ONLINE 2D SOCCER GAME USING SOCKS PROGRAMMING & AGENTS

MOHD KHAIRI BIN MOHD SHAUKHI

INFORMATION & COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNOLOGI PETRONAS
JULY 2007

Online 2D Soccer Game using Socks Programming & Agents

by

Mohd Khairi Bin Mohd Shaukhi

A project dissertation submitted to the
Information Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION COMMUNICATION TECHNOLOGY)

JULY 2007

Universiti Teknologi PETRONAS
Bandar Sri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Online 2D Soccer Game using Socks Programming & Agents

By

Mohd Khairi Bin Mohd Shaukhi

A project dissertation submitted to the
Information Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION COMMUNICATION TECHNOLOGY)

Approved by,

[illegible]
(Siti Rohkmah Mohd Shukri)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
July 2007

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

MOHD KHAIRI BIN MOHD SHAUKHI

ABSTRACT

A soccer game is designed to simulate a real soccer game that is played all over the world. The more realistic the game is, the more it will be played by people. The development of soccer games nowadays are moving towards creating 3D graphics that are almost human-like and artificial intelligence that are nearly perfect as a real soccer players. Some of them allow multiplayer gameplay which allows two people at different places connect to each other or to a game server and play the same game as opponent or teammates. This entire feature requires a personal computer which has fast processing power, a lot of memories, and a high-speed graphic processing chip. The objective of this project is to design a 2D soccer game which has the AI capability and multiplayer gameplay but only need minimum requirement to play it. To design such game, a lot of study and research had to be done in the field of artificial intelligence and networking. The scope of study also involves the research on the physics of the ball, such as the movement and interaction with other players. This system is developed using Visual Studio 2005 with the installment of Microsoft .NET 2.0 framework. Coding is done using WinSocks programming for the multiplayer mode, GDI+ Programming (.NET APIs) for graphical details and Visual Basic language for AI and other features. All features in the game have meet the targeted requirements. There are still rooms for improvements mostly on the AI.

ACKNOWLEDGEMENT

First of all, I would like to express lots of thank to Allah S.W.T for the blessing and also for my parents and family members for their priceless support, encouragement, constant love, valuable advices and their understanding in completing this project.

I also would like to extend my hearties gratitude to my supervisor, Siti Rohkmah Mohd Shukri for her guidance, knowledge, support, advices, experience and feedback throughout the process of completing this project. Her kindness, valuable advice and useful feedback are really important and give big contribution towards my project.

I am indebted to many individuals who helping me during the process of completing this project. They are people of my respect who involve directly or indirectly throughout this project.

Last but not least, a token of appreciation to all my colleagues for all the supports and cooperation during the development of the project. The support and assistance coming from all parties involved in this project is really appreciated. I sincerely would like to apologize for any mistaken that I made accidentally during this project.

Thank you to all of you.

TABLE OF CONTENTS

ABBREVIATIONS AND NOMENCLATURES

FYP	Final Year Project
ICT	Information and Communication Technology
UTP	Universiti Teknologi PETRONAS
WWW	World Wide Web
GUI	Graphical User Interface
2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
GDI	Graphics Device Interface
API	Application Programming Interface
LAN	Local Area Network
IP	Internet Protocol
PC	Personal Computer
FIFA	Fédération Internationale de Football Association
VB	Visual Basic
COM	Component Object Model
CPU	Central Processing Unit
RAM	Random Access Memory
MB	Megabyte

LIST OF FIGURES

CHAPTER 1
Introduction

1.0 Background Information

There are a lot of games that apply artificial intelligence movement to interact with the environment and its surrounding. For a soccer game, the artificial intelligence resides in the player reaction when it touches the ball and deciding what to do after it reacts to the ball. After the reaction by the player, the ball has to designed to move according on how the player kicks it. In doing this, calculation has to be done to make sure that the ball move logically and accordingly. In this project, the tasks that need to be done is researching and understanding how artificial intelligence works in a soccer match and applying it into a 2D soccer game.

Soccer games nowadays provide the opportunity for the human players to play against others through the Internet or LAN network. The game is seen by two or more people far from each other on their computer. Different players are controlled in a different network the game. This makes the game a lot more interesting as the multiplayer mode give the chance for the human player to compete against another human player. To apply the ability to play against other human player on a LAN network, Windows Sockets API (Winsock) can be installed together with the soccer game. One computer will act as a host while another computer can connect to the host as a client. It works as a medium of communication so that the 2D soccer game can be played over the LAN network.

The ball movement in accordance to the player's reaction with it is also taken into account. In such a way, the ball must be programmed to move like a normal ball in a real soccer game. In a 2D soccer game, the ball is programmed and calculated in terms of velocity, direction, angle and speed during movement.

1.1 Problem Identification

Among the problems that had been identified were:

1. Creating the player's artificial intelligence.
2. Programming the network protocol to support the multiplayer mode for the 2D soccer game.
3. Calculations on the velocity, angle, speed and reaction of ball motion.
4. Lagging occurs when there are a lot of calculations involving artificial intelligence.

1.2 Objective

Among the objectives are:

1. To implement Winsock into a 2D soccer game so that it can be played across the network by two players
2. To study the interaction of the artificial intelligence players with the ball.
3. To create a real-like simulation of real soccer game played by two human players.

1.3 Scope of Study

The scopes of this project are:

1.3.1 Artificial Intelligence

In this online 2D soccer game, artificial intelligence is applied to the players which aren't controlled by the human players. Artificial intelligence is designed for the movement of the players and what the players will do if they are close to the ball. When they are close to the ball, they will kick it towards the opponent's goal. Other artificial intelligence that is applied to the players is how the players will move around the field. There are regions set on the field so that the AI players do not move around freely around the field, but they will move inside their region only so that the game play is not messed up. In doing so, the field is divided into 3 vertical parts with each side's AI player controlling one region independently.

1.3.2 Modular engine

Modular engine is build into this game so that it allows expandability onto almost all aspects of the game engine including non-player objects, AI players and the ball. Modular engine is the ability of the program to accept broad number of inputs and process it with one generic code. The inputs can therefore vary from different sources whilst still retaining compatibility and performance as well as coding simplicity. The engines are therefore set to have a lot of functions. Rather than running the whole code sequentially, the engine will run certain function based on the input being fed. The input can be retrieve from various sources. A good example of a modular engine is the movement function. The movement of the player controlled by human is by using mouse or keyboard. It is easier to set up the movement keys because the movement function is a modular engine. AI currently is not set up as a modular engine. Each player has its own AI rather than applying one general coding to all the AI players. If AI is set as a modular engine, all of the AI players will run one particular AI code multiple times, depending on the its role on the field. The workload for AI to run as normal coding or as a modular engine are practically the same, so applying the AI as a modular engine will be a study on optimizing the program later.

1.3.3 Ball Movement

To make this 2D soccer game as real as possible, the ball movement should be realistic. The study of the ball movement is based on trigonometric calculation of vectors. Calculations is done when the ball interact with other players on the field. Ball speed is currently set as a constant value upon contact with a decaying velocity as it moves. When the ball touches other objects, for example, other players, the ball should move accordingly as a real ball would do. It would bounce back or reflected to other sides.

1.3.4 Multiplayer Mode

Multiplayer mode is the ability of the 2D soccer game to be played across the LAN network by two different human players at two different computers. One will act as a host, another is a client. A client will connect to the host by putting the IP address of the host. To make this applicable, Winsock programming is used. One human player will control one player in one team while the other human player will control another in the opposite team. The computation engine is non-existent (not functional) on the client as when it is run in client mode, all it does is relay keystrokes and various inputs to the host's modular input system. All calculations will therefore be done on the hosts, and the results are returned to the client at fast intervals. This is done so that there is not much lagging created between host and client. If both are set to run calculations independently, synchronization errors have a chance of occurring and the game will not be synchronized. Theoretically, the response time between calculations on the host machine and the display on the client machine is very dependant upon the network latency of the connection rather than the bandwidth. Therefore, network solutions with low inherent latency are much more preferred rather than one with a higher bandwidth but lackluster latency.

1.3.5 Game Formation

Compared to a real soccer game, this game will only have 4 players for each team. 3 of them will be positioned as field players and 1 will be the goalkeeper. It is more reminiscent of futsal games instead, with the only variation being the implementation of goalkeepers one each side. So, on the field, the maximum player that is allowed is 8 players. The goalkeeper will stay at the goal line and will move only on the line. It will block the ball from getting across the line. Two of the field players will also have Artificial Intelligence while human player will control another field player. The human player will be able to choose which of the 3 field players to control according to who is nearest to the ball. The difference between the controlled player and other AI player is that there will be a yellow ring on the human-controlled player that will distinguish it between other players. When the human player switch to other player, the yellow ring will also move to the currently controlled player.

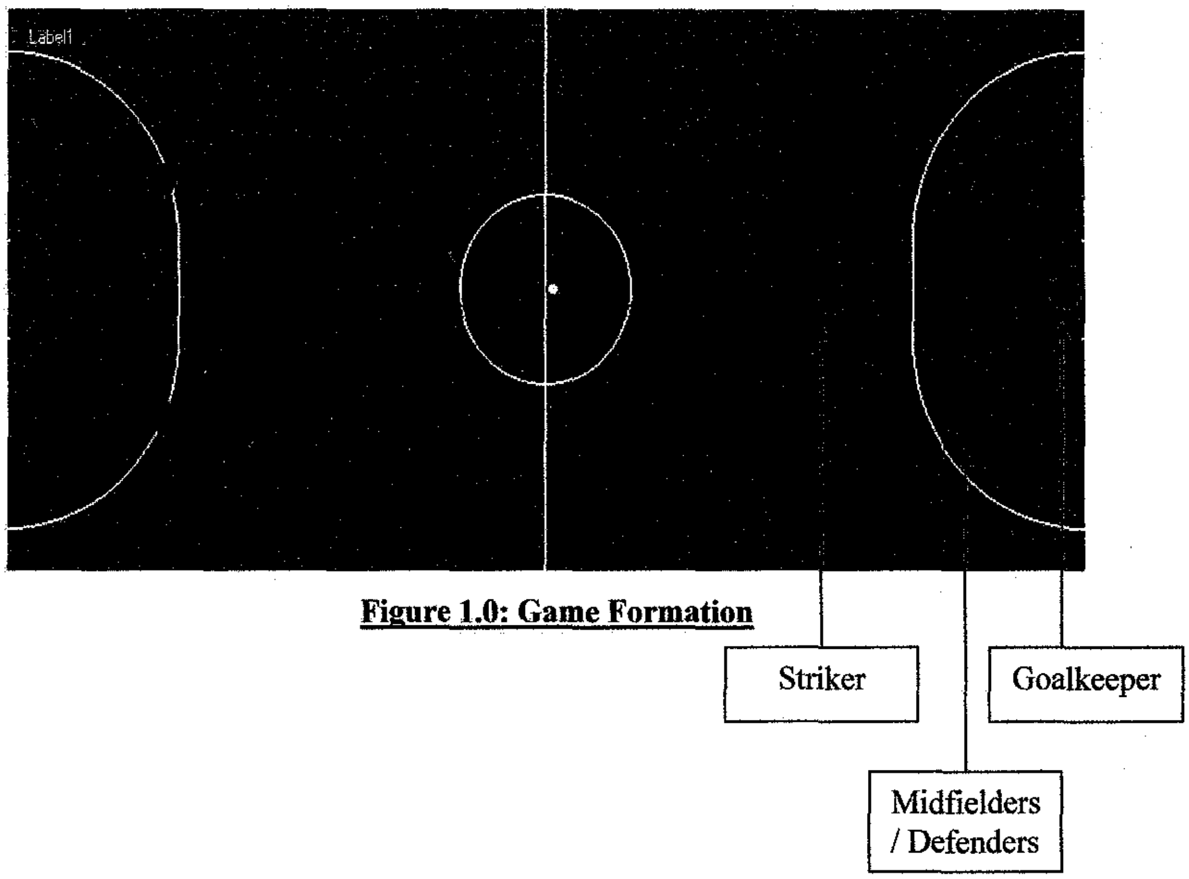


Figure 1.0: Game Formation

CHAPTER 2

Literature Review

2.0 What is a 2D soccer game?

A 2D soccer game is a PC game [1] which is played by one player simulating a soccer match [2]. The simulation is done by giving control to the human player a representation of him/her in the game in the form of a 2D character. A two-dimensional view of the game is the view from the top of the soccer field. The human player will see the field as he is seeing it from the sky with full view from one end of the goal to another. An example of a simple 2D soccer game with the same characteristics is Champion Soccer [3] and Championship Soccer [4]. People who tend to play 2D soccer game are interested in the quickness of response and the total control that they can have when playing a 2D soccer game instead of a 3D soccer game.

2.1 2D soccer game VS 3D soccer game

The major difference in deciding which of these two are better is the graphics. The graphics in a game is important to let the human players feel that the game is representing a real game. According to Wikipedia [5], 3D computer graphics are different from 2D computer graphics in that a three-dimensional representation of geometric data is stored in the computer for the purposes of performing calculations and rendering 2D images. So the response time for a 3D soccer game from the time a human player push the buttons to move the player in the field is slower than a 2D soccer game. Unless the PC which used to render the 3D soccer game is powerful, it has a slower response time. This is one of the major reasons why people tend to stick to 2D soccer game and sacrifices the beautiful graphics for lower ones [6]. The latest release from Electronic Arts (EA) Sports is their popular soccer franchise game; FIFA 07 [7] has the best 3D graphics compared to other 3D soccer games. But it uses a lot of resources especially the memory part of a host PC. Not all PCs can support this game and most of the current 3D soccer games. A popular 2D soccer game which highly popular around the 90's is Sensible Soccer [07]. It still releases the series until today, but in 3D. This game sets the standard for 2D soccer games which featured a zoomed-out bird's-eye view, editable national, club and custom teams and gameplay ahead of its time utilizing a relatively simple and user-friendly control scheme.

CHAPTER 3

Methodology

3.0 Procedure Identification

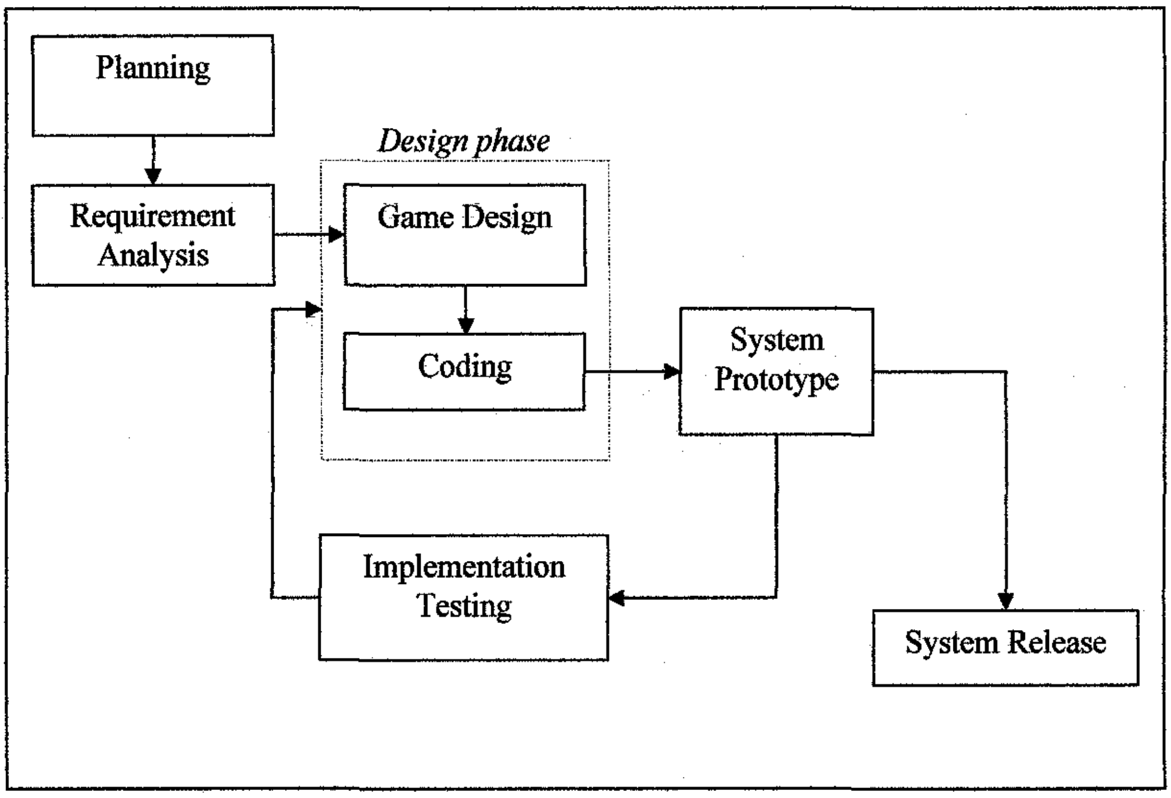


Figure 2.0: Prototype Model

The discussion will be focused on the method used in developing the system. Prototype Model consists of:

(i) Planning

(ii) Requirement Analysis

(iii) Design Phase – Coding

(iv) System Prototype

(vi) Implementation Testing,

(vii) System Release.

Each of the phases has its own role and reflects on how much the system will be progressing throughout the development stages of the system.

A prototyping based methodology performs the analysis, design and implementation phases concurrently and repeatedly in a cycle until the system is completed.

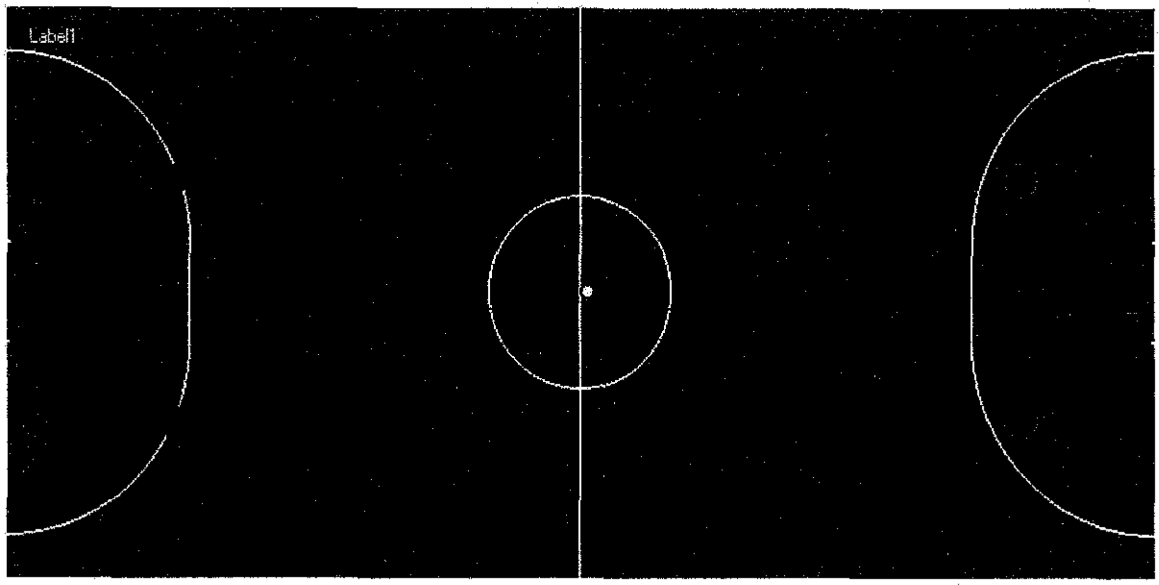
3.1 Planning

The author conducts planning for the project to identify the task for the system development. The author planned to do this project in 3 main phases.

3.1.1 First Phase

This first phase is to create the basic foundation of the soccer game. This includes the human-controlled player and the ball movement. The human-controlled player will be control using arrow keys on the keyboard. Other keys will be assigned to the player for the player to dribble or kick the ball. Only one player will be designed in this phase which is controlled by the human player. Both the ball and the player will be placed on a 2D soccer field with graphics. The graphics include the goal post, field line and penalty box.

The ball's movement is designed in this phase. This includes the velocity, reaction and gravity effects. Velocity is set so to match the keys that are pressed by the human players, or according to the mouse location if it is set as so. If the player kicks the ball, the ball will move further than when being passed or dribble. The friction effects is set so that the ball will slowed down gradually after being kick like a normal ball in a soccer game.



● ● - Player

○ - Ball

Figure 3.0: Basic Objects in Field

3.1.2 Second Phase

The second phase will focus on the network and AI players. Socks programming will be applied during this phase for the network play. Two game clients will be designed differently, one being the host, and another is the client. The host will run the game acting as the game host while the client will connect to the host. The client's game coding will not run the same as the host, but it will run as minimal as possible. The client will just send keystrokes to the host to control the player. The host will do all the calculation in the coding thus the client is running on low resources for the game. The host will need more resources to run the game. This network design is created to let computers with low end requirements can run the game as the client. In network playability, modular engine is designed so that it inputs the human players keystrokes from the client to the host allowing the client to act as a dummy terminal.

Main Classes

Two main engines are written for this game.

1. Main Engine

The main engine will cater mostly all of the calculations. The AI engine is written in here. All calls are made into this main engine. Another main event that the main engine holds is the input made by the human player using the keyboard or the mouse.

2. Network

Network engine will handle the networking part of the game. Both host and client have their own networking engine.

The reason this game has two separate classes is that to enable synchronizing when multiplayer mode is on. Calculations has to be made at the host to avoid lagging when a low end computer is used. For network play, modular engine is used. The host is set to accept only keystrokes from the client allowing the client to act as a dummy terminal.

AI Algorithm

1. Check Change Control

   The first step is to check which player is control by the human player. The human player can switch between the three field players.

2. Do AI
3. Calculate Player Angle
4. Move Player

   An engine module is called which moves avtive player objects based on keyinput struct state

5. Move Ball

   An engine module is called which moves the ball according to its angle and velocity characteristics

6. Move Goalkeeper

   The goalkeeper is set to move randomly but in its own territory

7. Collision Check

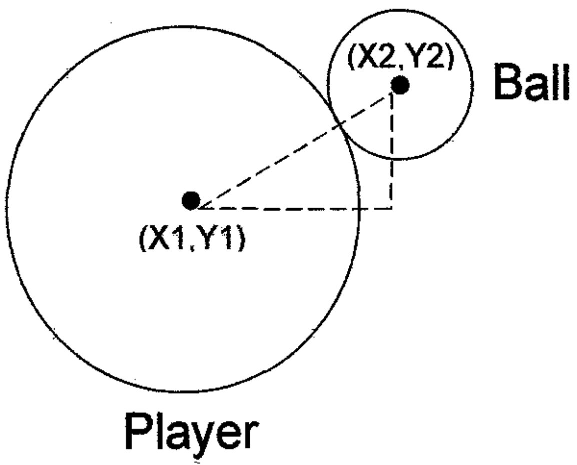
Check for collision between all players and the ball, returning the angle for the ball if collision occurs

If a goal is scored, subroutine reset is call. All players and ball position is reset and the game is restart.

Collision Detection

When the human-controlled player kicks the ball, this is how the calculation is made:

1. Center pixel (x and y) is took from both the ball and the player
2. Pythagoras formula is used to calculate the distance
3. If the distance is less than half of the ball and player's width, collision is detected.
4. The ball vector is set as (ball center x - player center x) divide by distance for x axis, (ball center y - player center y) divide by distance for y axis. .



```
'Get Player Centerpoint
Dim PCenterX As Double = picPlayer1.Location.X + (picPlayer1.Width / 2)
Dim PCenterY As Double = picPlayer1.Location.Y + (picPlayer1.Height / 2)
'Get Ball Centerpoint
Dim BCenterX As Double = picBall.Location.X + (picBall.Width / 2)
Dim BCenterY As Double = picBall.Location.Y + (picBall.Height / 2)

'Check collision
Dim P1Distance As Double = Math.Abs(((BCenterX - PCenterX) ^ 2 + (BCenterY - PCenterY) ^ 2) ^ 0.5)
If P1Distance <= (picPlayer1.Width + picBall.Width) / 2 Then
'Set Ball vector
```

```
BallVectorX = (BCenterX - PCenterX) / P1Distance
BallVectorY = (BCenterY - PCenterY) / P1Distance
BallExactVelocity = BallVelocitySetting
BallTimer.Tag = 0
End If
```

Figure 4.0: Collision Detection VB Codes

Drawing

Basic Circle Object Class is used to draw the ball and players. Vector format is used for the eclipse drawing. The reason vector drawing is used because it allows faster computation and smoother movement rather than using graphical picture such as .jpeg or .gif.

Input

Modular design allows input to be accepted from different sources such as keyboard or LAN. Subroutine sets input when called by outside sources. It can be called from keyboard presses or LAN data arrival.

Movement

Human player can use mouse or keyboard to control the computer player. Both of these inputs are already integrated into VB. The key mapping are:

Keyboard

W = Move player up

A = Move player left

S = Move player down

D = Move player rigt

X/Z = Change to next player

Mouse

Move cursor = Move player

Click = Change controls to keyboard

X/Z = Change to next player

For multiplayer, the keyboard press and release function is fetch and send to the host engine for calculation. The mouse will send its x and y position. Using the mouse is better for controlling the player because it allows 360 degrees of movement, but it is hard for inexperienced human player to use.

Player Change Control

Controlled player is set as P1. So when the human player presses X or Z , the P1 is change to the next player object.

Network

MSWinsock library is used in the socks programming. The protocol used to create the connection is TCPP. The network algorithm used is :

1. Host Listen for connecting IP
2. Client put the host's IP
3. Host detect connection
4. Host establish Connection
5. Game Start

Other Events

17 events are set in this game including engine, field, host, join, score and sound. Main sound which is the cheering supporters are stream continuously until an event happen such as a goal or the ball when outside the field. If a goal is scored, sound event is change.

3.4 System Prototype

In this phase, prototypes will be produced to a certain extent of functional requirements. The system prototype is not an end product; it will be refined for each requirements being added to make the system fully functional. This prototype will be tested for implementation in the next phase to ensure that the specific requirement works effectively and efficiently.

3.5 Implementation Testing

In this phase, the prototypes that are produced to a certain extent will be deployed in the application server and testing will be done to ensure that functional requirements are working. This will save times to debug any problem compare to implementing full system and checking for every bug.

The testing will be done based on the functional requirements and its integration with other functional requirements. With this method of testing, the implementation of the system will be done increasingly. Testing will be done in 2 stages, Alpha testing and Beta testing.

3.5.1 Alpha Testing

Alpha testing will be conducted by the author during the development phase and also when the game is fully operational. The objective of this test is to eliminate as much errors as possible that are visible to the developer before releasing it to the public for beta testing.

3.5.2 Beta Testing

Beta testing is conducted after alpha testing by a group of beta tester which is voluntary to anyone who wants to test it. There are two groups of beta testers which are gamers that have been playing soccer game for a long time and those who new to this genre.

3.6 System Release

This is the phase where the final system is expected to be completed, hopefully without a bug. This product will then be deployed as a multiplayer game with the ability of two people playing in one soccer match through a LAN network. The system will be implemented in the real environment and will be considered as a released title.

Three prototypes were released prior to the three phases that were planned. The final released was compiled into a Windows Installer Package. The size is 10Mb. The installer will install all the necessary files into the PC including the game itself, sounds, and also MSWinsock.reg. An administrator account is needed to install the game into a PC because the MSWinsock registry need to be embedded into the System32. User can choose the installation directory of the game. Icons and other elements are also included after the installation. The final name of the game is Winball.

3.7 System Requirements

Development Tools

-Visual Studio 2005

-Adobe Photoshop

- .NET2.0

Coding

-Visual Basic

-WinSocks Programming

-GDI+ Programming (.NET APIs)

Host and Client

-2GHz CPU Processor

-.NET 2.0 Framework

-Windows 2000/XP/Vista

-256MB RAM

-20MB Hard disk Space

CHAPTER 4
Result and Discussion

4.0 Result

This report describes the basic theories that need to be applied into the 2D Online Soccer Game. The theories include the collision detection of the ball when the ball interacts with other objects on the soccer field. Also included is how the human-player controls the player's movement in the game. Both of these theories have been studied and developed in the prototype release of the game. According to the methodology chosen by the author, the design phase which include the game and coding design have passed and are going into prototyping stage for the first phase of the game. It covers creating the basic foundation of the soccer game including the human-controlled player and the ball movement

4.1 Discussion

Two more phases need to be covered after the first phase has finished. It covers the computer players' artificial intelligence, network programming and graphics details. The players' artificial intelligence will involve studies on object's interaction in VB and mathematical formulas of the players' movement. Two types of games will be created to handle the socks programming architecture. Main host will be developed first and the coding will be strip down to as minimal as possible for the connecting client to use. The client will only have to calculate keystrokes calculation for the game. Graphical details will be applied to make the game more interesting and can compete with other 2D online soccer game. Extra graphics and sounds for the supporters will be added. Graphics includes a scoreboard that will update when a goal is scored and shows the two teams' logo. Sound effects which will be added are the supporters' chant and the sound effect then the players' dribbles or kick the ball. Interesting and eye catching graphical user interface (GUI) menu will be designed into the game.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

The development of this 2D online soccer game is to study and apply artificial intelligence into the players so that it can interact with other objects and players. The author also does study on Socks programming to be applied into the game to allow multiplayer gameplay. There are issues after the product has been released, mainly on applying more AI players on the field. The lagging issue has been resolved by using multithreading, and although more players can be applied into the game, AI issues will still be a problem. More AI means more logical coding that need to be added into each player. As discussed earlier, the AI engine is not a modular engine, meaning each player has its own AI codes. Creating more players with different AI may result in a messy gameplay and also coalition in the logic programming aspects. If this project is to be continued at a later stage, a good recommendation is to make the AI engine to be a modular engine. A general AI coding need to be research and written that can be applied to each of the players. This will make the game run smoothly and the AI players will move accordingly.

REFERENCES

[1] Stahl, T, What is A PC Game, Chronology of the History of Video Games [http://www.thocp.net/software/games/golden_age.htm; accessed on 26th February 2007]

[2] Cage, N, article entitled Basic Soccer Game Rules - History from British universities in 1855 [http://www.about-soccer.com/soccer-game-rules.shtml; accessed on 26th February 2007]

[3] Koffiepad, R, *MobyGames Database*, Champion Soccer for MSX [http://www.mobygames.com/game/champion-soccer; accessed on 26th February 2007]

[4] Gold, L, *MobyGames Database*, Championship Soccer for Atari 2600 [http://www.mobygames.com/game/championship-soccer; accessed on 26th February 2007]

[5] Wikipedia, definition of "*3D Computer Graphics*" article [http://en.wikipedia.org/wiki/3D_computer_graphics; accessed on 26th February 2007]

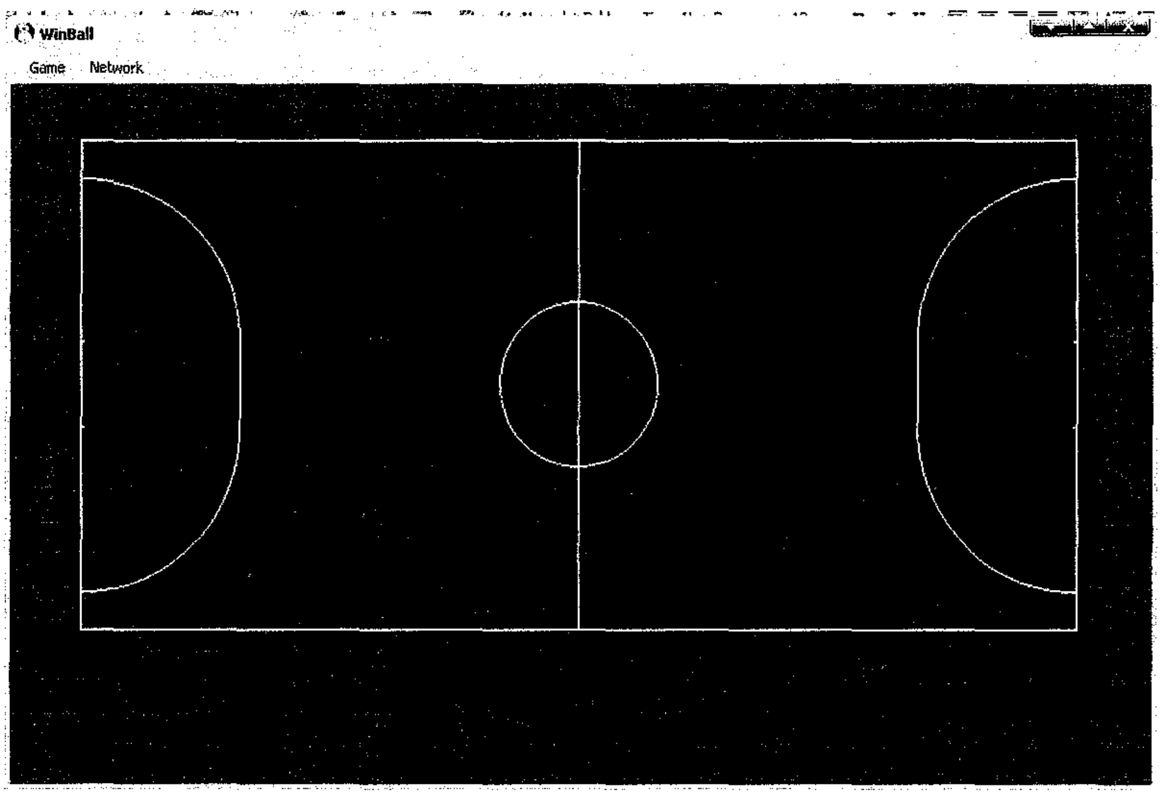
[6] Teggo, R, article entitled *2D via 3D approach* [http://spritecraft.teggo.com/features/2dvia3d.shtml; accessed on 26th February 2007]

[7] Light, M, Overview of FIFA 07 [http://www.fifa07.ea.com/home.asp?lang=us; accessed on 26th February 2007]

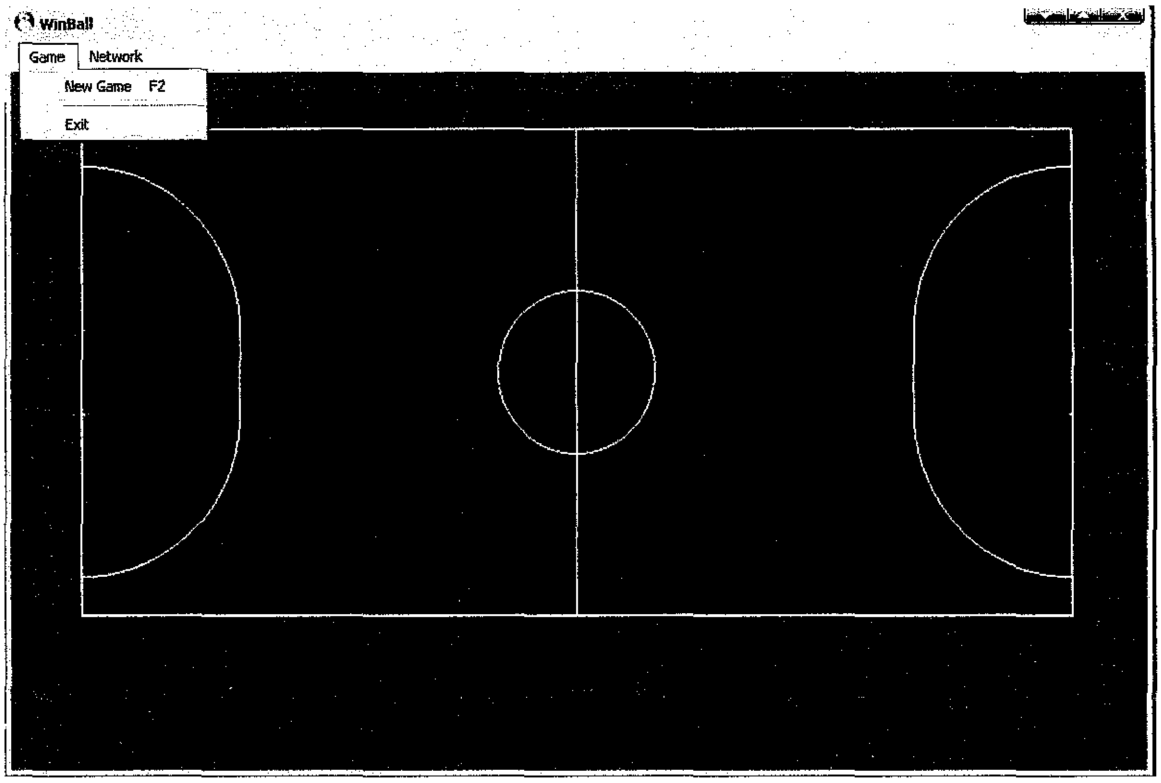
[8] Wikipedia, definition of "*Visual Basic*" article
[http://en.wikipedia.org/wiki/Visual_basic; accessed on 2nd May 2007]

APPENDICES

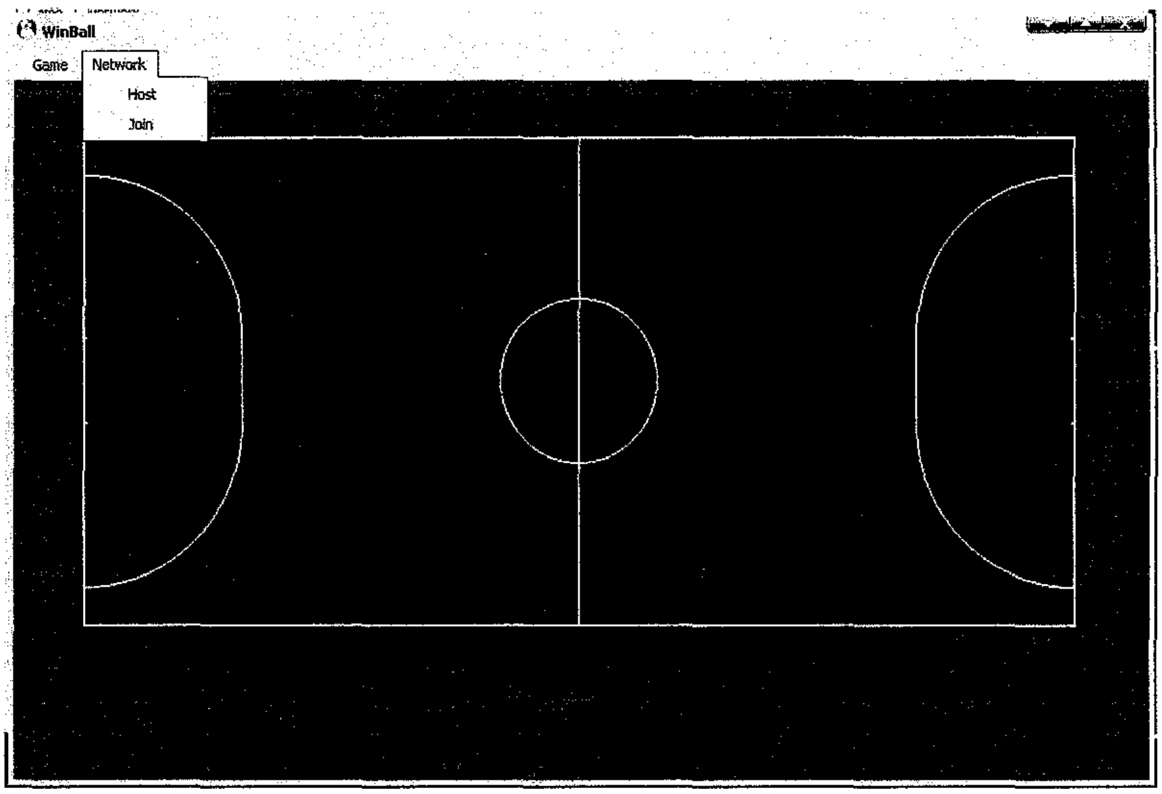
APPENDIX 1: Main GUI of the game



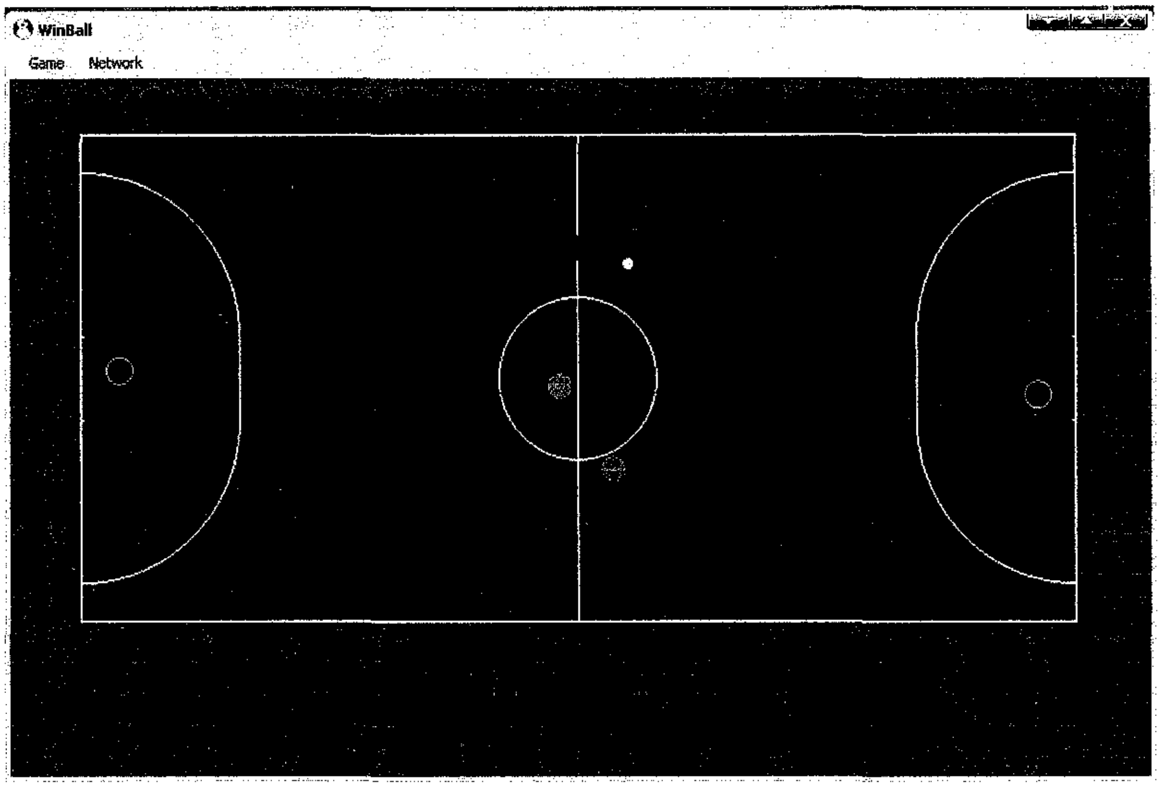
APPENDIX 2: Game Menu



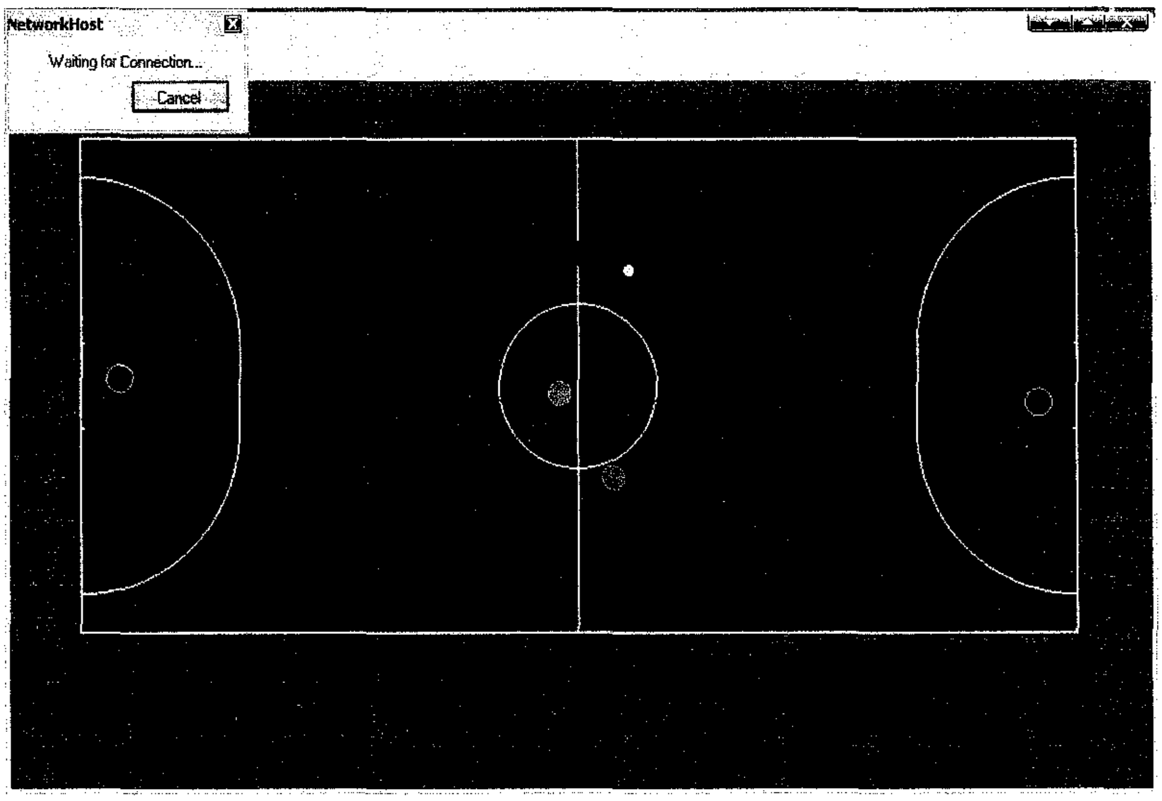
APPENDIX 3: Network Menu



APPENDIX 4: Single Player



APPENDIX 5: Host Waiting Box



APPENDIX 7: frmMain.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmMain
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Dim resources As System.ComponentModel.ComponentResourceManager = New
System.ComponentModel.ComponentResourceManager(GetType(frmMain))
        Me.EngineTicker = New System.Windows.Forms.Timer(Me.components)
        Me.Field = New System.Windows.Forms.PictureBox
        Me.mnuMenu = New System.Windows.Forms.MenuStrip
        Me.GameToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem
        Me.mnuNewGame = New System.Windows.Forms.ToolStripMenuItem
        Me.ToolStripSeparator1 = New System.Windows.Forms.ToolStripSeparator
        Me.mnuExit = New System.Windows.Forms.ToolStripMenuItem
        Me.NetworkToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem
        Me.mnuHost = New System.Windows.Forms.ToolStripMenuItem
        Me.mnuJoin = New System.Windows.Forms.ToolStripMenuItem
        Me.lblP1Score = New System.Windows.Forms.Label
        Me.lblSeparate = New System.Windows.Forms.Label
        Me.lblP2Score = New System.Windows.Forms.Label
        Me.NetworkTicker = New System.Windows.Forms.Timer(Me.components)
        Me.SoundSystem = New AxEASYSOUNDLib.AxESound
        Me.wskNetSend = New AxMSWinsockLib.AxWinsock
        Me.wskNetListen = New AxMSWinsockLib.AxWinsock
        CType(Me.Field, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.mnuMenu.SuspendLayout()
        CType(Me.SoundSystem, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.wskNetSend, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.wskNetListen, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'EngineTicker
        '
        Me.EngineTicker.Interval = 10
        '
        'Field
        '
        Me.Field.BackgroundImage = CType(resources.GetObject("Field.BackgroundImage"), System.Drawing.Image)
        Me.Field.Location = New System.Drawing.Point(53, 65)
        Me.Field.Name = "Field"
        Me.Field.Size = New System.Drawing.Size(760, 360)
        Me.Field.TabIndex = 0
        Me.Field.TabStop = False
        '
        'mnuMenu
        Me.lblSeparate.Font = New System.Drawing.Font("Courier New", 36.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lblSeparate.Location = New System.Drawing.Point(409, 449)
        Me.lblSeparate.Name = "lblSeparate"
        Me.lblSeparate.Size = New System.Drawing.Size(52, 54)
        Me.lblSeparate.TabIndex = 4
        Me.lblSeparate.Text = "-"
        '
        'lblP2Score
        '
        Me.lblP2Score.AutoSize = True
        Me.lblP2Score.Font = New System.Drawing.Font("Courier New", 36.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lblP2Score.Location = New System.Drawing.Point(467, 449)
        Me.lblP2Score.Name = "lblP2Score"
        Me.lblP2Score.Size = New System.Drawing.Size(52, 54)
        Me.lblP2Score.TabIndex = 5
        Me.lblP2Score.Text = "0"
        '
        'NetworkTicker
        '
        Me.NetworkTicker.Interval = 10
        '
        'SoundSystem
        '
        Me.SoundSystem.Enabled = True
        Me.SoundSystem.Location = New System.Drawing.Point(729, 465)
        Me.SoundSystem.Name = "SoundSystem"
        Me.SoundSystem.OcxState = CType(resources.GetObject("SoundSystem.OcxState"), System.Windows.Forms.AxHost.State)
        Me.SoundSystem.Size = New System.Drawing.Size(60, 60)
        Me.SoundSystem.TabIndex = 7
        '
        'wskNetSend
        '
        Me.wskNetSend.Enabled = True
        Me.wskNetSend.Location = New System.Drawing.Point(795, 497)
        Me.wskNetSend.Name = "wskNetSend"
        Me.wskNetSend.OcxState = CType(resources.GetObject("wskNetSend.OcxState"), System.Windows.Forms.AxHost.State)
        Me.wskNetSend.Size = New System.Drawing.Size(28, 28)
        Me.wskNetSend.TabIndex = 6
        '
        'wskNetListen
        '
        Me.wskNetListen.Enabled = True
        Me.wskNetListen.Location = New System.Drawing.Point(829, 497)
        Me.wskNetListen.Name = "wskNetListen"
        Me.wskNetListen.OcxState = CType(resources.GetObject("wskNetListen.OcxState"), System.Windows.Forms.AxHost.State)
        Me.wskNetListen.Size = New System.Drawing.Size(28, 28)
        Me.wskNetListen.TabIndex = 1
        '
        'frmMain
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.Color.SteelBlue
        Me.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Center
        Me.ClientSize = New System.Drawing.Size(869, 537)
        Me.Controls.Add(Me.SoundSystem)
        Me.Controls.Add(Me.wskNetSend)
        Me.Controls.Add(Me.lblP2Score)
        Me.Controls.Add(Me.lblSeparate)
        Me.Controls.Add(Me.lblP1Score)
        Me.Controls.Add(Me.wskNetListen)
        Me.Controls.Add(Me.Field)
        Me.Controls.Add(Me.mnuMenu)
        Me.Cursor = System.Windows.Forms.Cursors.Cross
        Me.DoubleBuffered = True
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.MainMenuStrip = Me.mnuMenu
        Me.Name = "frmMain"
        Me.Text = "WinBall"
        CType(Me.Field, System.ComponentModel.ISupportInitialize).EndInit()
        Me.mnuMenu.ResumeLayout(False)
        Me.mnuMenu.PerformLayout()
        CType(Me.SoundSystem, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.wskNetSend, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.wskNetListen, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents EngineTicker As System.Windows.Forms.Timer
    Friend WithEvents Field As System.Windows.Forms.PictureBox
    Friend WithEvents wskNetListen As AxMSWinsockLib.AxWinsock
    Friend WithEvents mnuMenu As System.Windows.Forms.MenuStrip
    Friend WithEvents GameToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents mnuNewGame As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ToolStripSeparator1 As System.Windows.Forms.ToolStripSeparator
    Friend WithEvents mnuExit As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents NetworkToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents mnuHost As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents mnuJoin As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents lblP1Score As System.Windows.Forms.Label
    Friend WithEvents lblSeparate As System.Windows.Forms.Label
    Friend WithEvents lblP2Score As System.Windows.Forms.Label
    Friend WithEvents NetworkTicker As System.Windows.Forms.Timer
    Friend WithEvents wskNetSend As AxMSWinsockLib.AxWinsock
    Friend WithEvents SoundSystem As AxEASYSOUNDLib.AxESound

End Class
```

APPENDIX 8: Engine.vb

```
Public Class SphereEngine

  'Basic Circle Object class, used for balls and player counters
  Public Structure CircleObject
    Dim X As Single
    Dim Y As Single
    Dim Size As Int16
    Dim Angle As Int16
    Dim Velocity As Single
    Dim AITag As Int16
  End Structure

  'Player Input. Modular design allows input to be accepted from different sources, keyboard or LAN.
  Private Structure Inputs
    Dim Up As Boolean
    Dim Down As Boolean
    Dim Left As Boolean
    Dim Right As Boolean
    Dim ChangeActive As Boolean
    Dim Power As Boolean
  End Structure

  'Declares variables, based on above structures
  Public Player(1, 2) As CircleObject 'Player objects, 6 total, 3 per side
  Private KeyInput(1, 2) As Inputs      'Define 2 sets of keyinputs, one for each side
  Public Ball As CircleObject        'Define ball object
  Public Goalie(1) As CircleObject
  Public P1ChangeP As Short
  Public P2ChangeP As Short
  Public NetworkPlay As Boolean = False
  Public NetworkIsHost As Boolean = False

  Public Sub SetInput(ByVal Team As Int16, ByVal Player As Int16, ByVal Key As Int16, ByVal State As Boolean)
    'Subroutine sets input when called by outside sources.
    'Can be called from keyboard presses or LAN data arrival
    Dim character As String = Chr(Key)
    Select Case character
      Case "A"
        KeyInput(Team, Player).Left = State
      Case "a"
        KeyInput(Team, Player).Left = State
      Case "W"
        KeyInput(Team, Player).Up = State
      Case "w"
        KeyInput(Team, Player).Up = State
      Case "D"
        KeyInput(Team, Player).Right = State
      Case "d"
        KeyInput(Team, Player).Right = State
      Case "S"
        KeyInput(Team, Player).Down = State
      Case "s"
        KeyInput(Team, Player).Down = State
      Case "Z"
        If State = False Then Exit Select
        If Team = 0 Then P1ChangeP = 1 Else P2ChangeP = 0
      Case "z"
        If State = False Then Exit Select
        If Team = 0 Then P1ChangeP = 1 Else P2ChangeP = 0
      Case "X"
        If State = False Then Exit Select
        If Team = 0 Then P1ChangeP = -1 Else P2ChangeP = 0
      Case "x"
```

```
            If State = False Then Exit Select
            If Team = 0 Then P1ChangeP = -1 Else P2ChangeP = 0
        End Select
    End Sub

    Private Function GetAngle(ByVal SourceX As Single, ByVal SourceY As Single, ByVal TargetX As Single, ByVal TargetY As
Single) As Integer
        Dim TriX As Single = (TargetX - SourceX)
        Dim TriY As Single = (TargetY - SourceY)
        Dim Angle As Single
        Angle = Math.Atan(TriY / TriX)
        'Convert radians to degrees
        If TargetY >= SourceY Then
            If TargetX >= SourceX Then
                'Bottom Right
                Angle = (Angle / Math.PI) * 180
            Else
                'Bottom Left
                Angle = 180 + ((Angle / Math.PI) * 180)
            End If
        Else
            If TargetX >= SourceX Then
                'Top Right
                Angle = (Angle / Math.PI) * 180 + 360
            Else
                'Top Left
                Angle = 180 + ((Angle / Math.PI) * 180)
            End If
        End If
        Return Int(Angle)

    End Function

    Private Function CheckCollision(ByVal obj1 As CircleObject, ByVal obj2 As CircleObject, ByVal simple As Boolean) As Int16
        'Basic collision detection code detection function.
        'Call with 'simple' will skip the collision angle calculations
        'and just return yes or no for collision
        'Simple disabled will return the collision angle referring
        'to the trajectory of the second object after collision
        Dim Obj1CenterX As Single = obj1.X + (obj1.Size / 2)
        Dim Obj1CenterY As Single = obj1.Y + (obj1.Size / 2)
        Dim Obj2CenterX As Single = obj2.X + (obj2.Size / 2)
        Dim Obj2CenterY As Single = obj2.Y + (obj2.Size / 2)
        Dim Distance As Single = ((((Obj2CenterX - Obj1CenterX) ^ 2) + ((Obj2CenterY - Obj1CenterY) ^ 2)) ^ 0.5)
        If Not simple Then
            If Distance <= ((obj1.Size + obj2.Size) / 2) Then
                Return Int(GetAngle(Obj1CenterX, Obj1CenterY, Obj2CenterX, Obj2CenterY))
            Else
                Return -1
            End If
        Else
            If Distance <= ((obj1.Size + obj2.Size) / 2) Then Return 1 Else Return 0
        End If
    End Function

    Public Sub Reset()
        'Subroutine call resets positions for all objects on the field
        Ball.Velocity = 0
        Ball.Angle = 0

        Player(0, 0).X = 190
        Player(0, 0).Y = 175
        Player(0, 0).AITag = 0

        Goalie(0).X = 20
        Goalie(0).Y = 175
        Goalie(0).AITag = 0

        Player(1, 0).X = 570
        Player(1, 0).Y = 175
    Player(1, 0).AITag = 0

    Player(0, 1).X = 100
    Player(0, 1).Y = 100
    Player(0, 1).AITag = 1

    Player(0, 2).X = 100
    Player(0, 2).Y = 250
    Player(0, 2).AITag = 2

    Player(1, 1).X = 660
    Player(1, 1).Y = 100
    Player(1, 1).AITag = 1

    Player(1, 2).X = 660
    Player(1, 2).Y = 250
    Player(1, 2).AITag = 2

    Goalie(1).X = frmMain.Field.Width - 40
    Goalie(1).Y = 175
    Goalie(1).AITag = 0

    Dim i As Int16
    Dim j As Int16
    For i = 0 To 1
        For j = 0 To 2
            With Player(i, j)
                .Angle = 0
                .Size = 20
            End With
        Next
        With Goalie(i)
            .Angle = 0
            .Size = 20
        End With
    Next

    Ball.X = 380
    Ball.Y = 175
    Ball.Size = 10

    frmMain.P1Active = 0
    frmMain.P2Active = 0
End Sub

Public Function Tick() As Boolean
    'Calculation engine tick algorithm. Calls appropriate subs in order
    If (Not NetworkPlay) Or (NetworkPlay And NetworkIsHost) Then
        Call CheckChangeControl()
        Call DoAI()
        Call CalcPlayerAngle()
        Call MovePlayers()
        Call MoveBall()
        Call MoveGoalie()
        If CheckBorders() = False Then Return False : Exit Function
        Call DoCollide()
        If NetworkPlay And NetworkIsHost Then Call SendEngineVars()
    End If
    frmMain.Field.Refresh()
    Application.DoEvents()
    Return True
End Function

Private Sub CheckChangeControl()
    Dim i As Short = 0
    If P1ChangeP <> 0 Then
        For i = 0 To 2
            Player(0, i).AITag += P1ChangeP
            If Player(0, i).AITag = 3 Then Player(0, i).AITag = 0 Else If Player(0, i).AITag = -1 Then Player(0, i).AITag = 2
            If Player(0, i).AITag = 0 Then frmMain.P1Active = i
        Next
        P1ChangeP = False
    End If
    If P2ChangeP <> 0 Then
        For i = 0 To 2
            Player(1, i).AITag += P2ChangeP
            If Player(1, i).AITag = 3 Then Player(1, i).AITag = 0 Else If Player(1, i).AITag = -1 Then Player(1, i).AITag = 2
            If Player(1, i).AITag = 0 Then frmMain.P2Active = i
        Next
        P2ChangeP = False
    End If
End Sub

Private Sub DoAI()
    Dim i As Short
    Dim j As Short
    For i = 0 To 1
        For j = 0 To 2
            CalcAIPlayer(Player(i, j), (i = 0))
        Next
    Next
End Sub

Private Sub CalcAIPlayer(ByRef Pl As CircleObject, ByVal IsP1 As Boolean)
    Dim BallCenterX As Single = Ball.X + 5
    Dim BallCenterY As Single = Ball.Y + 5
    Dim PCenterX As Single = Pl.X + 10
    Dim PCenterY As Single = Pl.Y + 10
    Dim PTargetX As Single
    Dim PTargetY As Single
    Dim Angle As Integer
    If (IsP1 And Pl.AITag = 0) Or (Not IsP1 And Pl.AITag = 0 And NetworkPlay And NetworkIsHost) Then Exit Sub

    If IsP1 Then
        Angle = GetAngle(BallCenterX, BallCenterY, frmMain.Field.Width - 1, frmMain.Field.Height / 2)
        Select Case Pl.AITag
            Case 1 ' Player 1 (Mid)
                If Angle < 180 Then
                    Pl.Angle = 180
                    If BallCenterX > PCenterX + 10 And BallCenterY > PCenterY + 6 Then
                        If Angle >= 180 Then Angle -= 180 Else Angle += 180
                        PTargetY = BallCenterY - 6 * Math.Sin(Angle)
                        PTargetX = BallCenterX - 10 * Math.Cos(Angle)
                        Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
                    Else
                        If BallCenterX < PCenterX + 10 Then
                            Pl.Angle = 180
                        Else
                            Pl.Angle = 270
                        End If
                    End If
                ElseIf Angle > 180 Then

                    If BallCenterX > PCenterX + 10 And BallCenterY < PCenterY - 6 Then
                        If Angle >= 180 Then Angle -= 180 Else Angle += 180
                        PTargetY = BallCenterY + 6 * Math.Sin(Angle)
                        PTargetX = BallCenterX - 10 * Math.Cos(Angle)
                        Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
                    Else
                        If BallCenterX < PCenterX + 10 Then
                            Pl.Angle = 180
                        Else
                            Pl.Angle = 90
                        End If
                    End If
                Else
                    If PCenterX + 10 > BallCenterX - 5 Then
                        Pl.Angle = 0
                    Else : Pl.Angle = 180
                    End If
            End If
            If (PCenterX > frmMain.Field.Width / 3 And PCenterX < frmMain.Field.Width * 2 / 3) Or (Ball.X >
frmMain.Field.Width / 3 And Ball.X < frmMain.Field.Width * 2 / 3) Or (PCenterX < frmMain.Field.Width / 3 And (Pl.Angle < 91 Or
Pl.Angle > 269)) Or (PCenterX > frmMain.Field.Width * 2 / 3 And (Pl.Angle > 89 And Pl.Angle < 271)) Then
              Pl.Velocity = 1.4142
            Else
              Pl.Velocity = 0.1
            End If

          Case 2 ' Player 1 (Back)
            If Angle < 180 Then
              Pl.Angle = 180
              If BallCenterX > PCenterX + 10 And BallCenterY > PCenterY + 6 Then
                If Angle >= 180 Then Angle -= 180 Else Angle += 180
                PTargetY = BallCenterY - 6 * Math.Sin(Angle)
                PTargetX = BallCenterX - 10 * Math.Cos(Angle)
                Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
              Else
                If BallCenterX < PCenterX + 10 Then
                  Pl.Angle = 180
                Else
                  Pl.Angle = 270
                End If
              End If
            ElseIf Angle > 180 Then

              If BallCenterX > PCenterX + 10 And BallCenterY < PCenterY - 6 Then
                If Angle >= 180 Then Angle -= 180 Else Angle += 180
                PTargetY = BallCenterY + 6 * Math.Sin(Angle)
                PTargetX = BallCenterX - 10 * Math.Cos(Angle)
                Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
              Else
                If BallCenterX < PCenterX + 10 Then
                  Pl.Angle = 180
                Else
                  Pl.Angle = 90
                End If
              End If
            Else
              If PCenterX + 10 > BallCenterX - 5 Then
                Pl.Angle = 0
              Else : Pl.Angle = 180
              End If
            End If
            If (PCenterX < frmMain.Field.Width / 3) Or (Ball.X < frmMain.Field.Width / 3) Or (PCenterX > frmMain.Field.Width *
1 / 3 And (Pl.Angle > 89 And Pl.Angle < 271)) Then
              Pl.Velocity = 1.4142
            Else
              Pl.Velocity = 0.1
            End If
        End Select
      Else
        Angle = GetAngle(BallCenterX, BallCenterY, 1, frmMain.Field.Height / 2)
        Select Case Pl.AITag
          Case 0 ' Player 2 (Forward)
            If Angle < 180 And BallCenterX < frmMain.Width / 3 Then
              Player(1, 0).Angle = 0
              If BallCenterX < PCenterX - 10 And BallCenterY > PCenterY + 6 Then
                If Angle >= 180 Then Angle -= 180 Else Angle += 180
                PTargetY = BallCenterY - 6 * Math.Sin(Angle)
                PTargetX = BallCenterX + 10 * Math.Cos(Angle)
                Player(1, 0).Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
              Else
                If BallCenterX > PCenterX - 10 Then
                  Player(1, 0).Angle = 0
                Else
                  Player(1, 0).Angle = 270
                End If
              End If
            ElseIf Angle > 180 And BallCenterX < frmMain.Width / 3 Then
            If Angle < 180 Then
               Pl.Angle = 0
               If BallCenterX < PCenterX - 10 And BallCenterY > PCenterY + 6 Then
                  If Angle >= 180 Then Angle -= 180 Else Angle += 180
                  PTargetY = BallCenterY - 6 * Math.Sin(Angle)
                  PTargetX = BallCenterX + 10 * Math.Cos(Angle)
                  Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
               Else
                  If BallCenterX > PCenterX - 10 Then
                     Pl.Angle = 0
                  Else
                     Pl.Angle = 270
                  End If
               End If
            ElseIf Angle > 180 Then

               If BallCenterX < PCenterX - 10 And BallCenterY < PCenterY - 6 Then
                  If Angle >= 180 Then Angle -= 180 Else Angle += 180
                  PTargetY = BallCenterY + 6 * Math.Sin(Angle)
                  PTargetX = BallCenterX + 10 * Math.Cos(Angle)
                  Pl.Angle = GetAngle(PCenterX, PCenterY, PTargetX, PTargetY)
               Else
                  If BallCenterX > PCenterX - 10 Then
                     Pl.Angle = 0
                  Else
                     Pl.Angle = 90
                  End If
               End If
            Else
               If PCenterX - 10 > BallCenterX + 5 Then
                  Pl.Angle = 180
               Else : Pl.Angle = 0
               End If
            End If
            If (PCenterX > frmMain.Field.Width * 2 / 3) Or (Ball.X > frmMain.Field.Width * 2 / 3) Or (PCenterX <
frmMain.Field.Width * 2 / 3 And (Pl.Angle < 91 Or Pl.Angle > 269)) Then
               Pl.Velocity = 1.4142
            Else
               Pl.Velocity = 0.1
            End If
         End Select
      End If
   End Sub

   Private Sub CalcPlayerAngle()
      Dim PCenterX As Single
      Dim PCenterY As Single
      Dim i As Int16
      Dim j As Int16
      For i = 0 To 1
         For j = 0 To 2
            If Player(i, j).AITag = 0 And i = 0 Then
               If frmMain.UseMouse Then
                  PCenterX = Player(i, j).X + 10
                  PCenterY = Player(i, j).Y + 10
                  Dim TriX As Single = (frmMain.MouseX - PCenterX)
                  Dim TriY As Single = (frmMain.MouseY - PCenterY)
                  Dim Angle As Single
                  Angle = Math.Atan(TriY / TriX)
                  'Convert radians to degrees
                  If frmMain.MouseY >= PCenterY Then
                     If frmMain.MouseX >= PCenterX Then
                        'Bottom Right
                        Angle = (Angle / Math.PI) * 180
                     Else
                        'Bottom Left
                        Angle = 180 + ((Angle / Math.PI) * 180)
                     End If
                  Else
                     If frmMain.MouseX >= PCenterX Then
```

```
                'Top Right
                Angle = (Angle / Math.PI) * 180 + 360
            Else
                'Top Left
                Angle = 180 + ((Angle / Math.PI) * 180)
            End If
        End If
        Player(i, j).Angle = GetAngle(PCenterX, PCenterY, frmMain.MouseX, frmMain.MouseY)
        Player(i, j).Velocity = 1.4142
    Else
        If KeyInput(i, j).Up And KeyInput(i, j).Left Then
            Player(i, j).Angle = 225
        ElseIf KeyInput(i, j).Up And KeyInput(i, j).Right Then
            Player(i, j).Angle = 315
        ElseIf KeyInput(i, j).Right And KeyInput(i, j).Down Then
            Player(i, j).Angle = 45
        ElseIf KeyInput(i, j).Down And KeyInput(i, j).Left Then
            Player(i, j).Angle = 135
        ElseIf KeyInput(i, j).Up Then
            Player(i, j).Angle = 270
        ElseIf KeyInput(i, j).Right Then
            Player(i, j).Angle = 0
        ElseIf KeyInput(i, j).Down Then
            Player(i, j).Angle = 90
        ElseIf KeyInput(i, j).Left Then
            Player(i, j).Angle = 180
        End If
        Player(i, j).Velocity = 1.4142
        If Not (KeyInput(i, j).Up Or KeyInput(i, j).Left Or KeyInput(i, j).Right Or KeyInput(i, j).Down) Then
            Player(i, j).Velocity = 0
        End If
    End If
ElseIf Player(i, j).AITag = 0 And i = 1 And NetworkPlay And NetworkIsHost Then
    If frmMain.P2UseMouse Then
        PCenterX = Player(i, j).X + 10
        PCenterY = Player(i, j).Y + 10
        Dim TriX As Single = (frmMain.P2MouseX - PCenterX)
        Dim TriY As Single = (frmMain.P2MouseY - PCenterY)
        Dim Angle As Single
        Angle = Math.Atan(TriY / TriX)
        'Convert radians to degrees
        If frmMain.P2MouseY >= PCenterY Then
            If frmMain.P2MouseX >= PCenterX Then
                'Bottom Right
                Angle = (Angle / Math.PI) * 180
            Else
                'Bottom Left
                Angle = 180 + ((Angle / Math.PI) * 180)
            End If
        Else
            If frmMain.P2MouseX >= PCenterX Then
                'Top Right
                Angle = (Angle / Math.PI) * 180 + 360
            Else
                'Top Left
                Angle = 180 + ((Angle / Math.PI) * 180)
            End If
        End If
        Player(i, j).Angle = GetAngle(PCenterX, PCenterY, frmMain.P2MouseX, frmMain.P2MouseY)
        Player(i, j).Velocity = 1.4142
    Else
        If KeyInput(i, j).Up And KeyInput(i, j).Left Then
            Player(i, j).Angle = 225
        ElseIf KeyInput(i, j).Up And KeyInput(i, j).Right Then
            Player(i, j).Angle = 315
        ElseIf KeyInput(i, j).Right And KeyInput(i, j).Down Then
            Player(i, j).Angle = 45
        ElseIf KeyInput(i, j).Down And KeyInput(i, j).Left Then
            Player(i, j).Angle = 135
        ElseIf KeyInput(i, j).Up Then
```

```
End Sub

Private Function CheckBorders() As Boolean
   'Checks ball position at borders
   Dim x As Boolean = True
   If Ball.X < 0 Then
      If Ball.Y > (frmMain.Field.Height / 2) - 32 And Ball.Y + 10 < (frmMain.Field.Height / 2) + 32 Then
         PlaySound(4)
         frmMain.lblP2Score.Text += 1
         x = False
      Else
         PlaySound(7)
         x = False
      End If
   ElseIf Ball.X > frmMain.Field.Width Then
      If Ball.Y > (frmMain.Field.Height / 2) - 32 And Ball.Y + 10 < (frmMain.Field.Height / 2) + 32 Then
         PlaySound(5)
         frmMain.lblP1Score.Text += 1
         x = False
      Else
         PlaySound(7)
         x = False
      End If
   ElseIf Ball.Y < 0 Then
      PlaySound(7)
      x = False
   ElseIf Ball.Y > frmMain.Field.Height Then
      PlaySound(7)
      x = False
   End If
   If (Ball.X < (frmMain.Field.Width / 3)) Or (Ball.X > (frmMain.Field.Width * 2 / 3)) Then
      PlaySound(3)
   End If
   Return x
End Function

Private Sub DoCollide()
   'Check for collision between all players and the ball,
   'returning the angle for the ball if collision occurs.
   Dim Angle As Int16
   Dim i As Int16
   Dim j As Int16
   For i = 0 To 1
      For j = 0 To 2
         Angle = CheckCollision(Player(i, j), Ball, False)
         If Angle <> -1 Then
            Ball.Angle = Angle
            Ball.Velocity = 3
            PlaySound(6)
         End If
      Next
      Angle = CheckCollision(Goalie(i), Ball, False)
      If Angle <> -1 Then
         Ball.Angle = Angle
         Ball.Velocity = 3
         PlaySound(6)
         PlaySound(1)
      End If
   Next
End Sub

Private Sub SendEngineVars()
   Try
      Dim SendString As String = "Engine;"
      Dim i As Short = 0
      Dim j As Short = 0
      For i = 0 To 1
         For j = 0 To 2
            SendString &= Player(i, j).X & ";"
            SendString &= Player(i, j).Y & ";"
```

APPENDIX 9: NetworkClient.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class NetworkClient
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.cmdConnect = New System.Windows.Forms.Button
        Me.txtIP = New System.Windows.Forms.TextBox
        Me.lblStatus = New System.Windows.Forms.Label
        Me.cmdCancel = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'cmdConnect
        '
        Me.cmdConnect.Location = New System.Drawing.Point(15, 51)
        Me.cmdConnect.Name = "cmdConnect"
        Me.cmdConnect.Size = New System.Drawing.Size(75, 23)
        Me.cmdConnect.TabIndex = 0
        Me.cmdConnect.Text = "Connect"
        Me.cmdConnect.UseVisualStyleBackColor = True
        '
        'txtIP
        '
        Me.txtIP.Location = New System.Drawing.Point(15, 25)
        Me.txtIP.Name = "txtIP"
        Me.txtIP.Size = New System.Drawing.Size(156, 20)
        Me.txtIP.TabIndex = 1
        Me.txtIP.Text = "127.0.0.1"
        Me.txtIP.TextAlign = System.Windows.Forms.HorizontalAlignment.Center
        '
        'lblStatus
        '
        Me.lblStatus.Location = New System.Drawing.Point(12, 9)
        Me.lblStatus.Name = "lblStatus"
        Me.lblStatus.Size = New System.Drawing.Size(159, 13)
        Me.lblStatus.TabIndex = 2
        Me.lblStatus.Text = "Connection Idle"
        Me.lblStatus.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
        '
        'cmdCancel
        '
        Me.cmdCancel.Location = New System.Drawing.Point(96, 51)
        Me.cmdCancel.Name = "cmdCancel"
        Me.cmdCancel.Size = New System.Drawing.Size(75, 23)
        Me.cmdCancel.TabIndex = 3
```

APPENDIX 10: NetworkHost.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class NetworkHost
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.cmdCancel = New System.Windows.Forms.Button
        Me.lblWait = New System.Windows.Forms.Label
        Me.SuspendLayout()
        '
        'cmdCancel
        '
        Me.cmdCancel.Location = New System.Drawing.Point(93, 32)
        Me.cmdCancel.Name = "cmdCancel"
        Me.cmdCancel.Size = New System.Drawing.Size(75, 23)
        Me.cmdCancel.TabIndex = 0
        Me.cmdCancel.Text = "Cancel"
        Me.cmdCancel.UseVisualStyleBackColor = True
        '
        'lblWait
        '
        Me.lblWait.Location = New System.Drawing.Point(12, 9)
        Me.lblWait.Name = "lblWait"
        Me.lblWait.Size = New System.Drawing.Size(156, 16)
        Me.lblWait.TabIndex = 1
        Me.lblWait.Text = "Waiting for Connection..."
        Me.lblWait.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
        '
        'NetworkHost
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(180, 67)
        Me.Controls.Add(Me.lblWait)
        Me.Controls.Add(Me.cmdCancel)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "NetworkHost"
        Me.Text = "NetworkHost"
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents cmdCancel As System.Windows.Forms.Button
    Friend WithEvents lblWait As System.Windows.Forms.Label
End Class
```