

---

# CSC200

Fall 2005

## Project 03

### Tic-Tac-Toe

Due: November 21, 2005

---

## 1 Purpose

We have just completed a project involving a text based BlackJack game. This project is yet another game, the classic Tic-Tac-Toe. This game has historical implications. By making the supercomputer play tic-tac-toe, Matthew Broderick saved the world from imminent Thermonuclear destruction (Wait, was that a movie? It seemed real to me.) Well anyway, the purpose of this project is to give further practice in the ideas behind programming computer games, especially representation of game objects in a computer environment.

## 2 Background

Tic-Tac-Toe is a game involving a 3 by 3 grid. The game is played between two players. One is 'X' and the other is 'O'. The 'X' player starts by placing an 'X' in one of the nine spots on the grid. The 'O' player then gets to place an 'O' in one of the remaining 8 blank spots on the grid. The game continues until either the whole grid is filled up or one player has 3 of their symbols in a row: diagonally, vertically, or horizontally. I assume everyone has played tic-tac-toe.

So how can we implement Tic-Tac-Toe in C++? For blackjack we represented the deck with an array and the player hands with arrays. What kind of structures could we use to represent things for Tic-Tac-Toe? The obvious answer is to use a 2D array to represent the game board. What should the base type of this 2D array be? With a 2D array representing our board, we can write functions for each piece of the game.

## 3 Requirements

For this project you will need to write a program in C++ which implements a simple tic-tac-toe game as described above. The following must be present in your project:

- One source code file named `tictactoe.cpp`
- Use `cin` (or `getline`) and `cout` to create a simple user interface to show the tic-tac-toe board after each turn. Allow the user to enter next move by picking a number between 0-8 as shown in the sample output below.
- Game is played by the rules stated in the background section above. (ie. Player 'X' starts first, etc.)
- Program uses a 2D array to represent the board.

- Program uses a while loop to allow multiple games to be played (clearing the board each time).
- At least three function (you must determine the correct return type and parameters) :
  - print\_board()
  - winning\_board()
  - clear\_board()

The program output should look something like:

Welcome to Tic-Tac-Toe! Enter moves as follows:

```
0|1|2
```

```
-----
```

```
3|4|5
```

```
-----
```

```
6|7|8
```

```
  | |
```

```
-----
```

```
  | |
```

```
-----
```

```
  | |
```

Player X, enter move: 0

```
X| |
```

```
-----
```

```
  | |
```

```
-----
```

```
  | |
```

Player O, enter move: 1

```
X|O|
```

```
-----
```

```
  | |
```

```
-----
```

```
  | |
```

Player X, enter move: 8

```
X|O|
```

```
-----
```

```
  | |
```

```
-----
```

```
  | |X
```

Player O, enter move: 4

```
X|O|
```

```
-----
```

```
 |O|
```

```
-----
```

```
 | |X
```

Player X, enter move: 7

```
X|O|
```

```
-----
```

```

|0|
-----
|X|X

Player 0, enter move: 6
X|0|
-----
|0|
-----
0|X|X

Player X, enter move: 2
X|0|X
-----
|0|
-----
0|X|X

Player 0, enter move: 5
X|0|X
-----
|0|0
-----
0|X|X

Player X, enter move: 3

The game resulted in a tie. Here is the final board:
X|0|X
-----
X|0|0
-----
0|X|X

Would you like to play again? (Y/N)

```

## 4 Grading

The following table shows the scoring that will be used for this project:

Area	Max Points
Compiles in Dev-C++	50
Report and Source Code Documentation	25
Correctness	25

There are a couple chances for extra credit on this project:

- A function called `place_move()` which encapsulates a single move. (10 Points).
- Extend the game to allow play against a computer opponent. (Would need to allow an option for this at the beginning of the gameplay as well as a decision system for the computer's moves.) There is an "Unbeatable Algorithm" for Tic-Tac-Toe, which you can find with a little googling. (40 Points)