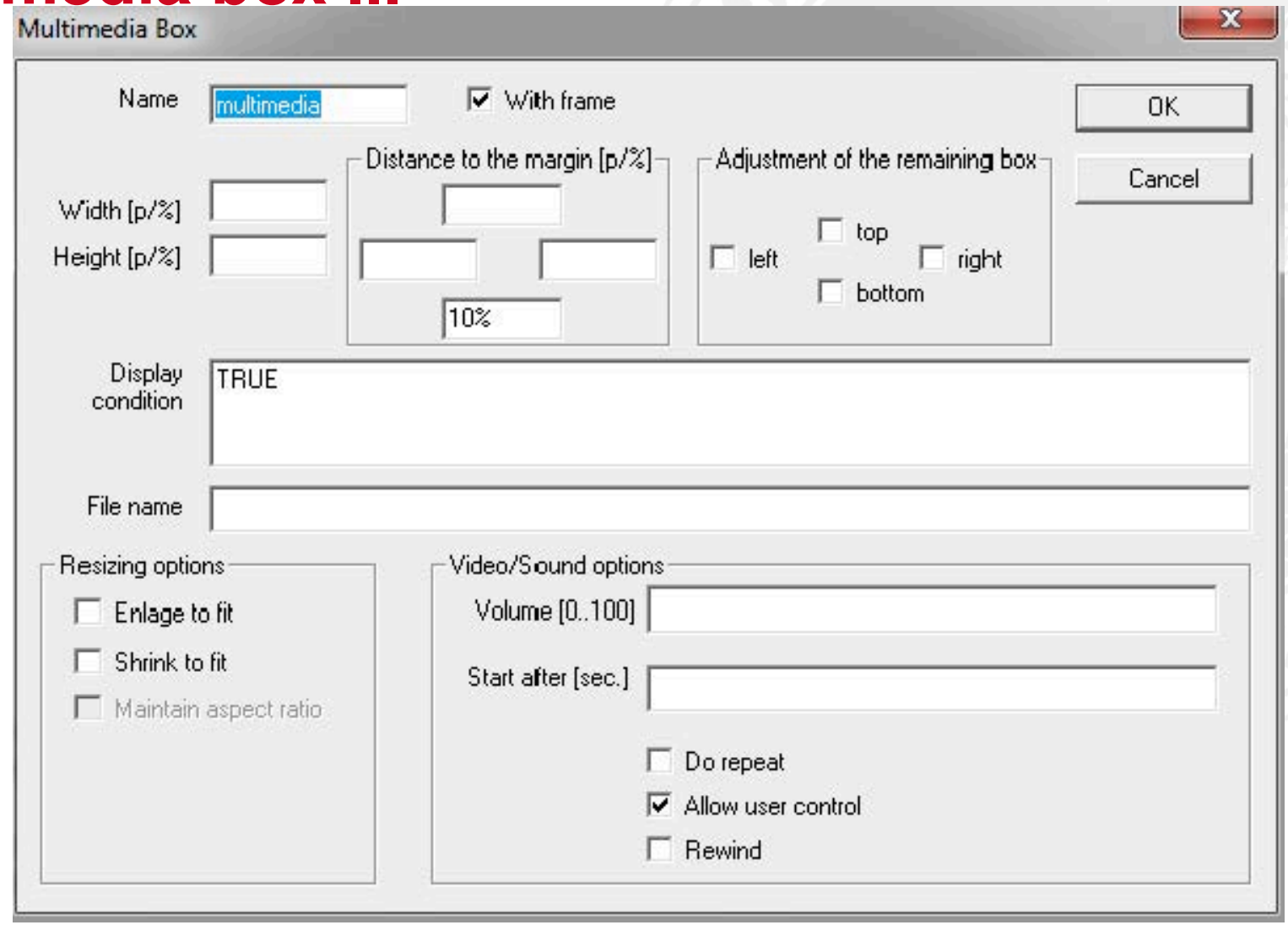
→ z-Tree wiki

- The multimedia box can contain images, movies or sounds.
- The content, i.e. the image, movie or sound of the multimedia box is stored in an external file. This file must be accessible at the client's computer.
- The best way to achieve this is to put the files on a server share and map this share to the same drive letter on each of the client computers.
- Movies and sound are played as long as the box is visible. This allows to to turn it on and off.

Multimedia box II

- The following formats should work on a standard installation:
  - Image: jpg, gif, png, bmp.
  - Movie: mpg, avi.
  - Audio: wav, mp3.
- The multimedia box is a Box. It can be contained in Screens and Container Boxes. It cannot contain other elements.

Multimedia box III



Mark Pigors

University of Cologne

Slide-show I

→ `slideshow.ztt`

- From z-Tree version 3.3.0 it is possible to include a slide-show in the treatment, i.e. a sequence of pictures displayed in sequence.
- To create a slide-show, the following procedure should be followed:
  1. In the Background, create a new Table and name it “slides” (or as you like).
  2. In the Background, create a new Program, running on the globals table. In this program, write: `slides.new`. This simply generates a new empty record in the table “slides”.
  3. In the Active Screen of a stage, click on “Treatment” → “New Box” → “New Slide Show”
  4. Within the slide-show, click on “Treatment” → “Slide Show” → “New Slide Sequence”

Slide-show II

5. Within the slides sequence, click on “Treatment” → “Slide Show” → “New Slide”
6. Within the slide, click on “Treatment” → “New Box” → “New Plot Box”
7. Within the plot box, click on “Treatment” → “Graphics” → “New rect..”

External programs

- From z-Tree version 3.3.0 it is possible to call external programs, from z-Tree or from z-Leaf.
- In the Background, or within a stage, click on “Treatment” → “New External Program”

Advanced programming examples

- Maria Bigoni's advanced examples show some possibilities of z-Tree:
  - Dutch auction (graphical version)
  - English auction (graphical version)
  - A real effort task: pick the right color

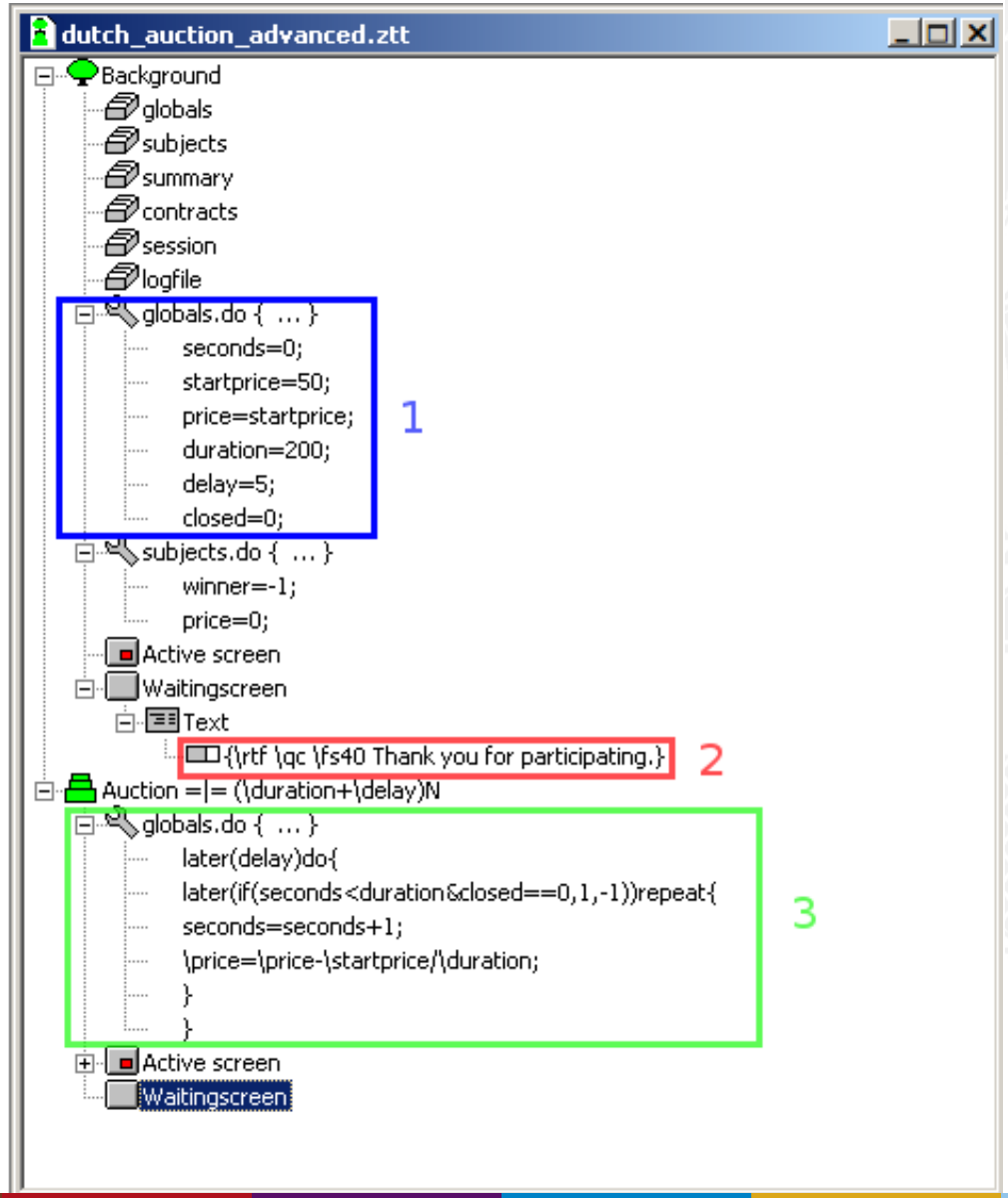
Graphical Dutch auction

Mark Pigors

University of Cologne

1. Define the global variables
2. Set the message for the waiting screen
3. Let the clock run, using:
   - the `later()do` statement
   - the `later()repeat` statement

→ `dutch_auction_advanced.ztt`



Mark Pigors

University of Cologne

- The clock is based on:
  - Plot Pie
  - Plot Point
  - Plot Text
- For the Input:
  - Transform a Rectangle in a button, by adding a plot input.
  - Add a program to trigger the consequences of the subject's action.
- Use a Multimedia box to insert the picture of the object of the auction
- Final message:
  - Use the Display condition to show different messages to the winner of the auction and to the other subjects.
  - With a program within a plot input, you let the subjects leave the stage when they click on the final message.

Graphical English auction

Mark Pigors

University of Cologne

- Price increases in time
- Subject's action: leave the auction
- plot on the right, showing the number of remaining participants, and the time when each of the others left the auction:
  - save the number of remaining subjects at each point in time in a user defined `table` or in the `contracts table`
  - plot the content of this table using a graph, which is one of the `plot items`

→ `english_auction_ advanced.ztt`

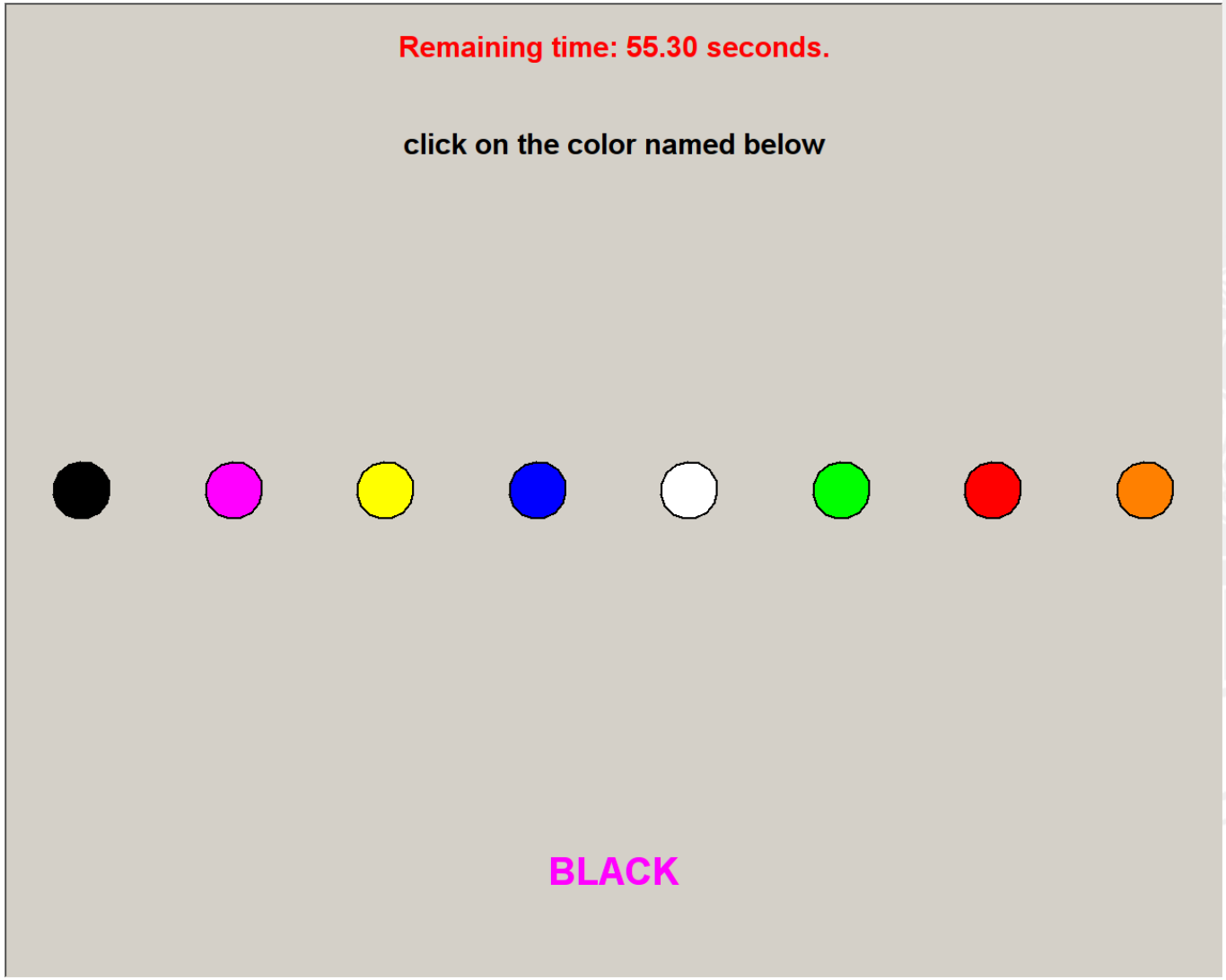
Mark Pigors

University of Cologne

Real effort task: Pick the right color

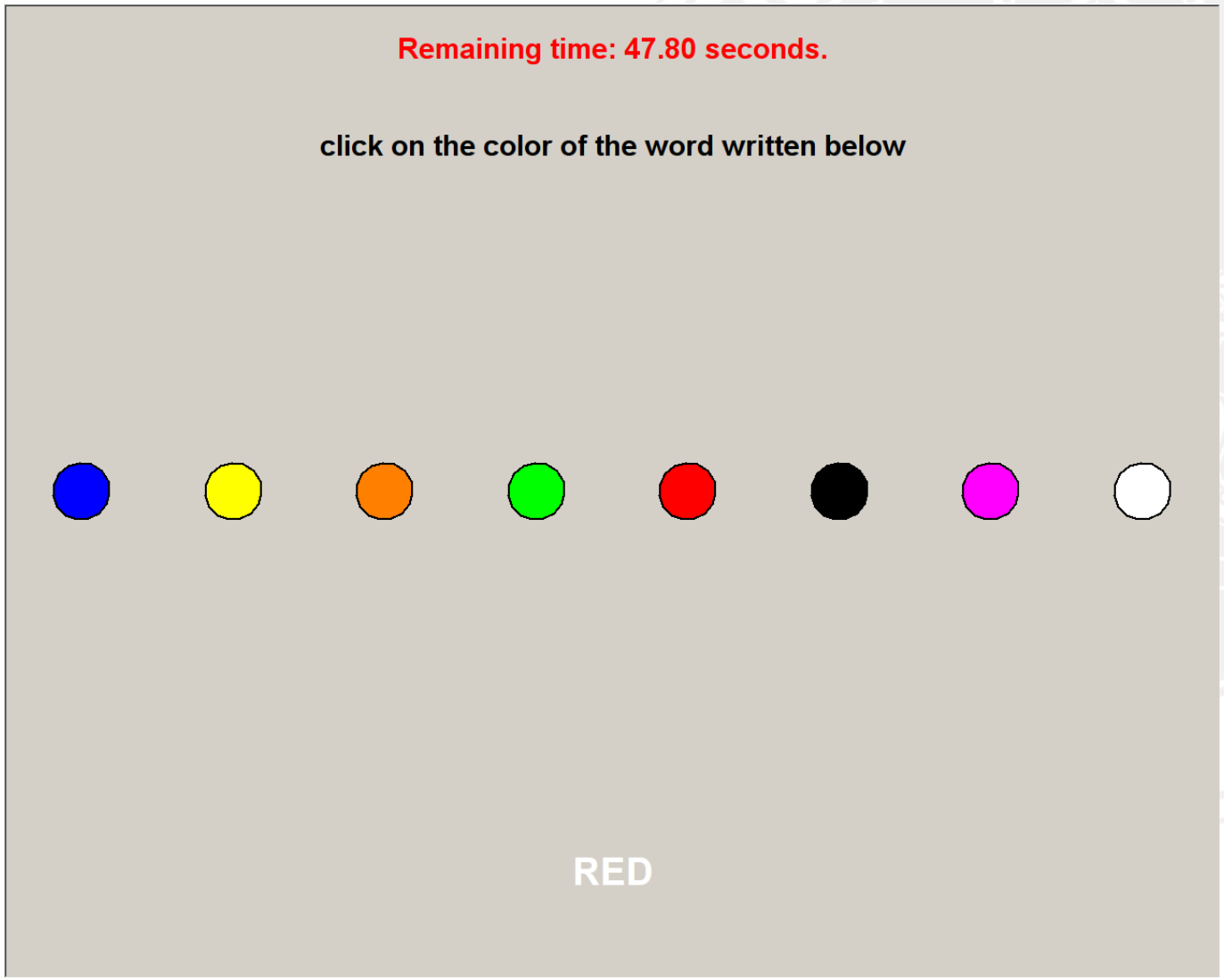
- The task:
  - correctly answer as many questions as possible
  - in a given time interval (60 seconds)
- Two different questions:
  - click on the *color of the word* written on the screen
  - click on the *color corresponding to the word* written on the screen
- To answer:
  - subject must click on one of eight alternative colors, presented in random order on the screen

→ `colors.ztt`



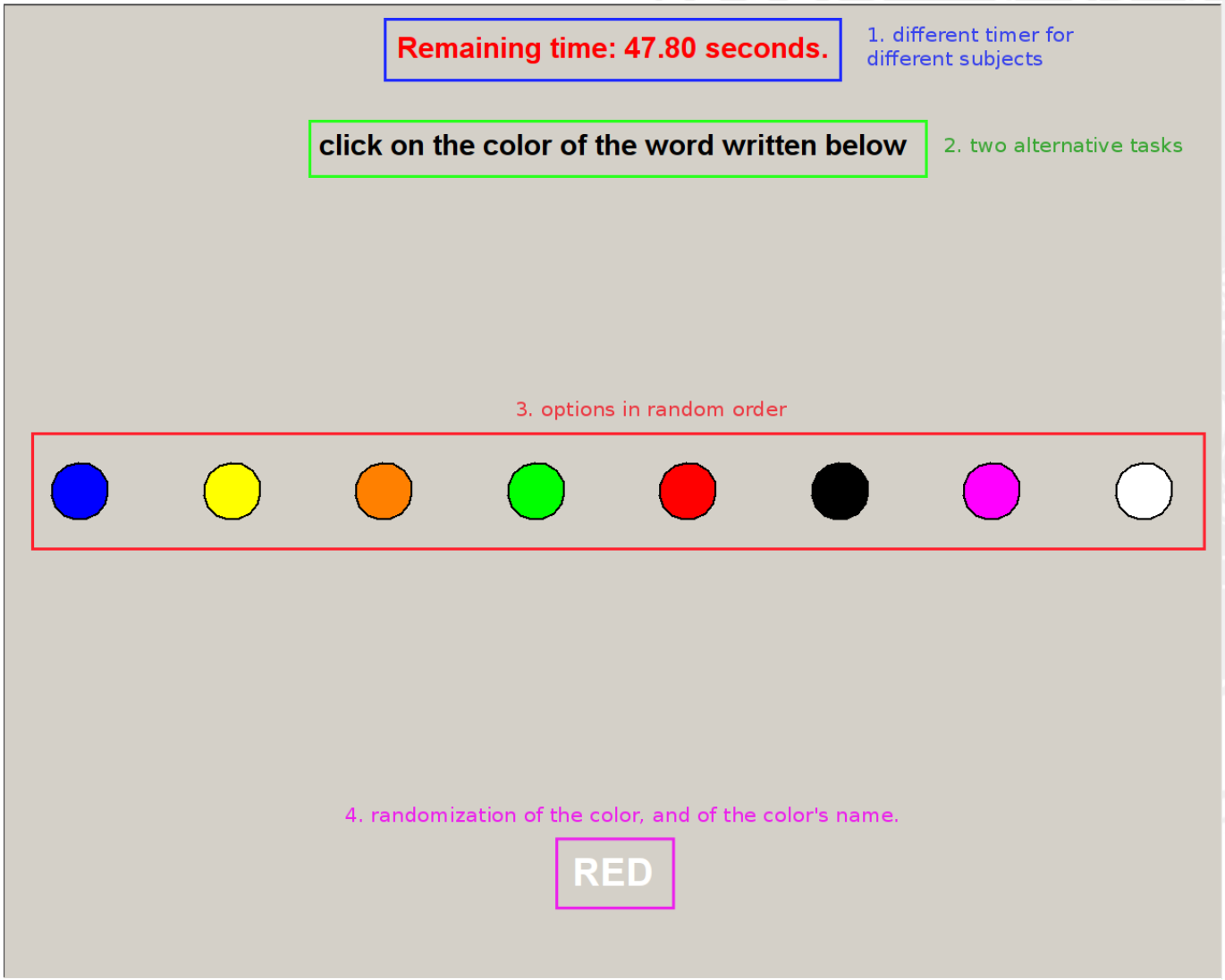
Mark Pigors

University of Cologne



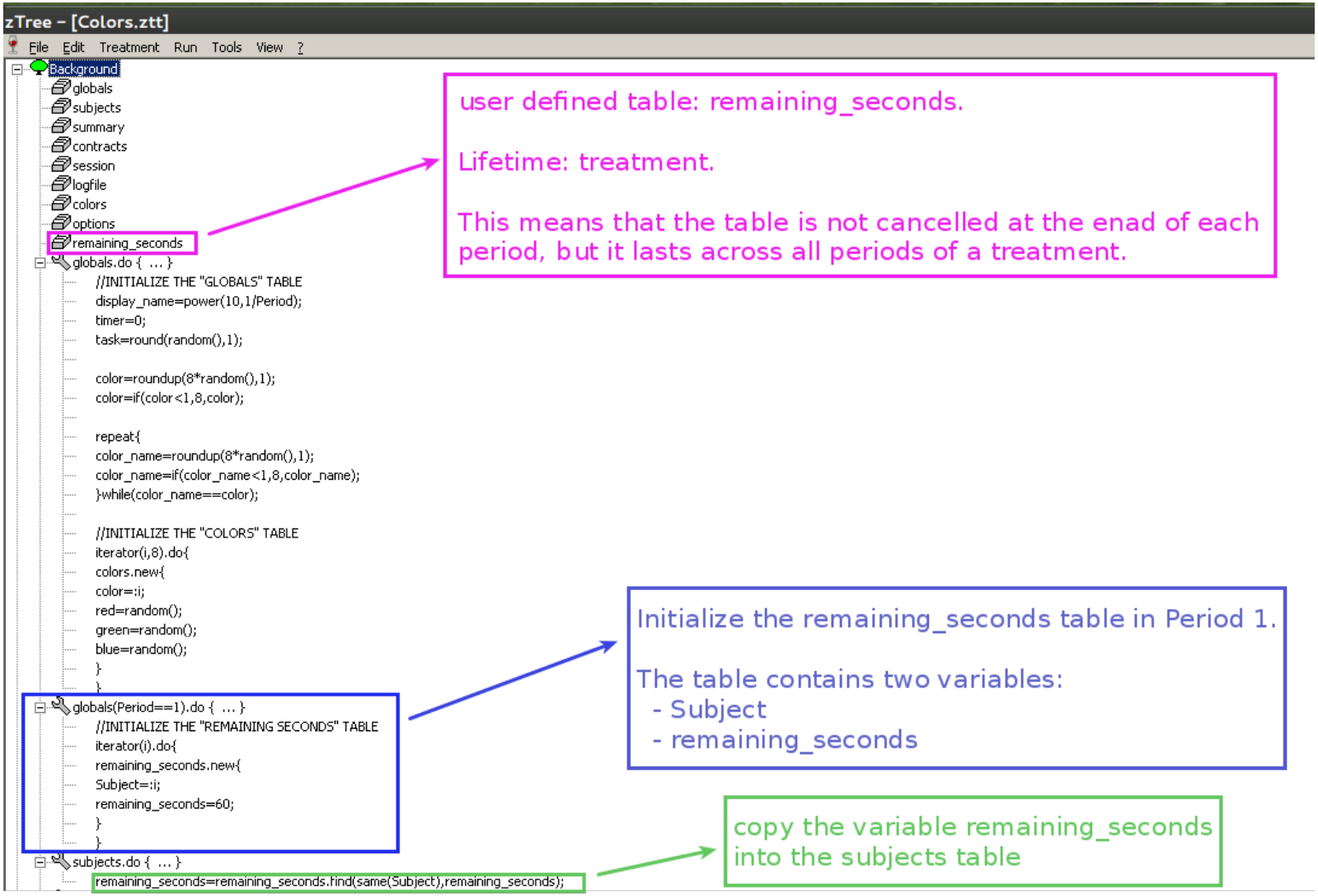
Mark Pigors

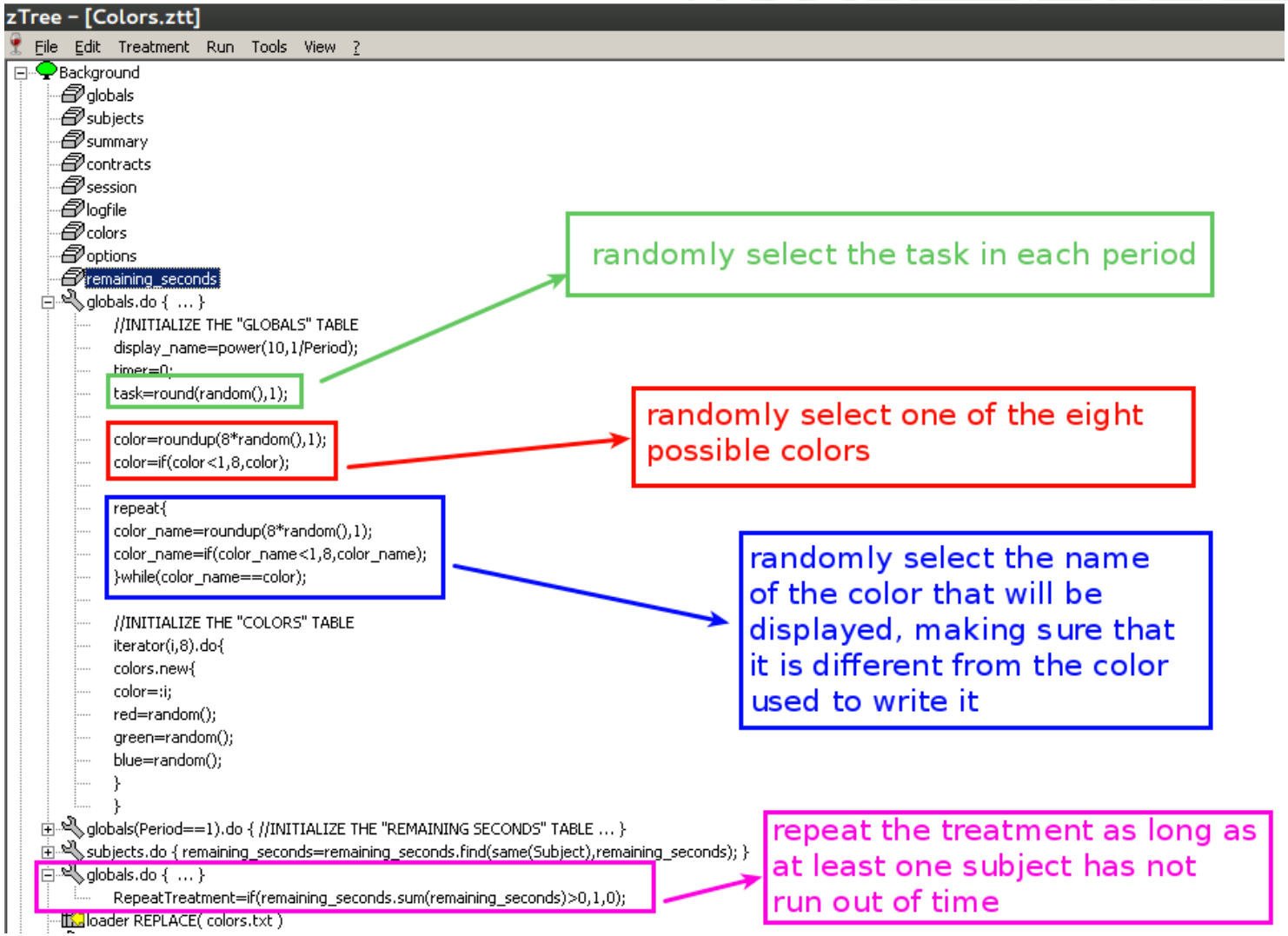
University of Cologne

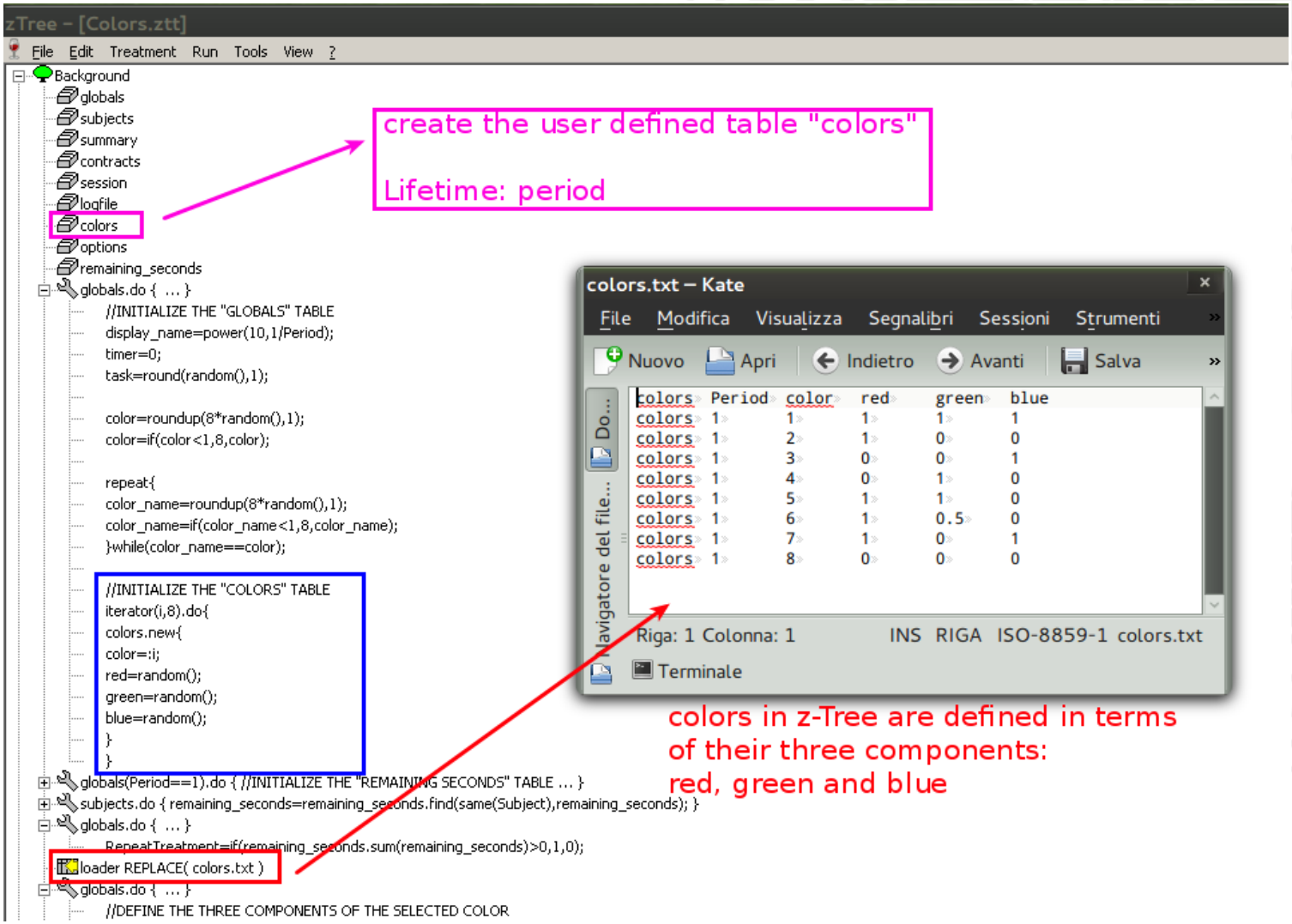


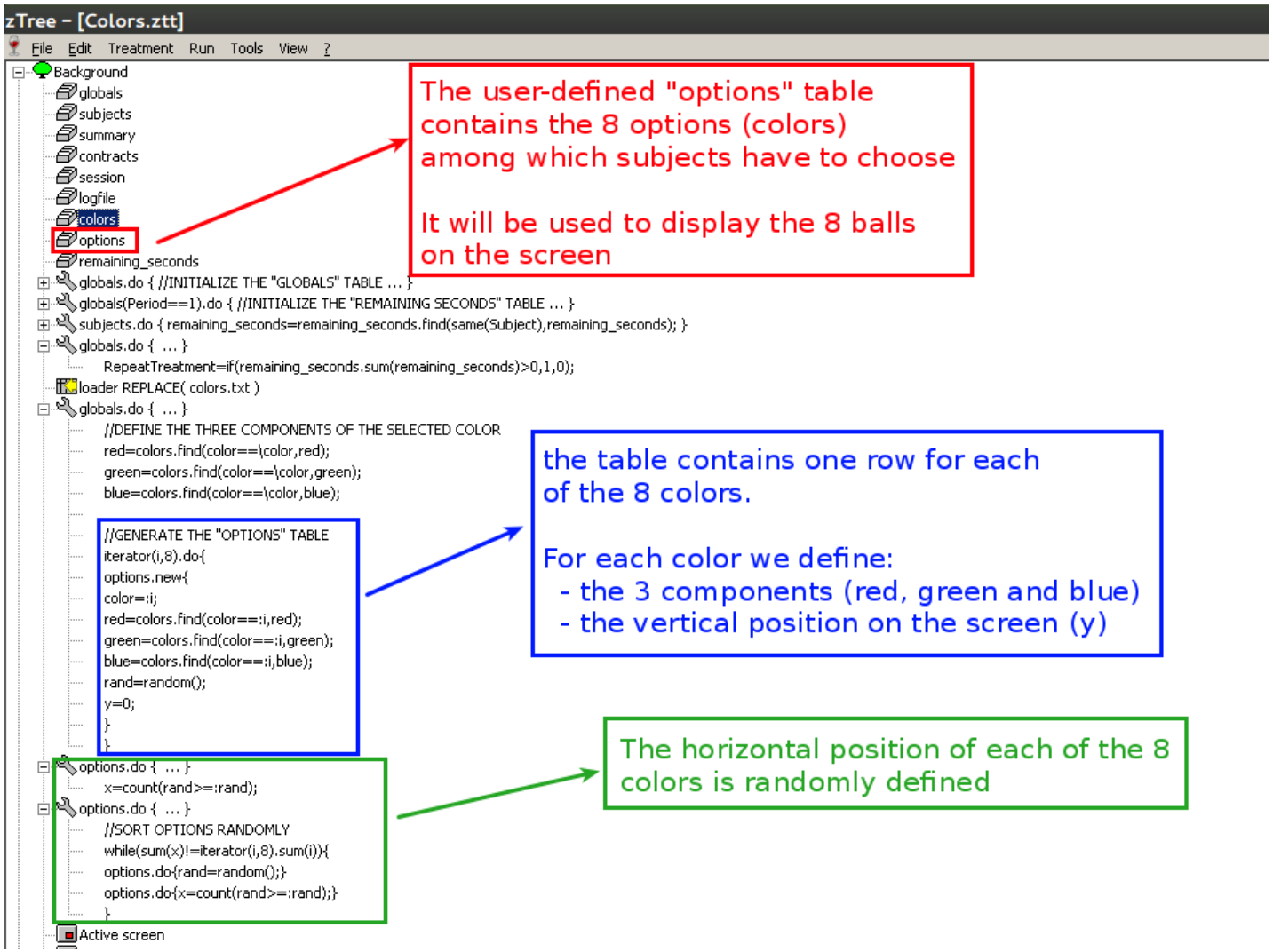
Mark Pigors

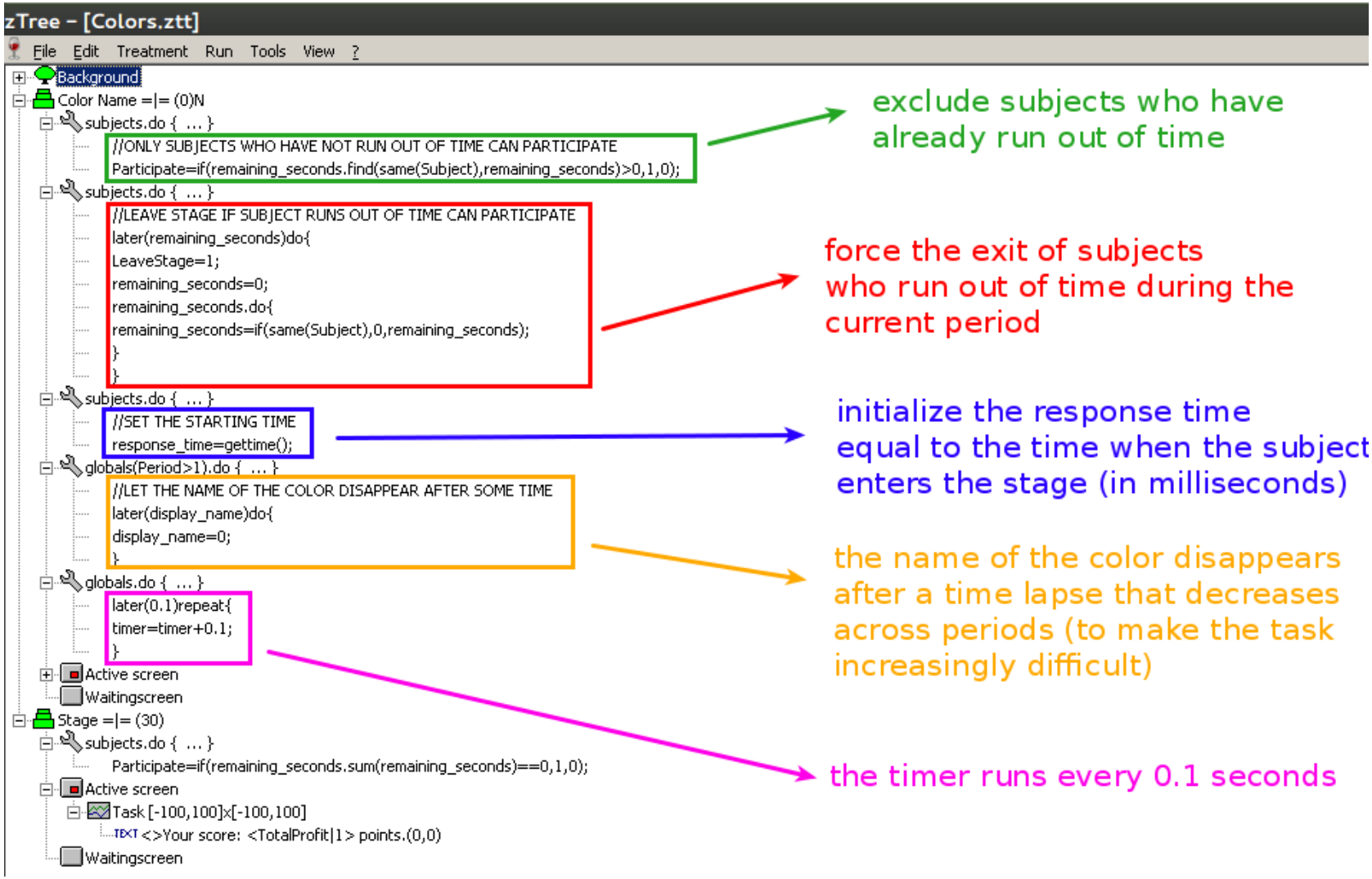
University of Cologne


zTree – [Colors.ztt]
File Edit Treatment Run Tools View ?
Background
globals
subjects
summary
contracts
session
logfile
colors
options
remaining_seconds
globals.do { ... }
//INITIALIZE THE "GLOBALS" TABLE
display_name=power(10,1/Period);
timer=0;
task=round(random(),1);
color=roundup(8*random(),1);
color=if(color<1,8,color);
repeat{
color_name=roundup(8*random(),1);
color_name=if(color_name<1,8,color_name);
}while(color_name==color);
//INITIALIZE THE "COLORS" TABLE
iterator(i,8).do{
colors.new{
color=:i;
red=random();
green=random();
blue=random();
}
}
globals(Period==1).do { ... }
//INITIALIZE THE "REMAINING SECONDS" TABLE
iterator(i).do{
remaining_seconds.new{
Subject=:i;
remaining_seconds=60;
}
}
subjects.do { ... }
remaining_seconds=remaining_seconds.find(same(Subject),remaining_seconds);
user defined table: remaining_seconds.
Lifetime: treatment.
This means that the table is not cancelled at the enad of each period, but it lasts across all periods of a treatment.
Initialize the remaining_seconds table in Period 1.
The table contains two variables:
- Subject
- remaining_seconds
copy the variable remaining_seconds into the subjects table


zTree – [Colors.ztt]
File Edit Treatment Run Tools View ?
Background
globals
subjects
summary
contracts
session
logfile
colors
options
remaining_seconds
globals.do { ... }
//INITIALIZE THE "GLOBALS" TABLE
display_name=power(10,1/Period);
timer=0;
task=round(random(),1);
color=roundup(8*random(),1);
color=if(color<1,8,color);
repeat{
color_name=roundup(8*random(),1);
color_name=if(color_name<1,8,color_name);
}while(color_name==color);
//INITIALIZE THE "COLORS" TABLE
iterator(i,8).do{
colors.new{
color=:i;
red=random();
green=random();
blue=random();
}
}
globals(Period==1).do { //INITIALIZE THE "REMAINING SECONDS" TABLE ... }
subjects.do { remaining_seconds=remaining_seconds.find(same(Subject),remaining_seconds); }
globals.do { ... }
RepeatTreatment=if(remaining_seconds.sum(remaining_seconds)>0,1,0);
loader REPLACE(colors.txt)
randomly select the task in each period
randomly select one of the eight possible colors
randomly select the name of the color that will be displayed, making sure that it is different from the color used to write it
repeat the treatment as long as at least one subject has not run out of time

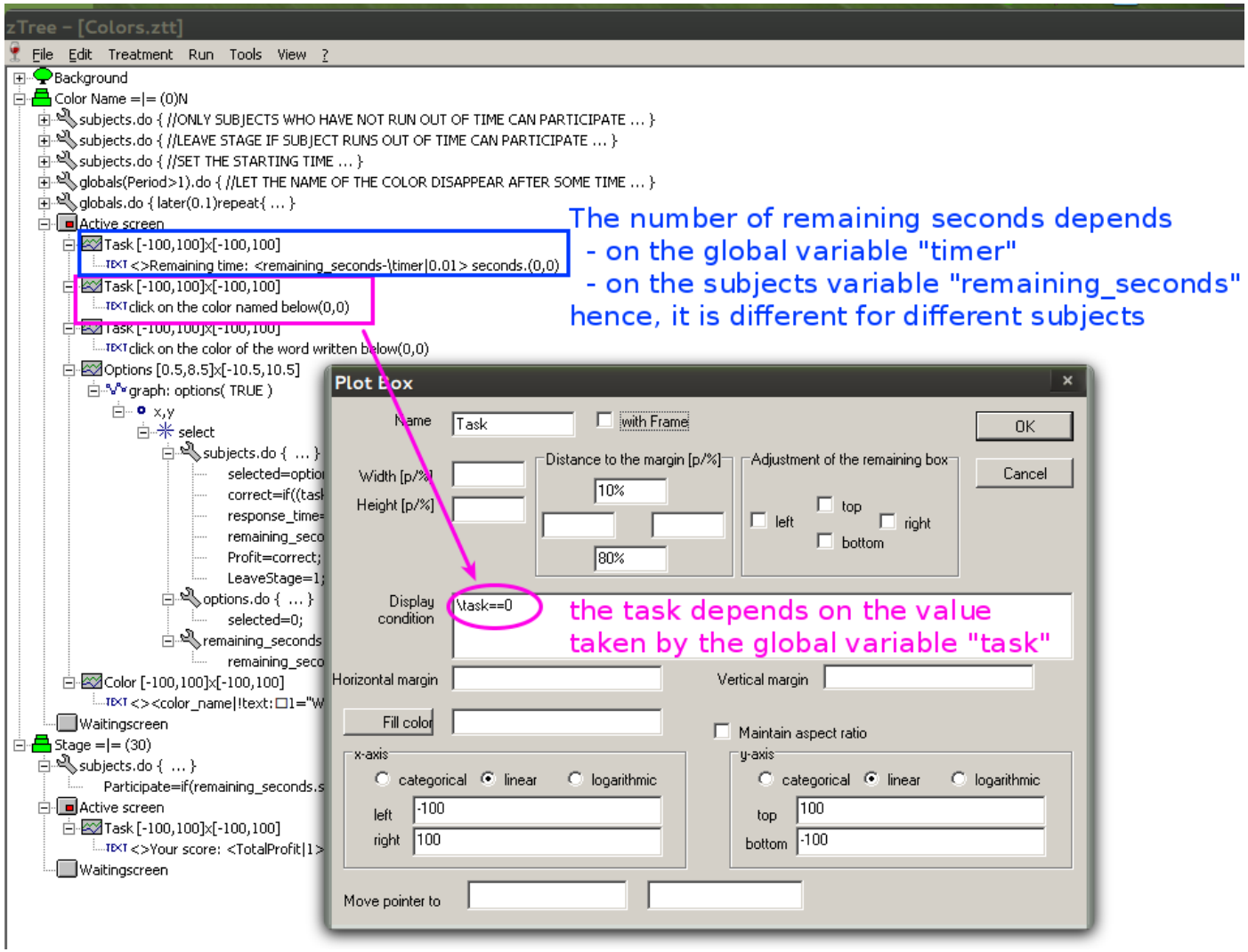

zTree – [Colors.ztt]
File Edit Treatment Run Tools View ?
Background
globals
subjects
summary
contracts
session
logfile
colors
options
remaining_seconds
globals.do { ... }
//INITIALIZE THE "GLOBALS" TABLE
display_name=power(10,1/Period);
timer=0;
task=round(random(),1);
color=roundup(8*random(),1);
color=if(color<1,8,color);
repeat{
color_name=roundup(8*random(),1);
color_name=if(color_name<1,8,color_name);
}while(color_name==color);
//INITIALIZE THE "COLORS" TABLE
iterator(i,8).do{
colors.new{
color=:i;
red=random();
green=random();
blue=random();
}
}
globals(Period==1).do { //INITIALIZE THE "REMAINING SECONDS" TABLE ... }
subjects.do { remaining_seconds=remaining_seconds.find(same(Subject),remaining_seconds); }
globals.do { ... }
RepeatTreatment=if(remaining_seconds.sum(remaining_seconds)>0,1,0);
loader REPLACE(colors.txt)
globals.do { ... }
//DEFINE THE THREE COMPONENTS OF THE SELECTED COLOR
create the user defined table "colors"
Lifetime: period
colors.txt – Kate
File Modifica Visualizza Segnalibri Sessioni Strumenti
Nuovo Apri Indietro Avanti Salva
Do...
Navigatore del file...
colors Period color red green blue
colors 1 1 1 1 1
colors 1 2 1 0 0
colors 1 3 0 0 1
colors 1 4 0 1 0
colors 1 5 1 1 0
colors 1 6 1 0.5 0
colors 1 7 1 0 1
colors 1 8 0 0 0
Riga: 1 Colonna: 1 INS RIGA ISO-8859-1 colors.txt
Terminale
colors in z-Tree are defined in terms
of their three components:
red, green and blue


zTree – [Colors.ztt]
File Edit Treatment Run Tools View ?
Background
globals
subjects
summary
contracts
session
logfile
colors
options
remaining_seconds
globals.do { //INITIALIZE THE "GLOBALS" TABLE ... }
globals(Period==1).do { //INITIALIZE THE "REMAINING SECONDS" TABLE ... }
subjects.do { remaining_seconds=remaining_seconds.find(same(Subject),remaining_seconds); }
globals.do { ... }
RepeatTreatment=if(remaining_seconds.sum(remaining_seconds)>0,1,0);
loader REPLACE(colors.txt)
globals.do { ... }
//DEFINE THE THREE COMPONENTS OF THE SELECTED COLOR
red=colors.find(color==\color,red);
green=colors.find(color==\color,green);
blue=colors.find(color==\color,blue);
//GENERATE THE "OPTIONS" TABLE
iterator(i,8).do{
options.new{
color=:i;
red=colors.find(color==:i,red);
green=colors.find(color==:i,green);
blue=colors.find(color==:i,blue);
rand=random();
y=0;
}
}
options.do { ... }
x=count(rand>=:rand);
options.do { ... }
//SORT OPTIONS RANDOMLY
while(sum(x)!=iterator(i,8).sum(i)){
options.do{rand=random();}
options.do{x=count(rand>=:rand);}
}
Active screen
The user-defined "options" table contains the 8 options (colors) among which subjects have to choose
It will be used to display the 8 balls on the screen
the table contains one row for each of the 8 colors.
For each color we define:
- the 3 components (red, green and blue)
- the vertical position on the screen (y)
The horizontal position of each of the 8 colors is randomly defined



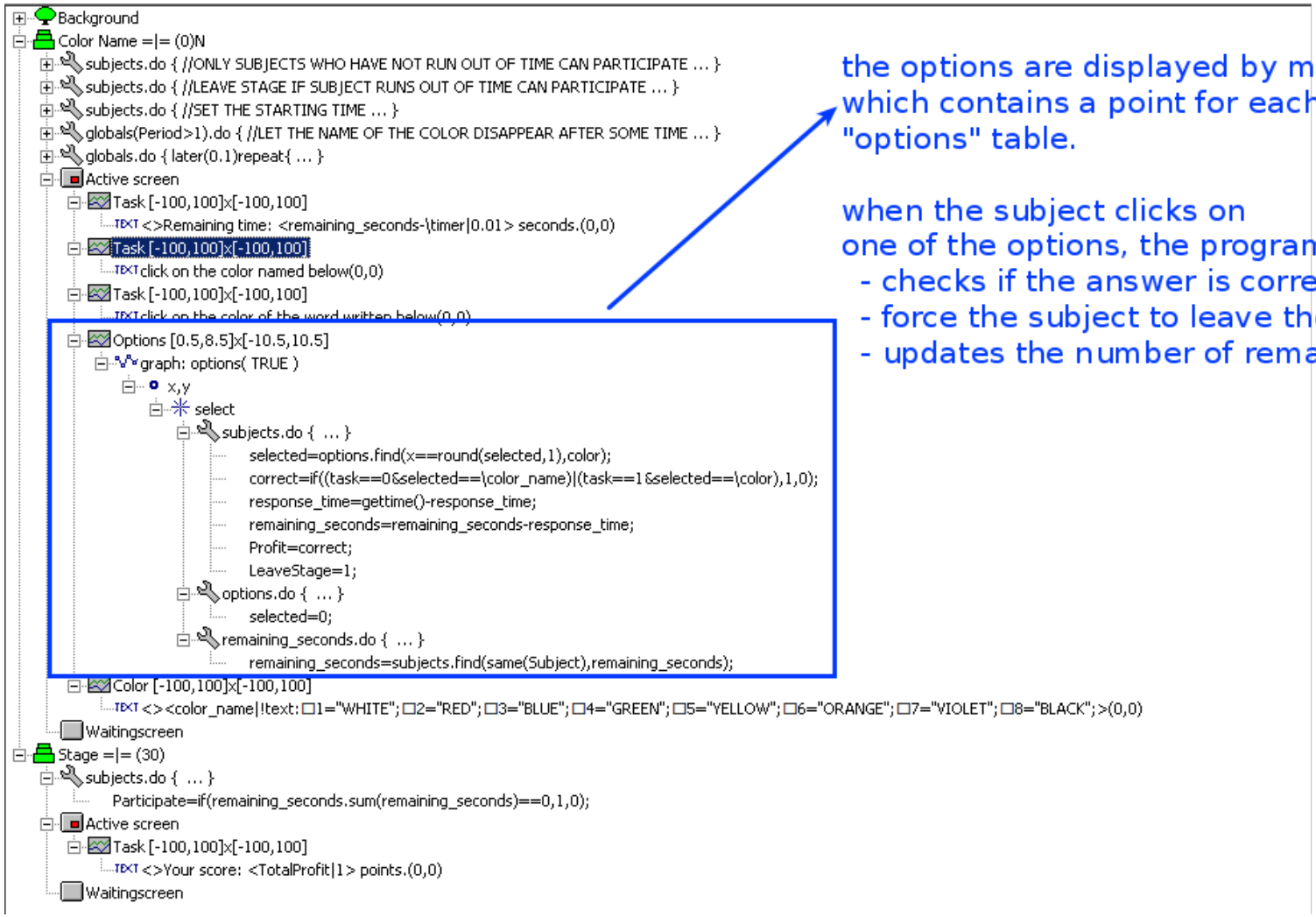
Mark Pigors

University of Cologne



Mark Pigors

University of Cologne

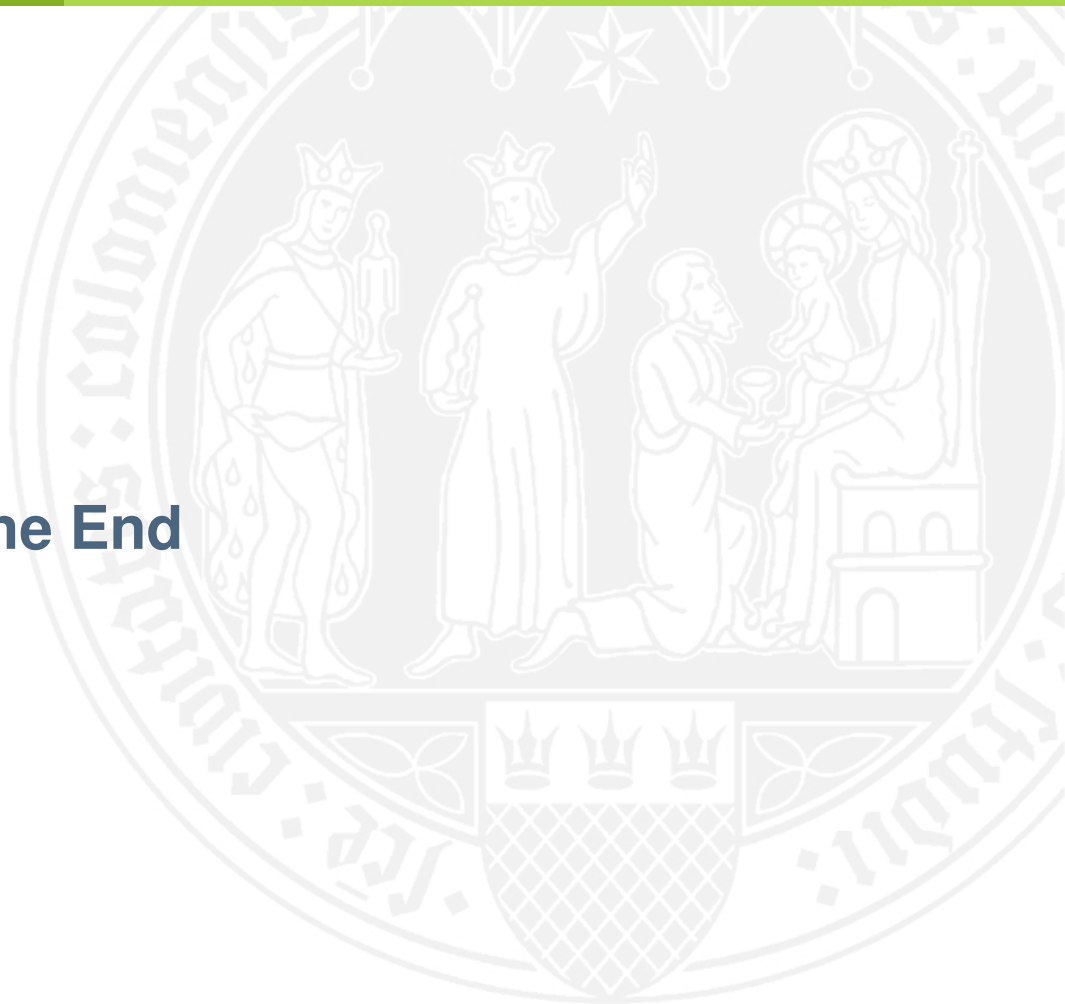


the options are displayed by means of a graph which contains a point for each element of the "options" table.

when the subject clicks on one of the options, the program
- checks if the answer is correct
- force the subject to leave the stage
- updates the number of remaining seconds

Mark Pigors

University of Cologne



The End