z-Tree の contracts table の特性とその応用

飯　田　善　郎

要　旨

本稿は経済実験ソフトとして世界中で広く用いられている z-Tree の変数の特徴について，特に contracts table に注目して説明する。contracts table はオークションにおける金額のオファーやオファーの受託など，オークション実験でクライアントが随時行う行動を逐一記録するための変数テーブルとして実装されていた。昨今の z-Tree は機能拡張により被験者間のチャットや，画面上のオブジェクトの操作による意思決定をさせることが可能になっており，それらの結果の記録にも contracts table は適している。実験研究の進歩に伴って実験の設計が多様で複雑になっている昨今，contracts table の重要性が増しているといえる。本稿ではまずオークション実験のプログラムを例に contracts table の特徴とプログラムの際に注意すべき点を述べ，それらを踏まえて日本語でチャットをする際生じる問題の対処方法，インタラクティブグラフィックを用いた実験の記録に用いる際に注意すべき点について述べ，z-Tree を利用する実験者に便宜を提供することを目的とする。

キーワード：実験経済学，実験ソフト，変数，オークション，z-Tree

1.　初めに

z-Tree（Zunch Toolbox for Reattmade Economc Experlments）は，Konstanz 大学の Urs Fischbacher によって開発され，世界中で広く用いられている経済実験ソフトウエアである。（http://www.ztree.uzh.ch/en.html）サーバーマシンに z-Tree を，クライアントマシンにはクライアントソフトである z-leaf を起動させ，Windows のフォルダ共有機能もしくは TCP/IP によるネットワークで両者をつなぐことで，被験者実験に求められる様々な処理，例えば被験者への情報提示，入力受付，受け付けた入力データの処理，被験者へのフィードバック，実験結果の出力などを実現する。実験者は意図した実験を実現するために z-Tree 上で実験設計を行うが，経済実験に求められる多くの処理があらかじめ z-Tree の機能として組み込まれており，実験者が改めて学ばなければならない要素は非常に少ない。

しかしながら，変数概念だけはその特殊性ゆえに z-Tree 固有の特徴として理解しておかなければならない。z-Tree において変数はその性質ごとに異なる table 上にあるものとして区別され，実験者はプログラム上で変数を使用する際に，どの種類の変数を用いるかをあらかじめプログラムボックス上のプルダウンメニューから選択する必要がある。この変数の分類は，プログラミングの労力を軽減することを意図しており，実際多くの場合そのように機能する。しかし，異なる種類の変数間で数値をやり取りする場合，直接それが出来る場合と table 関数と呼

ばれる関数を介さなくてはいけない場合があり，その関係性の見通しは悪い。また，変数の値がどの時点まで保持され，どこでリセットされるか，あるいはどの時点でプログラムボックス内のステートメントが再実行されて変数が上書きされるかについても，開発者によって公開されているマニュアルに十分な説明がない状態である。そのために著者は主な4種類の変数table，すなわちsubjects, globals, session, summaryそれぞれの変数の基本的な概念と相互関係について整理し，提示した。（飯田，2006）

そこで扱わなかったのがcontracts tableである。これは，contracts tableが市場取引実験を意図して後から実装されており，実際の実験設計上でも他のテーブルとの関係の重要性は薄いと考えたためである。

しかしその後のバージョンアップに伴いz-Treeはチャット機能や，画面上の図などのオブジェクトを直接クリック，ドラッグすることによる操作等を実装してきており，それらの記録にはcontracts tableが適している。経済実験が複雑化，多様化している昨今，研究者が意図した実験を実現するためにcontracts tableの重要性が増しているといえる。そこで本稿は改めてcontracts tableに目を向け，その特徴とプログラム上の注意点について述べたい。

2. オークション実験に関連するz-Tree上の機能の概要

本稿ではまずダブルオークションのプログラムを例にcontracts tableの説明を進める。ダブルオークションはプログラム例としてz-Treeのホームページに公開されている。（http://www.ztree.uzh.ch/en/examples.html）なお，プログラムの仕組みの解説としてはマニュアルが十分に機能すると考えられるため，本稿ではあくまでそれをcontracts tableの特徴を説明するための例として用いる。

ダブルオークションにおいて売り手は販売希望価格，売り手は購入希望価格を他者に公開でき，その価格情報は売り手，買い手に分けて価格順にリスト表示される。被験者は自分の公開した価格に応じてくれる他者を待つことも，他者が示した購入・販売希望価格を受け入れて売買を成立させることもできる。こうした情報提示や売買の意思決定は被験者がそのためのボタンを画面上で押した瞬間に実行される。このやり取りをz-Tree上で実現する際，contract list boxとcontract creation boxをステージツリーに配置する。（メニューにはcontract grid boxもあるがこれは変数の説明という点では言及する必要が無いので省略する。）contract creation boxは，オークションでの価格提示で用いる。このボックスに入力受付アイテムと決定ボタンを設置することで，被験者が入力した数値をcontracts table上の変数に送る機能を実現する。入力された数値はcontract list boxを用いて表示する。contract list boxはリスト表示の機能を持ち，contracts table上の変数を，提示順，価格の高い順など，指示した順序でソートして表示できる。表示する情報の条件を付して特定の情報を表示しないように指示することもできる。また

contract list box には被験者が画面上のリストの中の数値をクリックしてからボタンを押すことで，他者の価格提示を受け入れるといった，画面上のリスト内の数値に対するアクションを可能にするボタンを設置できる。

囚人のジレンマや公共財実験のような，処理をシーケンシャルに行えばよいゲームを z-Tree で実現しようとするときは，ステージツリー上の入力ステージで戦略の入力を行わせ，次の結果表示のステージでステージの冒頭にプログラムボックスを配し，そこで結果を計算した上で表示すればよい。一方で，オークション実験では，価格提示や提示価格の受け入れといった被験者の意思決定は即座に計算して画面表示に反映する必要がある。contracts table 上の変数への数値の新規の代入や変更はしたがって，被験者の意思決定，すなわち操作があるやいなや行われなければならない。図 1 は被験者が売り手として価格提示を行う処理のステージツリーを表している。contract creation box（図上では contract maker と表示されている）にぶら下げられた入力アイテム「提示する販売価格：IN」が提示価格の入力を受け付ける。入力された値は入力決定ボタンを押した瞬間に contracts table の変数 Price に格納される。（contracts creation box の入力で受け付けられるのは contracts table 上で定義されている変数だけであり，また contracts table 上の変数の入力が出来るのも contract creation box においてのみである。Standard box 内の入力アイテムで contracts table 上の変数を入力先に指定しても無視される。）入力した変数以外の contracts table の変数の値は，ボタンアイテムの下にプログラムボックスを配置し，そこに変数の値を代入するステートメントを書くことで，ボタンが押されると同時に数値が代入される。図 1 では Seller，Buyer そして Creator の値がプログラムボックス（ステージツリーではスパナ型のアイコンで表示される）のステートメントを通じて contracts table 上の変数に書き込まれている。これらの内容の詳述は後の 4 節に行う。

なお，ボタンアイテムに付随したプログラムボックスで使用できる変数は contracts table に属する変数に限らない。例えば買い手が売り手の提示した価格を受け入れるボタンを押した場合，取引の成立にともなってその価格がオープンだということを示す contracts table 上のフラグ変数 Buyer の -1 の値を購入した被験者の被験者番号 Subject に書き換えるが，同時に得点の

図 1　contract creation box を通じた提示価格の入力とそれに伴う処理

ような，被験者が各自持っている変数も書き換える必要がある。通常被験者が各自個別に持つ値はsubject table上に置いて管理する。こうしたcontracts tableに属さないtableの変数もボタンに付随するプログラムボックス内で扱うことができる。

3. contracts tableの意義

contracts creation boxあるいはcontract list boxに属さないボタンにもプログラムボックスを付することはできる。その場合でもボタンが押されるやいなやボックス内のステートメントが実行され，数値が書き換えられる。すなわち操作にともなって各種の数値をリアルタイムで書き換えることは，他のテーブルの数値でも可能である。ならば，contracts tableの意義は何処にあるだろうか。contracts tableの意義のひとつはcontract list boxと連動して実現するオークションのコントロールの簡便さである。contracts tableに定義された変数の数値は，全ての被験者に共有され，contract list box上の表示に反映され，他の被験者の意思決定のアクセスに供される。subjects table上の変数を用いてこれを実現するには，ボタンを押した被験者の変数の数値だけでなく，他の被験者が持つ変数の数値を書き換えて回るプログラムを書かなくてはならない。globals table上の変数であれば全員が簡単に共有できるが，今度は被験者をグループに分けたい場合に，全員に同じ変数が共有されるというglobals table変数の特長が逆にネックになる。その点contracts table上の変数は，contract list boxのdisplay conditionの欄に必要な条件を加えれば，同一グループなど，特定の条件を満たす被験者の間だけで情報を共有したり，グループ内だけで共有した数値にアクセスさせたりすることが容易に実現できる。さらに，contract list boxを通じて表示することで被験者は受け入れるオファーを画面上のリストから直接クリックして選ぶことができる。これらの機能がなければ，実験者は各被験者に見せるべき情報を選んでリストして表示し，買い手や売り手がそのリストの値のどれを選んだかを把握し，その値を適切な変数に送り込むプログラムを自ら書かなくてはならない。

もう一つの，そしてきわめて重要なcontracts tableの特徴が，古い変数の値を保持しながら新しい変数の値を無制限に書き足してゆけることである。オークション実験で被験者たちが何回価格を提示するか実験者が事前に知ることはできない。他のtableの変数を使ってオークションをやろうとするならば，新しい価格が提示されても古い価格が記録に残るように，十分な数の変数を用意しておく必要がある。一方contracts tableは特定の数値の入力の度に新しく入力された数値の格納先を作るという仕様になっている。このため実験者は事前に行列変数などで格納先を確保したり，その格納先の数をどれだけ用意しておいたらいいかで悩んだりする必要がないのである。ただしこの仕様は実験者にとっては混乱のもととなる。その点について次節以降随時説明する。

4. contracts table におけるデータの記録のされ方

まず，実験結果としての contracts table 内の変数の推移を実験者がどのように把握するかという点から述べる。contracts table に現れる実験中の被験者の行動は，z-Tree のプルダウンメニューから Run>Contracts Table を選択することで専用のウィンドウが開き，随時観察することができる（図 2）。実験後に結果を確認するにあたっては z-Tree が実行終了にともなって各変数の値を書き出す xls ファイルを参照する。contracts table 上の変数の特徴はこのウィンドウや出力される表によく表れている。

subjects table の変数であれば，初期設定で宣言された変数名が列ラベルとして横に並び，被験者番号（z-Tree が各クライアントに自動的に付与して subjects table 上の Subject という変数に格納される）が行ラベルとして縦に並んだ表が埋められる形でファイルが書き出される。実験者はそれを見ることで，各被験者がどのような意思決定をしたか（特定の変数がどのような値をとったか）またそれらによって結果的に点数など，他の変数がどのような値になったかを知ることができる。

contracts table 上の変数の書きだされ方はこれと大分と異なる。変数名が列ラベルとして書き出されるのは同じであるが，行に決まったラベルはない。そもそも行ラベルは，被験者番号のように，あらかじめ行の数が決まっていればそれに対応して決めることができるが，contracts table の行は被験者が contract creation box を通じて変数に数値を新たに入力する度に書き足される。例えばダブルオークションで売り手の誰かが価格提示を行うとする。図 1 のプログラム例で売り手の被験者が入力アイテムから数値 Price を入力し，ボタンを押すと，その数値が新たに格納される場所として contracts table に記録の行が一行増え，そこに contracts table で利用が宣言されている他のすべての変数が代入される場所も作られる。これは xls ファイルの表では価格が入力される度に行が一行増え，変数 Price の列にその価格が書き込まれるという形で表れる。もしボタンに付属するプログラムボックスで Price 以外の contracts table 変数の値が記述されていれば，それらの値もその行のそれぞれの変数の列に書き込まれる。

subject table と違い，このとき，誰が Price を提示したかは自動的に記録されない。このため，誰がどのような役割で行動したかは，それを記録するステートメントを実験者が自分で書く必

contracts table

Period	Buyer	Seller	Price	Creator
1	-1	2	300	2
1	-1	4	280	4
1	-1	2	250	2

図 2　売り手 2，4 が価格提示を行っている場合の contract table の例

要がある。今の場合売り手が価格提示をしたので，その被験者番号を変数 Seller に代入する。図 1 のステートメント Seller=:Subject; がそれである。

z-Tree では変数の前のコロン「：」はスコープオペレーターと呼ばれ，これは「当該レコードの」を意味する。Subject は subject table 上に自動で定義される変数なので，この場合の当該レコードは「入力の決定ボタンを押した本人の」を意味し，ボタンを押した本人の subject table 上の変数 Subject の値がここに代入される。ここで注意すべきは，Subject が subjects table 上の変数であることである。contracts table 上のプログラムボックスから subjects table の変数を呼び出すときには，どの Subject の変数を呼び出すか指定しなくてはならない。この例ではスコープオペレーターがその役割を果たしているが，もしボタンを押した本人でない誰かの subject table 上の変数の値を呼び出したい場合は，後述のテーブル関数を用いてそれを呼び出す必要がある。

図 2 は，被験者番号（Subject）が 2 と 4 のプレイヤが売り手として価格 300，280，250 を提示した場合の contracts table の内容である。プレイヤ 2 は最初 300 を提示しているがプレイヤ 4 が 280 を提示したのでさらに 250 に値下げしている。（Seller が価格提示したときは Buyer の初期値は -1 で，これを，後に本当に買い手がついた時にその買い手の被験者番号に書き換える。こうすることで，ある価格（Price）での取引が成立した時，売り手は何番の被験者で（Seller）買い手は何番の被験者（Buyer）かが記録される。ダブルオークションでは売り手・買い手どちらも価格提示できるので，売り手，買い手，価格の記録だけ残すと，後から確認するときにどちらがその価格を提示したのかが分からなくなる。Creator はその取引で価格提案をしたのがどちらなのかの記録として機能する。）

このように contracts table では記録の軸は contract creation box を通じて入力された数値（chat box を用いる場合は chat box を通じて入力された文字列。これについては後述する。）である。subject table であればプレイヤである被験者の被験者番号を軸として記録されるため，「誰が」が記録されそこねることはあり得ないが，contracts table ではそうなっていないため，誰が主体として意思決定をした（変数に値を入力した）かは，実験者自身がきちんと記録に残るように配慮してプログラムボックス内のステートメントを書く必要がある。

5. contracts table 変数の留意点

z-Tree の変数の特徴は，変数名はひとつなのにそれが実は複数の変数を指すことがある，というところにある。subjects table では同じ変数名が，試行回数と被験者の人数の数だけ存在し，同じ名前でありながらそれぞれ別の変数として扱われるが（詳説は飯田，2006 参照），contracts table も同様の特徴と問題がある。ダブルオークションで複数の売り手が価格 Price を提示した場合，変数名自体は Price ひとつだが，Price という変数は代入された回数の数だけ存

在し，contracts table 上に並べられている。Price が新たに提示される度に contracts table は一行書き足され，初期設定で使用を宣言されている contracts table 上の他の変数を収める場所も新たに作られる。すなわち，他の変数も変数名自体はひとつなのだが，実は Price が被験者によって提示された数だけ（contract creation box を通じて数値が入力された数だけ）存在する。従って，contracts table 上の変数を操作したいと考えた時，どの変数を操作するかを指示する必要がある。

contract creation box を通じて数値を入力し，入力を確定するボタンにプログラムボックスを付している場合，そこで contracts table 内の変数を特に指示なく記述すれば，それは入力にともなって新しく作られる行に属する変数を意味する。例えば売り手がダブルオークションで contract creation box から価格 Price を入力するプログラムを作成するとき，ボタンのプログラムボックスにその価格がオープンである（買い手 Buyer が決まっていないことを示す）フラグとして Buyer=-1; と記入すれば，入力した価格 Price を含む新しい行が contracts table に書き加えられ，その行の Buyer の列にはその時点では -1 という数値が代入される。

被験者が contract list box 上のリストの数値のどれかをクリックした上で決定ボタンを押したとき，そのボタンに付属するプログラムボックスの中で contracts table 上の変数を特に指示なく記述すれば，それはクリックした数値が記録されているのと同じ行に含まれる変数を意味する。例えば上述の例で売り手がそれぞれ 250，280，300 の価格を提示していて，被験者番号 3 の買い手が contract list によって画面上に表示されている数値のうち 250 をクリックした上で購入決定するボタンを押したとしよう。このとき上述のように誰が購入者か記録を残すためにプログラムボックスに

```
Buyer=:Subject;
```

と記述してあったとすると，値が書き換えられるのは contracts table の Price が 250 の行の Buyer の数値である。（図３　最下行）

このように数値を入力する，あるいはリストに表示されている数値をクリックするというように，被験者が直接ある数値に対して働きかけをした場合，ボタンに付随するプログラムボッ

contracts table

Period	Buyer	Seller	Price	Creator
1	-2	2	300	2
1	-1	4	280	4
1	3	2	250	2

図３　売り手２の価格 250 が買い手３によって受け入れられた場合の例

クスでcontracts table上の変数を記述すれば，それは働きかけがあったその数値と同じ行の変数のことを指す。

ではそのような働きかけを行っていない変数を操作したい場合どうすればよいか。それを実現するにはcontracts.do{　}ステートメントと条件文の併用が必要になる。上述の例では売り手2は価格300を提示し，後にその価格では売れないと見て250に下げて提示している。300という価格は，250という価格が提示された以上，買い手には魅力がないため，ほとんど意味はない。それでも何らかの理由で高い価格提示の方を受け入れる買い手がいる可能性を考え，300という提示もオープンのまま残すとしよう。既述のように買い手がついていないことを表すフラグはBuyer=-1であるので，Priceが300の行でも250の行でもBuyerの値は-1だった（図2）。今被験者番号3の買い手が250をクリックして購入のボタンを押したとする。そのボタンにBuyer=:Subject;というステートメントがあるプログラムボックスがぶら下がっているので，ボタンを押すと同時にPriceが250の行のBuyerの値は，購入を決定した被験者の被験者番号に書き換えられる。これにより価格250の掲示はオープンでなくなる。しかし価格300のBuyerの値はまだ-1のままで，すなわちオープンなままである。売り手2は300もしくは250でその財を売ってもいいという意思表示をしたわけだが，その財はもう売れてしまったので，300の方の提示もクローズしなくてはならない。つまり300の行のBuyerの値も-1から別の値に書き換える必要がある。書き換える数値としてSubject番号を入れると，この財があたかも2回販売されたかのような記録になるので，提示したが受け入れられずに終わった記号として-2を記入するとしよう。問題は，300という数値には誰も働きかけをとっていないため，ボタンに付随するプログラムで単にBuyerと書いても，価格300を提示した時のBuyerの値にはアクセス出来ないということである。そこでcontracts.doというステートメントの出番となる。このステートメントは

```
contracts.do{　}
```

という構文で書き，z-Treeはこのステートメントに出会うと，contracts table上のすべての変数に対して，{　}内で記述されたステートメントを実行する。これを使って売り手2が提示した過去の価格をクローズさせるに，以下のステートメントを書く。

```
contracts.do{
  if( Seller == :Seller & Buyer == -1 ) {
    Buyer = -2;
  }
}
```

ここで :Seller は，買い手が価格 250 をクリックしてボタンを押したことで，価格 250 の行の Seller の値がピックアップされる。「：」がスコープオペレーターと呼び，「当該レコードの」を意味することを前にも説明した。変数が subject table 上にある時は「当該」は「ボタンを押した本人」を意味するが，contracts table 上にある変数の場合は，「リスト上でクリックされている行の」を意味する。つまりこのステートメントに行き当たると，z-Tree は contracts table 上の Seller の値をすべて見て回り，250 の価格提示をした A の被験者番号と一致する行かどうかを判断する。すると被験者 2 が行った 250 と 300 の価格提示の行がその条件をクリアする。ついでそのデータの中でまだ買い手がついていない，すなわち Buyer の値が -1 になっている行かどうかを判断する。価格 300 の行が両方の条件を満たす。するとこのステートメントによってその行の Buyer の値は -2 に書き換えられる（図 3　3 行目）。

このように，アクティブでない contracts table 上の変数を書き換えるには，どの条件を満たす行で実行するのかを条件文で示した上で，書き換えの実行内容を書いたステートメントを，contracts.do を用いて全ての contracts table 上の変数に対して実行することが必要になる。

6.　table 間の相互参照

contracts table のプログラムボックスから他の table の変数を参照する際には，常にどの変数を指しているのか明確にする必要がある。globals table は全ての被験者に共通の変数を格納する場所なので，そこで変数が定義されていれば，その値はひとつしかない。したがって，単に globals table の変数名を書けばその値が参照される。

contracts table のプログラムボックスで subjects table の変数の値を参照する場合には，被験者がたった一人でないかぎり，table 上のどの Subject の数値かを指示する必要がある。参照するのがボタンを押した本人の変数であるならば，スコープオペレーターをつけるだけで subjects table 上の変数を直接参照できる。本人ではない誰かの subject 変数を見つけてくるには，テーブル関数を用いる。すなわち，特定の条件を満たす被験者の subjects table の変数を参照するには subjecs.find（条件，変数）と記述する。（テーブル関数についてはマニュアルおよび飯田，2006 を参照されたい。）contract creation box，contract list box につけたボタンには，contracts table のプログラムボックスだけでなく，subjects table，globals table のプログラムボックスもぶら下げることが出来る。これらのボックスの中で contracts table の変数を参照することも，テーブル関数を用いることで，可能である。その場合にはもちろん，複数ある変数の値のどれを参照するのか，テーブル関数を用いて適宜指定してやる必要がある。

7.　contracts table 内の変数の初期値に関する注意点

contracts table の初期値について注意するべき点がある。通常，変数の宣言と初期値の設定はステージツリーの先頭にある Background アイコンに各テーブルのプログラムボックスを配置し，その中で記述することで実現する。例えば subjects table のプログラムアイコン内で

```
Score=0;
```

と記述すると，subject table の中に被験者の数だけ Score という変数が用意され，その値はゼロにセットされる。contracts table の場合，利用する変数を Background であらかじめ宣言し，初期値を設定しても本当にその値にセットされるわけではない。

contracts table の変数は，そもそも contract creation box 内で新しい数値が入力されて，それに伴って新しい記録の行が追加された時に初めて記入できるようになる。変数の中身はボタンに付したプログラムボックスで値が記されていればそれが代入されるが，それをしなければ空欄となってしまう。空欄の変数の数値を参照した場合，基本的にはゼロが返ってくる。既述の例のように，売り手が価格を提示するときに，まだ買い手がついていないことを表すフラグとして Buyer の初期値を -1 にするとしよう。そのために background のプログラムボックスで Buyer=-1; と宣言してあったとする。しかしこの初期値の設定は，変数の宣言という意味はあるが，初期値の設定という意味では機能しない。新しい価格が contract creation box を通じて入力されたとき，z-Tree は新しく提示された価格 Price の値を入れる行を作るのに付随して background で使用を宣言されている Buyer の値を入れる場所も作る。しかしその初期値が -1 なのだと background まで戻って確認し，代入するということをやらないのである。従って実験者はボタンにプログラムを付してその中で Buyer=-1; と初期値を改めて設定しなくてはならない。このように正しく中身を定義してあれば問題ないが，それを怠って Background で宣言した初期値が入っているはずだと思い込んで contracts table の変数の数値を参照しても，意図せず未定義のままということになる。

8.　インクリメントが出来ない理由と実現する方法

contracts table 上の変数で，数値が従前の値から積み上がってゆくというような計算をする際には注意が必要である。例えば新しい価格が提示される度にその価格提示へ 1，2，3…という番号をつけてやろうと考え，OfferID という変数を contracts table 上に作ったとする。番号の割り付けはボタンプログラムに

OfferID＝OfferID＋1;

というステートメントを書いてやることで実現しそうに思える。しかし実際にはうまくいかない。最初の価格提示の際には，OfferID の値は特に設定しないかぎりゼロなのでそこに 1 が加算されて OfferID は 1 になる。そして次に新しい価格が提示され，contracts table にその価格を含む変数を記録する行が作成された瞬間，そこにまた別の新しい変数 OfferID が作られる。新しい OfferID の値はひとつ前の OfferID と同じでやはりゼロになってしまう。このためこの書き方では OfferID は何度価格提示が行われてもゼロに 1 を足すことを繰り返すだけなのである。

これを解決する方法はいくつかある。最も手間がかからないのは，インクリメントを他のテーブルの変数で行い，その値を contracts table の変数に渡す方法である。初期設定で globals table に OfferCouner という変数を用意し，価格提示のボタンに globals table のプログラムを付して，そこに

OfferCounter＝OfferCouner＋1;

というステートメントを書いておく。globals table の変数は宣言したものが一つだけで，そのピリオドの間はリセットされないので，ボタンが押される度にインクリメントしてゆく。この globals table のプログラムボックスのすぐ下に contracts table のプログラムボックスを付けて，そこに

OfferID＝OfferCounter

と書く。OfferCounter という変数は globals table 上にひとつしかないので，単に変数名を書くだけで，contracts table から globals table の変数を参照できる。こうしてボタンが押される度に増えてゆく番号を格納した OfferID が実現する

また，テーブル関数を使って，直前の値を見つけてきて OfferID においておいた上でインクリメントするという方法もある。すなわち，contracts table のプログラムボックスに

OfferID＝maximim(OfferID);
OfferID＝OfferID＋1;

このように書くのである。maximum は該当する変数の最大値を返すテーブル関数である。最初の価格提示がされたとき，OfferID はどこにも記録がないので，1 行目で最大値としてゼロが返され，OfferID はゼロになる。2 行目でそのゼロに 1 が足され，1 回目の価格提案の

OfferID は 1 になる。2 回目の価格提示が行われると，既存の OfferID の最大値は 1 なので 1 行目はこれを取り出して新しい OfferID に代入する。2 行目でそこに 1 が加算されるので 2 回目の提案の OfferID は 2 になる。この繰り返しである。このように contracts table でもインクリメントの式を書くことはできるが，「直前の値を探してきて代入しておく」という作業が必要になる。これは同じ変数名でも contract creation box を通じて値が代入される度に新しく別の変数が作られていて，新規の変数はその前に作られた変数と同じ変数名でも別のものであるという，contracts table の変数の特性上必要になるわけである。

9. グループ分割について

多くの経済実験において，被験者は特定の人数ごとにグループに分けられ，相互に戦略を決定する。一般的には subject table 上に Group という変数を用意し，被験者それぞれにグループ番号を割当てる。割当ては実験者がそのためのステートメントを書いてもいいし，z-Tree のメニューから割当てを指示することもできる。無論，これだけで z-Tree が被験者を Group で切り分けて計算してくれるわけではないので，利得計算など他者の戦略が必要なときは他者の戦略を探すテーブル関数に Group==:Group という条件文を付して，同じグループ内の数値に限って計算を行う。contracts table においてグループ分割をする際に留意するべき点は，プログラムのステートメント，あるいはメニューから subject table 上の Group の割当てを行っても，それは subject table 上のことで，contracts table には引き継がれないということである。従って何もしなければ，価格提示の際に別のグループに自分の価格が提示されたり，他グループの価格が自分の画面に提示されるということが起きる。これを避けるためには，まず subject table 上の変数 Group を正しく割り当てた後，contracts table 上のグループ変数として別の変数，例えば ConGroup を作り，価格入力のボタンに contracts table のプログラムボックスをぶら下げ，そこに

```
ConGroup=:Group;
```

と書く。これで価格入力ボタンを押すたびに ConGroup にはボタンを押した本人のグループ番号が引き渡される。そして価格を表示する contract list box の Display Condition に

```
ConGroup==:Group
```

と書いておく（似ているがこちらは = が二つ並んでいて，代入ではなく条件式であることに注意されたい）と，各クライアントの画面には当人のグループ番号と同じグループ番号の誰か

が提示した価格しか表示されなくなる。被験者は画面に表示されている価格にしかアクションを取れないので，価格の受け入れがグループをまたいで起きてしまうという問題もこれで防がれる。

10. チャットについて

contracts table は被験者同士でリアルタイムの数値のやりとりを可能にする。この数値がメッセージになれば，リアルタイムのメッセージのやり取りになり，チャット機能が実現する。z-Tree 3.4 から，変数でストリングが扱えるようになり，chat box が実装されることで，z-Tree 上で被験者同士にチャットをさせることが可能になった。

chat box はボックスの下辺に入力スペースが有り，入力したメッセージがボックスの上辺から入力された順にリスト表示される。表示メッセージにはメッセージの発信者など，各種のラベルをつけることもでき，同一グループ内のメッセージだけ表示するというような，表示内容の制限も可能である。

ただしこの chat box は，日本語で利用しようとするときやや煩わしい仕様になっている。マクロソフト日本語 IME では，スペースキーで変換，エンターキーで確定というキーの初期設定になっていて，多くの日本人はそれに慣れている。ところが z-Tree のチャットはリターンキーでメッセージを発信する仕様になっているため，ひらがなを入力し，スペースキーで変換，一旦確定のつもりでエンターキーを押すと，文章の途中でもエンターキーに反応してメッセージが送信されてしまうのである。

この問題を解決するには，ちょっとした工夫が必要になる。決定ボタンを押さないと入力した値が送信されず，値の送り先が chat box と同じ contracts table である box すなわち，contract creation box を使ってメッセージの入力を受け付け，chat cox はチャット内容の表示にだけ使うのである。

chat box を表示専用に使うには，box 内の Input var を空欄にすればよい。（図 4，①）

contract creation box を文章の入力に使うには，アイテムの Input にチェックを入れ，Layout 欄に

```
!string
```

と入力することで可能になる（図 5）。

ところで z-Tree と Windows のバージョンアップは，しばしば日本のユーザーに日本語文字化けの問題を突きつけてくる。チャットが使えるバージョン 3.4 以降で現在公開されているバージョンは，item のラベルを日本語で書くと文字が化けて表示されるという問題がある（図

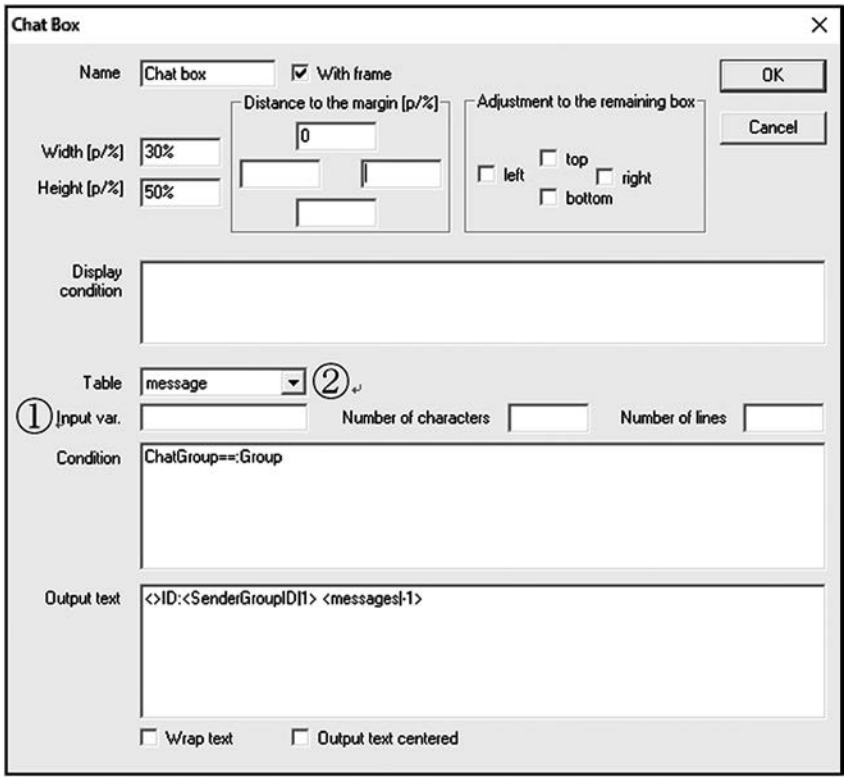

図４　日本語チャットに適した設定１：**chat box**

5 の Label 欄）。自分が書いたものが書いた瞬間には読めないため，不便があるが，実験中のクライアント画面では正しく表示される（図 6）。実験の遂行自体は問題なく可能なので，チャット機能が必要な場合はこれを受け入れるしかないのが現状である。チャット機能が特に必要なければ，この問題がないバージョン 2.1 や 3.3 も公開されているので，そちらを使う方がいいかもしれない。

デフォルトでは chat box を通じて入力されたテキストは contracts table に記録されてゆく。これはすべての記録を残す必要があり，事前にどれだけ記録スペースが必要かはわからないという点でピットマーケットの価格提示と同じであることから自然であろう。しかし，チャットさせつつ同時に contracts table に別の記録が必要な実験を行う場合，チャット内容と別の内容が同じ contracts table に書き出されるため，後で分離せねばならず，やや不便である。この問題は自分で新しく table を定義し，その table にチャット内容を書き出すことで回避できる。z-Tree はデフォルトの table のほかに，メニューの Treatment->New Table で実験者が独自の Table を作成できる。これを用いて例えば message という table を作成し，設定項目を contract table と同じにする（図 7）。そのうえで chat box の Table 欄に message を指定し，（図 4，②），同じくチャットで使う contract creation box の Table 欄も同様に指定すれば，message table に

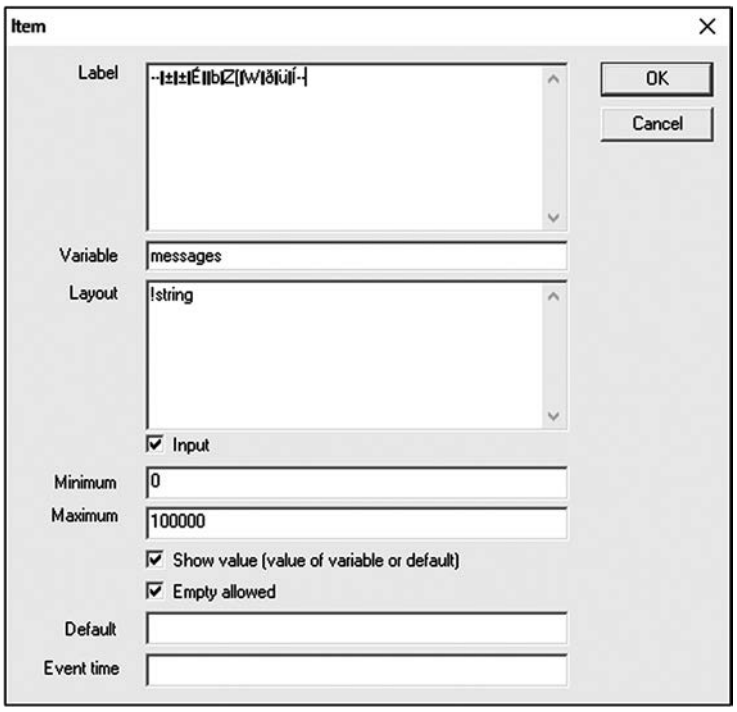

図５　日本語チャットに適した設定２：**contract creation box** の入力アイテム

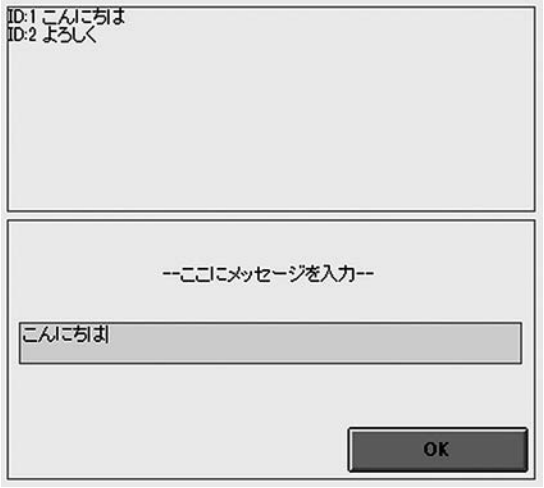

図６　日本語チャット画面例

チャットの内容が書き込まれる。こうすることで xls ファイル上でも contracts table と message table は分けて書き出される。このような日本語チャットの設定を施したプログラム例を筆者のウェブページに掲載している。（http://www.cc.kyoto-su.ac.jp/~iida01/sub3.htm）

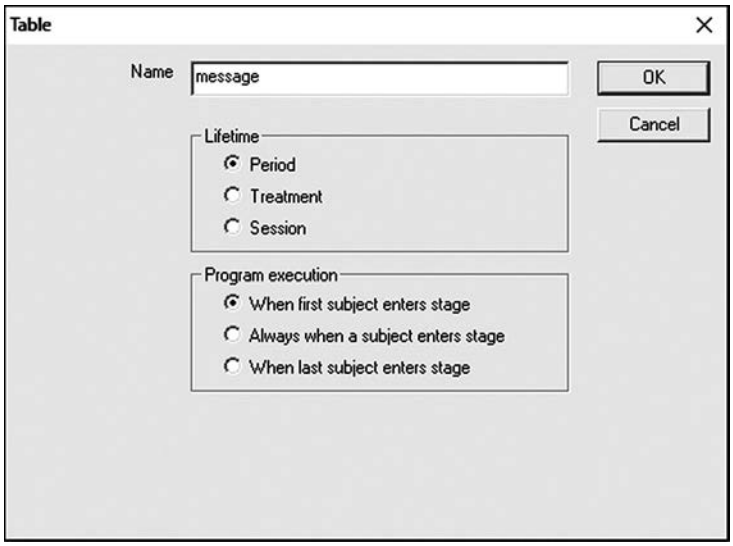

図7　新規の table の設定

11. インタラクティブグラフィックと contracts table

z-Tree には plot box 内に線や多角形や円のグラフィックを描く機能があり，さらに比較的新しい機能としてそのグラフィックを被験者が直接クリック，ドラッグして操作する機能が実装されている。これは2次元の戦略空間における自分と相手の戦略を直感的に把握させ，その上で戦略を選ばせたいときに有意な機能である。

この機能として New, Select, Drag という微妙に異なる3種類が用意されており，plot input box の，Action オプションでどれを使うかを選択する。

Action オプションの最初にある New は，ラジオボタンで選択すると x, y 座標の変数の入力欄と，subjects と contracts のテーブルの選択のプルダウンメニューが現れる。変数の入力欄に変数名を入力しておくと，実行時に被験者が画面上でクリックした場所の座標がその変数に送られる。このオプションを選んだ場合，被験者の入力を受け付ける plot input ボックスは表示するオブジェクトではなく plot box に直接ぶら下げなければならない。この仕様から，New は特定のオブジェクトへの操作を目的にしているというよりは，plot box 内のどの点をクリックしたかを伝えるのが機能の主旨だと考えられる。ただ，このマウスでクリックした場所をプレイヤのオブジェクトの座標に送るステートメントを加えると，クリックした場所に自分のオブジェクトが移動しているように見えることから，被験者にあたかも画面上でオブジェクトを操作しているかのような感覚を持たせることはできる。

次の Select は，特定のオブジェクトの下にぶら下げて設置し，x と y の入力欄に，そのオブ

ジェクトの位置を入力しておくことで，画面上でオブジェクトをクリックしてマウスポインタを動かすと，マウスのボタンを離した瞬間にポインタの位置にオブジェクトが移動する。Drag は，Select より必要な設定が多いが，オブジェクトをクリックしてマウスを動かすと，マウスポインタにオブジェクトが追随するので，より直感的にオブジェクトを動かしていると実感できる。Select と Drag の plot input ボックスはどちらも複数のオブジェクトにそれぞれつけることができるので，画面上の複数のオブジェクトを自分でクリック，ドラッグして操作するというような使い方もできる。

例えば被験者が画面上のどこに自分のアバターを置くかで自分の戦略を決定するが，実験終了の時間が来るまではいつでも何度でも自由にそれを変えることができる実験を行うとしよう。そのうえで被験者のすべての画面上の操作を記録したいとする。その場合も事前に被験者が何回アバターを動かすか分からないので，記録先として contracts table もしくは自分で定義した互換 table を用いる必要がある。またそうする場合，contract creation box や chat box 通じての記録ではないので，新しいアクションが被験者によってなされるたびに contracts table にそれを記録する行の追加するプログラムが必要になる。plot input ボックスにはプログラムボックスをぶら下げることでクリックやドラッグするたびに特定の作業を行わせることができる。（ボタンにプログラムボックスをぶら下げることで，画面上のボタンを押した瞬間に実行される命令を書くことができるが，それと同様である。）そこに contracts.new{ } と記述することで contracts table に新たに 1 行記録スペースが作られ，{ } 内のステートメントが実行される。そこで contracts table 内の変数に被験者番号とアバターの位置，操作の瞬間の時間を引き渡し，記録することで，被験者が行ったすべての行動を記録に残すことができるのである。図 8 はそのような処理を行うプログラム例，図 9 はその実行画面である。（このプログラム例も上記の筆者の Web ページに掲載している。）一点注意すべきはこの contracts.new{ } を contracts table ではなく subjects table のプログラムボックスで使う必要があることである。これはそもそも仕様として contracts table 上で contracts.new が機能しない仕様であるためだが，subjects table のプログラムボックスの中で実行することによって，スコープオペレーターだけでクリック，ドラッグした当該被験者の subject table 上の数値を拾って contracts table に引き渡すことができるという利便があることを考えると，意味のある仕様だといえるだろう。

12. 終わりに

z-Tree の変数 table のひとつである contracts table の性質とそれに由来・関連する注意点について述べてきた。contracts table の特徴であり，またそのために理解が難しいのが，subjects table において同じ変数が複数の異なる変数を表すというのとは違う意味で同じ変数名が複数の変数を指すことだろう。subjects table では同じ変数が被験者全員に与えられるため同じ名前

```
Object: ObjectX,ObjectY
  drag ( (xstart, ystart) , (xmouse, ymouse) (xobject, yobject) )
    subjects.do { ... }
        ObjectX=round(xobject-(xstart-ObjectX),1);
        ObjectY=round(yobject-(ystart-ObjectY),1);

        SMoveID=SMoveID+1;
    subjects.do { ... }
        contracts.new{

        ConSubject=:Subject;
        ConMoveTime=:SMoveTime;
        ConX=:ObjectX;
        ConY=:ObjectY;
        ConMoveID=:SMoveID;

        }
```

図８　オブジェクト操作と記録のプログラム例

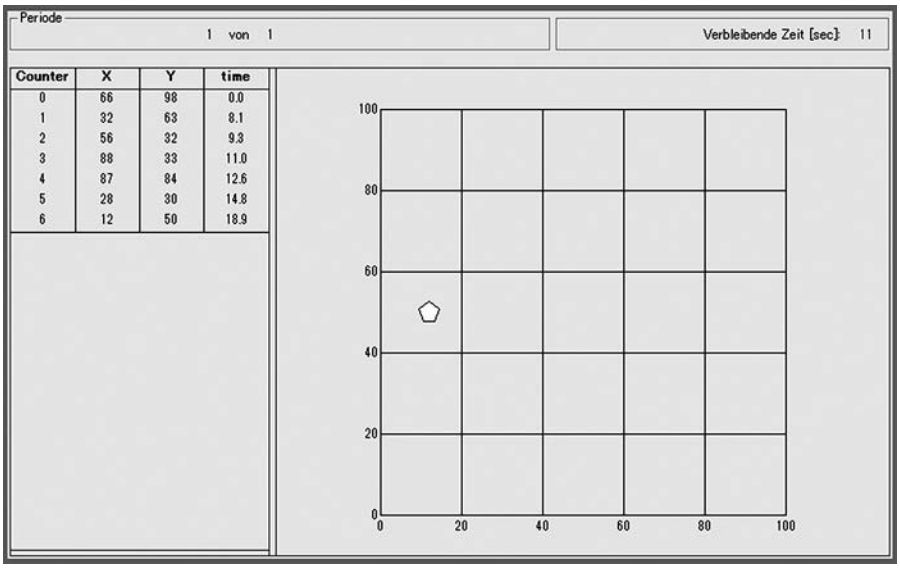

図９　オブジェクト操作の実行例

でありながら被験者の数だけ変数があるという状況が起きるが contracts table では被験者が contracts creation box や chat box を通じて新しいアクションを取るたび，あるいは contracts.new が実行されるたびに同じ名前の変数が増えてゆくのである。それゆえにステートメントを書くときにある変数名を記述したらそれがどの変数を意味するのかをプログラムする側は常に把握し，望む変数を参照するためにどのようにそれを指示するのかに通じている必要がある。

他の table から contracts table の変数を参照する際にももちろんこの点に留意する必要があ

り，また contracts table の変数に他の table から数値を参照するときにも同様に留意が必要である。

逆に言えばこの点を把握さえすれば，contracts table にまつわるプログラミングの困難さはかなりの部分が排される。z-Tree の設計者が意図した変数の概念をイメージとして理解できれば，それはほぼ達成できていると言っていいだろう。

11 節で述べたインタラクティブグラフィックの機能は，被験者が実験時間内に自由に戦略を選び，その結果がリアルタイムで報酬に反映するような，連続時間実験に応用できる。ただし，これは z-Tree にとってはすべての瞬間で被験者の行動を監視し，画面を書き換える作業を続けることになるため，非常に負荷が大きくなる。遅滞のない実験を実現するには，サーバーとクライアントの能力と通信環境に十分なゆとりが必要なうえ，不必要な負荷をできるだけ避ける設計を心がける必要がある。これらのノウハウについては検証途上の要素が多いため，今後知見を蓄積し，まとめてゆくこととしたい。

参考文献

「経済実験ソフト z-Tree の変数概念」2006，飯田善郎，京都産業大学論集．社会科学系列 23, 147–160

Features and applications of the contracts table in z-Tree

Yoshio IIDA

Abstract

This paper explains features of z-Tree variables, focusing on the contracts table, which is one of five types of variable tables. The contracts table contains variables and was initially implemented to record subjects' offers and decisions within auction experiments. z-Tree has been expanding functions. The contracts table is suitable for recording the results of some of the expanded functions, such as online chat and interactive graphics. As experimental design becomes increasingly diverse and complicated, the importance of this table increases. First, this paper explains features of the contracts table and how to apply these features to recently expanded functions. Second, the paper discusses ways of coping with problems with chatting in Japanese and the points that programmers should note when they use the contracts table for recording the results of continuous-time interactive game experiments.

Keywords: experimental economics, software for experimental economics, z-Tree, variables, auction