Основы z-Tree

Всеволод Корепанов

на базе «zTree Workshop»
Justin Esarey
(http://www.slideserve.com/charisma/ztree-workshop-fundamentals-of-ztree)

Что такое zTree?

- **Z**urich **T**oolbox for **R**eadymade **E**conomic **E**xperiments (Цюрихский инструментарий для подготовки и проведения экономических экспериментов)
- Состоит из 2 программ:
  - zTree (среда программирования и сервер исполнения)
  - zLeaf (клиентская программа для игроков)
- Создаётся многопользовательская игра. Примеры: «Общественное благо», переговоры, конкуренция, аукционы и т.п.

Технология z-Tree

- Клиент – Сервер,
- Игроки – Экспериментатор

Необходимое для работы

- zTree and zLeaf Programs (zTree.exe and zLeaf.exe) Версия 3.3.12
  - http://www.iew.uzh.ch/ztree/index.php, web-поисковики
- zTree Tutorial Manual (ztree21tutorial.pdf)
  - Руководство/путеводитель по возможностям z-Tree
- zTree Reference Manual (ztree21ref.pdf)
  - Справочная информация
- Итак…

Процесс игры

- Процесс игры = шаги, через которые последовательно или синхронно проходят все игроки
- *Сцена* (*Stage*) – точка принятия решения игроком; экран(ы), на котором игрок может выбрать своё действие
- *Сценарий* (*Treatment*) – логически-связанная последовательность сцен, через которые проходят игроки. Возможно, повторяющаяся (*Periods*).
- *Сессия* (*Session*) – игры с одним сценарием с одними и теми же игроками.
- В конце сессии *анкетирование*

Сценарий (Treatment) и сцена (Stage)

- Сценарий определяет:
  - количество игроков, разбиение на группы;
  - порядок сцен;
  - количество периодов.
- Каждая сцена определяет:
  - условия входа и выхода игрока со сцены;
  - интерфейс: таблицы, графики, поля, формы для принятия решения;
  - логику: подготовительные вычисления – подготовка/агрегация данных с предыдущей сцены, реакция интерфейса на действия игрока.
- Что из себя представляет сценарий в zTree?

zTree - [mtas.ru_games_gl_zTreeProgramFree.ztt]

File Edit Treatment Run Tools View ?

- Background
  - globals
  - subjects
  - summary
  - contracts
  - s_history
  - assist
  - assistEndowment
  - session
  - globals.do { array groups[]; ... }
  - subjects.do { grSubject = 0; ... }
  - subjects.do { // base game ... }
  - globals.do { // def params, history, Interface and helpers ... }
  - Active screen
  - Waitingscreen
- Выбор действия =|= (-1)N
  - subjects.do { // for calculator ... }
  - subjects.do { // 1 - setting context ... }
  - Active screen
    - Справка
    - Экран игры
      - Кнопки
      - Общий
        - Несвободный интерфейс
        - Свободный интерфейс
        - Информация
  - Waitingscreen
- Результат шага =|= (0)A
  - subjects.do { // Check modifications ... }
  - subjects(MakeAdviceRun == 1 & Period == 1).do { // initialize on first Period ... }

Фоновая сцена: глобальные переменные, общие элементы экрана

У каждого элемента есть параметры

Обычные сцены

zTree - [Treatment Name.ztt]

File Edit Treatment Run Tools View ?

- Background
  - globals
  - subjects
  - summary
  - contracts
  - session
  - subjects.do { ... }
    - // Programs executed at the beginning of each period
    - // (before any stages)
  - Active screen
    - Header
  - Waitingscreen
    - What is shown to subjects as they wait for the next stage to begin (in all stages)
- Stage 1 =|= (30)
  - subjects.do { ... }
    - // Programs executed at the beginning of the stage
    - // (before the screen is shown)
  - Active screen
    - The screen shown to subjects during the stage
  - Waitingscreen
- Stage 2 =|= (30)
  - subjects.do { ... }
    - // Programs executed at the beginning of the stage
    - // (before the screen is shown)
  - Active screen
    - The screen shown to subjects during the stage
  - Waitingscreen

Порядок сцен,

порядок исполнения

start | Inbox - Outloo... | Adobe Acroba... | zTree Worksh... | ztree | zTree - [Treat... | Macromedia Fi... | 6:00 PM

Данные игроков и др. данные

- Данные хранятся в реляционных таблицах.
- Строка таблицы называется «запись», столбцы – «переменные», столбец-переменная имеет имя. Т.о. все записи одной таблицы содержат одинаковый набор переменных. Например, одна запись в таблице subjects содержит все переменные одного из игроков.
- Базовые таблицы: globals (глобальные переменные исследования), subjects (переменные игроков), session (переменные, неизменяемые в течение сессии) и др.

Пример данных игры

globals table

Period	NumPerioc	RepeatTre	groups[1]	groups[2]	groups[3]	GroupsNu	types[1]	types[2]	types
1	1	0	3	0	0	1	9	25	

subjects table

Period	Subject	Group	Profit	TotalProfit	Participate	grSubject	flag	Endowmer	
1	1	1	0	6000000	1	1	0	115	
1	2	1	0	6000000	1	2	0	115	2
1	3	1	0	6000000	1	3	0	115	

s_history table

olds[1]	olds[2]	olds[3]	Period	time	face	gain	Group	grSubject
0	0	0	1	0	1	0	1	1
0	0	0	1	0	2	0	1	2
0	0	0	1	0	3	0	1	3

Создадим простую игру «Общественное благо»

- Идея игры «Общественное благо» (Public Goods):
  - Группа людей, размером N, у каждого имеется денежный фонд (Endowment)
  - Каждый выбирает долю своего фонда для вклада (Contribution)
  - Суммарный вклад группы (общественное благо) умножается на X > 1, затем возвращается в группу
  - Выигрыш игрока = (Фонд – Вклад) + (X* Суммарный_вклад) / N

Шаги игры «Общественное благо»

- Игроки разбиваются на группы по N
- Игроки получают их фонды
- Игроки делают выбор своих вкладов
- Подсчёт размера суммарного вклада игроков и умножение на X
- Сообщение выигрыша (оставшийся фонд + доля общего блага) каждому игроку
- (Повтор шагов?)

Сцены игры «Общественное благо»

- Игроки разбиваются на группы по N
- Игроки получают их фонды

Background Stage

- Игроки делают выбор своих вкладов

Stage 1

- Подсчёт размера суммарного вклада игроков и умножение на X
- Сообщение выигрыша каждому игроку

Stage 2

Терминология z-Tree для сцен

- Program: набор операций для назначения переменных, расчёта платежей, сохранения информации и т.п. **Каждая Program привязана к конкретной таблице! Program выполняется для каждой записи этой таблицы!**
- Active Screen: Экран, на котором игроки получают информацию и принимают решение
- Waiting Screen: экран “заполнитель”, который видят игроки после принятия решения на Сцене, пока ждут когда другие игроки закончат эту Сцену

Сцены игры «Общественное благо» (2)

- Игроки разбиваются на группы по N ← program
- Игроки получают их фонды ← program

Background Stage

- Игроки делают выбор своих вкладов ← active screen

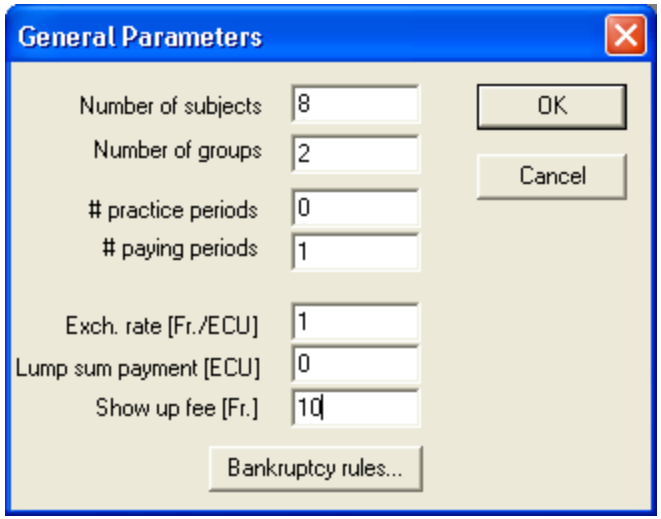
Stage 1

- Подсчёт размера суммарного вклада игроков и умножение на X ← program
- Сообщение выигрыша каждому игроку ← active screen

Stage 2

Шаг 1: Установка общих параметров фоновой сцены (пар-ров сценария)

- Double-click на **Background** => Окно настройки параметров
- **Number of Subjects**
- **Number of Groups**
- **Practice Periods**
- **Paying Periods**
  1 – один период для сценария



Шаг 2: Специфичные параметры фоновой сцены

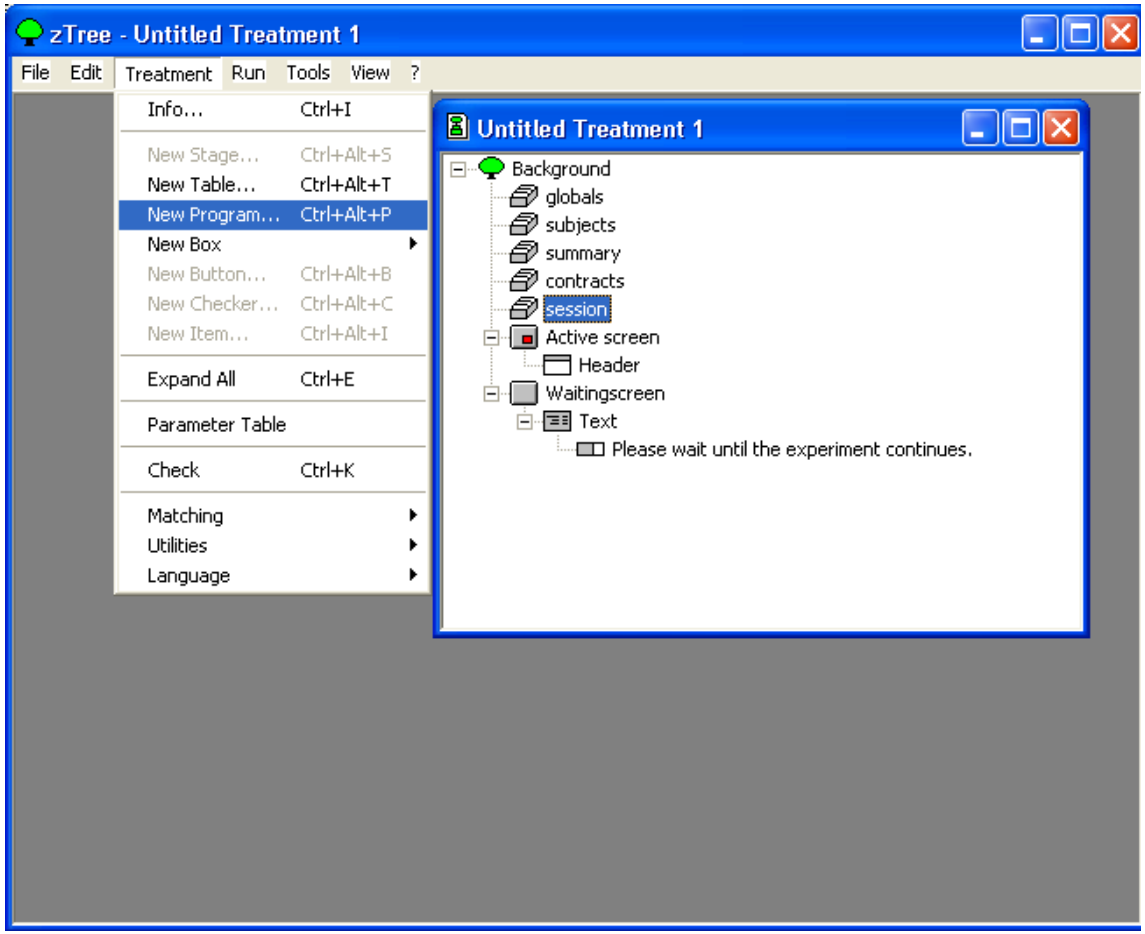
Какие переменные должны быть в игре…

- Количество игроков (общий)
- Количество периодов (общий)
- X – показатель эффективности
- Размер фондов

Общие параметры конкретно нашей игры

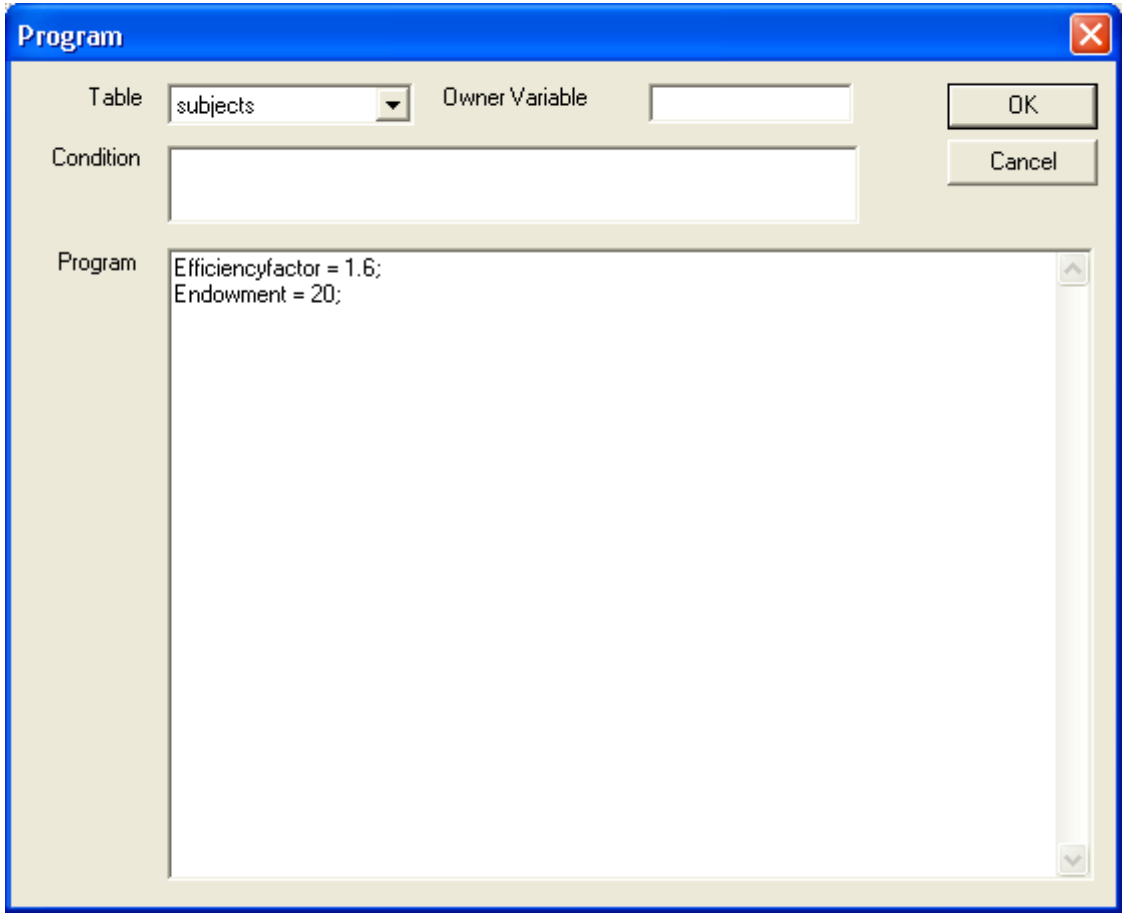
Создание программы для определения переменных

- Курсор на последнюю таблицу (**Session)**
- Меню **Treatment-> New Program**



Создание программы для определения переменных (2)

- В поле **Table** выбрать **Subjects**
- Создать имена для X и Фонда
- Синтаксис:
  - variablename = number;
- Нажать **OK**



zTree - Untitled Treatment 1

File Edit Treatment Run Tools View ?

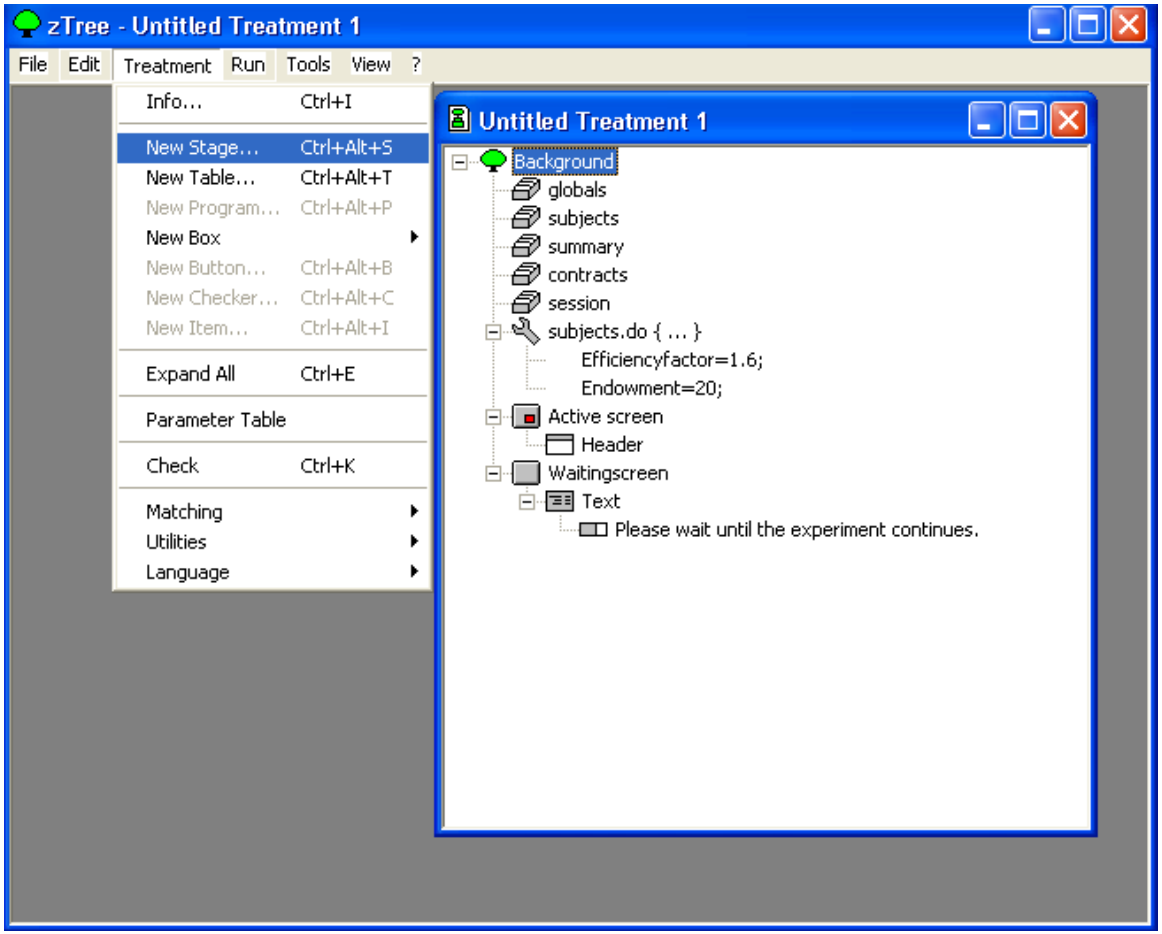
Untitled Treatment 1

- Background
  - globals
  - subjects
  - summary
  - contracts
  - session
  - subjects.do { ... }
    - Efficiencyfactor = 1.6;
    - Endowment = 20;
  - Active screen
    - Header
  - Waitingscreen
    - Text
      - Please wait until the experiment continues.

start | Inbox - Outloo... | Adobe Acroba... | zTree Worksh... | ztree | zTree - Untitle... | Macromedia Fi... | 4:10 PM

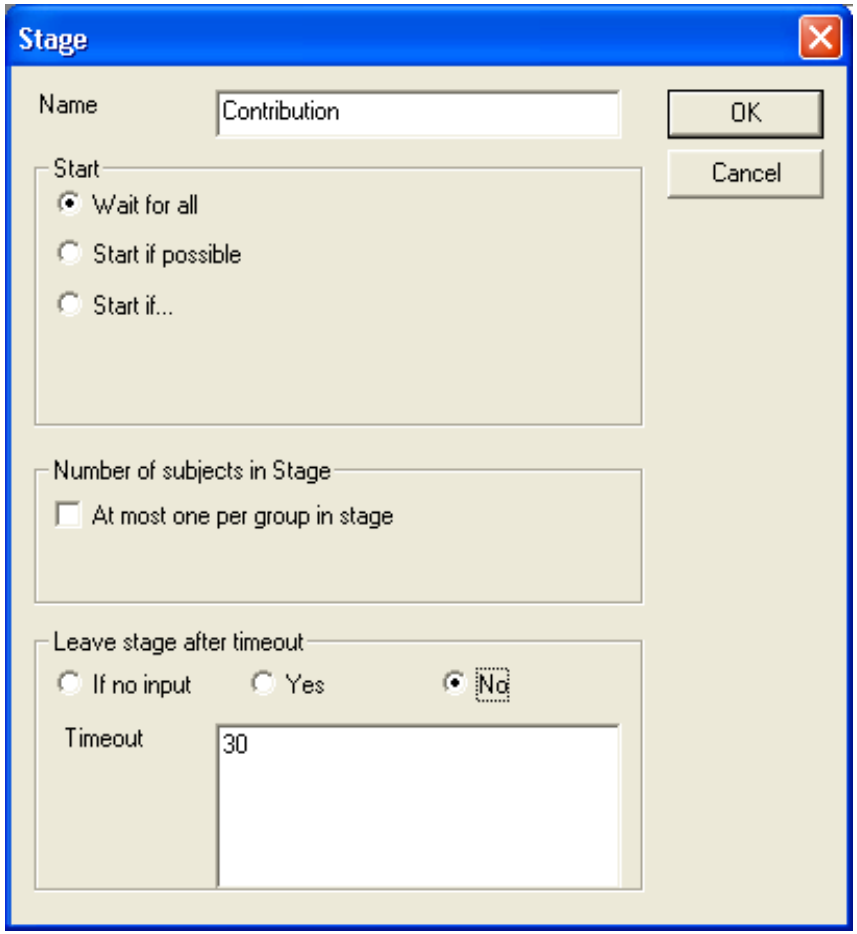
Шаг 3: Сцена принятия решения о вкладах

- Курсор на **Background**
- Меню
- **Treatment -> New Stage**



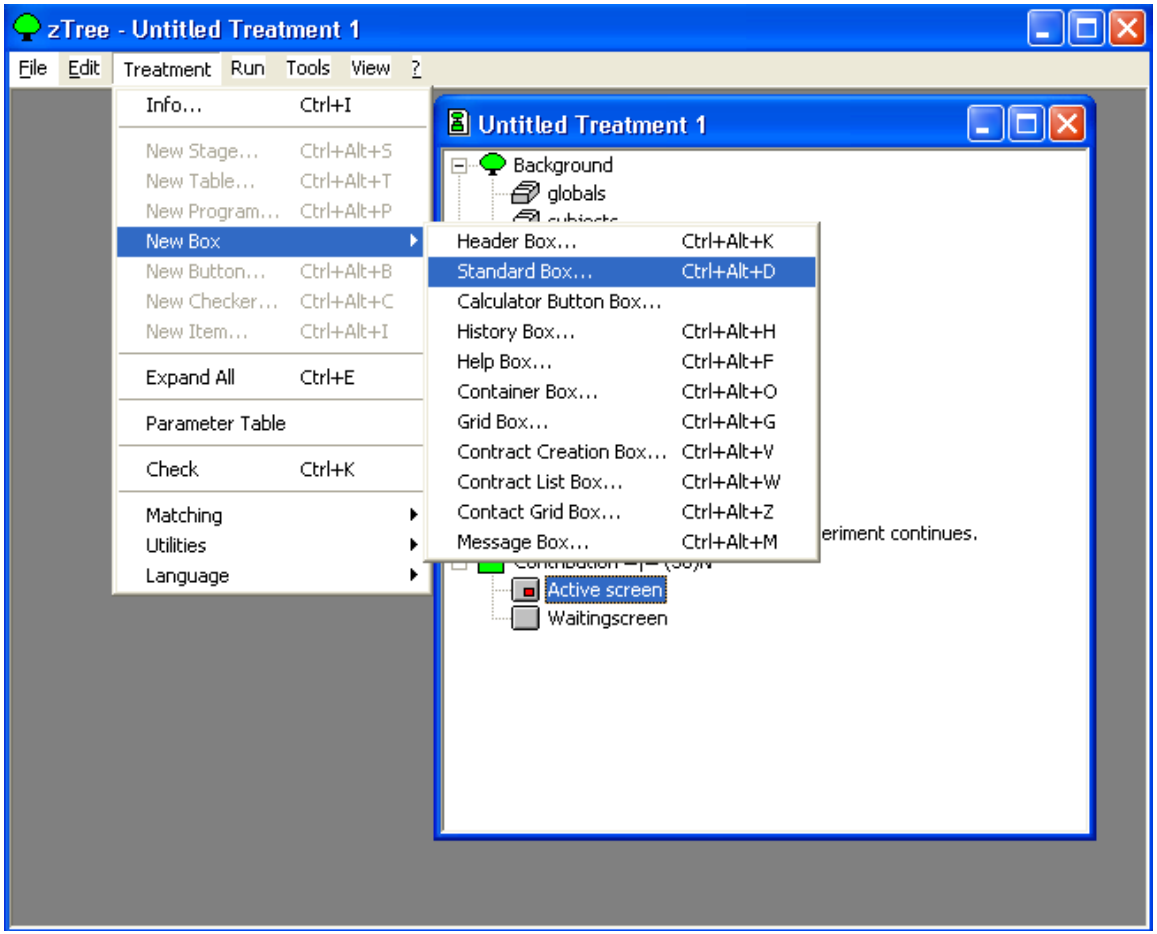
Создание новой сцены

- Имя сцены, напр. “Contribution”
- В области **Start**, выбрать условие входа **Wait for all** (т.о. все игроки входят на сцену вместе)
- Оставить **At most one…** без галочки (для поочерёдного принятия решения)
- В области **Leave stage after timeout**, выбрать **No** (делает несущественным временное ограничение на принятие решения)
- **Timeout** , 30 (секунд) или -1 – убрать таймаут



Создание Экрана для ввода игроками Вклада

- Курсор на **Active Screen**
- Меню **Treatment → New Box → Standard Box**



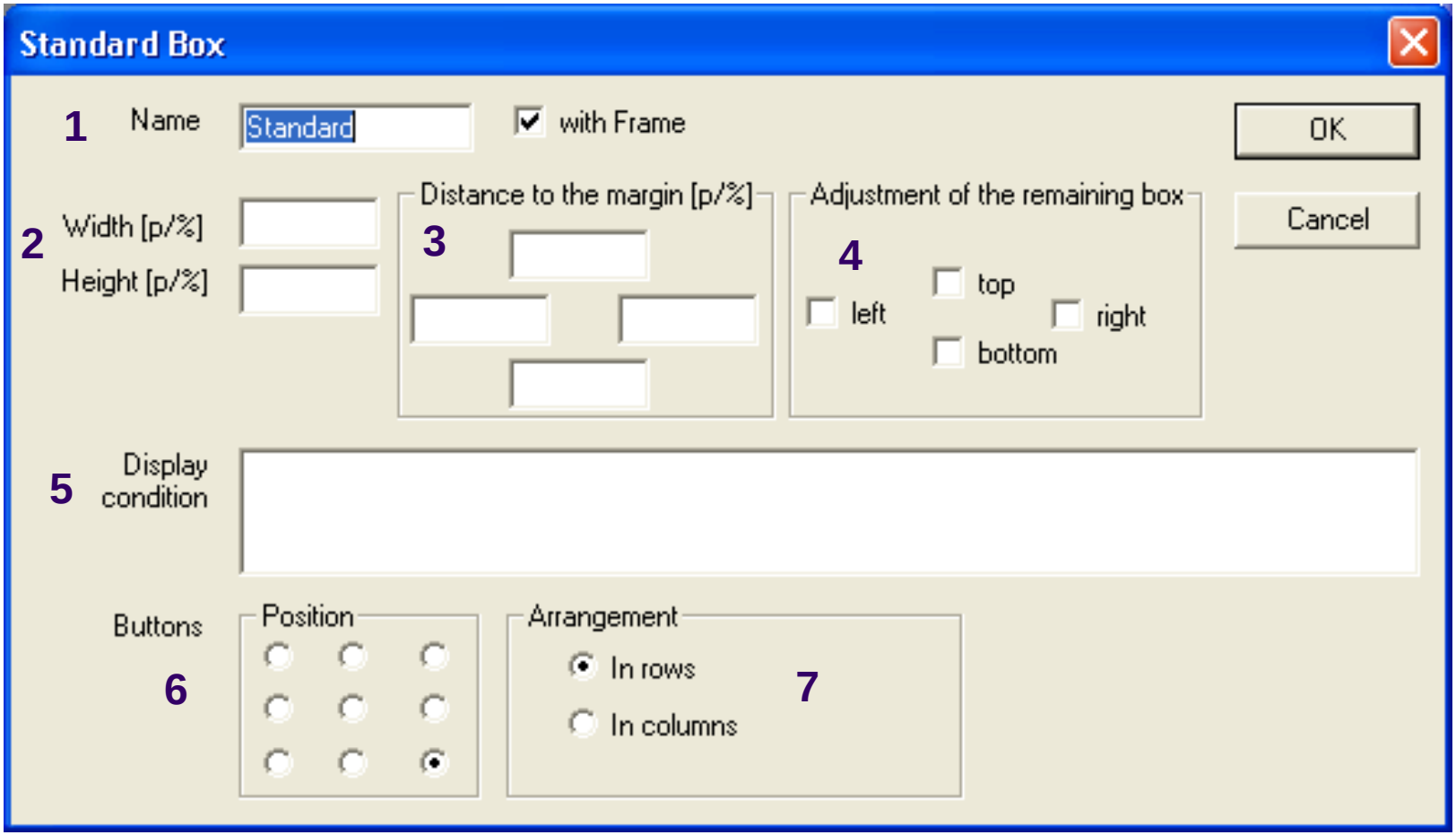
Терминология построения интерфейса

- Boxes – прямоугольные контейнеры для отображения однородной информации: текста, переменных, кнопок, radio-button, slider, графиков, чатов, мультимедиа, контрактов,…, таблиц истории и **других контейнеров**;
- Созданы не только для содержания, но и для разбиения экрана на части;
- Standard Box: Контейнер для отображения полей с текстом, полей со значениями переменных и кнопок
- Item – поле с текстом и значением/вводом переменной

Правила построения интерфейса

- Последовательно-иерархичное определение контейнеров;
- Для каждого контейнера доступна некая *область* его расположения на экране;
- Контейнер задаёт место своего расположения в его *области*, свои размеры и *область* для следующего контейнера;
- *Область* расположения для первого контейнера – весь экран (за исключением фонового);
- *Область* расположения контейнеров внутри другого контейнера – его прямоугольник.

Новый контейнер Standard Box



Параметры Контейнера Standard Box

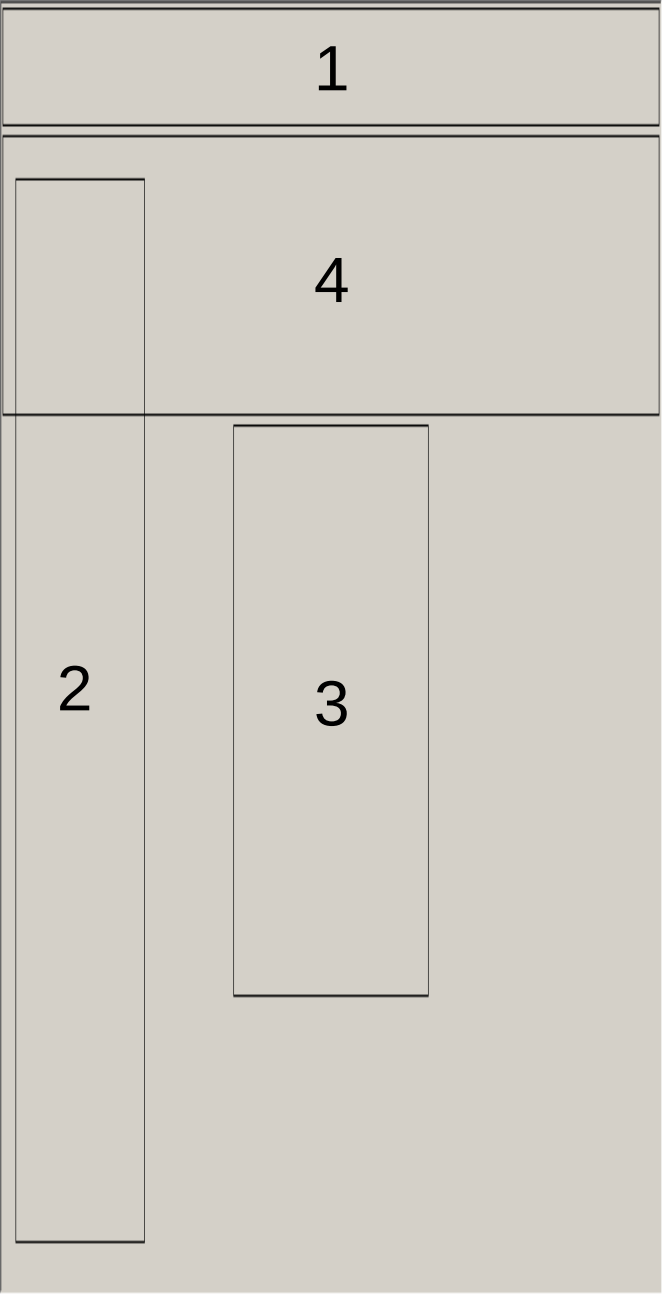
1. Метка (для внутр. использования)
2. Ширина, Высота Конт. в пикселях или процентах от *оставшегося* экрана
3. Отступы от краёв *оставшегося* экрана
4. Изменение оставшегося экрана (обрезать ли оставшийся экран для следующих контейнеров своими размерами)
5. Условие отображения
6. Положение кнопок (где размещать кнопки в этом контейнере)
7. Порядок кнопок (как расположить кнопки)

1

4

2

3



Name 1 ☑ with Frame

Width [p/%]
Height [p/%] 10%

Distance to the margin [p/%]: 0

Adjustment of the remaining box: ☑ top ☐ left ☐ right ☐ bottom

Name 2 ☑ with Frame

Width [p/%] 20%
Height [p/%]

Distance to the margin [p/%]: 20p; 20p; 20p

Adjustment of the remaining box: ☐ top ☐ left ☐ right ☐ bottom

Name 3 ☑ with Frame

Width [p/%] 30%
Height [p/%] 50%

Distance to the margin [p/%]

Adjustment of the remaining box: ☐ top ☐ left ☐ right ☑ bottom

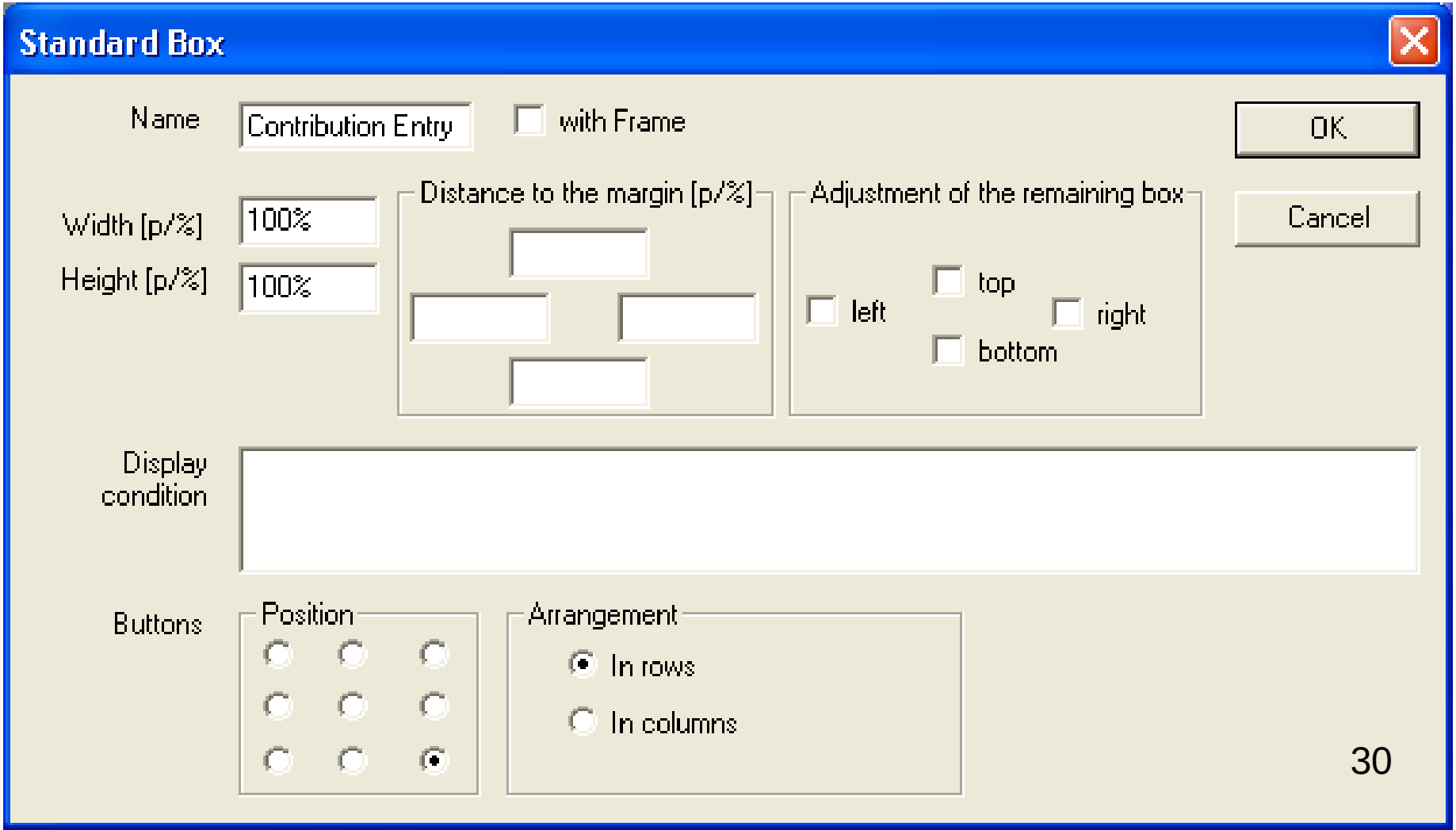
Name 4 ☑ with Frame

Width [p/%]
Height [p/%]

Distance to the margin [p/%]

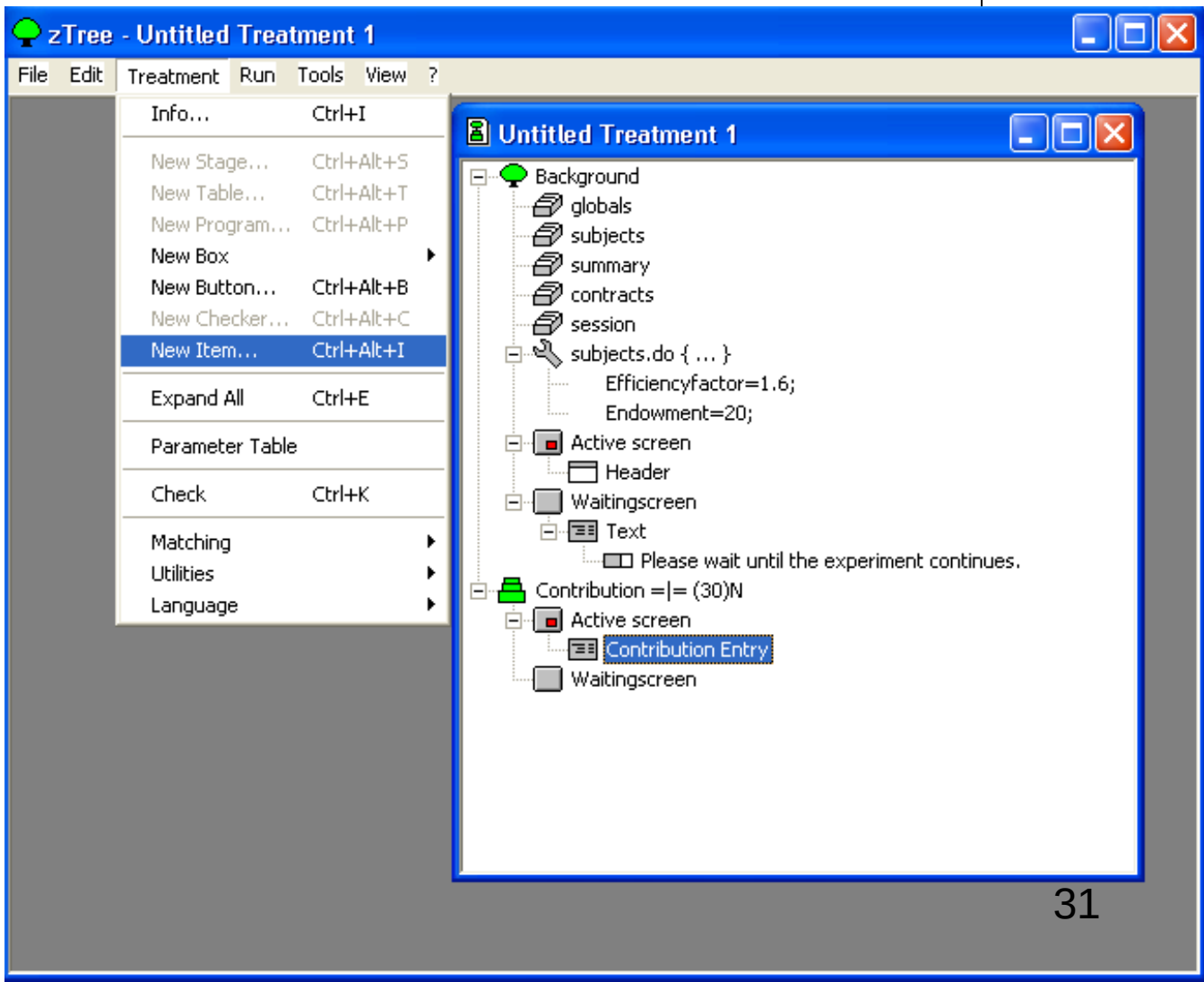
Adjustment of the remaining box: ☐ top ☐ left ☐ right ☐ bottom

Устанавливаем параметры

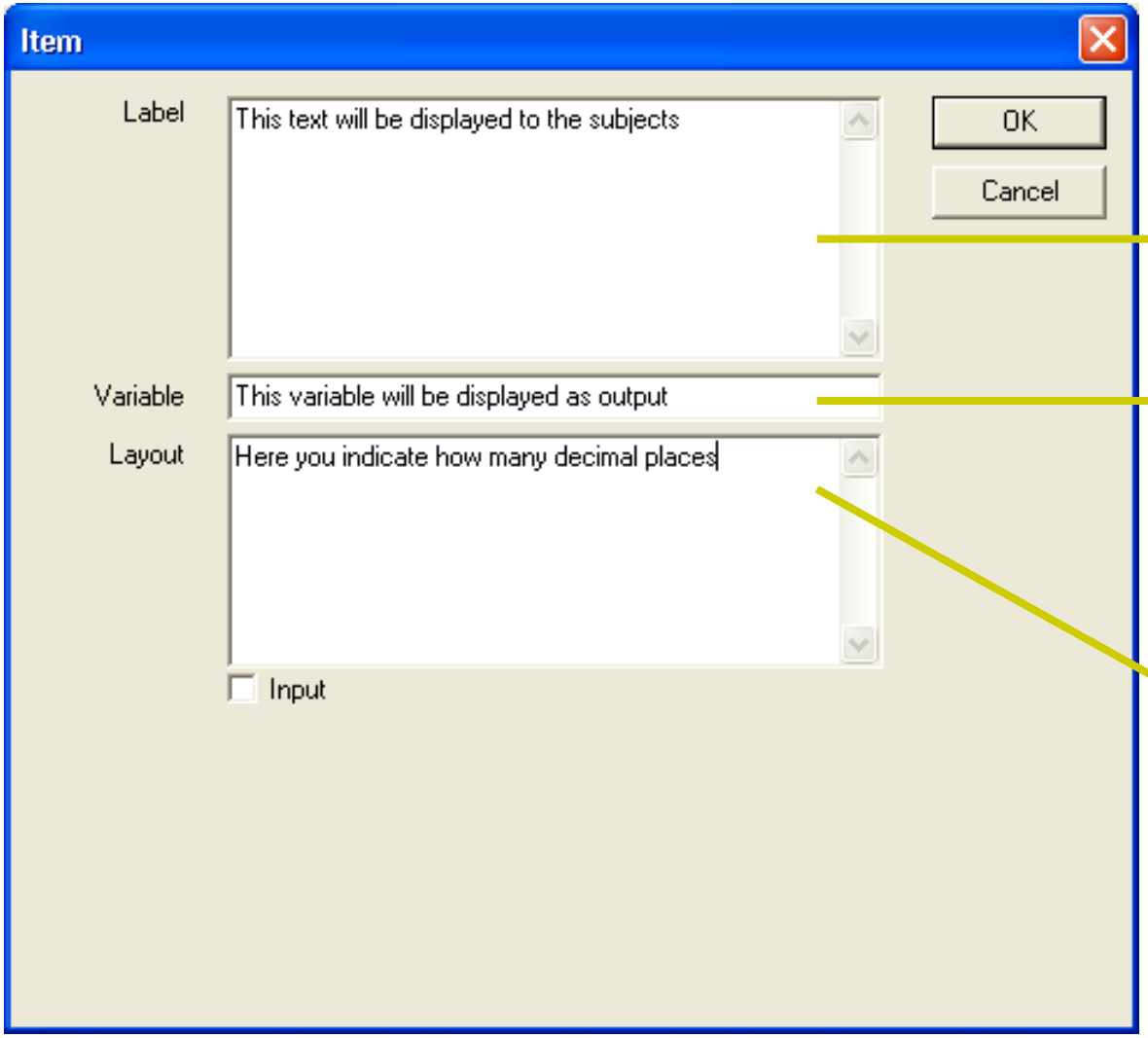


Добавляем поле в Контейнер

- Курсор на новый контейнер
- Меню **Treatment-> New Item**



Параметры поля



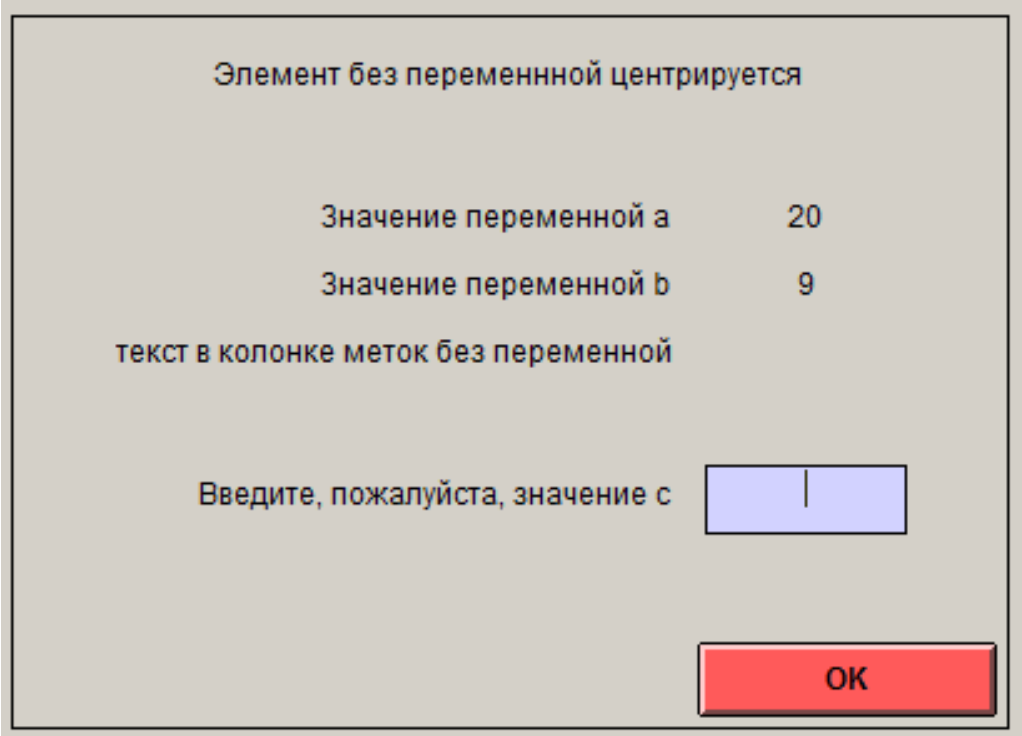
Метка поля

Переменная для отображения её значения

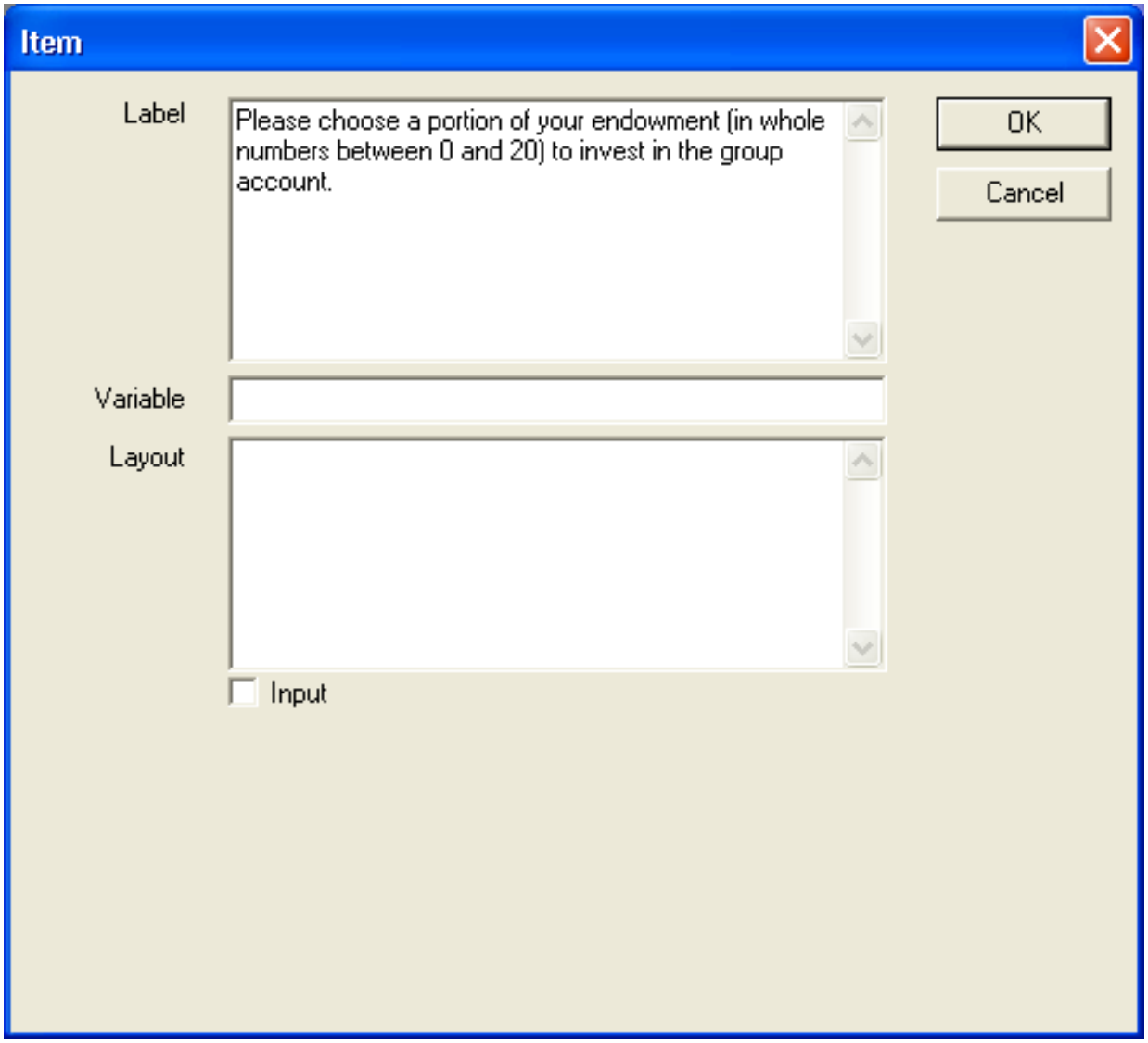
Точность/формат отображения переменной

Возможности отображения информации с помощью полей

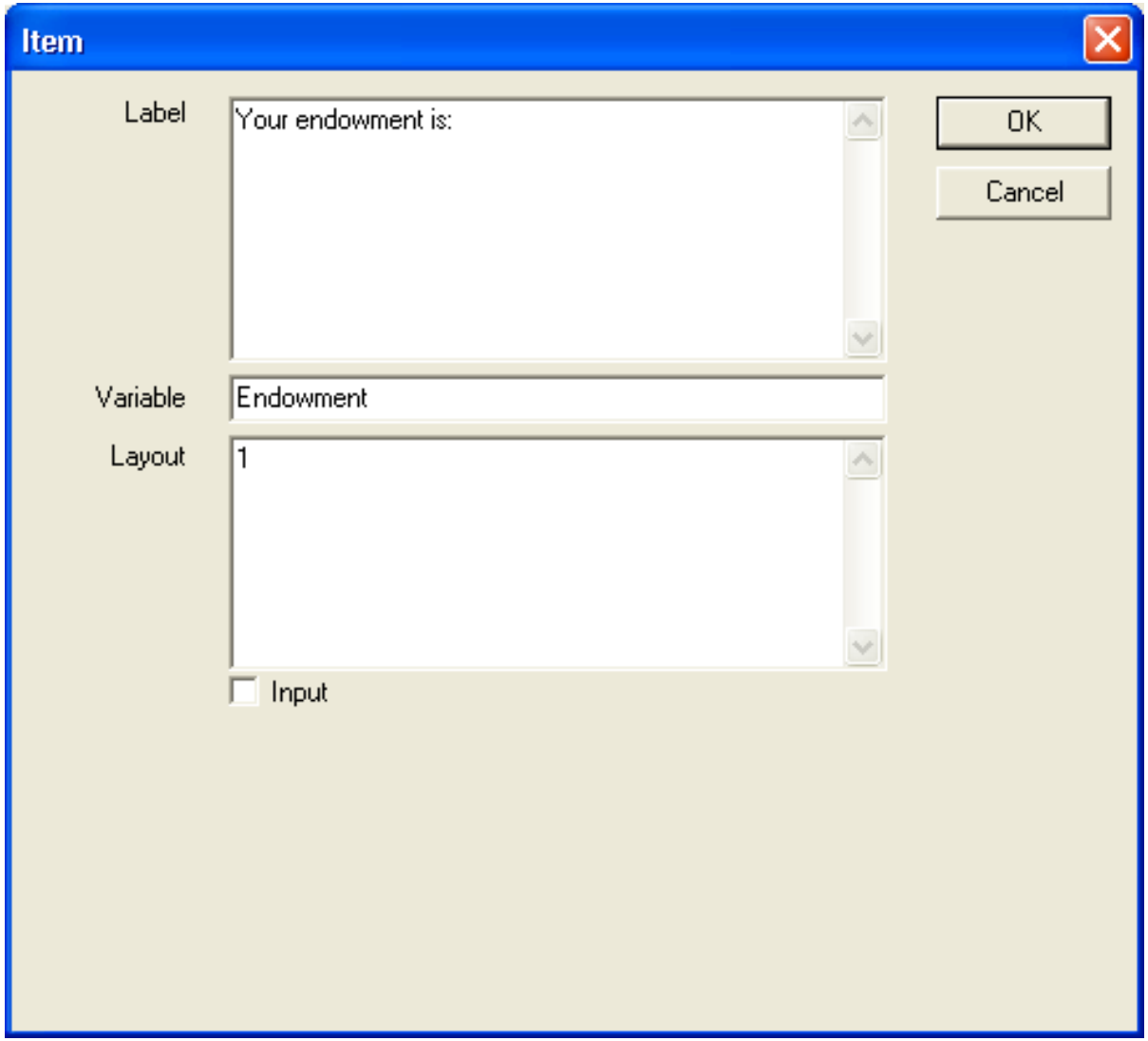
- Standard
  - Элемент без переменнной центрируется
  -
  - Значение переменной a: OUT(a)
  - Значение переменной b: OUT(b)
  - текст в колонке меток без переменной: OUT(_)
  -
  - Введите, пожалуйста, значение c: IN(c)
  -
  - OK



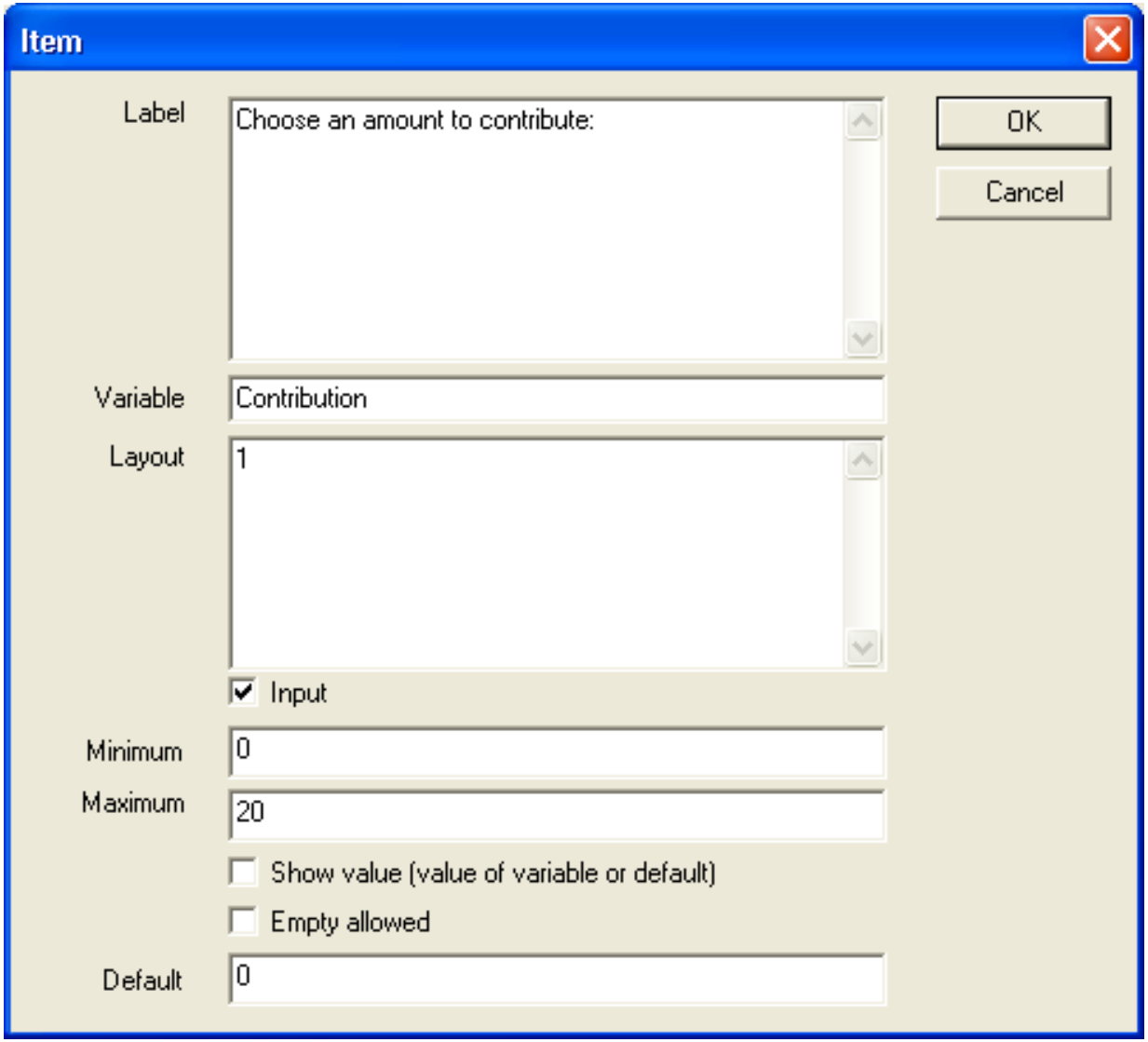
Заголовок контейнера



Показываем Фонд

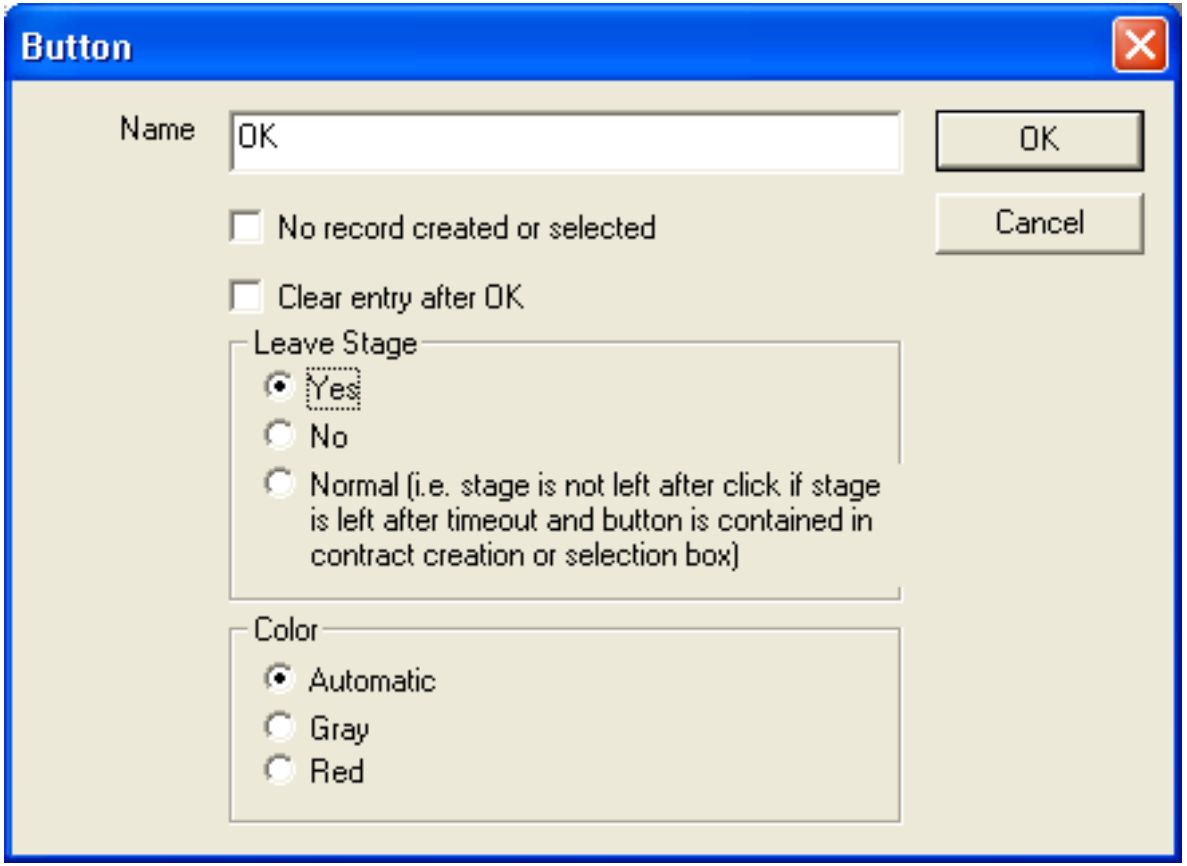


Поле ввода для Вклада

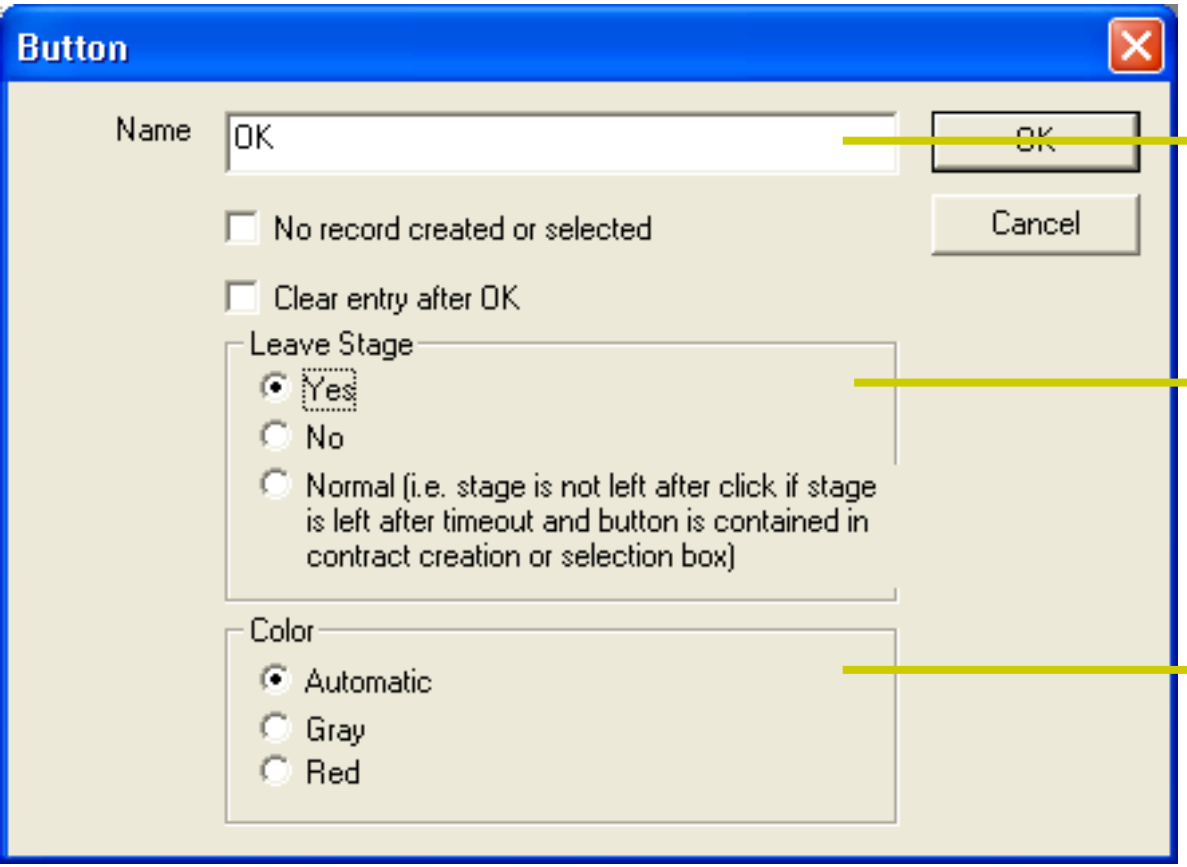


Кнопка для завершения Сцены

- Курсор на последнее поле контейнера
- Меню **Treatment-> New Button**
- Настроить параметры как показано



Параметры кнопок



Отображаемая метка

Покидать ли сцену при нажатии

Цвет кнопки

zTree - [Untitled Treatment 1]

File Edit Treatment Run Tools View ?

- Background
  - globals
  - subjects
  - summary
  - contracts
  - session
  - subjects.do { ... }
    - Efficiencyfactor=1.6;
    - Endowment=20;
  - Active screen
    - Header
  - Waitingscreen
    - Text
      - Please wait until the experiment continues.
- Contribution =|= (30)N
  - Active screen
    - Contribution Entry
      - Please choose a portion of your endowment (in whole numbers between 0 and 20) to invest in the group account.
      - Your endowment is:: OUT(Endowment)
      - Your investment:: IN(Contribution)
      - OK
  - Waitingscreen

start | Inbox - Outloo... | Adobe Acrobat... | zTree Worksho... | Macromedia Fir... | zTree - [Untitle... | Sign On | 9:58 PM

Шаг 3: Подсчёт общественного блага

- Создадим новую сцену “Результаты”
- В новой сцене создадим программу для таблицы *subjects* со следующим кодом:
  - Pgood = Efficiencyfactor * sum(same(Group), Contribution);
  - Функция *sum*: sum(same(Group), Contribution) – просуммировать “*Contribution*” для каждого из игроков (для каждой записи в *subjects*) моей группы.

Ещё программы

- Создадим другую программу после последней и введём след. код:
  - N = count(same(Group));
  - Эта операция создаёт для каждого игрока переменную *N* с количеством игроков в его группе
- Следующая программа:
  - Profit = Pgood / N + (Endowment – Contribution);
  - *Profit* это переменная, существующая по умолчанию в каждой записи *subjects*, которую *zTree* использует для сохранения платежа игроку в каждом периоде

zTree - [PG Example.ztt]

File Edit Treatment Run Tools View ?

- Background
  - globals
  - subjects
  - summary
  - contracts
  - session
  - subjects.do { ... }
    - Efficiencyfactor=1.6;
    - Endowment=20;
  - Active screen
    - Header
  - Waitingscreen
    - Text
      - Please wait until the experiment continues.
- Contribution =|= (30)N
  - Active screen
    - Contribution Entry
      - Please choose a portion of your endowment (in whole numbers between 0 and 20) to invest in the group account.
      - Your endowment is:: OUT(Endowment)
      - Your investment:: IN(Contribution)
      - OK
  - Waitingscreen
- Display Outcomes =|= (30)
  - subjects.do { ... }
    - Pgood=Efficiencyfactor*sum(same(Group), Contribution);
  - subjects.do { ... }
    - N = count(same(Group));
  - subjects.do { ... }
    - Profit = Pgood/N+(Endowment-Contribution);
  - Active screen
  - Waitingscreen

start | Inbox - Outlook Exp... | Adobe Acrobat Prof... | zTree Workshop.ppt | Macromedia Firewor... | zTree - [PG Example... | 10:29 PM

Шаг 4: Вывод результатов

- Создать Standard box “Итоговые выплаты”
  - Сделать контейнер на весь экран
- Добавить элементы:
  - Заглавие для результатов
  - Часть фонда, которую игрок оставил себе
  - Величина вклада игрока
  - Величина суммарного вклада игроков
  - Показать долю вклада игрока
  - Величина выигрыша игрока в этом периоде
  - Кнопка завершения сцены

zTree - [PG Example.ztt]

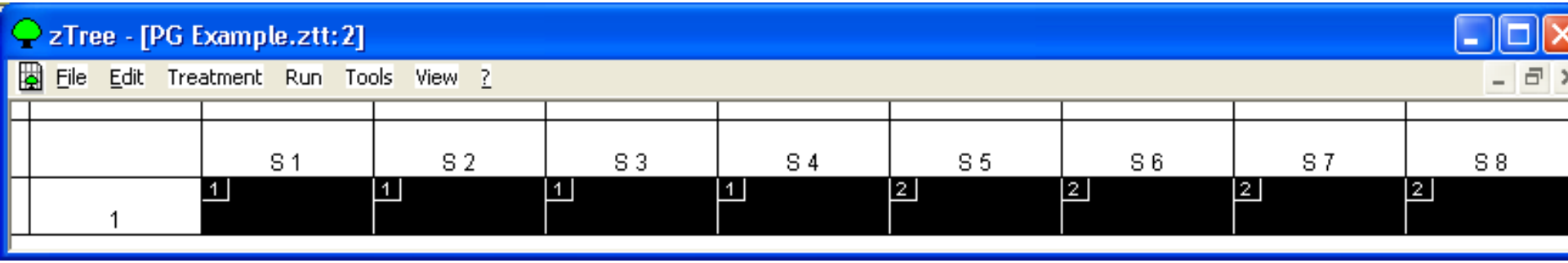
File Edit Treatment Run Tools View ?

- Background
  - globals
  - subjects
  - summary
  - contracts
  - session
  - subjects.do { ... }
    - Efficiencyfactor=1.6;
    - Endowment=20;
  - Active screen
    - Header
  - Waitingscreen
    - Text
      - Please wait until the experiment continues.
- Contribution =|= (30)N
  - Active screen
    - Contribution Entry
      - Please choose a portion of your endowment (in whole numbers between 0 and 20) to invest in the group account.
      - Your endowment is:: OUT(Endowment)
      - Your investment in the group account:: IN(Contribution)
      - OK
  - Waitingscreen
- Display Outcomes =|= (30)N
  - subjects.do { ... }
    - Pgood=Efficiencyfactor*sum(same(Group), Contribution);
  - subjects.do { ... }
    - N = count(same(Group));
  - subjects.do { ... }
    - Profit = Pgood/N+(Endowment-Contribution);
  - Active screen
    - Results
      - Results of your investment
      - Amount of endowment kept:: OUT(Endowment-Contribution)
      - Amount of endowment invested in group account:: OUT(Contribution)
      - Total investment in group account by all group members:: OUT(sum(same(Group), Contribution))
      - Your share of the group account:: OUT(Pgood/N)
      - Your total earnings:: OUT(Profit)
      - OK
  - Waitingscreen

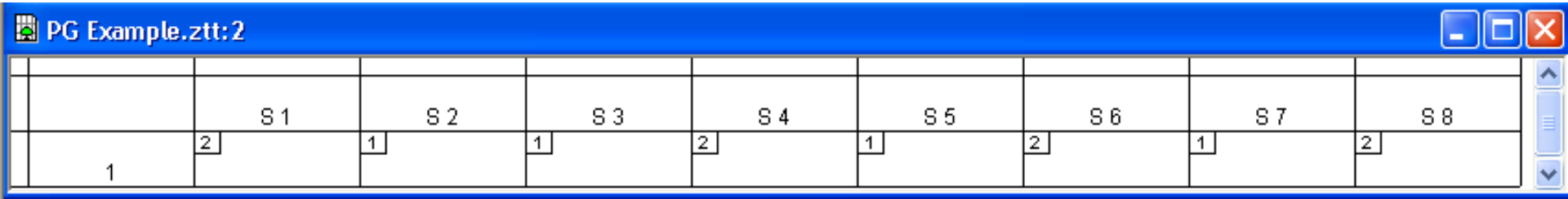
start | Inbox - Outloo... | Adobe Acrobat... | zTree Worksho... | ztree | zTree - [PG Ex... | Sign On | 11:29 PM

Настройка механизма группировки игроков

- Меню **Treatment→Parameter Table**
- Выбрать все ячейки относящиеся к игрокам на 1 периоде



- Select **Treatment→Matching→Stranger**



Это всё!

- Сохраняем файл **File→Save**
- Всё готово для запуска экспериментов

УГЛУБЛЕНИЕ В ZTREE

Динамичный повтор сценария

- Задаём количество периодов = 1 в свойствах фоновой сцены;
- В последней сцене создаём программу для таблицы globals, в которой устанавливаем переменную *RepeatTreatment* как Вам нужно:
  - 1 – повторять сценарий,
  - 0 – не повторять сценарий.

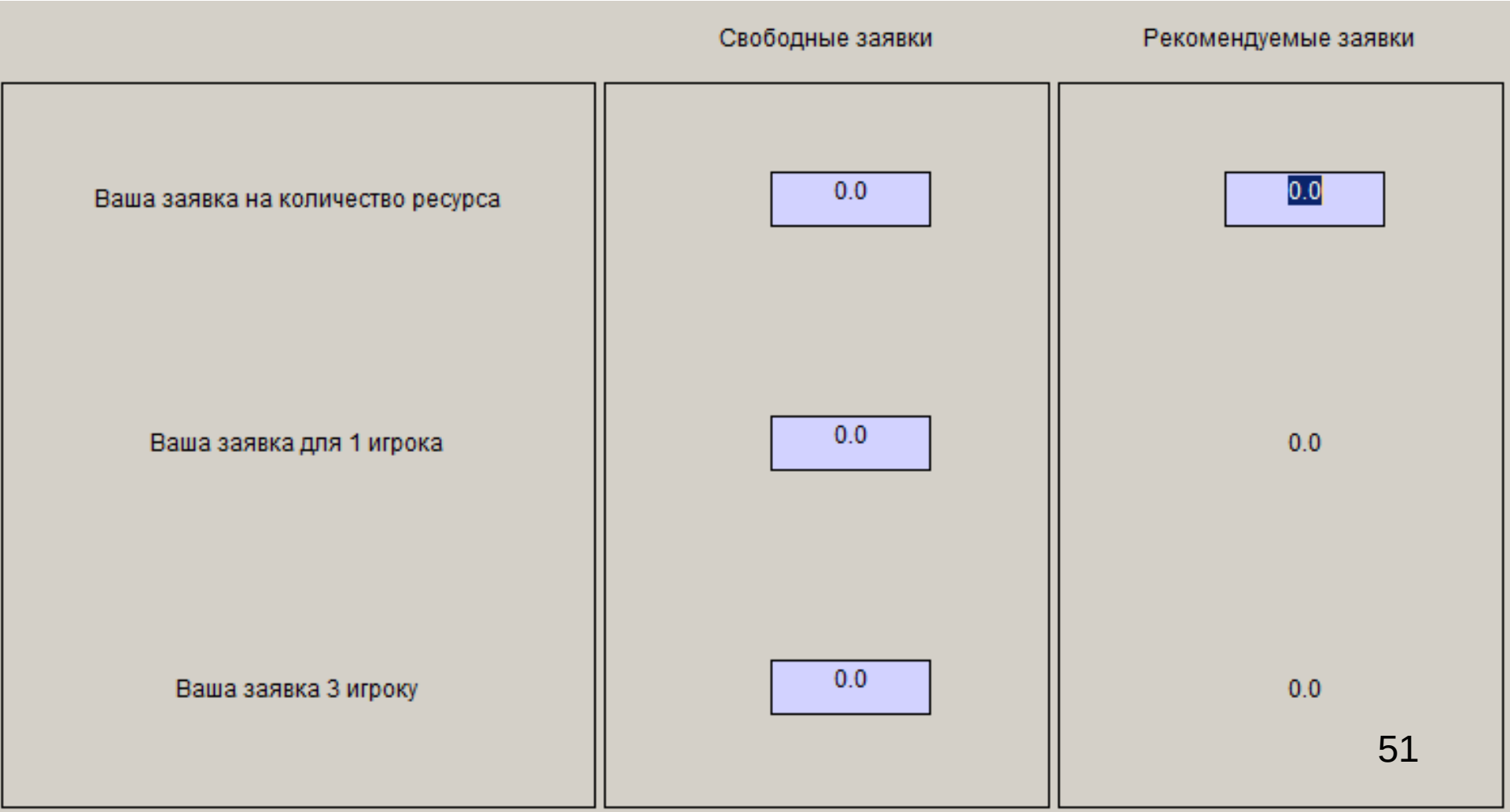
Динамичный интерфейс сцен. Экраны

- ***Проблема***:
  - Сделать несколько экранов на сцене с возможностью переключения между ними.
- ***Решение***:
  - Каждый экран – контейнер;
  - Условное отображение и условные размеры экранов-контейнеров.

Динамичный интерфейс сцен. Поля

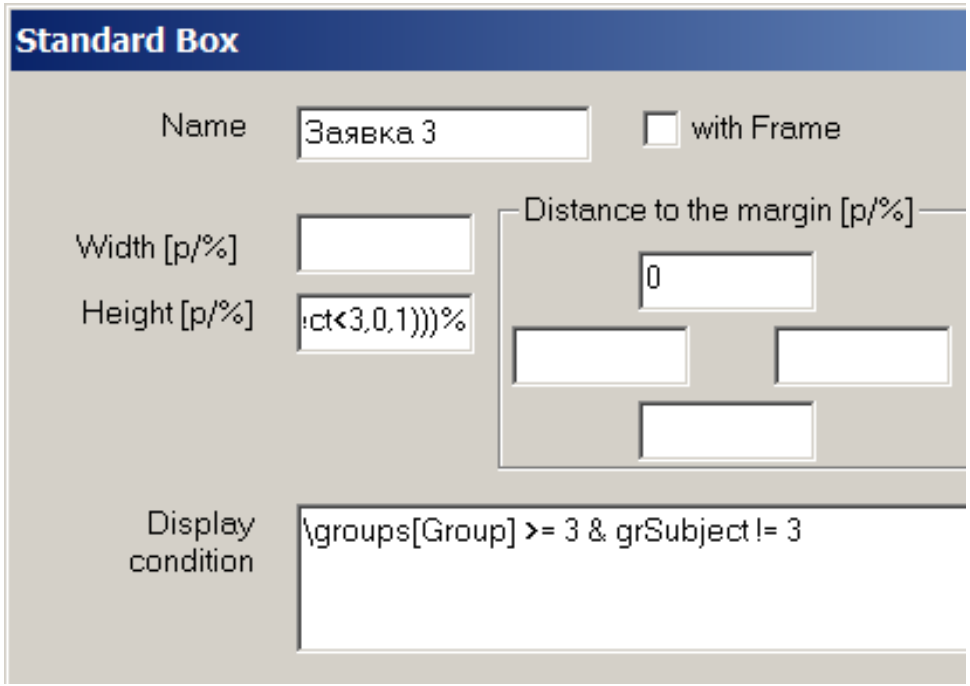
- При создании интерфейса нет возможности использовать для этого циклы
- ***Проблема***:
  - вывод полей в количестве равным количеству игроков в группе.
- ***Решение***:
  - Создаём поля для случая максимального количества игроков;
  - Каждое поле в контейнере;
  - Условное отображение контейнера, условные размеры контейнера.

Пример динамического интерфейса полей



Пример динамического интерфейса полей (2)

- **Параметры для 3 поля**:
  - Height: if(grSubject == 3, 0,
    100 / (\groups[Group] – 2 – if(grSubject<3,0,1)))%
  - Display Condition:
    \groups[Group] >= 3 & grSubject != 3



Пример динамического интерфейса полей (3)

Общий контейнер

Контейнер заголовков

Контейнер текста

Контейнер свободных заявок

Контейнер реком. заявок

Динамичный интерфейс. Калькулятор

- ***Проблема***:
  - Игрок должен быть обеспечен возможностью проверки своих действий – д.быть подсказки.
- ***Решение***:
  - Кнопка «Подсказка» содержит программу, которая может посчитать подсказку;
  - Кнопка не вызывает выход со сцены;
  - Нужен контейнер для отображения подсказок;
  - Поля *Item* автоматически обновляют значения переменных, которые с ними связаны.

Язык программирования. Табличные функции

- [*Table*.]*func*(*args*) – пробегает по всем записям таблицы *Table* (если *Table* не указана явно, то текущей таблицы); args может содержать переменные таблицы.
  - sum(1), count() – количество записей таблицы;
  - sum(a+b) – суммирование сумм a и b всех записей.
- [*Table*.]*func*(*cond*, *args*) – пробегает по тем записям, которые удовлетворяют условию *cond.*
  - subjects.sum(Group == 1, a) – суммирование *a* среди игроков в группе №1;
  - find(b == 12, c+e) – (c+e) из первой записи в которой b = 12.

Язык программирования. Оператор контекста (scope) - ‘:’

- Посчитать: $x_i = \sum_j a_j b_j$
  - Код: x = sum(a * b)
- Посчитать: $x_i = \sum_j a_i b_j$
  - Код: x = sum(:a * b)
- Пусть A,B,C – таблицы с переменной v

  x = v + B.sum(v * :v – C.product(v - :v - ::v))

  A B A C B A
- sum(Group == :Group, X) – суммирование X среди игроков моей группы – sum(same(Group), X)
- функция same(x) ≡ x == :x

Язык программирования. Массивы

- array arrayvar[ ] ; *// массив с индексами от 1 до количества игроков*
- array arrayvar[n]; *// массив с индексами от* 1 *до* n
- array arrayvar[x, y] ; *// массив с индексами от* x *до* y
- array arrayvar[x, y, d]; *// массив с индексами от* x *до* y *с шагом* d

Язык программирования. Условный оператор

- if (*condition*) { *true_statements* }
- if (*condition*) { *true_statements* } *else* { *false_statements* }

- *Функция* if:
  - if(a, x, y) – *если* a *верно*, *то* x, *иначе* y.

Язык программирования. Циклы

- while(*condition*) { *statements* }
- repeat { *statements* } *while* (*condition*);
- Итераторы:
  - iterator(*varname*) *// от 1 до кол-ва игроков*
  - iterator(*varname, n*) *// от 1 до n*
  - iterator(*varname, x, y*) *// от x до y*
  - iterator(*varname, x, y, d*) *// от x до y с шагом d.*
  - Итератор создаёт таблицу *varname* с записями от x до y.

Итератор, пример.

- Два способа посчитать сумму квадратов 1+4+9+16+25 = 55 :
  - squareSum = iterator(i, 1, 5).sum(i*i);

или

  - squareSum = 0;

    iterator(i,5).do {

    :squareSum = :squareSum + i*i;

    // итератор имеет собственный контекст!

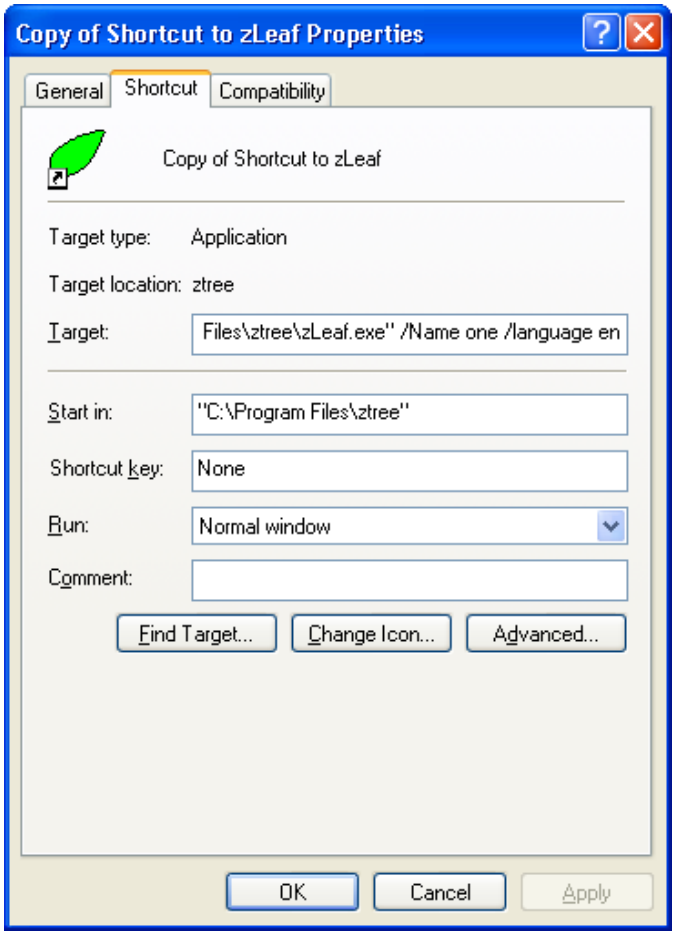
    }

Проведение эксперимента

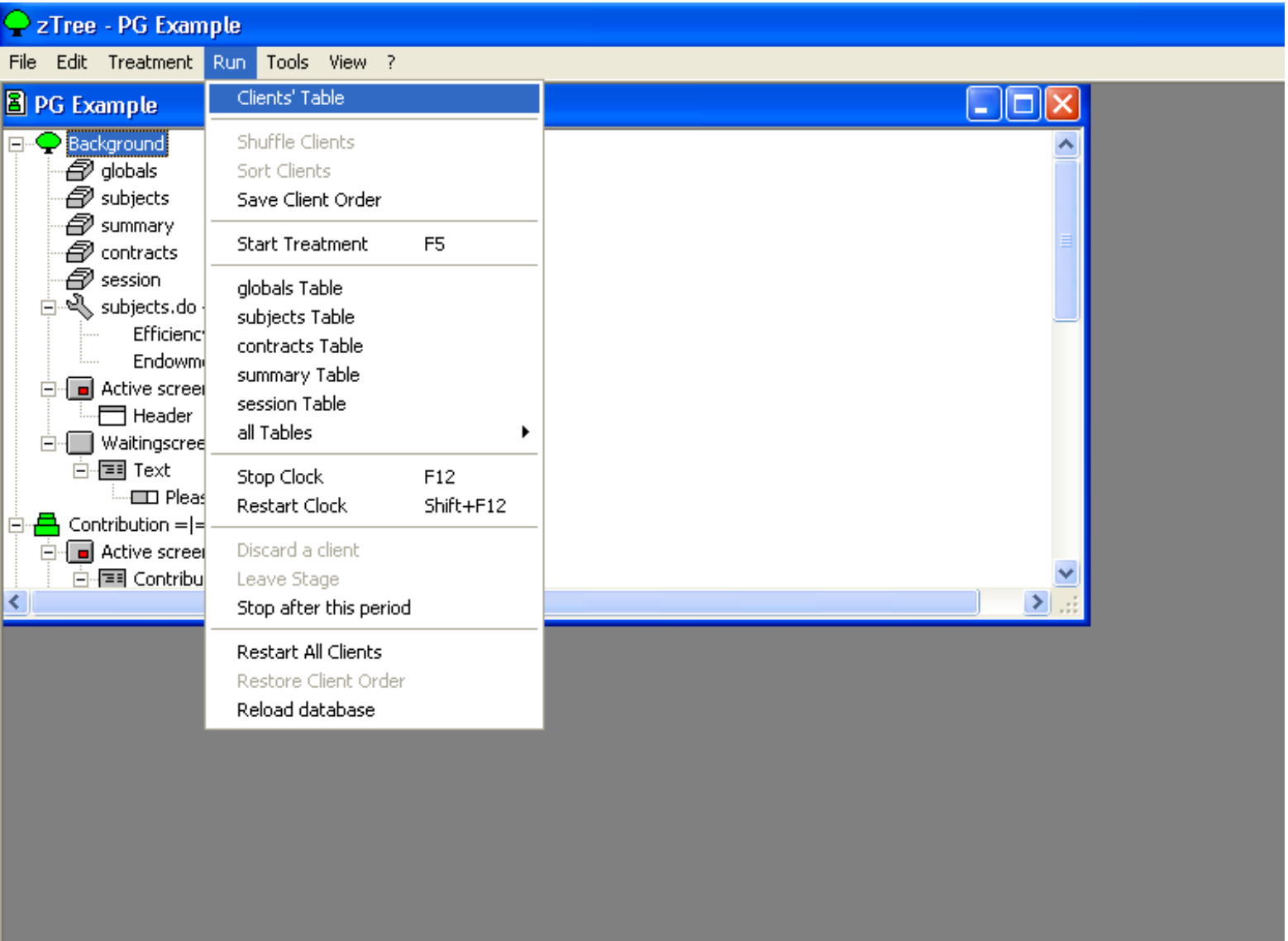
- Открываем zTree и файл с игрой на сервере Экспериментатора
  - Настраиваем механизм группировки (если нужно)
- Открываем zLeaf на каждом компьютере игроков
  - убедиться, что имена клиентов устанавливаются корректно при запуске zLeaf
  - Проверить, что все клиенты приконнектились
- Запускаем эксперимент
  - Следим за таблицей Subjects и окном Clients
- Запускаем анкетирование
- Сбор и обработка данных

Установка языка и имени клиентов zLeaf

- Создать ярлык zLeaf для каждого игрока
- В поле **Target** добавить:
  - /Name <имя> /language ru
- Давайте клиентам разные имена



Проверка, что клиенты присоединились: окно Clients’ Table



меню **Run** -> **Clients’ Table**

Сохранение данных игры

- zTree сохраняет все таблицы в файл <дата_время>.xls – дата и время запуска эксперимента.
- В файле <дата_время>.sbj – информация о клиентах игроков

Как тестировать игры самому?

- Открыть файл игры в zTree
- Создать столько ярлыков zLeaf сколько нужно для тестирования
  - только имена клиентам нужно дать разные
- Запустить игру в zTree как обычно, переключаясь между клиентами по Alt-Tab