TicTacToe SDK

Учебный комплекс программ для разработки интеллекта, играющего в крестики-нолики

Краткое руководство пользователя

Автор: Бреслав А. А.

2007 год

1 Назначение и возможности библиотеки

Язык программирования: Delphi 5

Операционная система: Windows XP

Библиотека предназначена для обучения основам искусственного интеллекта для настольных игр на примере игры в крестики-нолики. Поле имеет ограниченный, но большой размер, игра ведется до пяти знаков в ряд.

Библиотека дает обучающемуся возможность создавать реализации интеллекта любой сложности, причем не требует заниматься ни графикой, ни другими посторонними вопросами.

2 Как это работает

Комплект поставки состоит из

1. Программы-сервера TicTacToe.exe
2. Модуля TicTacToe.pas
3. Программы-клиента, управляемой вручную Client.exe
4. Исходного кода программы-клиента, управляемой вручную Client.dpr
5. Программы-клиента с искусственным интеллектом Stupid.exe

2.1 Сервер и клиенты

Крестики-нолики – игра для двоих. В нашем случае оба игрока – это программы, которые могут «думать» сами или управляться человеком. Чтобы две программы могли общаться между собой и никто не мог сжульничать, используется третья программа – *сервер*[1].

Каждая программа-игрок посылает свои ходы на сервер и получает от него информацию о ходах противника. Программа, подключившаяся первой, играет крестиками. Сервер следит за тем, чтобы ходы делались по очереди и правильно (нолик или крестик ставился только в пустую клетку в пределах поля) и определяет победителя. Кроме того, в окне сервера отображается игровое поле, имена игроков и сообщения о состоянии игры (рис. 1).

В окне сервера можно начать новую игру (при этом текущая игра прервется и оба игрока проиграют).

1 Программы, работающие с сервером (не только с нашим, но и вообще с любым сервером), часто называют *клиентами* этого сервера.

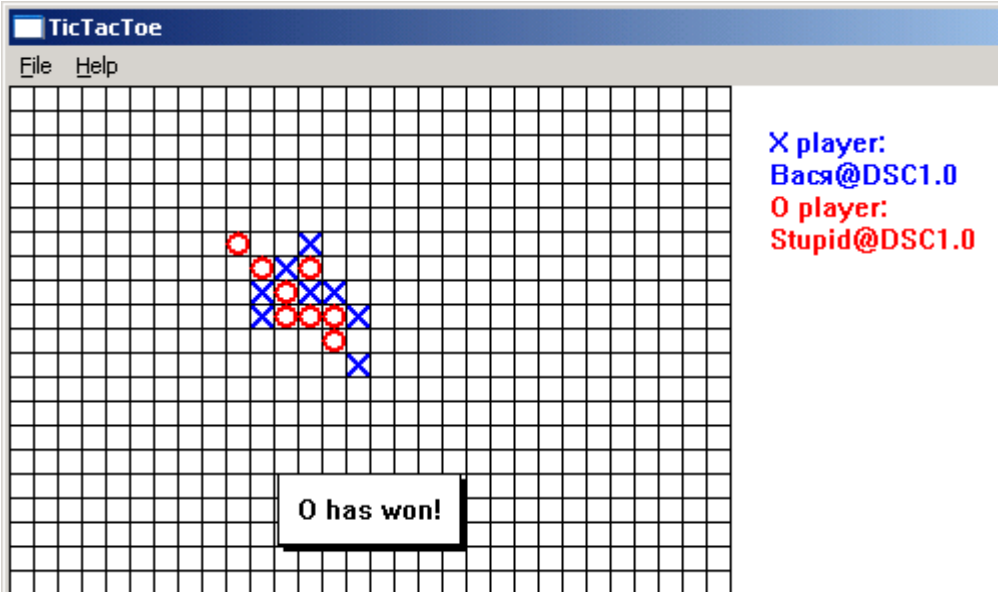

Рис. 1: Окно сервера

2.2 Как запустить игру

Для того, чтобы запустить игру, нужно скачала запустить сервер (TicTacToe.exe), на экране появится пустое поле и сообщение о том, что сервер ждет подключения первого игрока.

Затем нужно последовательно запустить двух клиентов (игроков) – первый из них будет играть крестиками, второй – ноликами. Клиенты не обязательно должны быть одинаковыми.

Два человека

Для того, чтобы поиграть самому (а не учить компьютер) используется стандартный клиент (Client.exe) окно этой программы представляет собой игровое поле, над которым отображается сообщение о состоянии игры. Щелкая мышкой в клетку поля, пользователь может поставить свой знак (крестик или нолик) в эту клетку. Поставить знак в занятую клетку или за пределы поля невозможно. Чтобы прервать игру, достаточно просто закрыть программу.

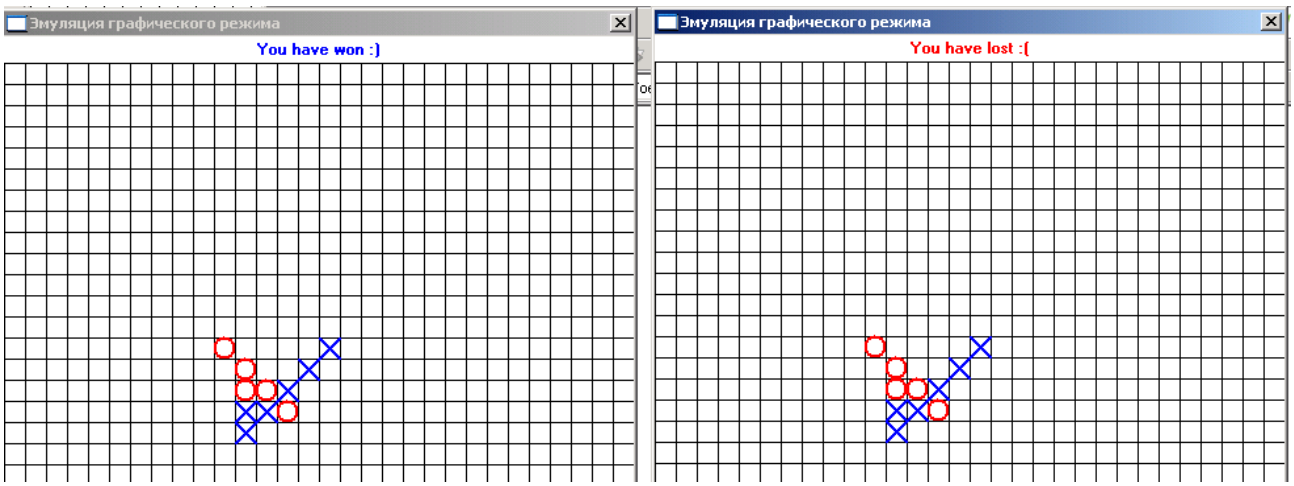

Рис. 2: Игра вдвоем

Итак, чтобы поиграть вдвоем (на одном компьютере), достаточно, запустив сервер, запустить два раза Client.exe (появится два окна с одинаковыми игровыми полями), после чего каждый

играет в своем окне (рис. 2).

Отметим также, что, несмотря на то, что каждая программа-клиент показывает игроку игровое поле, сервер тоже его показывает. Работа сервера не зависит от того, какие клиенты к нему подключены.

Человек и программа

Если для того, чтобы один человек мог поиграть с другим (или сам с собой), нужно было запускать два одинаковых клиента, то для того, чтобы человек мог поиграть с программой, клиенты должны быть разными. В комплект поставки входит очень простая «думающая» программа Stupid.exe. Несмотря на свою простоту, эта программа все же играет по правилам, не делает ошибочных ходов и стремится выиграть. Кроме того, она отображает в окне игровое поле и некоторую информацию, которую использует алгоритм принятия решений (рис. 3).

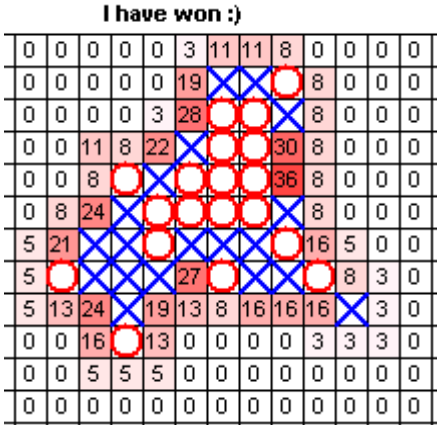

Рис. 3: Окно программы Stupid

Также в алгоритме, используемом этой программой используется генератор случайных чисел, поэтому с ней трудно сыграть две одинаковые партии.

Итак, чтобы сразиться с программой Stupid, нужно (после запуска сервера) запустить ее, а затем программу Client.exe, тогда Stupid будет играть крестиками, а человек – ноликами (если запустить программы в другом порядке, будет по-другому).

Две программы

Как легко догадаться, чтобы две программы играли между собой, достаточно запустить сервер, а потом – две эти программы. Например, если запустить Stupid.exe два раза, то алгоритм Stupid будет играть сам с собой (и сам у себя выиграет). Программы обычно думают довольно быстро, поэтому наблюдать за из игрой трудно. Чтобы замедлить ход игры, программу Stupid можно попросить делать задержку после каждого хода. Для этого нужно, запуская Stupid, в командной строке указать время задержки в миллисекундах. Например,

```
...> Stupid.exe 1000
```

Если запустить Stupid таким образом, после каждого хода она будет ждать одну секунду (1000 миллисекунд). ВНИМАНИЕ: это касается только программы Stupid, а не любой программы, написанной с помощью нашей библиотеки.

3 Как написать свою реализацию клиента

Основное назначение этого проекта – возможность легко создавать свои «думающие» программы. Такая программа не должна делать ничего, кроме принятия решений о ходах и передачи этих решений на сервер. Обычно же, если нужно написать игру, приходится писать графический интерфейс, правила и т. д.

На самом деле все, что делает какую бы то ни было программу клиентом для игры в крестики-нолики – это возможно общаться с сервером. Если есть способ подключиться и передавать ходы, можно написать и аналог программы Client.exe, и (при определенном старании) «думающую» программу.

3.1 Модуль TicTacToe.pas

Чтобы писать клиентов на Delphi, нужно подключить к программе модуль TicTacToe.pas – при запуске программы он ищет сервер и подключается к нему, кроме того в этом модуле есть процедуры и функции, позволяющие обмениваться данными с сервером.

Итак, самый простой клиент, написанный на Delphi, выглядит так:

```
uses TicTacToe;
begin
end.
```

Кажется, что эта программа ничего не делает, однако это неправда: при запуске она подключается к серверу и получает от него размеры поля и информацию о том, какими знаками играет – крестиками или ноликами. Эта информация необходима любому клиенту, получить ее можно так:

```
WriteLn('Ширина поля: ', FieldWidth);
WriteLn('Высота поля: ', FieldHeight);
if Me = csCross then
  WriteLn('Я играю крестиками')
else WriteLn('Я играю ноликами');
if He = csCross then
  WriteLn('Мой противник играет крестиками')
else WriteLn(' Мой противник играет ноликами');
```

Функции FieldWidth и FieldHeight возвращают ширину и высоту поля. Функция Me возвращает константу csCross, если вызывающая ее программа играет крестиками и csCirlce, если ноликами. Функция He делает наоборот – то есть сообщает, какими знаками играет противник (эта функция нужна только для удобства, без нее легко можно было бы обойтись).

После получения основной информации и поле нужно дождаться начала игры. Дело в том, что игра не может начаться пока оба противника не подключатся к серверу, а для ноликов – пока крестики не сделают первый ход. Поэтому необходимо подождать пока все будет готово перед тем как начать играть.

Для этой цели используется процедура WaitForgameStart, которой в качестве параметра можно передать имя игрока. Например,

```
WaitForGameStart('Вася');
```

Когда эта процедура заканчивает свою работу, игра началась и ход принадлежит вызвавшей ее программе, а имя игрока отображается в окне на сервере.

Когда игра началась, крестики могут сразу сделать ход, а вот ноликам, наверняка нужно узнать, куда поставлен первый крестик. Это можно сделать, вызвав функцию CrossFirstTurn. Эта функция возвращает запись типа TTurn, который определяется в модуле TicTacToe следующим образом (определять этот тип самостоятельно не нужно):

```
type
  TTurn = packed record
    x, y : Word;
    status : TPlayerStatus;
  end;
```

Поля x и y – это координаты ячейки, в которую поставлен знак (в случае CrossFirstTurn – это крестик), ячейки нумеруются с нуля, так что левая верхняя имеет координаты (0,0), а правая нижняя – (FieldWidth – 1, FieldHeight – 1).

В поле status передается состояние игры после данного хода, это поле может принимать следующие значения:

- YourTurn – Ваш ход, игра продолжается;
- YouHaveWon – Вы выиграли (либо Ваш последний ход был выигрышным, либо противник сдался);
- YouHaveLost – Вы проиграли (либо противник сделал выигрышный ход, либо сервер прекратил игру);
- YourMistake – Вы сделал ошибочный ход (поставили знак за пределы поля и в занятую клетку, Вам засчитывается техническое поражение).

Итак, чтобы узнать, каков был первый ход крестиков, можно сделать так:

```
WriteLn(CrossFirstTurn.x, ' ', CrossFirstTurn.y);
case  CrossFirstTurn.status of
  YourTurn: WriteLn('Игра продолжается');
  YouHaveWon: WriteLn('Я выиграл');
  YouHaveLost: WriteLn('Я проиграл');
  YourMistake: WriteLn('Этого не может быть!');
end;
```

Если функцию CrossFirstTurn вызывают крестики, она возвращает бессмысленный результат.

Далее остается только делать ходы. Для этого существует функция MakeTurn, ей передаются координаты клетки, в которую Вы хотите поставить свой знак, она передает координаты на сервер, ждет пока противник сделает ход и возвращает результат этого хода в виде той же записи TTurn.

```
hisTurn := MakeTurn(x, y);
```

В записи hisTurn поле status означает состояние игры после вашего хода, если Вы выиграли или сделали ошибку, и после хода противника в остальных случаях. Если status говорит о Вашем ходе, то поля x и y ничего не значат (противник не делал хода), иначе в них содержатся координаты ячейки, в которую противник поставил свой знак.

ВНИМАНИЕ! Каждый вызов MakeTurn делает ход, поэтому следующий пример приводит к ошибочному ходу (техническое поражение):

```
WriteLn(MakeTurn(x, y).x, ' ', MakeTurn(x, y).y);
```

ЗАМЕЧАНИЕ. Поскольку сервер может прервать игру в любой момент, да и противник может сдаться или сделать ошибку когда угодно, стоит предусмотреть эти возможности. К счастью, для этого достаточно всего лишь **проверять поле status у каждого хода (даже у первого хода крестиков)**.

3.2 Пример

Здесь мы рассмотрим пример простого клиента, делающего случайные ходы.

```
program Simple;

uses TicTacToe;

var
  f : array[0..100, 0..100] of TCellState; // Игровое поле
  i, j : Integer;
  hisTurn : TTurn;
begin

  for i := 0 to FieldWidth - 1 do
    for j := 0 to FieldHeight - 1 do
      f[i, j] := csEmpty; // Пустая клетка

  WaitForGameStart('Simple');

  if Me = csCircle then begin
    // На случай ошибки противника или прерывания игры
    if CrossFirstTurn.status <> YourTurn then
      Exit;
    f[CrossFirstTurn.x, CrossFirstTurn.y] := csCross;
  end;

  Randomize;
  while true do begin
    // Поиск свободной клетки
    repeat
      i := Random(FieldWidth);
      j := Random(FieldHeight);
    until f[i, j] = csEmpty;

    f[i, j] := Me;
    hisTurn := MakeTurn(i, j);

    if hisTurn.status <> YourTurn then
      Exit;
    f[hisTurn.x, hisTurn.y] := He;
  end;
end.
```