教育用ネットワーク・ゲームの一構成モデル

片　山　益　男

Development of a network-game model for education

KATAYAMA Masuo

要　約

インターネットを利用した教育用の競争型ゲーム（制約資源配分を含むもの）を実現するための１メカニズムを考案し，Windows IIS webサーバーと，ASPスクリプトを用いて，その動作を確認した。参加者のアクセスのみで自動的にゲームを進行できる特徴を持つ。そのメカニズムの内容を中心に説明した。

Abstracts

This paper presents a mechanism of network-game for education. Its characteristics are: 1) uses popular web servers, 2) allows constrained resource allocation model, 3) can play with no human assistance except game players. The mechanism is confirmed to work by Windows IIS web-server and ASP scripts. This paper explains the details of its mechanism.

1　はじめに

複数のプレイヤーが制約資源の獲得を競うという本来の意味でのゲームを，ネットワークを用いて実現するという試みは決して新しいものではない。例えば，公衆電話回線によるデータ通信が開始された1970年代に，電電公社がDEMOSシステムの１ライブラリとしてビジネス・ゲームを提供していた。比較的最近の例としては，シンガポール国立大学で開発されたMAGNASがよく引用されているようである［1］。

しかし，それらはすべて完成品がgivenとして提供されているもので，自らその種のゲームを開発したいと思っても，筆者にとっては，それに関する情報を得ることができないまま今日に至った。インターネットの普及により，「信長の野望」といった人気ゲームのオンライン版も提供されるようになったが，その作成方法はブラックボックスのままであ

る。最近，多数の人々が参加できる，大規模な戦闘型オンライン・ゲームの構成法についての書物が出版されるようになったが［2］，少なくとも即時応答の必要度という点で，ここで想定した教育用ゲームのタイプとは異なっている。

webアプリケーションの基礎について，先に発表した「web利用自主学習システムの一構成モデル」［3］での経験をふまえ，多少の時間的ゆとりができた昨今，ネットワーク・ゲームの1プロトタイプをwebアプリケーションの一つとして作成することができたので，ここに報告する。

本稿の主旨は，インターネットを利用した教育用ゲームを手軽に構築するための一手段を提案することであり，ここでの中心的な内容は，コンピュータによりゲームを進行させるメカニズムの説明である。このメカニズムの動作確認のため，Windows IIS webサーバーと，ASPスクリプトを用いた。そのときのゲーム・コンテンツとして，既開発の「レコード店の経営」［4］を用いたが，それを組み込んだ内容説明は別の機会に譲る。

2　ネットワーク利用によるゲーム進行メカニズムの概要

今回開発した，ネットワーク利用によるゲーム進行メカニズムの概要をまず説明する。

1）ゲームの実施期間（サイクル数）を最初に設定する。例えば，3年の期間を想定し，1ヶ月単位でゲームを進行させるとすれば36サイクルとなる。

2）参加者を事前に確定する。すなわちゲーム開始後の追加参加を認めない。参加人数の上限は設けていないが，ゲームの内容とサーバーの能力から，数人～数十人を一応の想定範囲としている。

3）1サイクル内のゲーム進行を，（意思決定値の）入力モードと（実施結果の）参照モードに2分割し，各モードへのアクセス期間を設定する。各プレイヤーは入力モードの期間内に，意思決定値を入力（送信）する。その実施結果を参照モードの期間内に問い合わせる。

データを入力したとき，入力データの確認内容と，問い合わせのためのページが返送される。そのページを用いて問い合わせると，ゲーム実施結果と，次回の入力様式が返送される。

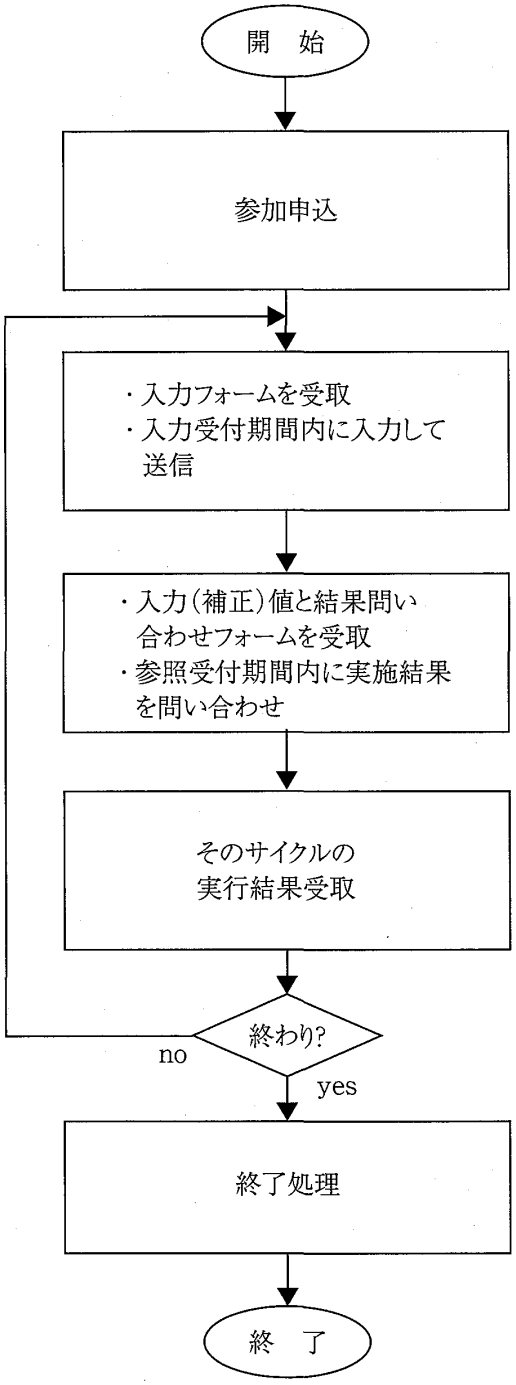
4）各モードのアクセス期間内にアクセスしないと，そのプレイヤーはゲームから脱落したと見なされ，以降のゲーム進行に参加できない。各モードのアクセス期間の長さは，ゲーム開始時に自由に設定できる；例えば1教室内で実施するときは5分，地理的に離れたプレイヤー間で実施するときは半日単位というように。（ゲーム実施途中で変更すること

も可能である。）

5）各プレイヤーの入力データや実施結果の値のうち，ゲーム進行と結果評価に必要なデータすべてをサーバー内に記録し，それらを用いて，制約資源の配分計算などをサーバーで行う。

以上に説明した，ゲーム進行の流れを図１に示す。

図１　ゲーム進行の流れ



ここでは1サイクルの内容を2段階に分割した場合について説明したが，ゲーム内容によっては，さらに多段階に分割することが必要な場合も生じるであろう。その場合でも，ここで説明するメカニズムをそのまま利用することができる。

次章で，このようなメカニズムを構築するに到った経過を説明し，4章で，モデルの詳細について説明する。

3　ネットワーク・ゲームの構成法を考案するまで

一口にネットワーク・ゲームといっても，多様な種類が想定され，各々の特徴により実現システムの構成法も大きく変化する。ここでは，今回想定したビジネス・ゲームのシステムをどのようにして具体化していったか，その試行錯誤過程を整理した結果を次に説明する。

3.1　ネットワーク・ゲームの特性

ネットワークを用いたゲームという場合，複数のプレイヤーがゲームに参加し，各プレイヤーの振る舞いに相互干渉が存在するという特徴を持たせることが基本的な要件になろう。本来，単独で楽しむ構造になっているゲーム・ソフトを，ネットワークを通じて配信する場合を含めて，ネットワーク・ゲームと呼ぶこともあるが，ここでは対象としない。

プレイヤー間に相互干渉が存在するゲームとして，囲碁・将棋・トランプ・チェスなどの対局型ゲーム，応答スピードを競う戦闘型ゲーム，有限資源の争奪を競うビジネス・ゲームなどを列挙できるが，今回は「教育」を主目的にしているので，とりあえず有限資源の配分をめぐるゲームを対象として取り上げることとした。典型的な例として，総需要一定の下で売上を競うもの，資材や資金の供給に制約があり，その確保を競うものなどがある。

教育を目的とする場合，ゲームを競うことにより何を教育するのか，ゲームに組み込まれた規則が教育目的に叶ったものであるかということが最も重要な事柄となる。ゲームを教育に用いることの有効性について，筆者の仮説を前に記した [5]。しかし，ここではネットワーク・ゲームをどのように構築すればよいかの探索を主目的としたので，教育目的への適合性については検討の主対象としなかった。その問題は，ゲーム・コンテンツの問題として別に扱うことができるからである。

3.2 ゲーム・モデルの限定

制約資源を複数の参加者にどのように配分するかというタイプのゲームということに対象を限定しても，その具体化には，さらに多くの選択肢がある。

（1）ネットワーク・システム，サーバーの種類

今日ではインターネット利用というのが，ごく自然な選択肢であるが，TCP/IPプロトコルを使っていても，閉じたLAN内でVisual Basicなどを使用して，ゲーム専用のサーバーを設定している例もある。

ここではwebアプリケーションの１つとしてwebサーバーの利用を前提とした。

（2）進行の時間管理

ネットワーク・ゲームの運用において，筆者が最も関心を持った項目の一つは，進行の時間管理をどうするかということであった。

速さを競うのが目的のゲームでは，誰かがアクセスすると，それを優先処理するという論理も許されるが，資源配分型のゲームでは，各プレイヤーが意思決定を行い，それらのデータを総合(プール)して資源配分が行われ，その結果をもとに各々の成果が計算され，それが次のステップの意思決定に利用されるという，一連のステップを繰り返すことにより進行させるのが典型的なパターンとなる。例えば複数社から見積を取り，最低価格のものを選択するという場合，〆切期限まで入力値を保持しておく必要がある。そのパターンを前提とすると，各意思決定結果のデータの提出に共通の〆切時刻を設定し，その時刻が過ぎてから資源配分計算を行うことが必要になる。そこで「〆切時刻が来たから，配分計算を行います」というトリガーの発生をどのような方法で実現するか，その手段を考える必要がある。

誰もが考えつく最も常識的な方法は，所定の時刻が来たら，自動的に配分計算を行わせるプログラムを作り，常時実行させておくという方法であろう。しかしwebアプリケーションで，そのようなプログラムを作成する方法を筆者は知らない。次に思いつくのは，進行管理を人間が分担する方法であろう。教室など，一個所に集まって一斉に実施する場合は，進行係の合図でタイム・コントロールが容易にできる。しかし，ネットワークを介して地理的に離れた場所にいるプレイヤー達に，どのような方法で合図を送ればよいのか？〆切時刻を予め設定しておき，時刻が来たら配分計算のプログラムを人間が起動するという方法も，次善の策として許容出来るかも知れない。しかし進行係の心的な負担は大きく，さらに，１ステップ半日というような長い時間かかるようなゲームでは非実際的な方法となる。やはり，自動的に進行できる方法が必要になる。

筆者が考えついた解決策は，ゲームのプレイヤーがトリガー役を兼ねるという方法であ

る。すなわち，プレイヤーがゲーム進行のために（サーバーに）アクセスしたとき，もし〆切時刻が過ぎておれば，ステップが1つ進んだという処理を自動的に行わせるようにする方法である。それにより，プレイヤー以外に人間の介在無しにゲームを進行させることができる。

ステップの進め方に関しては，〆切時刻を待たずとも，参加者全員の応答が有った時点で次に進めるという方法も可能である。しかしプレイヤーが地理的に離れているとき，「次に進みます」という合図をどのようにして送れば良いのか，そのための良い方法が思い浮かばないこと，プレイヤーが絶えず次のステップ開始に注意していることが必要になることなどから採用しないこととした。その代わりに，プレイヤーの習熟向上などにより，途中で〆切時刻を変更できる方法を含めることとした。その場合，現ステップの〆切時刻を常に表示しておく必要がある。

（3）参加のタイミング

ゲームへの参加を，いつでも自由に行えるようにするか，ゲームを開始すれば終了まで追加参加を認めない方式とするかにより，ゲームの構成方法がかなり変化する。利用者側から見れば前者の方が望ましいと思われるが，ゲームの構成方法（コンピュータ側で準備すべき運用規則）を設計する立場からは，かなり複雑になることが予想される。例えば，あるプレイヤーが参加した期間の成果をどのように評価するのか，他のプレイヤーとの成果比較をどのような方法で行うのか，参加のタイミングと資源配分計算のタイミングをどのように取り決めるのか，などの規則づくりが複雑になる可能性が高い。それらの設定条件を基にして，あるプレイヤーの参加から退場までに，どのような種類のデータを記録しておき，成果計算のときに使わねばならないかの見通しを立てることが必要になる。

実際に前者の方式を検討したことがないので，複雑性の比較について断定的なことはいえないが，ここでは，とりあえず後者の（追加参加を認めない）方式を採用することにした。

（4）サーバーの守備範囲

サーバー（ゲームの運用機能担当）の負荷をできるだけ少なくしたい（これはゲーム運用のためのプログラムを最小に押さえることにもなる）という立場から考えると，相互干渉型ゲームでは，干渉の結果を明らかにする部分だけをサーバーが担当すればよい。資源配分型ゲームの場合，各プレイヤーに配分結果を知らせるだけの仕事をすればよい。その他の部分は各プレイヤーが自主的に計算と記録を行うことになる。

この方針を採った場合の短所は，配分計算に必要なデータを，その都度サーバーに送る必要があること，ゲームの成果計算はすべてプレイヤー側に委ねられているので，結果の

正確性，信頼性が保証されないこと，などである。

配分計算にどれだけのデータが必要かは，どのような配分規則を設定するかに依存するが，例えば過去の売上実績を加味するという可能性は大きい。そうすれば過去の売上データをその都度送信する必要が出てくる。

それらの要因を考慮すると，結局，すべての計算をサーバーで行い，それに必要なデータもすべてサーバーに記録しておく方が実際的ではないかという判断に傾く。ここでは，ほとんどすべてをサーバー側で処理するという方針を採ることにしたが，参加者の規模が大きくなると，方針を変更する必要が出るかもしれない。

（5）配分結果や成果を知る期間の分離

基本的なwebアプリケーションでは，サーバーから各プレイヤーに情報を伝達できるのは，プレイヤー側からのアクセスに応答する場合だけである。それゆえ，配分結果や成果はプレイヤー側から問い合わせるという方式をとることが必要になる。

そうすると，結果問い合わせの期間と，意思決定データ送信の期間を分離する方が，運用規則を作りやすいことになる。同じ時間帯に，結果の問い合わせと，意思決定データの入力の両方のアクセスを許容すると，処理論理の複雑化，保存データ量の増大を招く可能性があるからである。

ここでは，データ入力モードと，結果の参照モードを設定し，それらを交互に切り替えることによりゲームを進行させる方式を採った。

（6）参加人数の規模

１つのゲームに何人までのプレイヤーを受け入れるかという問題である。まず，ゲーム・コンテンツの特性から，論理的に何人まで許容できるかを考える必要がある。例えば寡占業界をモデルとしたゲームに，数十人もの参加を許すような運営はナンセンスであろう。

次に，ゲームの特性からは多人数の参加が可能であっても，サーバーの負荷が耐えられるかどうかということを検討しておくことが必要となる。ゲームの構成法に依存するが，ある程度のデータはメモリに記録しておくことが，応答速度の点から必要になる。参加人数が多くなると，所要メモリ・サイズも増大する。

資源配分型ゲームの場合，参加人数が多くなると，あるプレイヤーの振る舞いが，他のプレイヤーに与える影響は小さくなる傾向があるのではないかという一般的な予想が成立する場合が多いであろう。例えば，あるプレイヤーが倒産したとき，参加者が10人なら，プレイヤー当たりの売上増加が10%見込まれるかも知れない。しかし，参加者が100人なら，その影響はわずか１%となる。しかし現実社会では多くの同業者があっても，寡占状態への進行が進む場合もあるので，ゲーム・コンテンツの作り方により，常に一般的な予想が

正しいとは限らない。

参加規模に関連する項目として，複数のゲームを同時進行できないかという要請が出るかも知れない。例えば1つのゲームへの参加人数が数人に限られるという条件が必要な場合，20～30人の授業クラスを運営していれば，同時に複数ゲームを実行したいという場合が出てくる。webアプリケーションでは，1つの（仮想）ディレクトリに1つのアプリケーションを対応させる仕組みになっているので，複数のディレクトリに同じゲームを設定しておけば，アクセス先を変更するだけで，同じゲームの同時進行は簡単に実現できる。その場合の制約は，サーバーが負荷に対応できるかどうかという点だけである。

（7）応答遅れ，データ・エラーへの対処方針

意思決定データの入力や，結果の問い合わせが〆切時刻までになされなかった場合，どのように対処するかを決めておくことも，ゲームの運営システム設計にあたっての必要条件となる。ある程度まで遅れを許容するのか否かのどちらかになるが，前者の場合，運営アルゴリズムが複雑になる。

ここでは，入力モード，参照モードともに，遅れを認めずにゲームから脱落させるという方針を取った。しかしネットワーク障害が原因で遅延した場合の対策は必要になると考えている。

データ・エラーについて，例えば識別情報の送受信でエラーが発生すれば，そのプレイヤーは以降のゲームを継続できなくなる（テスト試行期間の間に，1回だけ誤通信が原因と見られるデータ・エラーが発生した)。そのような場合の回復方法を準備しておくことも必要となる。

（8）プレイヤーの識別

各プレイヤーから，意思決定データの送信や，結果問い合わせのアクセスがあったとき，それが，どのプレイヤーからのものか識別しなければならない。その都度プレイヤーにユーザーIDの類の入力を求めるのは余分な労力を強いることになるので，何らかの対応策が必要になる。

ここでは，参加番号というものを設定し，サーバーからプレイヤーに返すwebページにその番号をFORM内に埋め込むことにより，データ入力や問い合わせ時に，自動的にそれがサーバーに送信されるという方法を用いた。その番号を配列データの添字番号として使うことにより，当該プレイヤーのデータ抽出を容易に行うことができる。ユーザーの登録レコードの中に，現在の参加番号を書き込むことにより，ユーザーIDなどとの対応付けができるようにした。ただし，この方法をとる場合，何らかの原因でサーバーから送られてきたページをプレイヤーが消失した場合，それ以降のゲームを続けられなくなるので，

回復手段を準備しておくことが必要になる。

4　モデルの詳細

4.1　モード切替とアクセス内容受付のタイミング

このモデルではプレイヤーからのアクセスを，参加申込み，意思決定結果の入力，（1サイクルの）ゲーム実施結果の問い合わせ，の3種とした。それらを正しく受け入れるために，ゲーム・システム（サーバー）の状態を，参加受付モード，入力モード，参照モードの3種に区別し，アクセスの種類とシステムのモードが一致するときに，必要な処理が実行されるようにした。

上述のように，モード切替をプレイヤーからのアクセスに基づいて行う方式を取ったので，どの時点でモード変更を実施するかを正しく設定することが，本稿でのキーポイントとなる。システムのモードと，アクセスの種類の組み合わせ，および各組み合わせ時にどのような処理を行うべきかを整理したものを図2に示す。

図2において，例えばアクセスｅは，システムの状態がサイクル＃1，参加受付モード（モード0）にあるとき，意思決定結果を入力するためにアクセスしたことを示す。その時刻は，システムで保持しているモード切替の〆切時刻をすでに過ぎているので，システムではモードを参加受付モードから入力受付モード（モード1）に変更した後，送られてきたデータを受け入れ，必要な処理を行うべきことを表している。

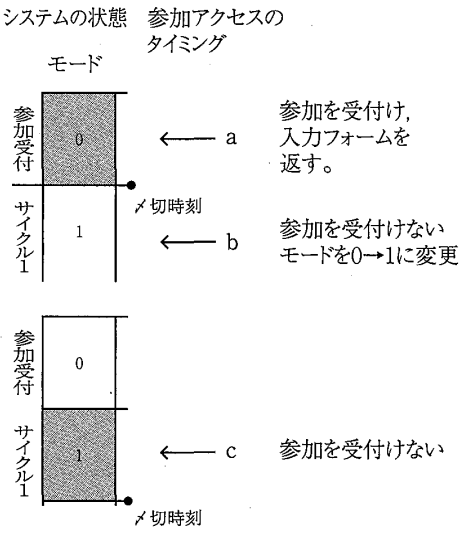
今一つの例として，アクセスｑは，システムの状態がサイクル＃ｉ，参照受付モード(モード2)にあるとき，サイクルｉの実行結果問い合わせのためにアクセスしたことを示す。その時刻は，システムで保持しているモード切替の〆切時刻を過ぎているので，本来はサイクル＃ｉ+1，入力モードに更新されているべきである。それゆえシステムの状態を本来の状態に更新するが，問い合わせは時刻遅れのため受け付けない（ゲームからの脱落となる)。

4.2　ゲーム進行のために必要となる保持データ

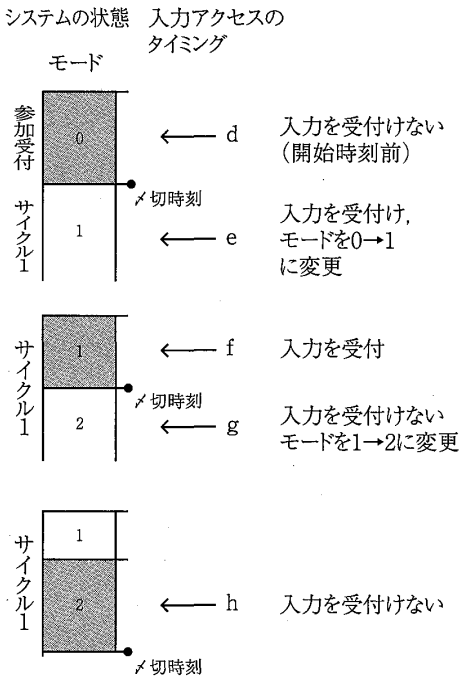
メモリ上で保持しているデータの種類を表1に示す。開発したプロトタイプはWindows2000のウェブ・サーバーIISを用いているので，具体的にはApplication変数に記録している。表1の中で「参加状況」は参加者数分の配列データとなっている（ASPのjoin，split 関数を用い，1つの文字列データとして記録している)。それらの初期値はゲーム開始時に設定する。

図2　システムの状態とアクセス種別・タイミングの組み合わせ

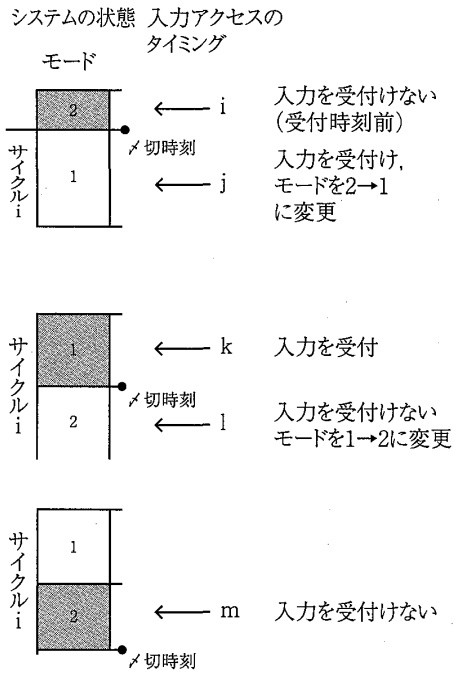
(1)参加受付



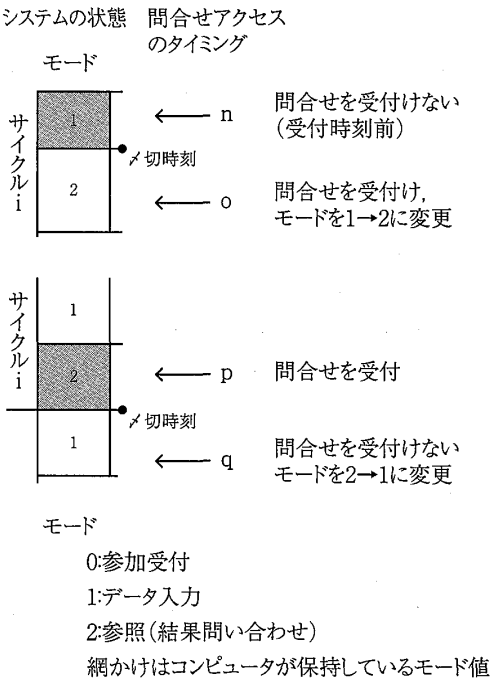
(2)入力1



(3)入力2



(4)問い合わせ(参照)



この外，ゲーム・コンテンツに関する多数のデータを保持する必要があるが，そのほとんどは，参加者数分の配列データとなる。

表1　システムで保持しているデータ

項目	変数名	説明
ゲームID	Application(“gameid”)	日時を基に作成。例：g2006/02/18_17
サイクル数	Application(“cyclelmt”)	1ゲームのサイクル数
入力期間幅	Application(“inspan”)	入力モードの長さ
参照期間幅	Application(“seespan”)	参照モードの長さ
サイクル番号	Application(“cycleno”)	現在のサイクル番号を記録
モード番号	Application(“pmode”)	現在のモード番号を記録
締切時刻	Application(“timelmt”)	現在のモードの終了時刻
次の締切時刻	Application(“nextlmt”)	次のモードの終了時刻
参加者数	Application(“joinsize”)	そのゲームの参加者数
継続者数	Application(“donesize”)	各モードでアクセスした人数
参加状況	Application(“s1”)	プレイヤ別現モードの参加状態（2以上なら脱落）
参加番号	Application(“joinno”)	ワーク・エリア
エラーマーク	Application(“errsw”)	ワーク・エリア

ゲーム・コンテンツ固有の部分は除く

4.3　参加申込み受付

ゲーム利用者は予め利用登録を行っておく必要がある。登録テーブル（データベース）にはユーザIDとパスワードが記録される。

プレイヤーが参加申込みを行うと，参加申込み様式（webページ）が返送される（リスト1：gjoin)。それにユーザIDとパスワードを入力して，参加受付モードの期間内に送信すると，ユーザー登録を確認し，ユーザー数を追加カウントした後（リスト2：gjoina)，ゲームの初回意思決定項目の入力様式が返送される。それには「ゲームID」「サイクル番号」「参加番号」が埋め込まれており，入力モードの〆切時刻も表示されている(リスト3：inform1a)。リストの途中に「ゲーム内容に依存した内容」という注釈行があるが，ゲーム・コンテンツとして処理すべき内容を，この部分に追加記入する必要があることを示す。以降のリストも同様。

4.4　参加受付モードから入力モードへの変更

プレイヤーが参加受付，または意思決定項目の入力のためにアクセスしたとき，入力モードへの変更時刻を過ぎているときは，まずモード変更の処理を行う。

モード番号を変更し，新しいモードの〆切時刻，その次のモードの〆切時刻を更新する。

サイクル番号を1に設定し，配列データの初期値を設定する。ゲーム進行のための骨格として必要な配列データは「参加状況」の1項目だけであるが，ゲーム・コンテンツを加えると，多項目の初期設定が必要になる。このタイミングで初期値設定するのは，このモード変更時点で「参加者数」が確定するからである。その人数分だけの配列データをjoin関数で1つの文字列として格納する（リスト4：mode0to1)。

4.5　意思決定データの受付

各プレイヤーが意思決定項目にデータを入力し，入力モードの期間内に送信すると，エラー・チェックを行った後，ゲーム・コンテンツに依存した内容の処理を行う（リスト5：inform1b)。エラー・チェックの種類としては，埋め込まれているデータが，その入力モードに対応した正しいものか，重複入力ではないか，などである。

処理の結果，入力受付したデータ値，およびゲーム・コンテンツで入力データの修正が許されている場合は，その修正値を返送する。そのとき，参加番号やサイクル番号などを埋め込んだ，実行結果の問い合わせ様式も含める（リスト6：query1a)。

4.6　入力モードから参照モードへの変更

データ入力または結果問い合わせのためのアクセスがあったとき，その時点が入力モードの終了時刻を過ぎていた場合に，参照モードへの変更を行う。モード番号を変更し，新しいモードの〆切時刻，その次のモードの〆切時刻を更新する（リスト7：mode1to2)。

ゲーム・コンテンツの内容によっては，この時点で制約資源の配分計算などを行い，そのサイクルの実行結果をメモリー，DBに記録する。

4.7　実行結果の返送

そのサイクルの実行結果問い合わせがあったとき，まずモード変更が必要なら，それを実施する。そして，問い合わせが正しい様式を用いたものか，参照期間内かなどをチェックしたのち（リスト8：query1b)，そのサイクルの実行結果（リスト9：gresulta）と，次のサイクルの意思決定値入力のための様式（リスト10：inform2a）を返送する。ゲームが最終サイクルであれば，終了処理を行ってゲームを終了する。

4.8　意思決定データの受付（第2サイクル以降)

第2サイクル以降のデータ受付時のチェック内容は，初回とは異なるので，異なるスクリプトを用いる（リスト11：inform2b)。すなわち，初回はモード0から1への変更要否

のチェックだけで良かったが，サイクル２以降は，モード１（入力）から２（参照）への変更が必要かと，モード２から１への変更が必要かの，２つのチェックを行わなければならない。

5　画面の例示

ゲーム・コンテンツを組み込んだ場合の，各段階の処理内容はかなり複雑になるので別稿にゆずるが，ここではサーバーにアクセスしたとき返送される２種類の様式を例示する。図３は実行結果，および同時に返送された入力様式，図４は入力結果の確認と問い合わせ様式の例である。

6　おわりに

インターネットを利用した教育用ゲームを手軽に作れないものかという願望がやっと実現できた。ゲーム・コンテンツのどの部分をどのステップで処理すれば良いかについて，かなり注意深い配慮が必要になるので，「手軽に」とまではいかないが，一応，ゲーム構築法の１つを透明化できたことで，マイルストーンの役割を果たし得たと思っている。

残されている最大の課題は，教育に本当に役立つゲーム・コンテンツをどのようにして作るのかという所にある。

この構築法の主要な部分は，二ヶ月あまりの作業により2004年３月に完成していたが，その後中断し，既開発内容の再学習を含めて一ヶ月あまりを費やして，細部の修正を終えたのが2006年２月となった。ここ数年の間に発表したwebアプリケーションはすべて，学生など他人の協力はなく，独力での学習に基づき開発したものである。プログラマー30才説というのがあるが，70才前後でも，この程度のことはできるという，１つの記録として記しておきたい。

参照文献

1．黒沢，村原，藤田，市川共著，「インターネットを利用したビジネスゲームの普及に関する調査研究とテキストの作成」（平成11年度科学技術融合振興財団調査研究助成報告書），2002
2．Nam Jaeook，葵七美訳，「NMORPG ゲーム・サーバー・プログラミング」，ソフトバンク，2005

3．拙稿，「web利用自主学習システムの一構成モデル」，大阪産業大学経営論集2巻2号，pp.15-49，2001
4．拙著，「経営システムと情報システム」，付章1，中央経済社，2000
5．拙稿，「Webページを用いた教育用ゲーム開発事例」，大阪産業大学経営論集5巻3号，pp.17～40，2004

図３　実行結果，および同時に返送された入力様式の例

query1b.asp
game-ID（ゲーム番号）：g2006/02/19_12　　サイクル番号：３　　参加番号：１
モード1から2へ変更しました。
gresulta.asp
　　このサイクルの実施結果は次の通りです。

基本需要枚数	632
実際販売枚数	568
売上高	1420
支払経費合計	590
人件費等	530
増設陳列台費用	0
広告費	0
支払利息	60
仕入枚数	650
仕入金額	975
在庫枚数	1805

inform2a.asp
ゲーム#：g2006/02/19_12　　サイクル#：４　　参加#：１

直前の値は次のようになっています。

現金	3412	買掛金	975
（入金予定）			
売掛金	0		
受取手形（今月）	0		
受取手形（来月）	0		
在庫数量	1805	借入金残高	6000

　上のデータを参考にして，ゲームをどう進めるか検討し，意思決定結果を次のフォームに入力し，送信してください。
　現在の時刻は　　2006/02/19 12：21：29
　入力受付期間は　2006/02/19 12：23：26　から
　　　　2006/02/19 12：25：26　の間です。

　決定結果を次に入力し，送信してください。
借入金増減額（千円）：[　　]（返済ならマイナスをつける）
販売方法：　　現金売　○　掛売　○　手形売　○
広告の有無：　　出す　○　出さない　○
陳列台台数：[　　]
仕入枚数：[　　]
[送信]

図4　入力結果の確認と問い合わせ様式の例

inform2b.asp
game-ID：g2006/02/19_12　　サイクル番号：4　　参加番号：1
query1a.asp
game-ID：g2006/02/19_12　サイクルNo.：4　参加No.：1

入力された意思決定データは次のように処理されました。
前サイクルの借入金増減額：　0

	入力値	修正値
販売方法	1	
広告	0	
陳列台数	4	4
仕入数量	550	550
在庫数量	2355	

現在の時刻は　　2006/02/19 12：22：57
ゲーム結果の問合せ受付期間は　2006/02/19 12：24：52　から
2006/02/19 12：26：52　の間です。
このページを誤って削除しないよう，保存しておくことをお勧めします。
受付時刻が来たら「参照」ボタンを押して，このサイクルの販売実績と，
次の入力フォームを受け取ってください。

[実施結果の参照]

リスト1　gjoin.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV= "Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>ゲーム参加申込</TITLE>
<STYLE TYPE="text/css">
</STYLE>
<SCRIPT LANGUAGE="javascript">
<!--
function fnSubmit() {
  if (fm.user1.value == "") {
    alert("ユーザーIDは必ず入力してください。");
```

```
    return false;
  }
  }
-->
</SCRIPT>
</HEAD>
<BODY><br>
ゲームへの参加申し込み<br><br>
<HR>
<FORM NAME="fm" METHOD="post" ACTION="gjoina.asp" onSubmit="return fnSubmit();">
<TABLE BORDER="0">
<TR>
<TD>ユーザーID：</TD>
<TD><INPUT NAME="user1" TYPE="text"></TD>
</TR>
<TR>
<TD>パスワード：</TD>
<TD><INPUT NAME="pass1" TYPE="password"></TD>
</TR>
</TABLE><br>
<INPUT TYPE="submit" VALUE="参加申込">
</FORM>
</BODY>
</HTML>
```

リスト2　gjoina.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>ゲーム参加申込</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
<% ' 参加申込時の処理
 If Request.Form("user1") <> "" Then
  ' 受付可能か否かのチェック
   Application("errsw") = 0
   Application("modechng") = 0
```

```
If Application("pmode") <> 0 Then
  Application("errsw") = 1
  Response. Write("すでにゲームが進行中か，ゲームが未実施状態です。<br>")
  Response. Write("次の開催時に参加ください。<br><br>")
End If
dif = DateDiff("n", Application("timelmt"), Now)
If dif > 0 Then
  Application("errsw") = 1
  Application("modechng") = 1
  Response. Write("すでにゲームの参加受付時刻を過ぎました。<br>")
  Response. Write("次の開催時に参加ください。<br><br>")
End If
'ユーザーID，パスワードの確認
  userid  = Request. Form("user1")
  passwrd = Request. Form("pass1")
  sql = "select * from user where id='" & userid & _
    "' and pwd='" & passwrd & "'"
  Set adoConn = Server. CreateObject("ADODB.Connection")
  Set adoRecs= Server. CreateObject("ADODB.Recordset")
  adoConn. Open  "netgame"
  adoRecs. Open sql, adoConn, adOpenStatic, adLockOptimistic, adCmdText
  If adoRecs. EOF Then
    Application("errsw") = 1
    Response. Write("ユーザーIDまたはパスワードが間違っています。<br><br>")
  Else
     If adoRecs("joinno").Value <> 0 Then
        Application("errsw") = 1
        Response. Write("すでにゲームに参加しています。<br><br>")
     End If
  End If
  adoRecs. Close
  adoConn. Close
'モード変更が必要な場合
If Application("modechng") = 1 Then
  Server. Execute("mode0to1.asp")
End If

'正常な場合，参加者数を1増加し，入力フォームを送り返す。
If Application("errsw") = 0 Then
  jno = Application("joinsize") + 1
  Application("joinsize") = jno
```

```
    sql = "update user set joinno = '" & jno & "'  where id = '" & userid & "' and pwd = '" &
passwrd & "' "
    Set adoConn = Server. Create Object("ADODB.Connection")
    adoConn. Open "netgame"
    adoConn. Execute sql
    adoConn. Close
    Server. Transfer("inform1a.asp")
  End If
 End If
 Session.Abandon
 %>
</BODY>
</HTML>
```

リスト3　inform1a.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>最初の入力フォーム</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY>
inform1a.asp<br><br>
<%= "game-ID：" & Application("gameid") %>
    <%= "サイクル番号：01" %>
    <%= "参加番号：" & Application("joinsize") & "<BR><BR><BR>" %>
<%
‘ゲーム内容に依存した内容（#1）
%>
<HR>
ゲームの初期値は次のようになっています。<br><br>
ゲーム内容に依存<br>
  上のデータを参考にして，ゲームをどう進めるか検討し，意思決定結果を<br>
次のフォームに入力し，送信してください。<br><br>
  現在の時刻は      <%= Now %><br><br>
  入力受付期間は  <%= Application("timelmt") %>  から<br><br>
                  <%= Application("nextlmt") %>  の間です。<br><br><br>
このページを誤って削除しないよう，保存しておくことをお勧めします。<br><br>
```

```
<HR>
  決定結果を次に入力し，送信してください。<br><br>
<FORM NAME="fm" METHOD="post" ACTION="inform1b.asp">
<INPUT TYPE="hidden" NAME="gid" VALUE="<%= Application("gameid") %>">
<INPUT TYPE="hidden" NAME="cno" VALUE="1">
<INPUT TYPE="hidden" NAME="jno" VALUE="<%= Application("joinsize") %>">
<%
‘ゲーム内容に依存した内容（#2）
%>
<INPUT TYPE="submit" VALUE="送信">
</FORM>
</body>
</html>
```

リスト4　mode0to1.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>モード変更 0 to 1</TITLE>
<STYLE TYPE="text/css">
</STYLE>
<%
Application Lock
Application("pmode") = 1
Application("cycleno") = 1
Application("timelmt") = Application("nextlmt")
Do until Application("timelmt") > Now
Application("timelmt") = DateAdd("n", Application("seespan"), Application("timelmt"))
Loop
Application("nextlmt") = DateAdd("n", Application("seespan"), Application("timelmt"))

‘ゲーム内容に依存した内容（#3）

’初期値設定
st = ""
zr = ""
n = Application("joinsize")
for i = 1 to n
  st = st & ","
```

```
  zr = zr & "," & "0"
next
Application("null") = st
Application("zero") = zr
Application("s1") = zr

‘ゲーム内容に依存した内容（#3）
%>
<%’ゲーム条件をDBに記録
  Set adoConn = Server. CreateObject("ADODB.Connection")
  Set adoRecs= Server. CreateObject("ADODB.Recordset")
    adoConn. Open  "netgame"
    adoRecs. Open "gamelog", adoConn, 3, 3, 2
    adoRecs. AddNew
    adoRecs("gameid"). Value = Application("gameid")
    adoRecs("cyclelmt"). Value = Application("cyclelmt")
    adoRecs("joinsize"). Value = Application("joinsize")

‘ゲーム内容に依存した内容（#4）

    adoRecs. Update
    adoRecs. Close
    adoConn. Close
%>
</HEAD>
<BODY><br>
<%= "モード0から1へ変更しました。<br><br>" %>
</BODY>
</HTML>
```

リスト5　inform1b.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>最初の入力フォーム</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
```

```
inform1b.asp<br><br>
<%
  gid1 = Request. Form("gid")
  cno1 = Request. Form("cno")
  jno1 = Request. Form("jno")

‘ゲーム内容に依存した内容（#5）

  jno1 = jno1 + 0
  Application("errsw") = 0
%>
<%= "game-ID：" & gid1 %>
<%= "サイクル番号：" & cno1 %>
<%= "参加番号：" & jno1 & "<BR><BR><BR>" %>
<%
 ’送信ボタンでデータが送られて来た場合，受付可能か否かのチェック
 ’すべての項目にデータが入力されているか？

‘ゲーム内容に依存した内容（#6）

 ’ゲームID が一致しているか？サイクル番号が正しいか？
  If Application("gameid") <> gid1 or cno1 <> "1" Then
    Application("errsw") = 1
    Response. Write("ゲームID またはサイクル番号が一致していません。<br>")
    Response. Write("間違った入力フォームを使っています。<br><br>")
    Response. Write("このゲームを続行できません。<br><br>")
  Else
    ’モード2（参照）になっているか？
    If Application("pmode") = 2 Then
      Application("errsw") = 1
        Response.Write("データ入力の受付期限を過ぎました。参照モードになっています。
<br><br>")
      Response. Write("このゲームを続行できません。<br><br>")
    End If
   ’〆切時刻を過ぎたか（dif>0）？ & モードは？
    dif = Date Diff("n", Application("timelmt"), Now)
    ’モード0（初期）の場合
    If Application("pmode") = 0 Then
      If dif >= 0 Then
      ’モード変更が必要，実行
        Server. Execute("mode0to1.asp")
```

```
    Else
      Application("errsw") = 1
      Response.Write("まだ入力受付時刻前です。<br><br>")
      Response.Write("ページを戻して，後刻入力してください。<br><br>")
    End If
  End If
  'モード1（入力）の場合
  If Application("pmode") = 1 Then
    If dif > 0 Then
      Application("errsw") = 1
      Response.Write("データ入力の受付期限を過ぎました。<br><br>")
      Response.Write("このゲームを続行できません。<br><br>")
      Server.Execute("mode1to2.asp")
    End If
  End If
  '脱落マークが付いているか？重複入力にならないか？
  If Application("errsw") = 0 Then
    wtr1 = split(Application("s1"),",")
    If wtr1(jno1) <> "0" Then
      Application("errsw") = 1
      If wtr1(jno1) = "1" Then
        Response.Write("すでに入力済みです。重複入力はできません。<br><br>")
      Else
        Response.Write("このゲームから脱落しています。続行できません。<br><br>")
      End If
    End If
  End If

'正常な場合，その参加者のゲーム進行処理をし，結果問い合わせフォームを送り返す。
If Application("errsw") = 0 Then
  '入力者数をカウント
  application("donesize") = application("donesize") + 1
  'その参加者に入力済みマークをつける。
  wtr1(jno1) = "1"
  Application("joinno") = jno1
  Application("s1") = join(wtr1,",")

'ゲーム内容に依存した内容（#7）

  Server.Transfer("query1a.asp")
End If
```

```
 End If
%>
</body>
</html>
```

リスト6　query1a.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>結果の問合せフォーム1</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
query1a.asp<br><br>
<%= "game-ID：" & Application("gameid") %>
   <%= "サイクルNo.：" & Application("cycleno") %>
   <%= "参加No.：" & Application("joinno") & "<BR><BR><BR>" %>

<%
j = Application("joinno")

‘ゲーム内容に依存した内容（#8）

%>
<HR>
入力された意思決定データは次のように処理されました。<br><br>
<%
‘ゲーム内容に依存した内容（#8）
%>
  現在の時刻は　　<%= Now %><br><br>
  ゲーム結果の問合せ受付期間は <%= Application("timelmt") %>　から<br><br>
                              <%= Application("nextlmt") %>　の間です。<br><br><br>
このページを誤って削除しないよう，保存しておくことをお勧めします。<br><br>

  受付時刻が来たら「参照」ボタンを押して，このサイクルの販売実績と，<br>
次の入力フォームを受け取ってください。<br><br>
<HR>
<FORM NAME="fm" METHOD="post" ACTION="query1b.asp">
```

```
<INPUT TYPE="hidden" NAME="gid" VALUE="<%= Application("gameid") %>">
<INPUT TYPE="hidden" NAME="cno" VALUE="<%= Application("cycleno") %>">
<INPUT TYPE="hidden" NAME="jno" VALUE="<%= Application("joinno") %>">
    <INPUT TYPE="submit" VALUE="実施結果の参照"><br><br>
</FORM>
</body>
</html>
```

リスト7　mode1to2.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>モード変更 1 to 2</TITLE>
<STYLE TYPE="text/css">
</STYLE>
<%
Application Lock
Application("pmode") = 2
Application("timelmt") = Application("nextlmt")
Do until Application("timelmt") > Now
Application("timelmt") = DateAdd("n", Application("inspan"), Application("timelmt"))
Loop
Application("nextlmt") = DateAdd("n", Application("inspan"), Application("timelmt"))

‘ゲーム内容に依存した内容（#9）
’資源配分計算
’各プレイヤーの実績計算

’アクセス済マークをクリア，未入力には6を記録
For i = 1 to n
  If wtr1(i) = 1 Then
    wtr1(i) = 0
  Elseif wtr1(i) = 0 Then
    wtr1(i) = 6
  Else
  End If
Next
Application("s1") = join(wtr1,",")
Application("donesize") = 0
```

```
Application UnLock
%>
</HEAD>
<BODY><br>
<%= "モード1から2へ変更しました。<br><br>" %>
</BODY>
</HTML>
```

リスト8　query1b.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>結果の問合せフォーム2</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
query1b.asp<br><br>
<%
  gid1 = Request. Form ("gid")
  cno1 = Request. Form ("cno")
  jno1 = Request. Form ("jno")
 %>
 <%= "game-ID（ゲーム番号）：" & gid1 %>
<%= "サイクル番号：" & cno1  %>
<%= "参加番号　：" & jno1 & "<BR><BR><BR>" %>
 <%
   Application ("errsw") = 0
 ' 送信ボタンでデータが送られて来た場合，受付可能か否かのチェック
  ' ゲームID が一致しているか？サイクル番号が正しいか？
  If (Application ("gameid") <> gid1) or (Application ("cycleno") <> cno1 + 0) Then
     Application ("errsw") = 1
     Response. Write ("ゲームID またはサイクル番号が一致していません。<br>")
       Response. Write ("間違った問合せフォームを使っているか，問い合わせの遅すぎです。
<br><br>")
  Else
     jno1 = jno1 + 0
     Application ("joinno") = jno1
   ' メ切時刻を過ぎたか（dif>0）？ & モードは？
```

```
    dif = DateDiff("n", Application("timelmt"), Now)
    ' モード1（入力）の場合
    If Application("pmode") = 1 Then
      If dif >= 0 Then
      ' モード変更が必要，実行
        Server. Execute("mode1to2.asp")
      Else
        Application("errsw") = 1
        Response. Write("まだ問合せ受付時刻前です。<br><br>")
      End If
  Else
   ' モード2（参照）の場合
    If dif >= 0 Then
        Response. Write("問合せの受付期限を過ぎました。<br><br>")
        Application("errsw") = 1
      ' モード変更が必要，実行。
        Server. Execute("mode2to1.asp")
    End If
  End If

   ' 脱落マークが付いているか？重複問合せにならないか？
   If Application("errsw") = 0 Then
     wtr1 = split(Application("s1"),",")
     If wtr1(jno1) <> "0" Then
       Application("errsw") = 1
       If wtr1(jno1) = "1" Then
         Response. Write("すでに問合せ済みです。再問合せはできません。<br><br>")
       Else
         Response. Write("このゲームから脱落しています。続行できません。<br><br>")
       End If
     End If
   End If
 End If

' 正常な場合，そのサイクルのゲーム結果を通知し，次の入力フォームを送り返す。
If Application("errsw") = 0 Then
 ' 問合せ者数をカウント
 application("donesize") = application("donesize") + 1
 ' その参加者に問合せ済みマークをつける。
 wtr1(jno1) = "1"
 Application("s1") = join(wtr1,",")
```

```
 Server.Execute("gresulta.asp")

'ゲーム内容に依存した内容（#10）

 If application("cycleno") >= application("cyclelmt") Then
    Server. Transfer("final.asp")
   Else
    Server.Transfer("inform2a.asp")
 End If
End If
%>
</body>
</html>
```

リスト9　gresulta.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>そのサイクルの実施結果を通知</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
gresulta.asp<br><br>
<% '= "game-ID：" & Application("gameid") %>
<% '= "サイクル番号：" & Application("cycleno") %>
<% '= "参加番号：" & Application("joinno") & "<BR><BR><BR>" %>
  このサイクルの実施結果は次の通りです。<br><br>
<%
'ゲーム内容に依存した内容（#11）
%>
</html>
```

リスト10　inform2a.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
```

```
<TITLE>入力フォーム2</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
inform2a.asp<br><br>
<%
  Application("errsw") = 0
  cno2 = 1 + Application("cycleno")
%>
<%= "ゲーム#：" & Application("gameid") %>
    <%= "サイクル#：" & cno2 %>
    <%= "参加#：" & Application("joinno") & "<BR><BR><BR>" %>
<%
j = Application("joinno")

‘ゲーム内容に依存した内容（#12）

%>
  上のデータを参考にして，ゲームをどう進めるか検討し，意思決定結果を<br>
次のフォームに入力し，送信してください。<br><br>
  現在の時刻は　　<%= Now %><br><br>
  入力受付期間は　<%= Application("timelmt") %>　から<br><br>
                  <%= Application("nextlmt") %>　の間です。<br><br><br>
<HR>
  決定結果を次に入力し，送信してください。<br><br>
<FORM NAME="fm" METHOD="post" ACTION="inform2b.asp">
<INPUT TYPE="hidden" NAME="gid" VALUE="<%= Application("gameid") %>">
<INPUT TYPE="hidden" NAME="cno" VALUE="<%= cno2 %>">
<INPUT TYPE="hidden" NAME="jno" VALUE="<%= Application("joinno") %>">
<br>
<%
‘ゲーム内容に依存した内容（#13）
%>
<INPUT TYPE="submit" VALUE="送信">
</FORM>
</body>
</html>
```

リスト11　inform2b.asp

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=shift_jis">
<TITLE>入力フォーム2</TITLE>
<STYLE TYPE="text/css">
</STYLE>
</HEAD>
<BODY><br>
inform2b.asp<br><br>
<%
  gid1 = Request. Form("gid")
  cno1 = Request. Form("cno")
  jno1 = Request. Form("jno")
  cno1 = cno1 + 0
  jno1 = jno1 + 0
  sale1 = Request. Form("sale")
  ad1 = Request. Form("ad")
  dai1 = Request. Form("dai")
  invol1 = Request. Form("invol")
  debt1 = Request. Form("debt")
%>
<%= "game-ID：" & gid1 %>
    <%= "サイクル番号：" & cno1 %>
    <%= "参加番号：" & jno1 & "<BR><BR><BR>" %>
<%
  Application("errsw") = 0

 ' 送信ボタンでデータが送られて来た場合，受付可能か否かのチェック

' ゲーム内容に依存した内容（#14）

  ' ゲームID が一致しているか？
  If Application("gameid") <> gid1 Then
    Application("errsw") = 1
    Response. Write("ゲームID が一致していません。<br>")
    Response. Write("間違った入力フォームを使っています。<br><br>")
  End If
  ' 現時刻は〆切時刻を過ぎているか（dif>0）の計算
  dif = DateDiff("n", Application("timelmt"), Now)
```

```
  If Application("cycleno") = cno1 Then
  'システムのサイクル番号はフォームと同じか？
    If Application("pmode") = 1 Then
    'モード1（入力）になっているか？
      If dif >= 0 Then
      '〆切時刻を過ぎているか
        'モード変更が必要，実行
         Server. Execute("mode1to2.asp")
         Application("errsw") = 1
         Response. Write("データ入力の受付期限を過ぎています。
モードを変更しました。<br><br>")
      End If
    Else
      Application("errsw") = 1
        Response. Write("データ入力の受付期限を過ぎました。参照モードになっています。
<br><br>")
    End If
  Else
    If Application("cycleno") = cno1 - 1 and Application("pmode") = 2 Then
    'サイクル番号がフォームより１つ小さく，モード２か？
      If dif >= 0 Then
      '〆切時刻を過ぎたか
        'モード変更が必要，実行
         Server. Execute("mode2to1.asp")
      Else
         Application("errsw") = 1
         Response. Write("まだ入力の受付時刻前です。<br><br>")
      End If
    Else
       Application("errsw") = 1
       Response. Write("サイクル番号が間違っています。<br><br>")
    End If
  End If
  '脱落マークが付いているか？重複入力にならないか？
  If Application("errsw") = 0 Then
     wtr1 = split(Application("s1"),",")
     If wtr1(jno1) <> "0" Then
       Application("errsw") = 1
       If wtr1(jno1) = "1" Then
         Response. Write("すでに入力済みです。重複入力はできません。<br><br>")
       Else
```

```
            Response. Write("このゲームから脱落しています。続行できません。<br><br>")
          End If
        End If
      End If

    ' 正常な場合，その参加者のゲーム進行処理をし，結果問い合わせフォームを送り返す。
    If Application("errsw") = 0 Then
      ' 入力者数をカウント
      application("donesize") = application("donesize") + 1
      ' その参加者に入力済みマークをつける。
      wtr1(jno1) = "1"
      Application("joinno") = jno1
      Application("s1") = join(wtr1,",")

  ' ゲーム内容に依存した内容（#15）

      Server. Transfer("query1a.asp")
    End If
%>
</body>
</html>
```