# Local Network Tic Tac Toe

#### Technical Report

Andromeda Weir, Areli Morales-Hernandez, Ben Price

## Introduction

Define objectives, importance in local multiplayer gaming,   
and the scope of the game based on socket programming.

#### Objectives

* Create a local network multiplayer tic-tac-toe game
* Implement a functional GUI for the game
* Define and program the underlying logic of the game
* Use socket programming to connect a host and opponent on the same network
* Learn about socket programming and its implementation

#### Importance

* It is important to keep local multiplayer gaming alive, as most multiplayer games are played online today
* Local multiplayer helps facilitate communication and bonding between peers
* The user experience of playing a tic-tac-toe game on one computer with two players can be awkward
* Solution: local network tic-tac-toe

#### Scope

* Users must be on same local network
* Users can play as many games as wanted
* Users must be able to run a python script on both machines

## Features

Detail each implemented feature and its significance in facilitating   
a multiplayer Tic Tac Toe game through socket communications.

#### Class Overview

The code base is split across 4 python files, each of which contains at least one class, listed below:

* Main.py
* TicTacToe.py
  1. GameEndException
  2. TicTacToe
* TicTacToeGUI.py
  1. Player
  2. TicTacToeBoard
* Connections.py
  1. formalConnectionInterface
  2. TicTacToeServer
  3. TicTacToeClient

TicTacToe class methods

* Constructor, optional parameters length and width
  1. Creates a new instance of the class, parameters length and width are used to determine the dimensions of the TicTacToe board, which can be made an arbitrary size. Defaults to 3 by 3 board
* setConnection, mandatory parameter Connection must implement formalConnectionInterface
  1. takes a reference to the established connection from one player to another to allow for the game object to make calls to network functions
* getBoardState:
  1. returns a 2 dimensional array representing the state of the game at time of call
* updateBoardState, mandatory parameter newBoardState must be a 2 dimensional array of ints
  1. called by a connection object to pass received information to the game, synchronizing the state between both players
  2. This function also checks to see if the new board state has a game ending Condition, and throws GameEndException if it does
* getBoardSize:
  1. returns an integer value representing the dimension of the board, useful only in non-standard dimensions
* takeTurn, mandatory parameters player, posX, posY:
  1. receives the int value of the player making a move, and the desired position that the player wishes to play at, throws ValueError if the move is illegal for any reason (not player’s turn, space already occupied)
* checkWin, mandatory parameters posX, posY, optional parameter checkTurn:
  1. checks to see if a move made at a given position results in a win condition
  2. if no parameter checkTurn is passed, function checks for win conditions on behalf of the player whose turn it currently is
  3. throws GameEndException when a Win or Tie condition is found
* resetGame:
  1. resets the state of the game to allow for a new game to be played  
       
        
       
        
       
        
       
     formalConnectionInterface class:
* Does not provide any methods of its own, only requires that its subclasses impliment method sendNewState

#### TicTacToeServer class methods

* Constructor, parameter game must be TicTacToe object
  1. creates, binds, and listens on a socket
  2. calls setConnection(self) on the TicTacToe object it is passed, allowing the two objects to call eachother’s public methods
* sendNewState, mandatory parameter boardState must be a 2d list of ints:
  1. translates the boardState into a byteArray object and sends it to the other player
* start:
  1. accepts a new connection to the server, and starts a thread that begins execution on method handle\_client
* handle\_client, mandatory parameter clientConnection
  1. listens for new data to be received, and calls TicTacToe.updateBoardState() with received data when the other player sends move data

TicTacToeClient class methods

* constructor, mandatory parameters hostIP, portNum, game:
  1. Initializes a connection to a server at a given host IP and port number, passes a reference to itself to the game once the connection is established, and creates a new thread beginning execution at recieveData
* sendNewState, mandatory parameter boardState must be a 2d array of ints:
  1. translates boardState to bytearray and sends to other player
* recieveData:

upon reciept of data from other player, updates the board state to synchronize the state between players

TicTacToeBoard methods  
  
 - Doesn’t have to be an exhaustive list, just the ones that are relevant to understanding how the class works and how they all fit together

## Implementation

Discuss technical insights, challenges faced, and significant   
decisions made during the socket-based development process.

#### Insights

* It is possible to make a decent UI in python using tkinter

#### Challenges Faced

* Communication between components
* Figuring out tkinter

#### Major Decisions

* Programming language to use for implementation: Python
* GUI library: tkinter
* Data structure representing game board: 2D array
* Version control: Github

## Results Analysis

Showcase outcomes from each feature and their contributions to   
effective local multiplayer gaming solely through socket programming.

## Conclusion

Summarize key findings.