The Mutton Project

Sheepshead Card Game Final Report

Group Members:
Adam Gritt – Project Leader and GUI
Mike Krautkramer – AI
Greg Schreiner – Communications
Nhat Nguyen – Help and MS Agent

Table of Contents

1 Initial Requirements

1.1 Overview

The purpose of this project is to create a computerized version of the sheepshead card game. The game will support computerized and human players.

1.2 Customer

Any person(s) who wish to learn or play a computerized version of the Sheepshead card game.

1.3 Goals

The general goal of this application is to teach and entertain people who wish to play sheepshead on the computer. Some of the specific goals include:

- Supply and AI that provides multiple levels of play.
- Provides a large rule base for user customization.
- Supports multiple communication types.
- Offers a robust and easy to use graphical user interface.
- Supply easy to follow and understand online help.
- Provides an interactive teaching tutorial.

1.4 System Requirements

1.4.1 Basic Functions

Ref. #	Function	Category
R1.1	Deals from a computerized deck of cards	Hidden
R1.2	Play using the same style as the human players (learning AI)	Hidden
R1.3	Connect to other computers for network play	Hidden
R1.4	Play based on user-selected/customized rules	Hidden
R1.5	Player communication	Evident

1.5 System Attributes

Attribute	Details and Boundary Constraints
Operating Systems Platforms	(detail) Microsoft Windows 9x, NT4.0, 2000 (detail) Possible Port to Linux
Interface Metaphor	(detail) Forms-Metaphor windows and dialog boxes (detail) Maximize for easy Keyboard navigation (detail) Maximize for easy Pointer navigation
Computer System Requirements	(boundary constraint) Runs on a Pentium 166Mhz or higher (boundary constraint) Runs on 800x600 resolution or higher (boundary constraint) Runs at 256 colors or higher

2 Front End

2.1 Requirements

2.1.1 Card Control

2.1.1.1 Overview

A third party control is needed which will cover the object Deck, Hand, and Card.

2.1.1.2 Customer

The Mutton Project – Senior Design Group

2.1.1.3 Goals

The goal is to find a third party card control that will contain Deck, Hand, and Card objects. The objects should have the properties and functions as stated below.

2.1.1.4 System Requirements

2.1.1.4.1 Basic Functions

Ref. #	Function	Category
R1.1	The deck object should have a Shuffle Command	Hidden
R1.2	The deck object should have an array of Card objects	Hidden
R1.3	The deck object should have an array of Hand objects	Hidden
R1.4	The deck object should have a Deal function	Hidden
R1.5	The deck object should have a setting for deck type (number and type of cards available)	Hidden
R1.6	The deck object should have a setting for type of deal (one or more cards at a time)	Hidden
R1.7	The deck object should have a Cut function	Hidden
R1.8	The deck object should have a Top Card attribute	Hidden
R1.9	The hand object should hold an array of Card objects.	Hidden
R1.10	The card object should have a value attribute (Ace, King, etc)	Hidden
R1.11	The deck object should have a Card Back attribute	Hidden
R1.12	The card object should have a type attribute (Clubs, Spades, etc)	Hidden
R1.13	The control must be able to display the cards on the screen and allow the user to move them around.	Evident

2.1.2 Graphical User Interface

2.1.2.1 Overview

These requirements are the basic requirements that the GUI for the Mutton Project should perform.

2.1.2.2 Customer

Any person(s) who wish to learn or play a computerized version of the Sheepshead card game.

2.1.2.3 Goals

The general goal of the GUI is to provide a user interface that is easy to use and allows the user to customize to their liking. The GUI should also allow the user to access all main functionality of the Mutton Project.

2.1.2.4 System Requirements

2.1.2.4.1 Basic Functions

Ref. #	Function	Category
R1.1	A Menu bar should exist.	Evident
R1.2	Online help should be accessible	Evident
R1.3	A tutorial using MS Agent should be made available to the user	Evident
R1.4	Each Player's cards should be shown, with only the current machine player's card visible.	Evident
R1.5	A Chat window should be able to be attached or hanging	Evident
R1.6	A Scorecard should be shown on the display	Evident
R1.7	Allow ability to connect to other computers	Evident

2.1.2.5 System Attributes

Attribute	Details and Boundary Constraints
Screen Size	Must be able to support 800x600x256, be able to adjust for larger screen sizes.

2.2 Graphical User Interface

2.2.1 High Level Design

2.2.1.1 Introduction

The purpose of this document is to describe the design of the graphical user interface.

2.2.1.2 Applicable Documents

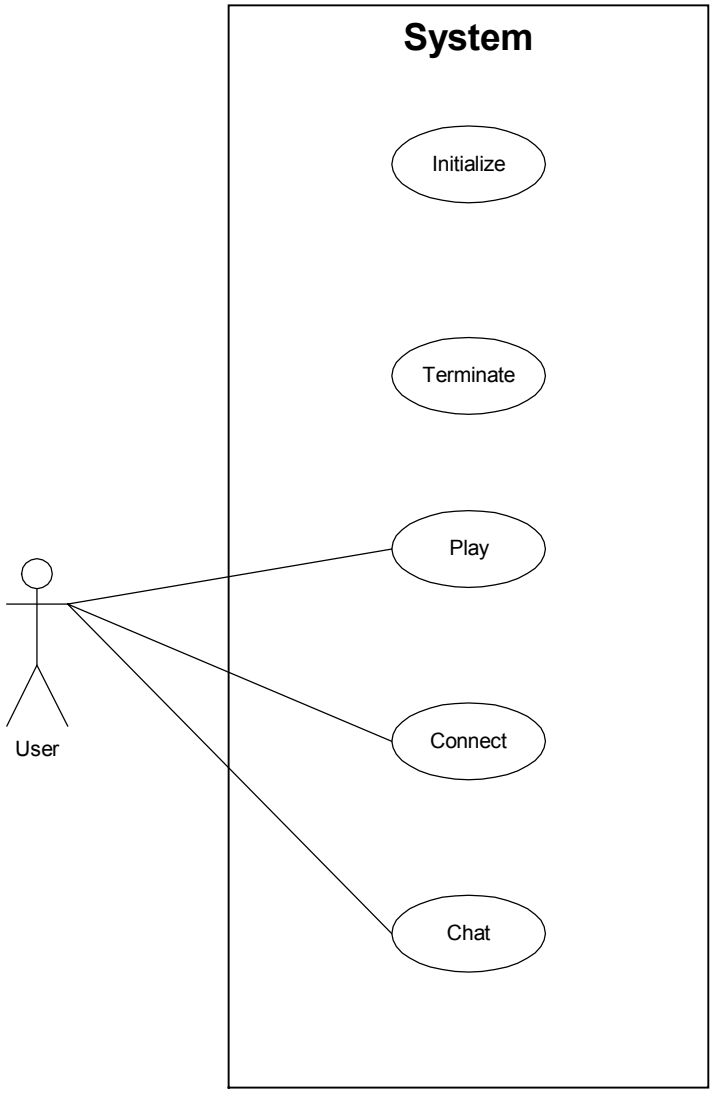
GUI Requirments:
SRD - Graphical User Interface

Technology Report

2.2.1.3 Glossary of Terms

User – Any user who wishes to play the game.
Application – The Sheepshead application.

2.2.1.4 High Level Use Case Diagram



2.2.1.5 High Level Use Cases

2.2.1.5.1 Initialize

2.2.1.5.1.1 Purpose/Overview

Initialize prepares the application window for display and sets all initial settings based on defaults or last used settings.

2.2.1.5.1.2 Actors

Application

2.2.1.5.1.3 Type

Primary and essential

2.2.1.5.1.4 Cross Reference

R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7

2.2.1.5.2 Terminate

2.2.1.5.2.1 Purpose/Overview
Terminate the application and save settings. Perform any object cleanup that is necessary.

2.2.1.5.2.2 Actors
Application

2.2.1.5.2.3 Type
Primary and essential

2.2.1.5.2.4 Cross Reference
R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7

2.2.1.5.3 Connect

2.2.1.5.3.1 Purpose/Overview
Allow user to connect to other Sheepshead games currently being played.

2.2.1.5.3.2 Actors
User

2.2.1.5.3.3 Type
Primary

2.2.1.5.3.4 Cross Reference
R1.7

2.2.1.5.4 Play

2.2.1.5.4.1 Purpose/Overview
The act of playing the sheepshead card game

2.2.1.5.4.2 Actors
User, Application

2.2.1.5.4.3 Type
Primary and essential

2.2.1.5.4.4 Cross Reference
R1.1, R1.2, R1.3, R1.4, R1.6

2.2.1.5.5 Chat

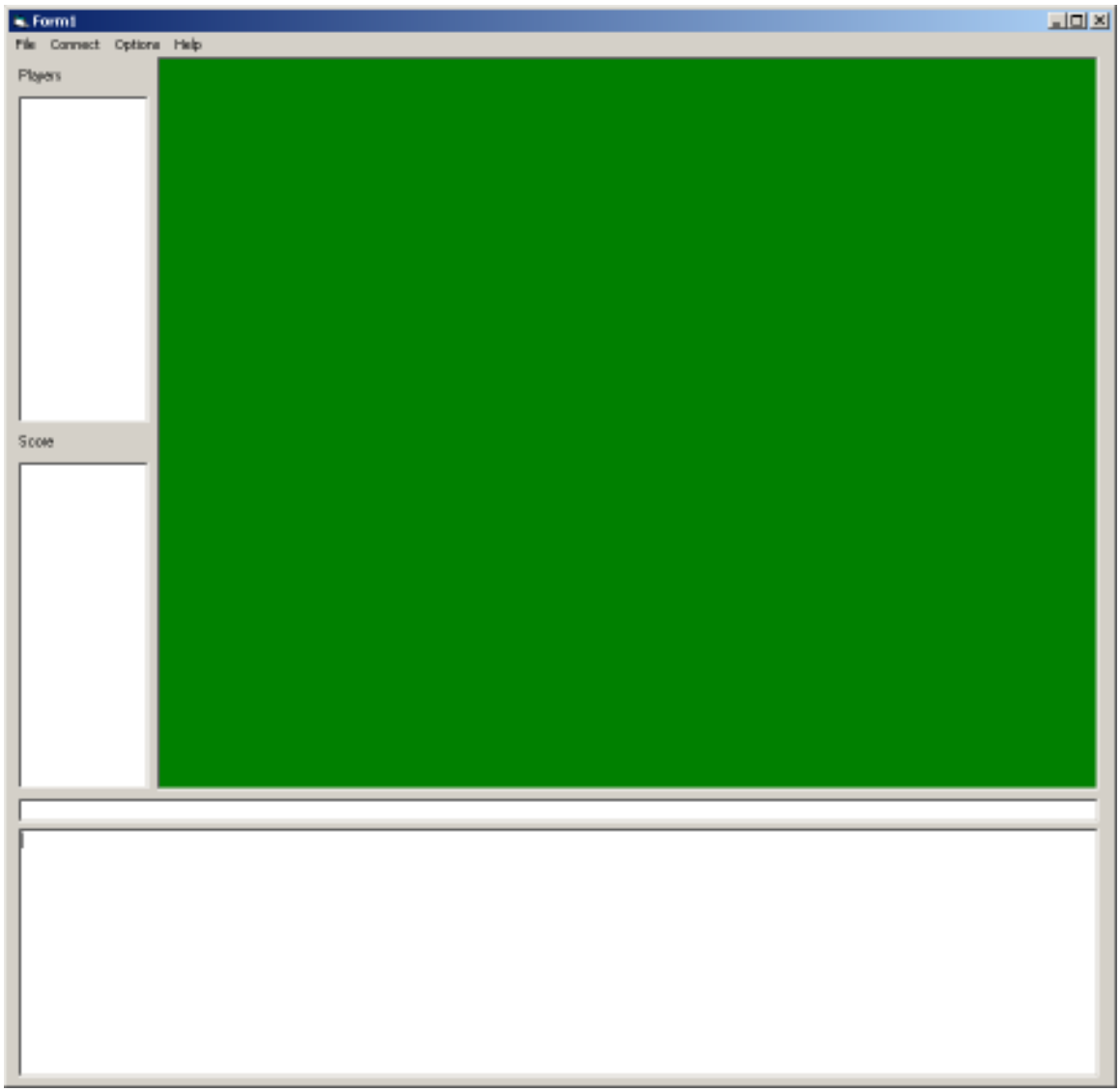
2.2.1.5.5.1 Purpose/Overview
Allow users to communicate with each other.

2.2.1.5.5.2 Actors
User, Application

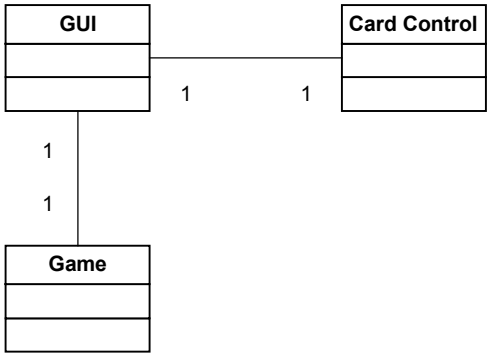
2.2.1.5.5.3 Type
Primary

2.2.1.5.5.4 Cross Reference
R1.5

2.2.1.6 Screen Prototypes



2.2.1.7 Conceptual Object Model



2.2.2 Expanded Use Cases

2.2.2.1 Introduction

The purpose of this document is to detail the design of the Graphical User Interface for the Mutton Project Sheepshead program.

2.2.2.2 Applicable Documents

GUI Requirements
SRD - GUI Requirements

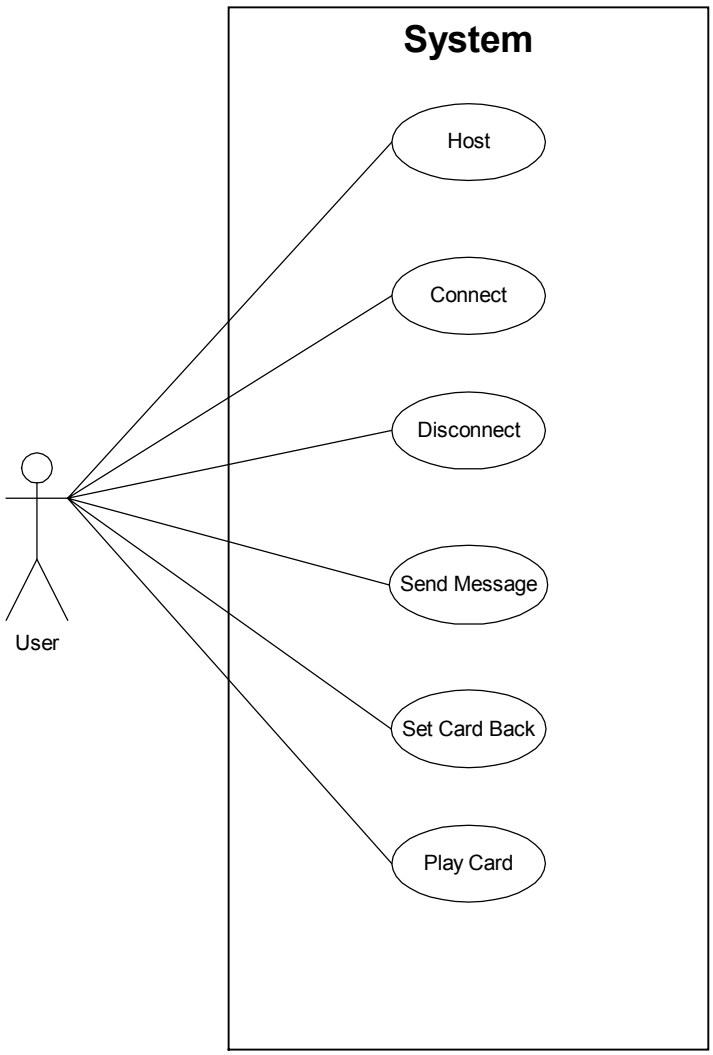
Card Requirements
SRD – Card Control

GUI HLD Document
HLD – Graphical User Interface

2.2.2.3 Glossary of Terms

User – Any person who plays the Sheepshead computer game.

2.2.2.4 Expanded Use Case Diagram



2.2.2.5 Expanded Use Cases

2.2.2.5.1 Host

2.2.2.5.1.1 Purpose/Overview

Allows the user to set up the computer as a Host computer for the game. Also is required for single player vs. AI mode.

2.2.2.5.1.2 Actors

User

2.2.2.5.1.3 Type

Primary and Essential

2.2.2.5.1.4 Cross Reference

R1.7

2.2.2.5.1.5 Typical Course of Events

Actor Action	System Response
Line 1: User chooses to host game.	Line 2: GUI informs system it is the Host.
	Line 3: GUI waits for system response of other users.
Line 4: User starts game once other players have joined or for stand-alone mode.	Line 5: GUI informs system to start game.

2.2.2.5.1.6 Alternative Courses

None.

2.2.2.5.2 Connect

2.2.2.5.2.1 Purpose/Overview

Allows the user to set up the computer as a Client computer for the game and connect to a remote Host.

2.2.2.5.2.2 Actors

User

2.2.2.5.2.3 Type

Secondary and Essential

2.2.2.5.2.4 Cross Reference

R1.7

2.2.2.5.2.5 Typical Course of Events

Actor Action	System Response
Line 1: User decides to connect to another game and specifies the type of connection.	Line 2: GUI informs system of type of connection to open.
	Line 3: GUI waits for system to respond.

2.2.2.5.2.6 Alternative Courses

Line 3: If connection timed out user is prompted on action to take.

2.2.2.5.3 Disconnect

2.2.2.5.3.1 Purpose/Overview

Allows the user to disconnect from a remotely hosted game and play in stand-alone mode.

2.2.2.5.3.2 Actors

User

2.2.2.5.3.3 Type
Primary and Essential

2.2.2.5.3.4 Cross Reference
R1.7

2.2.2.5.3.5 Typical Course of Events

Actor Action	System Response
Line 1: User chooses to disconnect	Line 2: GUI tells system to close connection.
	Line 3: GUI runs in Stand-Alone mode.

2.2.2.5.3.6 Alternative Courses
None.

2.2.2.5.4 Send Message

2.2.2.5.4.1 Purpose/Overview
Allows the users to communicate to each other while playing the game remotely.

2.2.2.5.4.2 Actors
User

2.2.2.5.4.3 Type
Secondary

2.2.2.5.4.4 Cross Reference
R1.5

2.2.2.5.4.5 Typical Course of Events

Actor Action	System Response
Line 1: User enter message to send.	Line 2: GUI gives message to system to pass on.
	Line 3: GUI receives message and displays it in the chat window.

2.2.2.5.4.6 Alternative Courses
None

2.2.2.5.5 Set Card Back

2.2.2.5.5.1 Purpose/Overview
Allows the user to customize the card backs.

2.2.2.5.5.2 Actors
User

2.2.2.5.5.3 Type
Secondary

2.2.2.5.5.4 Cross Reference
Cards – R1.11

2.2.2.5.5.5 Typical Course of Events

Actor Action	System Response
Line 1: User chooses to change card back.	Line 2: GUI displays selection of card backs.
Line 3: User chooses card back	Line 4: GUI tells system card back to use on local machine.

2.2.2.5.5.6 Alternative Courses
None

2.2.2.5.6 Play Card

2.2.2.5.6.1 Purpose/Overview
Player plays the card they wish.

2.2.2.5.6.2 Actors
User

2.2.2.5.6.3 Type
Primary and Essential

2.2.2.5.6.4 Cross Reference
R1.4, Cards – R1.13

2.2.2.5.6.5 Typical Course of Events

Actor Action	System Response
Line 1: User selects card to play.	Line 2: Card face colors are inverted and card is slid up.
Line 3: User Dbl Clicks on card or drags card to center of play.	Line 3: GUI informs system of what card was played and moves card on screen.

2.2.2.5.6.6 Alternative Courses

None.

2.2.2.6 Class Diagram

Not Applicable.

2.3 Help

2.3.1 High Level Design

2.3.1.1 Introduction

The purpose of this document is to describe the design of the help system that will implement Microsoft Agent and HTML Help.

2.3.1.2 Applicable Documents

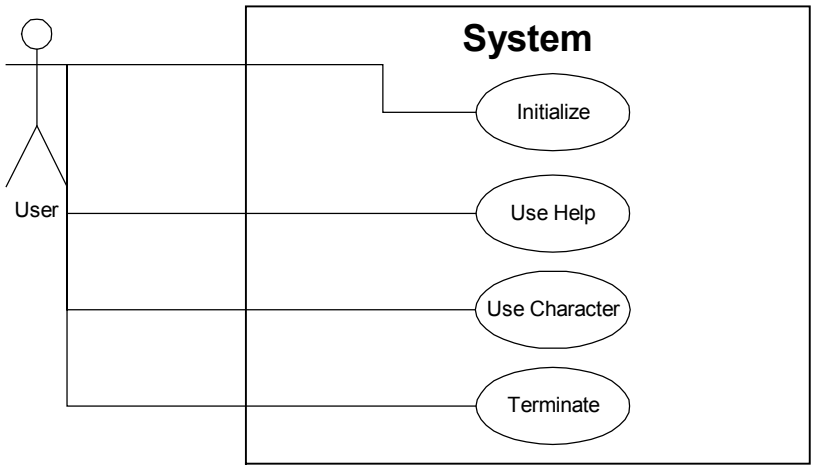
Technology Report

2.3.1.3 Glossary of Terms

User – The client of the help system consisting of the domain controller and the AI system.

System – The Sheepshead application.

2.3.1.4 High Level Use Case Diagram



2.3.1.5 High Level Use Cases

2.3.1.6 Intialize

2.3.1.6.1.1 Purpose/Overview

Instantiates the main Mutton Help object and its underlying Mutton Character and Mutton HTML Help objects.

2.3.1.6.1.2 Actors
User

2.3.1.6.1.3 Type
Primary

2.3.1.6.1.4 Cross Reference

2.3.1.6.2 Use Help

2.3.1.6.2.1 Purpose/Overview
Allows access to display and search the Mutton Help System (shown via the HTML Help Viewer).

2.3.1.6.2.2 Actors
User

2.3.1.6.2.3 Type
Primary

2.3.1.6.2.4 Cross Reference

2.3.1.6.3 Use Character

2.3.1.6.3.1 Purpose/Overview
Allows access to the various functionality offered by the Mutton Character object including hide, load character, move, play, show, speak, stop, and think.

2.3.1.6.3.2 Actors
User

2.3.1.6.3.3 Type
Primary

2.3.1.6.3.4 Cross Reference

2.3.1.6.4 Terminate

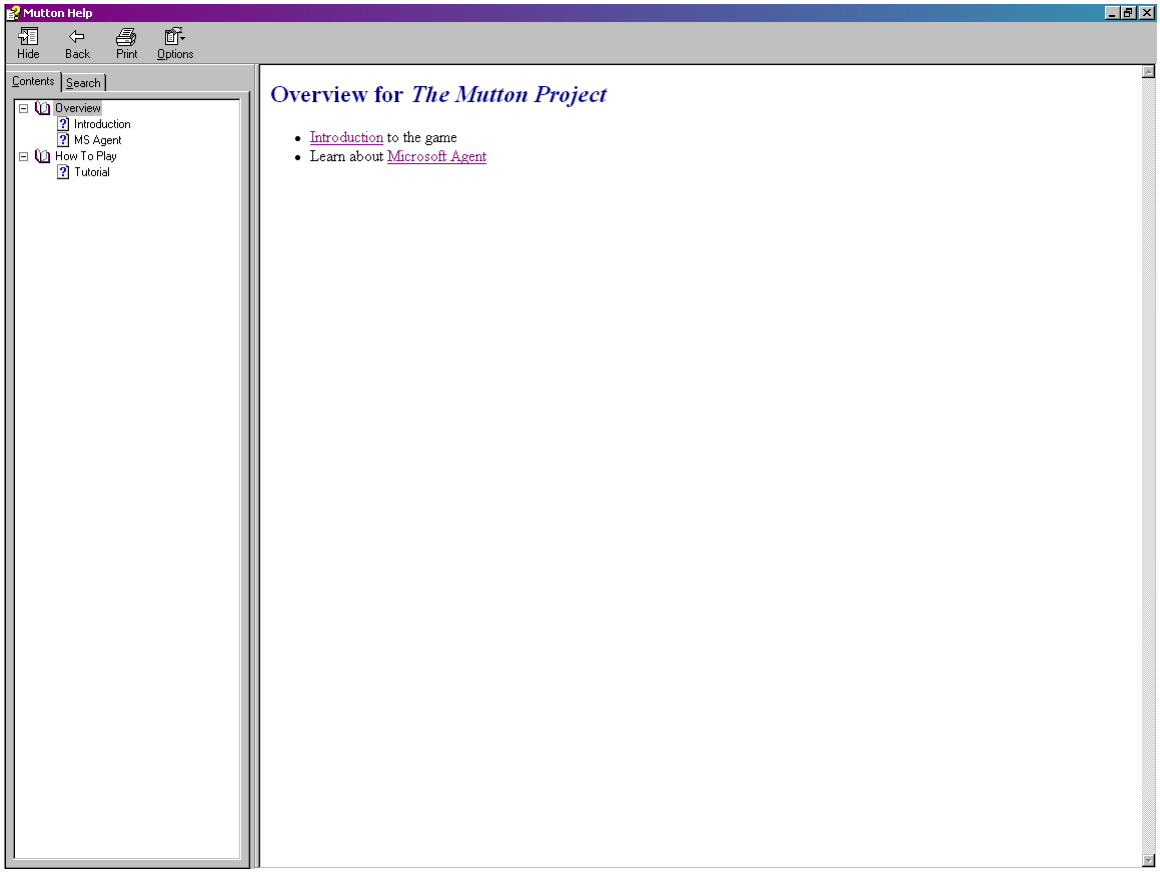
2.3.1.6.4.1 Purpose/Overview
Cleans up any necessary objects.

2.3.1.6.4.2 Actors
User

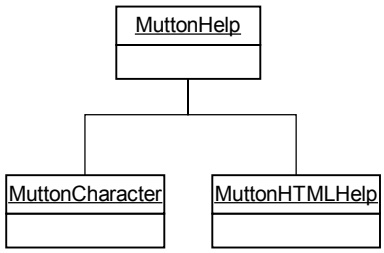
2.3.1.6.4.3 Type
Primary

2.3.1.6.4.4 Cross Reference

2.3.1.7 Screen Prototypes



2.3.1.8 Conceptual Object Model



2.3.2 Expanded Use Cases

2.3.2.1 Introduction

The purpose of this document is to describe in detail the design of the help system for *The Mutton Project*. The Mutton Help System uses HTML Help for the actual help files and Microsoft Agent for the visual interface (via an Agent character) to search the help

files. Microsoft Agent will also be the primary visual mechanism for interfacing with the user of the application.

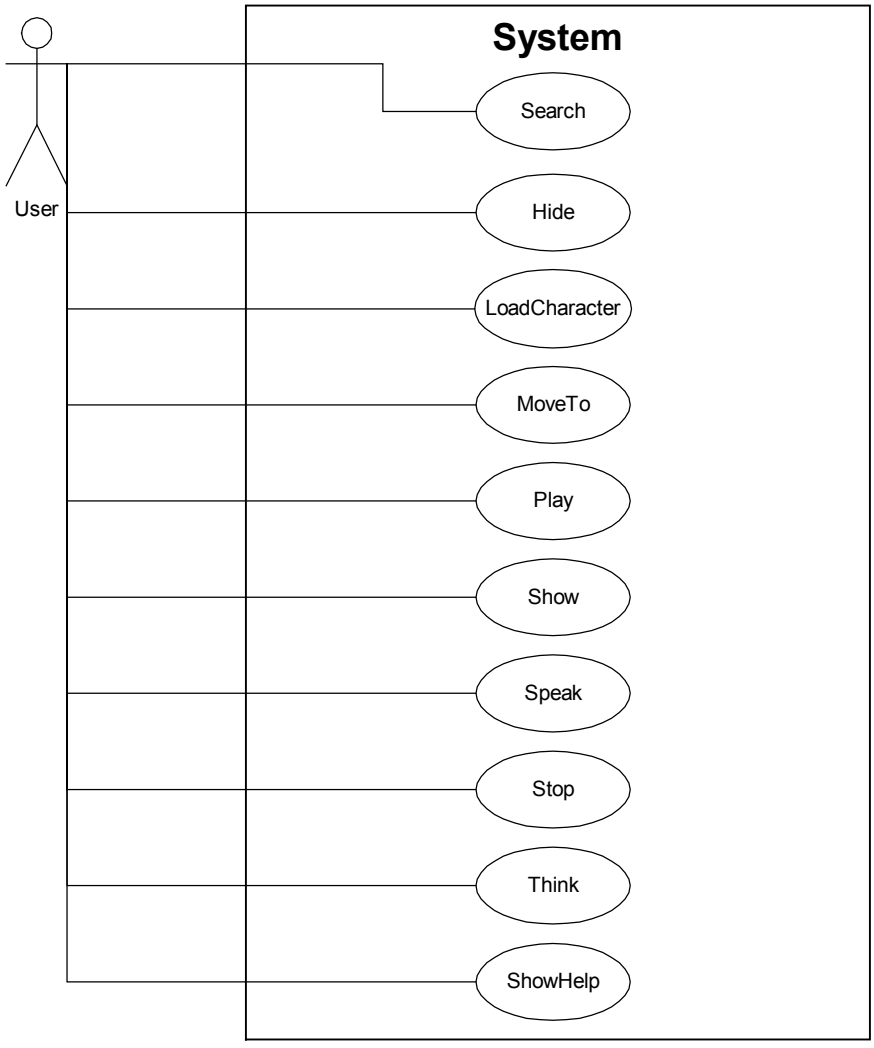
2.3.2.2 Applicable Documents

High Level Design for Help System:
HLD - Help System

2.3.2.3 Glossary of Terms

User – The client of the help system, primarily the game domain controller.
System – The Sheepshead application.

2.3.2.4 Expanded Use Case Diagram



2.3.2.5 Expanded Use Cases

2.3.2.5.1 Search

2.3.2.5.1.1 Purpose/Overview

Allows user of the application to query the HTML Help files via the MS Agent character

2.3.2.5.1.2 Actors
User

2.3.2.5.1.3 Type
Primary

2.3.2.5.1.4 Cross Reference

2.3.2.5.2 Hide

2.3.2.5.2.1 Purpose/Overview
Hides the current Agent character from view.

2.3.2.5.2.2 Actors
User

2.3.2.5.2.3 Type
Primary

2.3.2.5.2.4 Cross Reference

2.3.2.5.3 LoadCharacter

2.3.2.5.3.1 Purpose/Overview
Loads a new Agent character.

2.3.2.5.3.2 Actors
User

2.3.2.5.3.3 Type
Primary

2.3.2.5.3.4 Cross Reference

2.3.2.5.4 MoveTo

2.3.2.5.4.1 Purpose/Overview
Moves the Agent character to a specified Cartesian coordinate (origin is located at the top left corner of screen).

2.3.2.5.4.2 Actors
User

2.3.2.5.4.3 Type
Primary

2.3.2.5.4.4 Cross Reference

2.3.2.5.5 Play

2.3.2.5.5.1 Purpose/Overview
Plays a specified Agent animation.

2.3.2.5.5.2 Actors
User

2.3.2.5.5.3 Type
Primary

2.3.2.5.5.4 Cross Reference

2.3.2.5.6 Show

2.3.2.5.6.1 Purpose/Overview
Displays the Agent character on the screen.

2.3.2.5.6.2 Actors
User

2.3.2.5.6.3 Type
Primary

2.3.2.5.6.4 Cross Reference

2.3.2.5.7 Speak

2.3.2.5.7.1 Purpose/Overview
Makes the Agent character speak the specified text.

2.3.2.5.7.2 Actors
User

2.3.2.5.7.3 Type
Primary

2.3.2.5.7.4 Cross Reference

2.3.2.5.8 Stop

2.3.2.5.8.1 Purpose/Overview
Stops any Agent animation in progress.

2.3.2.5.8.2 Actors

User

2.3.2.5.8.3 Type
Primary

2.3.2.5.8.4 Cross Reference

2.3.2.5.9 Think

2.3.2.5.9.1 Purpose/Overview
Displays an Agent "think" balloon containing the specified text.

2.3.2.5.9.2 Actors
User

2.3.2.5.9.3 Type
Primary

2.3.2.5.9.4 Cross Reference

2.3.2.5.10 ShowHelp

2.3.2.5.10.1 Purpose/Overview
Displays the Mutton Help System using the HTML Help Viewer.

2.3.2.5.10.2 Actors
User

2.3.2.5.10.3 Type
Primary

2.3.2.5.10.4 Cross Reference

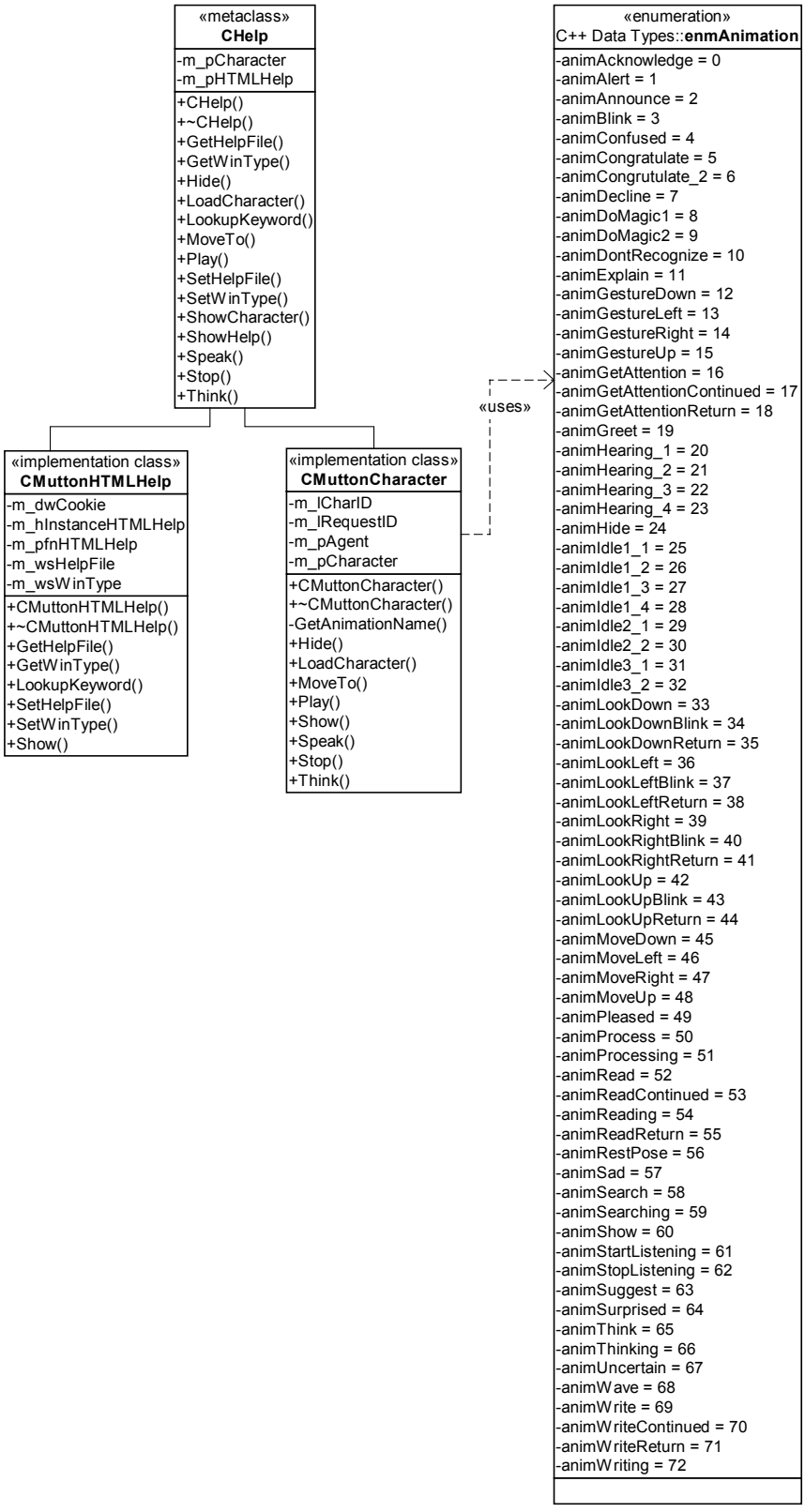
2.3.2.5.10.5 Typical Course of Events

Actor Action	System Response
Search	Agent character provides a prompt requesting search word(s) to feed to the MuttonHTMLHelp object.
Hide	Agent character will be hidden from view.
LoadCharacter	Current Agent character is unloaded and a new one is loaded.
MoveTo	Agent character is moved to the specified location utilizing movement animations.
Play	Specified animation is played for the

	current Agent character.
Show	Agent character becomes visible.
Speak	Agent character speaks the specified text.
Stop	Clear the animation queue—stop any processing animation.
Think	Display “think” balloon with the specified text.
ShowHelp	Mutton Help System is shown via the HTML Help Viewer.

2.3.2.5.10.6 Alternative Courses

2.3.2.6 Class Diagram



3 Back End

3.1 Requirements

3.1.1 Engine

3.1.1.1 Overview

These requirements are the basic requirements that the Engine for the Mutton Project should perform.

3.1.1.2 Customer

Any person(s) who wish to learn or play a computerized version of the Sheepshead card game.

3.1.1.3 Goals

The general goal of the Engine is to power the AI and Communications of the Mutton Project.

3.1.1.4 System Requirements

3.1.1.4.1 Basic Functions

Ref. #	Function	Category
R1.1	Provide a learning AI	Evident
R1.2	Provide an AI that works on different levels of difficulty	Hidden
R1.3	Provide an AI that can play based on user defined rules	Evident
R1.4	Support TCP/IP protocol	Evident
R1.5	Support IPX/SPX protocol	Evident
R1.6	Support Modem connection	Evident
R1.7	Support Direct Connect connection	Evident
R1.8	Support “Hot Seat” connections	Evident
R1.9	Allow the AI to interface with the tutorial	Hidden

3.1.1.4.2 Extended Functions

Ref. #	Function	Category
R2.1	Provide an AI that can learn from previous games	Hidden
R2.2	Provide an AI that will keep track of other player’s moves, and use those as the basis for it’s own moves.	Hidden
R2.3	Provide logic to explain why moves were made for the tutorial.	Evident
R2.4	Provide a chat window so the players can communicate.	Evident
R2.5	Provide the Host with administrative rights (manage players, boot players, etc.)	Evident
R2.6	Provide the ability to build games and play on a dedicated server.	Evident
R2.7	Provide the ability to play with any mix of human and computer players, and different types of connections.	Evident

R2.8	Provide the ability to generate different house rules.	Evident
R2.9	Provide a computer player to fill in for any players that loose their connection to the game.	Evident Hidden

3.1.1.5 System Attributes

Attribute	Details and Boundary Constraints
Operating System Platforms	Windows 95, 98 and NT
Communication Protocols Supported	TCP/IP, IPX/SPX, Direct connection (null modem), and Modem
Response	AI should respond within ten seconds.

3.1.2 Player

3.1.2.1 Overview

This outlines the basic requirements for the player in a game

3.1.2.2 Customer

Any person(s) who wish to learn or play a computerized version of the Sheepshead card game.

3.1.2.3 Goals

The general goal of the player is to provide a placeholder for the Sheephead player within the game and allow the customer the ability to play the game.

3.1.2.4 System Requirements

3.1.2.4.1 Basic Functions

Ref. #	Function	Category
R1.1	Allow the Player to have a hand to play cards from	Evident
R1.2	Allow player to play a card while enforcing rules on the card played	Evident
R1.3	Allow for Human, AI, and a remote player	Evident
R1.4	Keep Track of cards taken	Evident
R1.5	Keep Score	Evident
R1.6	Give Player a name	Evident
R1.7	Allow for the addition of new remote players between games	Evident
R1.8	Allow for an AI player to fill in for a dropped remote player	Evident
R1.9	Allow player to pick and use blind cards	Evident
R1.10	Allow chat messages between all players	Evident

3.2 Communications

3.2.1 High Level Design

3.2.1.1 Introduction

The purpose of this document is to discuss the operation of the communications

3.2.1.2 Applicable Documents

Requirements Document:
SRD - Engine Requirements

3.2.1.3 Glossary of Terms

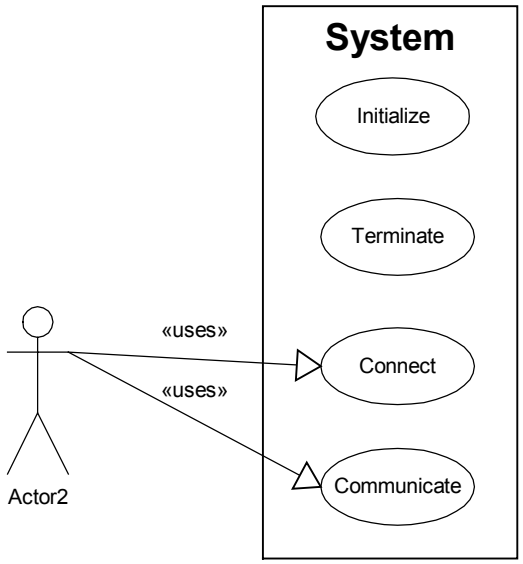
Host: A player that allows other players to connect to it for the game.

Player: One plays the game. In multiplayer play, referred to one who connects to the host's computer in order to play.

Session: Allows other players to connect to the computer—i.e. act as a host.

Message: Any data being sent between computers. Could be initialization data, internal game messages, or chat messages.

3.2.1.4 High Level Use Case Diagram



3.2.1.5 High Level Use Cases

3.2.1.5.1 Initialize

3.2.1.5.1.1 Purpose/Overview

Initialize communication to allow for connections to other computers

3.2.1.5.1.2 Actors
Application

3.2.1.5.1.3 Type
Primary and essential

3.2.1.5.1.4 Cross Reference
R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.1.5.2 Terminate

3.2.1.5.2.1 Purpose/Overview
Terminate any connections and perform cleanup

3.2.1.5.2.2 Actors
Application

3.2.1.5.2.3 Type
Primary and essential

3.2.1.5.2.4 Cross Reference
R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.1.5.3 Connect

3.2.1.5.3.1 Purpose/Overview
Set up a connection to another computer for multiplayer play

3.2.1.5.3.2 Actors
Player

3.2.1.5.3.3 Type
Primary and essential

3.2.1.5.3.4 Cross Reference
R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.1.5.4 Communicate

3.2.1.5.4.1 Purpose/Overview
Send messages for the game between computers

3.2.1.5.4.2 Actors
Application, Player

3.2.1.5.4.3 Type

Primary and essential

3.2.1.5.4.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7

3.2.1.6 Conceptual Object Model

3.2.2 Expanded Use Cases

3.2.2.1 Introduction

The purpose of this document is to discuss the operation of the communications

3.2.2.2 Applicable Documents

Requirements Document:
SRD - Engine Requirements

High Level Design document
HLD - Communications

3.2.2.3 Glossary of Terms

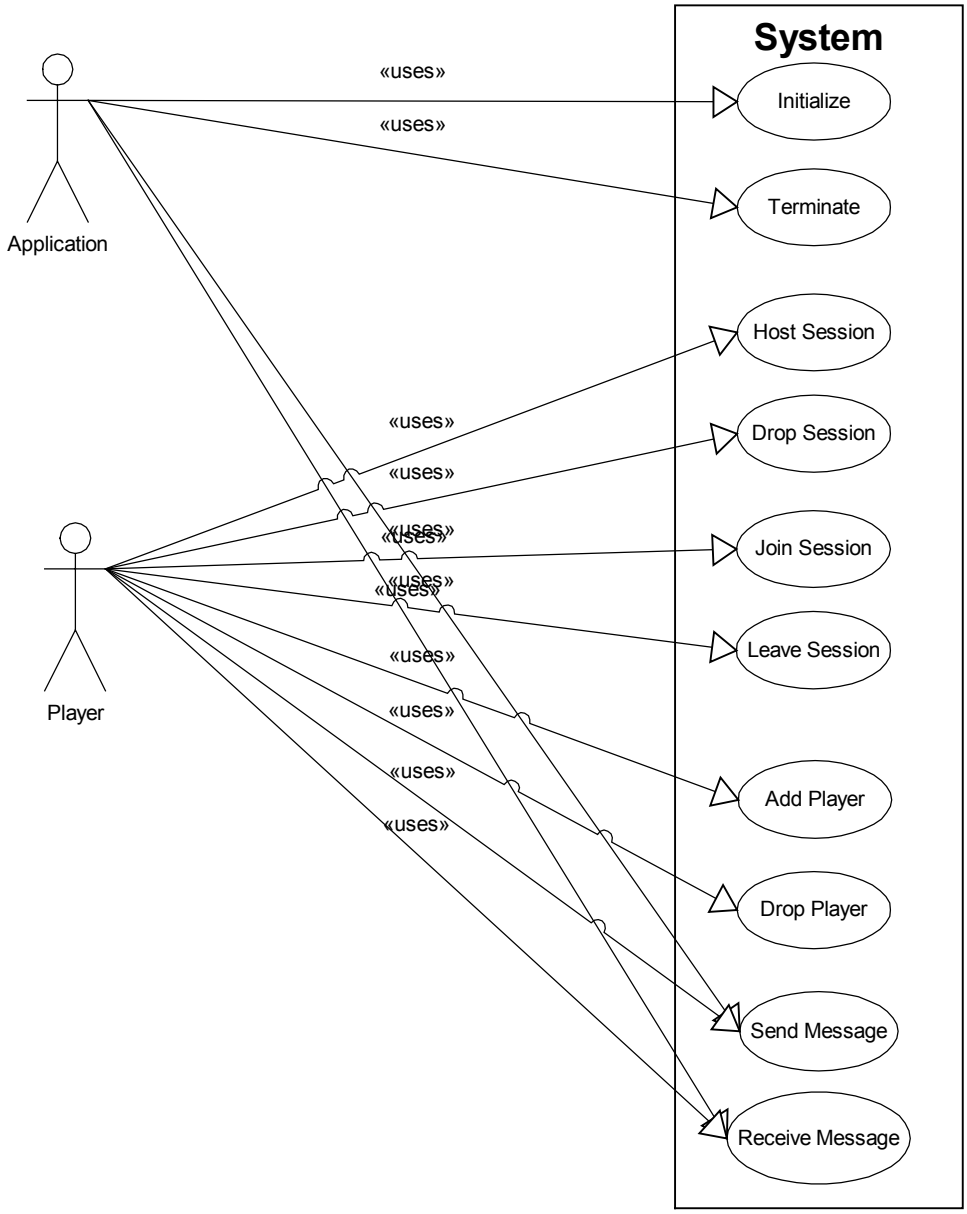
Host: A player that allows other players to connect to it for the game.

Player: One plays the game. In multiplayer play, referred to one who connects to the host's computer in order to play.

Session: Allows other players to connect to the computer—i.e. act as a host.

Message: Any data being sent between computers. Could be initialization data, internal game messages, or chat messages.

3.2.2.4 Expanded Use Case Diagram



3.2.2.5 Expanded Use Cases

3.2.2.5.1 Initialize

3.2.2.5.1.1 Purpose/Overview

Initialize communication to allow for connections to other computers. Enumerate connections available on computer to see if it is capable of multiplayer play.

3.2.2.5.1.2 Actors

Application

3.2.2.5.1.3 Type
Primary and essential

3.2.2.5.1.4 Cross Reference
R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.2.5.1.5 Typical Course of Events

Actor Action	System Response
Create Communication Class	Initialize memory.
Initialize	Return true if multiplayer capability exists.

3.2.2.5.1.6 Alternative Courses

3.2.2.5.2 Terminate

3.2.2.5.2.1 Purpose/Overview
Terminate any connections and perform cleanup

3.2.2.5.2.2 Actors
Application

3.2.2.5.2.3 Type
Primary and essential

3.2.2.5.2.4 Cross Reference
R1.4, R1.5, R1.6, R1.7, R1.8, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.2.5.2.5 Typical Course of Events

Actor Action	System Response
Terminate Game	Close Connections, Sessions, release memory and objects

3.2.2.5.2.6 Alternative Courses

3.2.2.5.3 Host Session

3.2.2.5.3.1 Purpose/Overview
Set up a host session to allow for other players to connect to the game.

3.2.2.5.3.2 Actors
Player

3.2.2.5.3.3 Type
Primary and essential

3.2.2.5.3.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.2.5.3.5 Typical Course of Events

Actor Action	System Response
Close All Connections	Close Connections
Close Sessions	Close Session
Cleanup	Release memory, destroy objects

3.2.2.5.3.6 Alternative Courses

3.2.2.5.4 Drop Session

3.2.2.5.4.1 Purpose/Overview

Disconnect any connected players and destroy host session. Don't allow others to connect to the game.

3.2.2.5.4.2 Actors

Player

3.2.2.5.4.3 Type

Primary and essential

3.2.2.5.4.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.5, R2.7

3.2.2.5.4.5 Typical Course of Events

Actor Action	System Response
Close Session	Close hosted connections and allow player to select a new game.

3.2.2.5.4.6 Alternative Courses

3.2.2.5.5 Join Session

3.2.2.5.5.1 Purpose/Overview

Allow the current computer to join a game on a different computer hosting a connection.

3.2.2.5.5.2 Actors

Player

3.2.2.5.5.3 Type

Primary and essential

3.2.2.5.5.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.2.5.5.5 Typical Course of Events

Actor Action	System Response
Join Session	Lists methods of connecting
Select Method	Find computers hosting game or allow user to select computer manually
Choose Host	Create Connection and initialize game data

3.2.2.5.5.6 Alternative Courses

3.2.2.5.6 Leave Session

3.2.2.5.6.1 Purpose/Overview

Disconnect current computer from the game. Break the current connection to the host.

3.2.2.5.6.2 Actors

Player

3.2.2.5.6.3 Type

Primary and essential

3.2.2.5.6.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7, R2.9

3.2.2.5.6.5 Typical Course of Events

Actor Action	System Response
Leave Session	Send message that player is leaving game, close connection and perform cleanup.

3.2.2.5.6.6 Alternative Courses

3.2.2.5.7 Add Player

3.2.2.5.7.1 Purpose/Overview

Joins a new player to the current game from a different computer.

3.2.2.5.7.2 Actors

Player

3.2.2.5.7.3 Type

Primary and essential

3.2.2.5.7.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7

3.2.2.5.7.5 Typical Course of Events

Actor Action	System Response

<none>	New Player has been Added – Raise Message with player info
Add New Player	Add New Player to Game

3.2.2.5.7.6 Alternative Courses

3.2.2.5.8 Drop Player

3.2.2.5.8.1 Purpose/Overview

Destroy connection from the computer the player is on if needed and drop player from game or allow AI to fill in. Also used for cleanup if connection between computers is broken.

3.2.2.5.8.2 Actors

Application, Player

3.2.2.5.8.3 Type

Primary and essential

3.2.2.5.8.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.5, R2.8

3.2.2.5.8.5 Typical Course of Events

Actor Action	System Response
<none>	Player has been Dropped – Raise Message with player info
Drop Player From Game	Drop Player From Game

3.2.2.5.8.6 Alternative Courses

If desired, after player is dropped, allow an AI player to fill in using the host computer.

3.2.2.5.9 Send Message

3.2.2.5.9.1 Purpose/Overview

Send a message to other players over the connection.

3.2.2.5.9.2 Actors

Application, Player

3.2.2.5.9.3 Type

Primary and essential

3.2.2.5.9.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7

3.2.2.5.9.5 Typical Course of Events

Actor Action	System Response
Send Message	Send Message to host and client computers via the connections that have been made

3.2.2.5.9.6 Alternative Courses

3.2.2.5.10 Receive Message

3.2.2.5.10.1 Purpose/Overview

Send a message to other players over the connection.

3.2.2.5.10.2 Actors

Application, Player

3.2.2.5.10.3 Type

Primary and essential

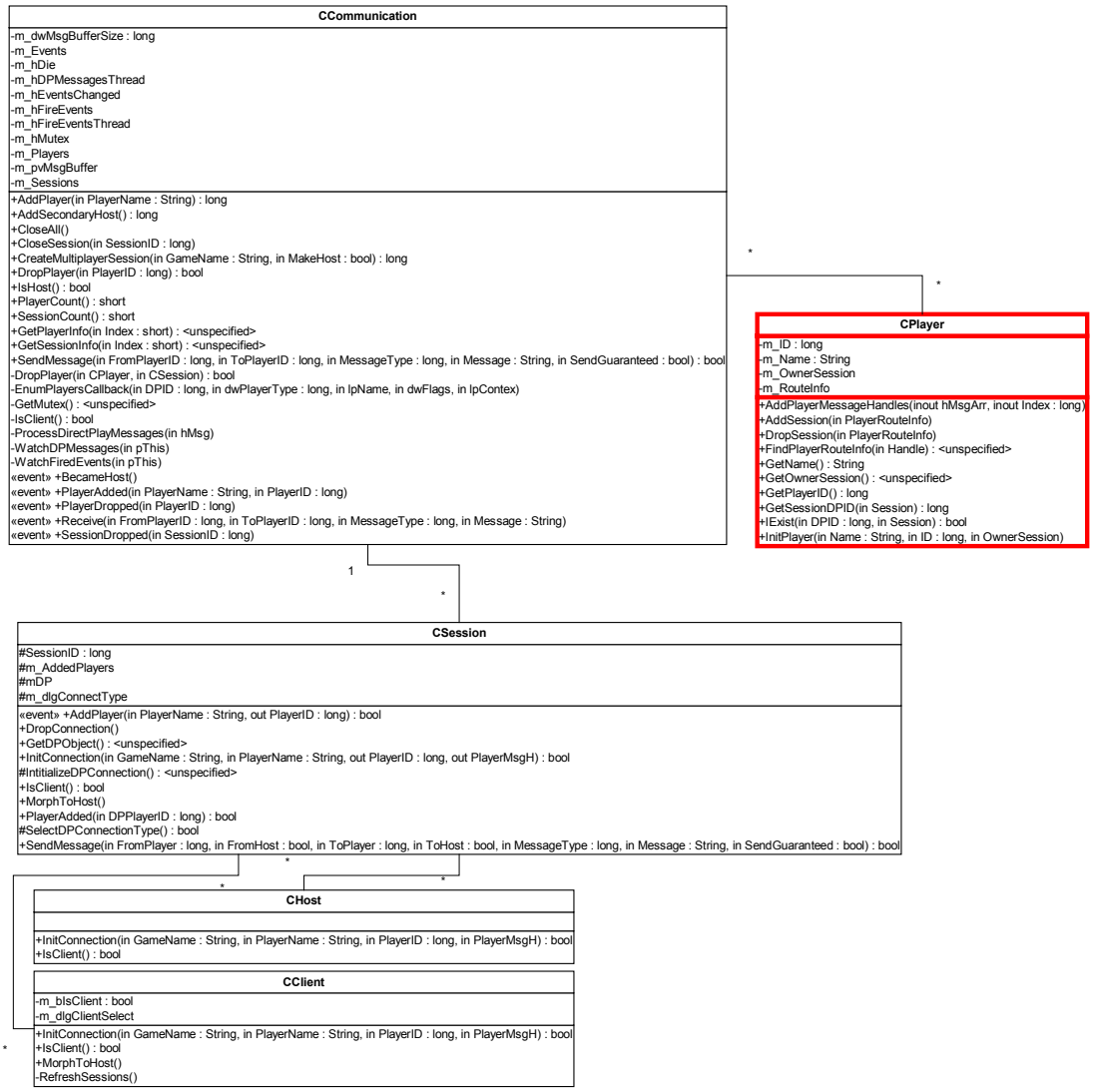
3.2.2.5.10.4 Cross Reference

R1.4, R1.5, R1.6, R1.7, R2.4, R2.5, R2.6, R2.7

3.2.2.5.10.5 Typical Course of Events

Actor Action	System Response
<None>	A message has been sent by another computer, raise event
Deal with Message	Parse message as to what type and act on it accordingly

3.2.2.6 Class Diagram



3.3 Artificial Intelligence

3.3.1 High Level Design

3.3.1.1 Introduction

This document provides the basic design for the artificial intelligence engine that will be used in the sheepshead program. The AI will be responsible for determining computer player's moves, suggesting moves to human players, and enforcing the rules of the game.

3.3.1.2 Applicable Documents

SRD – Engine Requirements
SRD - Engine Requirements

AI Design.vsd (Visio 2000)

3.3.1.3 Glossary of Terms

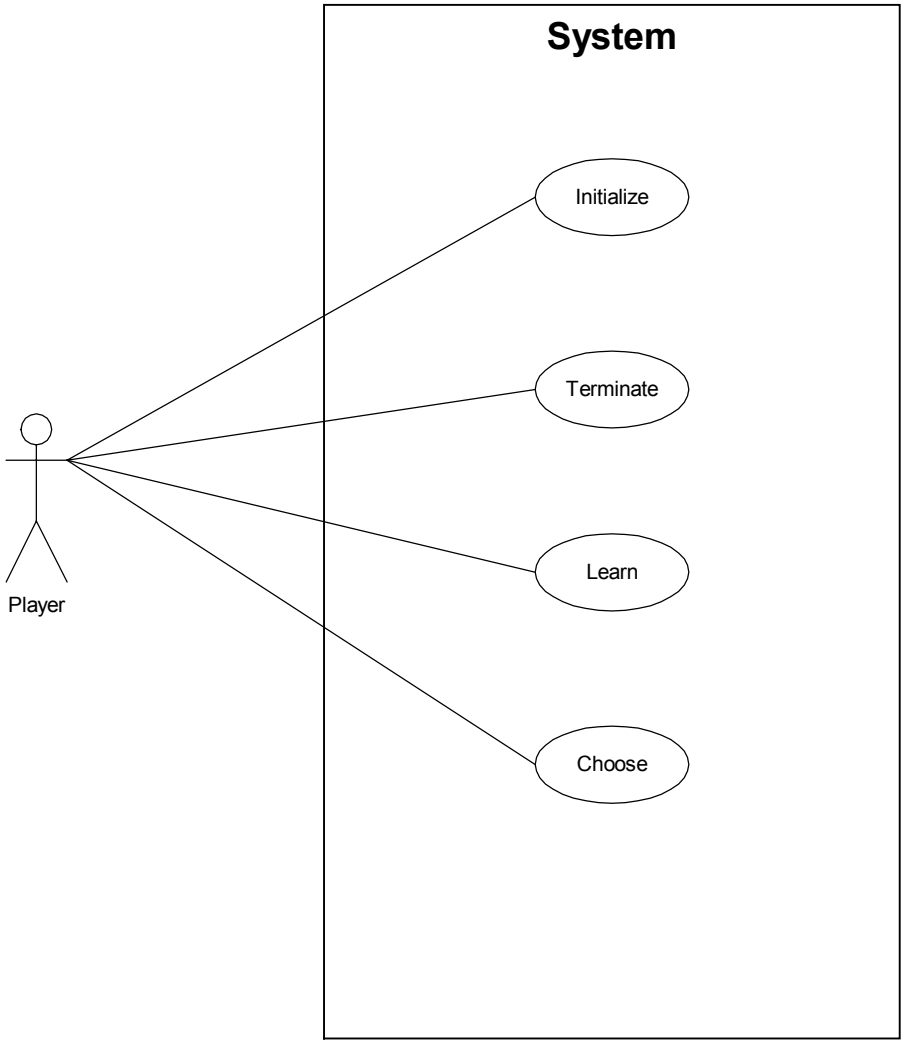
AI player – A player controlled by the computer.

Host – A human player that has set up a network/internet game.

Neural Network – A method of mimicking human brain functions on a computer. A neural network is made of multiple layers of neurons. Each neuron processes data, and ultimately contributes to the output of the neural net.

Player – Any player in the game. This includes AI players.

3.3.1.4 High Level Use Case Diagram



3.3.1.5 High Level Use Cases

3.3.1.5.1 Initialize

3.3.1.5.1.1 Purpose/Overview
Create the instance of the AI and initialize all of the variables.

3.3.1.5.1.2 Actors
Player

3.3.1.5.1.3 Type
Primary and Essential

3.3.1.5.1.4 Cross Reference
All

3.3.1.5.2 Terminate

3.3.1.5.2.1 Purpose/Overview
When the application is shut down all instances of the AI are closed and object clean-up is preformed.

3.3.1.5.2.2 Actors
Player

3.3.1.5.2.3 Type
Primary and Essential

3.3.1.5.2.4 Cross Reference
All

3.3.1.5.3 Choose

3.3.1.5.3.1 Purpose/Overview
The AI's turn comes up and the AI has to determine what card to play and play it.

3.3.1.5.3.2 Actors
Player, AI

3.3.1.5.3.3 Type
Primary and Essential

3.3.1.5.3.4 Cross Reference
R1.3, R2.2, R2.3, R2.7

3.3.1.5.4 Learn

3.3.1.5.4.1 Purpose/Overview
The neural net will update it's strategy, based on whomever wins the game.

3.3.1.5.4.2 Actors

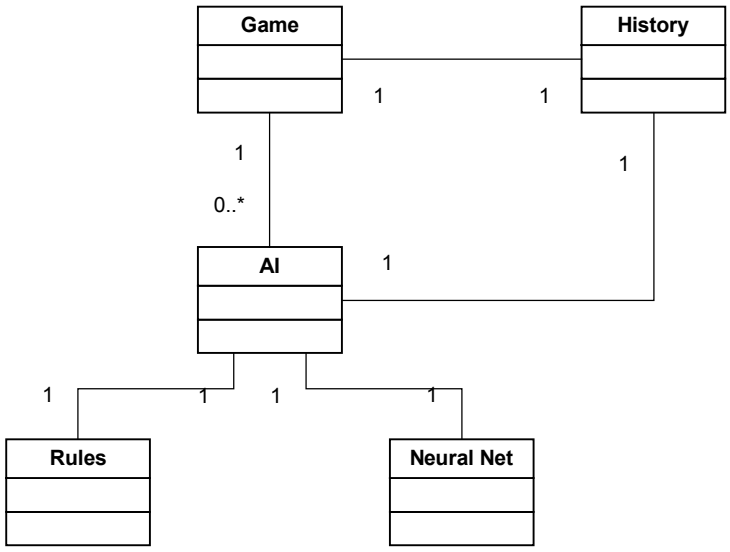
Players

3.3.1.5.4.3 Type
Secondary

3.3.1.5.4.4 Cross Reference
R2.1

3.3.1.6 Screen Prototypes
Not Applicable

3.3.1.7 Conceptual Object Model



3.3.2 Expanded Use Cases

3.3.2.1 Introduction
This document provides the extended design for the artificial intelligence engine that will be used in the sheepshead program. The AI will be responsible for determining computer player's moves, suggesting moves to human players, and enforcing the rules of the game.

3.3.2.2 Applicable Documents
AI High Level Design Document:
HLD – Artificial Intelligence

AI Design.vsd (Visio 2000)

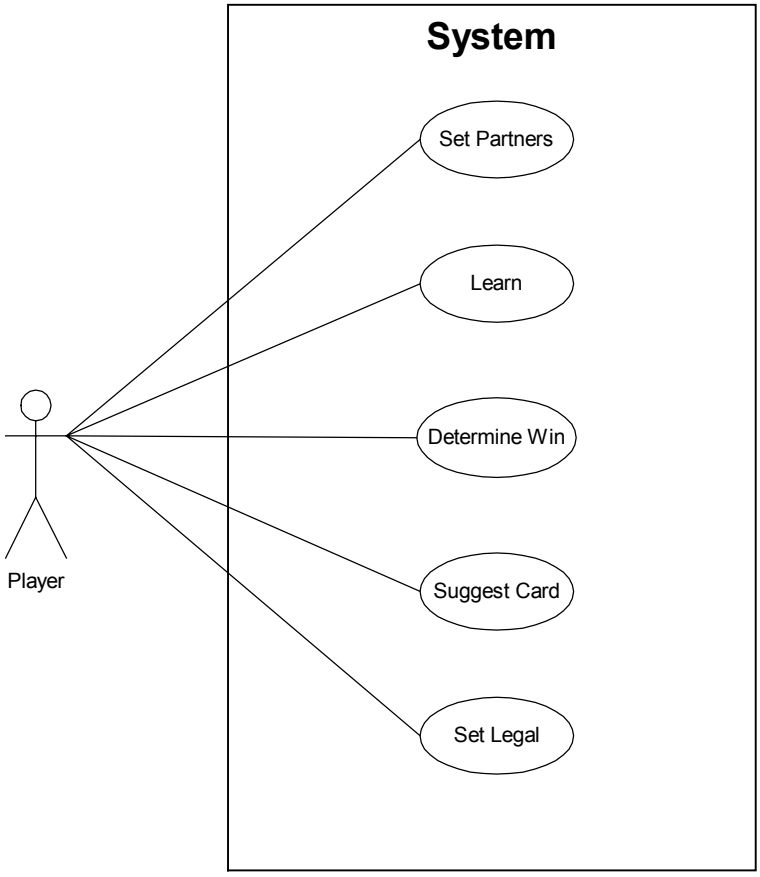
3.3.2.3 Glossary of Terms
AI player – A player controlled by the computer.

Host – A human player that has set up a network/internet game.

Neural Network – A method of mimicking human brain functions on a computer. A neural network is made of multiple layers of neurons. Each neuron processes data, and ultimately contributes to the output of the neural net.

Player – Any player in the game. This includes AI players.

3.3.2.4 Expanded Use Case Diagram



3.3.2.5 Expanded Use Cases

3.3.2.5.1 Suggest Card

3.3.2.5.1.1 Purpose/Overview

Given the history of the game to date, the AI will determine the best card to play, either by applying a rule, or by using the neural network.

3.3.2.5.1.2 Actors

Player – The player may request a sugession.
AI – The AI players will use this to determine their moves.

3.3.2.5.1.3 Type

Primary and Essential

3.3.2.5.1.4 Cross Reference

SRD – Engine Requirements.doc: R2.2

3.3.2.5.1.5 Typical Course of Events

Actor Action	System Response
The player asks for help, or the AI player's turn comes	The system maps the cards played to fuzzy values
	The system checks for a rule that applies to the game history. If a rule applies, apply that rule, map the fuzzy response to a card, and stop here.
	If the rules fail, format the history into the format required by the neural net.
	Apply the history to the neural net, and map the response to a card.

3.3.2.5.1.6 Alternative Courses

None

3.3.2.5.2 Determine Win

3.3.2.5.2.1 Purpose/Overview

Given the cards played in a trick, the AI will determine which player has won the trick, and how many points are awarded to that player. The winner is also the leader of the next trick.

3.3.2.5.2.2 Actors

Player: The use case begins after the last player in the trick has played their card.

3.3.2.5.2.3 Type

Primary and Essential

3.3.2.5.2.4 Cross Reference

3.3.2.5.2.5 Typical Course of Events

Actor Action	System Response
The last player for the trick plays their card.	The system loads the values for the cards.
	The system maps the cards to fuzzy values.
	The system determines who won the hand

	based on the highest ranked card played.

3.3.2.5.2.6 Alternative Courses

None

3.3.2.5.3 Set Legal

3.3.2.5.3.1 Purpose/Overview

Given the card that led the trick, and the cards in the player's hand, the system will determine which ones are legal to play for this trick.

3.3.2.5.3.2 Actors

Player: The player who's turn it is.

3.3.2.5.3.3 Type

Primary and Essential

3.3.2.5.3.4 Cross Reference

SRD – Engine Requirements.doc: R2.2

3.3.2.5.3.5 Typical Course of Events

Actor Action	System Response
The player's turn comes up.	The system loads the trump cards.
	The system returns which cards in the player's hand are legal, all if he has no cards matching the suit that was lead.

3.3.2.5.3.6 Alternative Courses

None

3.3.2.5.4 Set Partners

3.3.2.5.4.1 Purpose/Overview

The system will determine which players are partnered based on the loaded rules.

3.3.2.5.4.2 Actors

Players: Partners are based on all of the player's actions.

3.3.2.5.4.3 Type

Primary and Essential

3.3.2.5.4.4 Cross Reference

SRD – Engine Requirements.doc: R1.3

3.3.2.5.4.5 Typical Course of Events

Actor Action	System Response

The cards are dealt.	If the type of game states that the partners are determined by a pre-game action, check the rules for which players are partnered.
A player plays a card.	If the type of game states that the partners are determined by an in-game action, check the rules to see if the player's move has partnered him with someone else.

3.3.2.5.4.6 Alternative Courses

None

3.3.2.5.5 Learn

3.3.2.5.5.1 Purpose/Overview

The neural net will update it's strategy, based on whomever wins the game.

3.3.2.5.5.2 Actors

Players: The neural net can only update it's play based on a human player's move, not it's own.

3.3.2.5.5.3 Type

3.3.2.5.5.4 Cross Reference

SRD – Engine Requirements.doc: R2.1

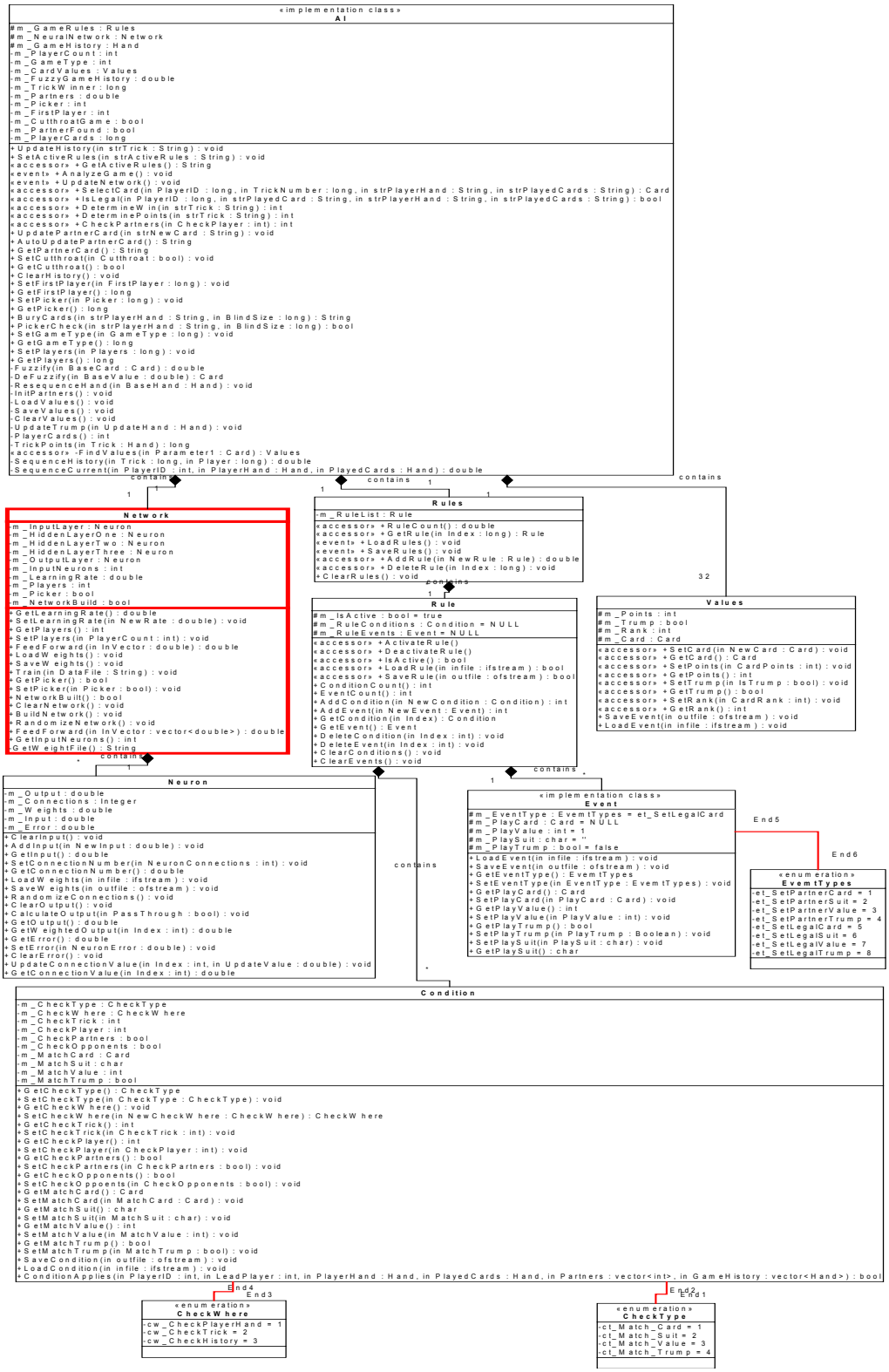
3.3.2.5.5.5 Typical Course of Events

Actor Action	System Response
One or more human players play a game.	The system stores the history and results of the game.
	While the system is idle, the neural net attempts to update its weight matrix based on the play of winners from previous games.

3.3.2.5.5.6 Alternative Courses

The neural net may not be able to converge on the data gathered. The training data will be accumulated. If the network does not converge once a set number of data points have been gathered, the oldest ones will be deleted.

3.3.2.6 Class Diagram



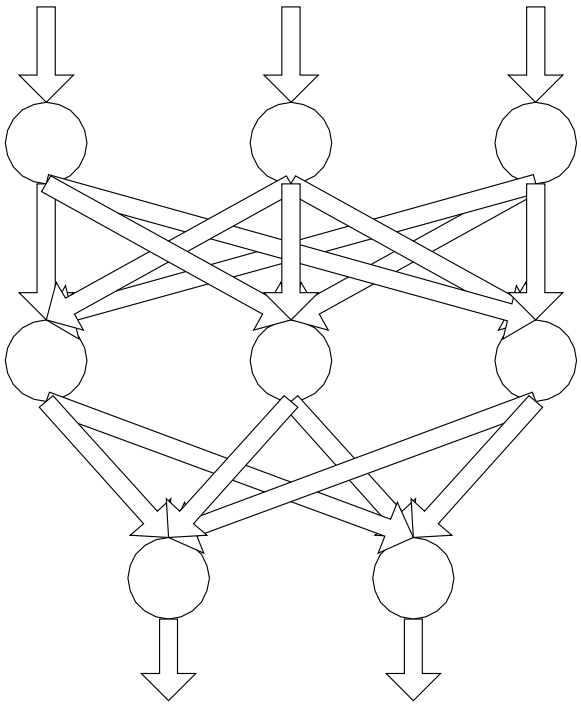
3.3.3 Special Notes

3.3.3.1 Neural Networks

As part of the artificial intelligence for this project, we will be implementing a neural network to take care of some of the decisions that are made during the game. A neural network is a computer simulation of how the human mind works. A neural network is made up of many neurons. Each of these neurons performs a small part of the calculations needed by the network. The calculations of each neuron can affect some or all of the other neurons by means of connections. The output from one neuron can be assigned as specific 'weight', and become part of the input for another neuron. The type of network we are using is a feed forward back propagation network,

3.3.3.2 Feed forward Back propagation Neural Network

In this type of network, the neurons are ordered into different layers. The first layer, known as the input layer, is where the values to be processed by the network are entered. All of the neurons in this layer are connected each neuron in the next layer by means of a weighted connection. That is, the output of each neuron in the input layer contributes to each neuron in the next layer, but the amount each neuron contributes varies depending on the weight. However there are no connections between any neurons in a layer, and no connections going back to a previous layer. An example of this type of network is shown.

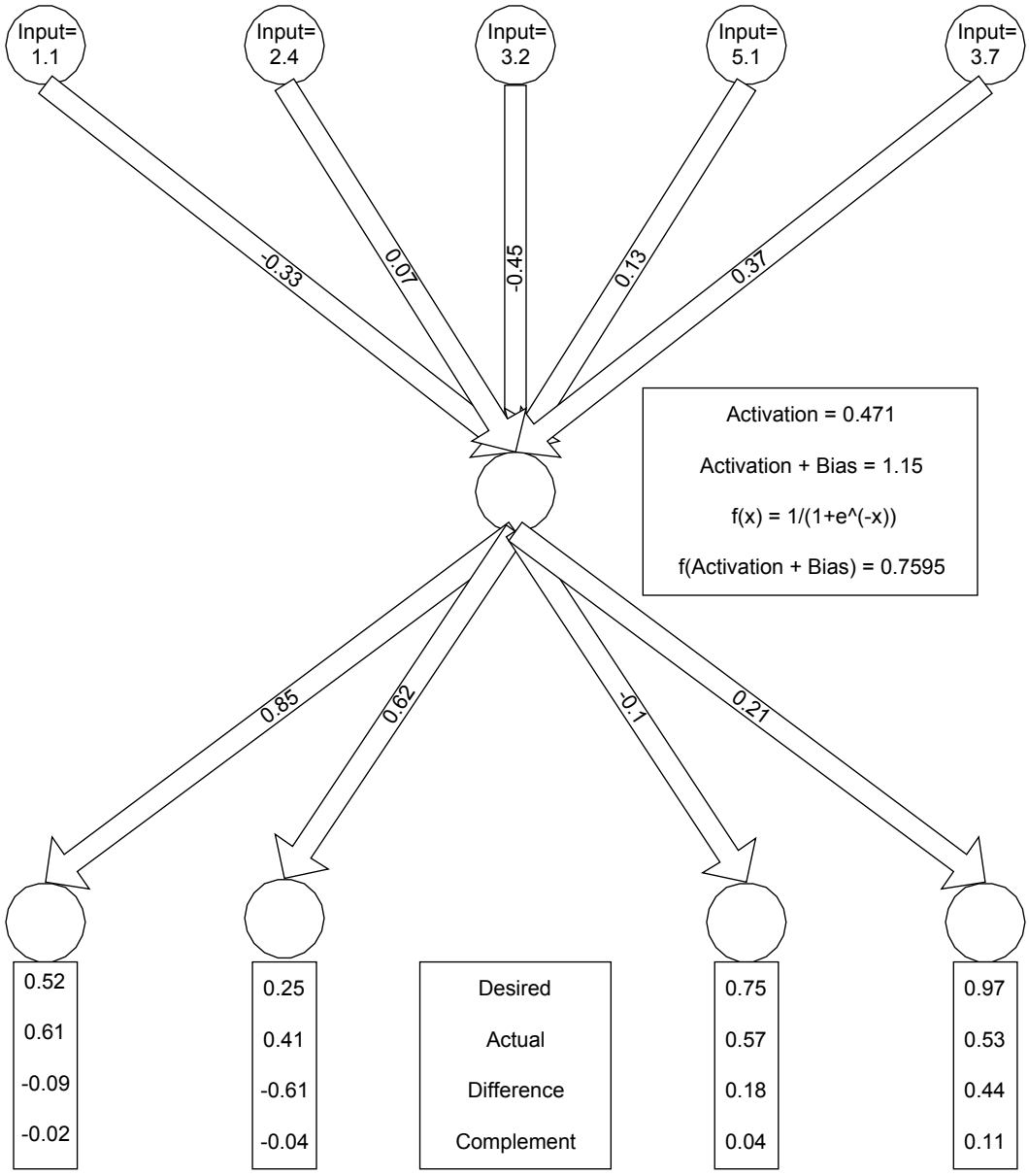


There may be any number of hidden, or processing, layers between the input layer and the final output layer. Once processing has reached the output layer, the network is done. Usually, given a certain input vector to the neural net, there is one specific output vector expected. It is doubtful that the network will produce the correct output on the first try. To adapt the network to produce the appropriate output vector for an input vector, the network is trained.

3.3.3.3 Training

In the feed forward back propagation network, the network is set up to produce the appropriate output by means of supervised training, where the network is provided with values for the input layer of neurons, and the expected values from the output layer. The weights are adjusted until the actual output is within some tolerance of the expected output. The algorithm used to adjust the weight matrix will be discussed here.

To illustrate how the network will learn, a sample network with inputs and connection weights is shown. For simplicity, only one hidden layer with one neuron in it is shown, but the idea can be extended to any neuron in any layer.



The input to the network is (1.1, 2.4, 3.2, 5.1, 3.7). The weights for the input neurons to the shown hidden layer neuron are (-0.33, 0.07, -0.45, 0.13, 0.37). Multiplying the input

values by the weights and summing results in a value of (1.1*-0.33) + (2.4*0.07) + (3.2*-0.45) + (5.1*0.13) + (3.7*0.37) = 0.471. This is the input value to the hidden layer neuron.

The hidden layer neuron may perform several operations on the input value to determine the output value. For example, a bias may be added to the input value. A bias is any number that is added to the input value. For the example, the bias will be 0.679. So, the activation plus the bias is: 0.417 + 0.679 = 1.15.

The neuron may also process the value by applying a threshold function. In this case, the threshold function used is $f(x) = 1/(1+e^{-x})$. This function is recommended if the output is to be in the range of zero to one. So, f(1.15) = 0.7595. This is the output of the neuron.

Again, the output value of the neuron is multiplied by each of the weights connecting it to the output neurons. The weights between the hidden neuron and the output neurons are (0.85, 0.62, -0.1, 0.21) for an output of (0.61, 0.41, 0.57, 0.53).

For this example, the expected output is (0.52, 0.25, 0.75, 0.97). Obviously, the actual output and the expected output do not match. To make the actual output of the network match the expected output of the network, the weights that connect the neurons must be adjusted. The method used to adjust the weights will be described next.

First, the difference between the desired and the actual output must be determined. Again, the desired output was (0.52, 0.25, 0.75, 0.97) and the actual was (0.61, 0.41, 0.57, 0.53). Taking the difference of these two vectors, we get the vector (-0.09, -0.61, 0.18, 0.44).

Next, we can determine the amount of error each weight contributed to the total error by multiplying the desired-actual vector by the actual output. This vector is (-0.549, -0.2501, 0.1026, 0.2332).

Finally, we must determine the amount of change that is needed. The current vector will also be scaled by the complement (1-value) of the actual output. This provides us with the first derivative, which indicates the amount of change. So, the complements of the actual output are (0.39, 0.59, 0.43, 0.47). Multiplying this times the previous vector results in (-0.02, -0.04, 0.04, 0.11).

Now, with one other piece of data, we can calculate how much to change the weights by. The other piece of data needed is the learning rate parameter. The learning rate parameter defines how much the changes we make are scaled by. The smaller it is, the more precise the final output will be, but the longer the network will take to train. In this example, the learning rate parameter is 0.2.

Given the output, the learning rate, and the complement vector, the amount of adjustment in each weight can be determined by multiplying these three values together. So, the adjustment to each weight between the hidden layer and the output layer would be: (0.2*0.7595*-0.02, 0.2*0.7595*-0.04, 0.2*0.7595*0.04, 0.2*0.7595*0.11) = (-0.003, -0.006, 0.006, 0.067). When this vector is added to the weights, we get adjusted weights of (0.847, 0.614, -0.094, 0.227).

Now, we have the adjusted weights for the hidden layer to the output layer, but what about any other layers there may be? The same principal can be extended to higher layers, since the output for each neuron is known, and the learning rate parameter remains constant. The only other thing we need is the error in the neuron's output. The actual

output of the neuron, the output's complement, and the error in each weight multiplied by the complement vector, all multiplied together will determine the amount of error in the neuron. For example, the neuron's output was 0.7595, the complement vector was (-0.02, -0.04, 0.04, 0.11), and the weights from the hidden neuron to the output neurons were (0.85, 0.62, -0.1, 0.21), resulting in the following calculation: (0.7595) * (1-0.7595) * (-0.02*0.85) * (0.62*-0.04) * (-0.1*0.04)*(0.21*0.11) = -0.0041. The error in the output of the neuron is –0.0041.

This value can be used to determine the error in the previous layer, whether it is another hidden layer or the input layer, in the same fashion as before. The only difference between a hidden layer and the input layer is that the hidden layer neurons may modify the input values they receive with threshold functions and biases. The input layer neurons just pass the values through, so their output is the same as their input.

4 Core

4.1 High Level Design

4.1.1 Introduction

This document is to describe the high level design of the overall project based on the components.

4.1.2 Applicable Documents

Requirements:
SRD – The Mutton Project
SRD - Engine Requirements
SRD - GUI Requirements

AI:
HLD - AI
EUC - AI

Communications:
HLD - Communications
EUC - Communications

GUI:
SRD - Card Control
HLD - GUI
EUC - GUI

Help:
HLD - Help System
EUC - Help System

Player:
SRD - Player
HLD - Player
EUC - Player

4.1.3 Glossary of Terms

User – Any person who wishes to play the Sheepshead computer game.

Player – A human or AI entity within the game.

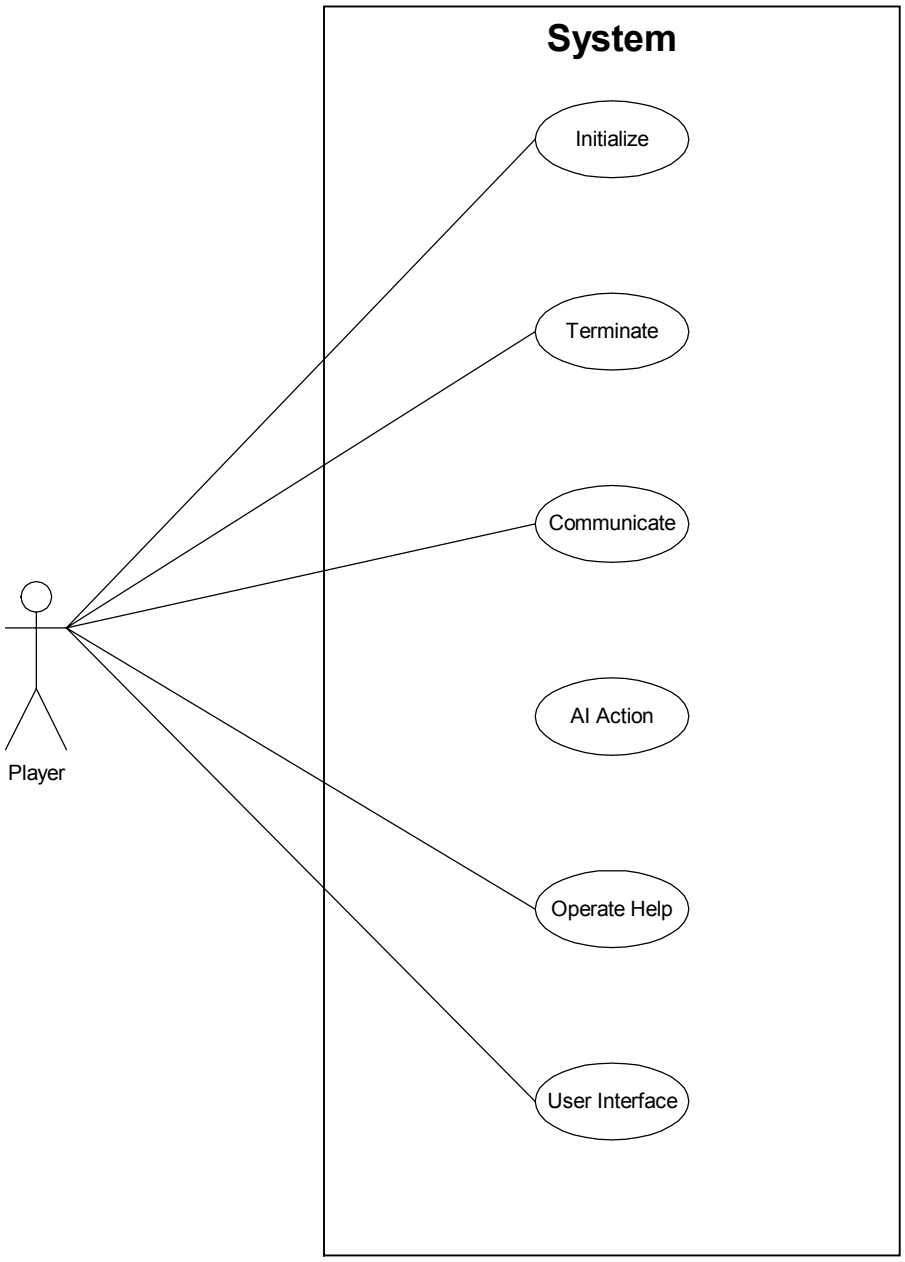
GUI – The Graphical User Interface presented to the user.

Comm – The communications object.

Help – The online help system.

System – The game object itself.

4.1.4 High Level Use Case Diagram



4.1.5 High Level Use Cases

4.1.5.1 Initialize

4.1.5.1.1 Purpose/Overview

The game object is created and instances of the AI, Comm., GUI, and Player are created.

4.1.5.1.2 Actors

User

4.1.5.1.3 Type
Primary and Essential

4.1.5.1.4 Cross Reference
All

4.1.5.2 Terminate

4.1.5.2.1 Purpose/Overview
Shutdown the application and close all instances of the GUI, AI, Comm, and Player.

4.1.5.2.2 Actors
User

4.1.5.2.3 Type
Primary and Essential

4.1.5.2.4 Cross Reference
All

4.1.5.3 Communicate

4.1.5.3.1 Purpose/Overview
Talk to another instance of the game on a different computer.

4.1.5.3.2 Actors
User, System, Comm.

4.1.5.3.3 Type
Primary and Essential

4.1.5.3.4 Cross Reference
GUI Req. R1.5; Engine Req. R1.4, R1.5, R1.6, R1.7, R1.8, R2.4

4.1.5.4 AI Action

4.1.5.4.1 Purpose/Overview
Any action the AI has to perform from acting as a tutorial to acting as a player and playing a card.

4.1.5.4.2 Actors
System, Help, AI

4.1.5.4.3 Type

Primary and Essential

4.1.5.4.4 Cross Reference

Engine Req. R1.1, R1.2, R1.3, R1.9, R2.1, R2.2, R2.3, R2.9

4.1.5.5 Operate Help

4.1.5.5.1 Purpose/Overview

Detail any actions that are performed upon the help by the user or by the system with the tutorial and agent.

4.1.5.5.2 Actors

System, User, Help

4.1.5.5.3 Type

Primary and Essential

4.1.5.5.4 Cross Reference

GUI Req. R1.2, R1.3; Engine Req. R2.3

4.1.5.6 User Interface

4.1.5.6.1 Purpose/Overview

Provide a way for the user to communicate to the system.

4.1.5.6.2 Actors

User, System

4.1.5.6.3 Type

Primary and Essential

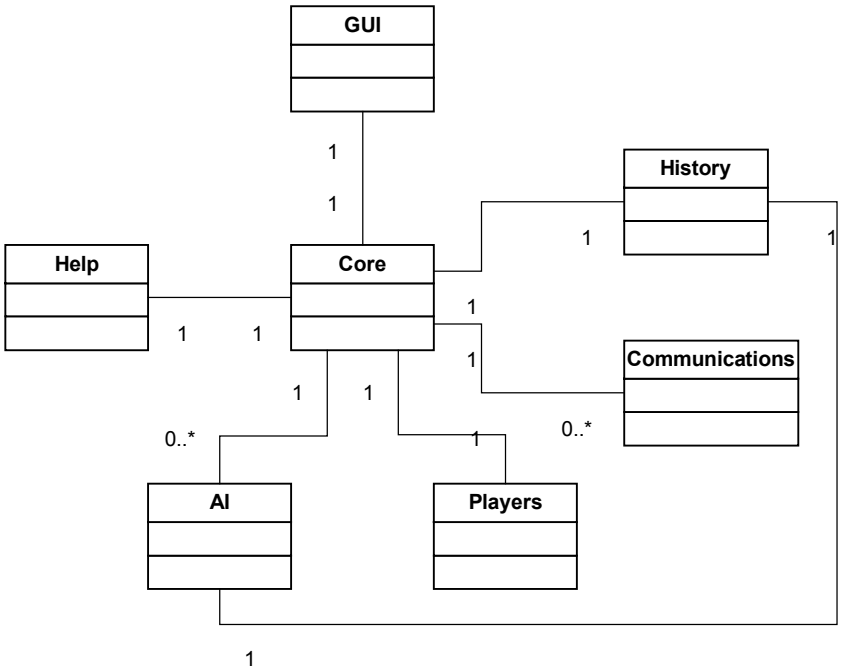
4.1.5.6.4 Cross Reference

GUI Req. R1.1, R1.4, R1.5, R1.6, R1.7; Engine Req. R2.4, R2.5, R2.8

4.1.6 Screen Prototypes

See HLD - GUI.doc

4.1.7 Conceptual Object Model



4.2 Expanded Use Cases

4.2.1 Introduction

The purpose of this document is to detail the design of the Game or System object of the Mutton Project.

4.2.2 Applicable Documents

Core High Level Design
HLD - Core

4.2.3 Glossary of Terms

User – Any person who wishes to play the Sheepshead computer game.

Player – A human or AI entity within the game.

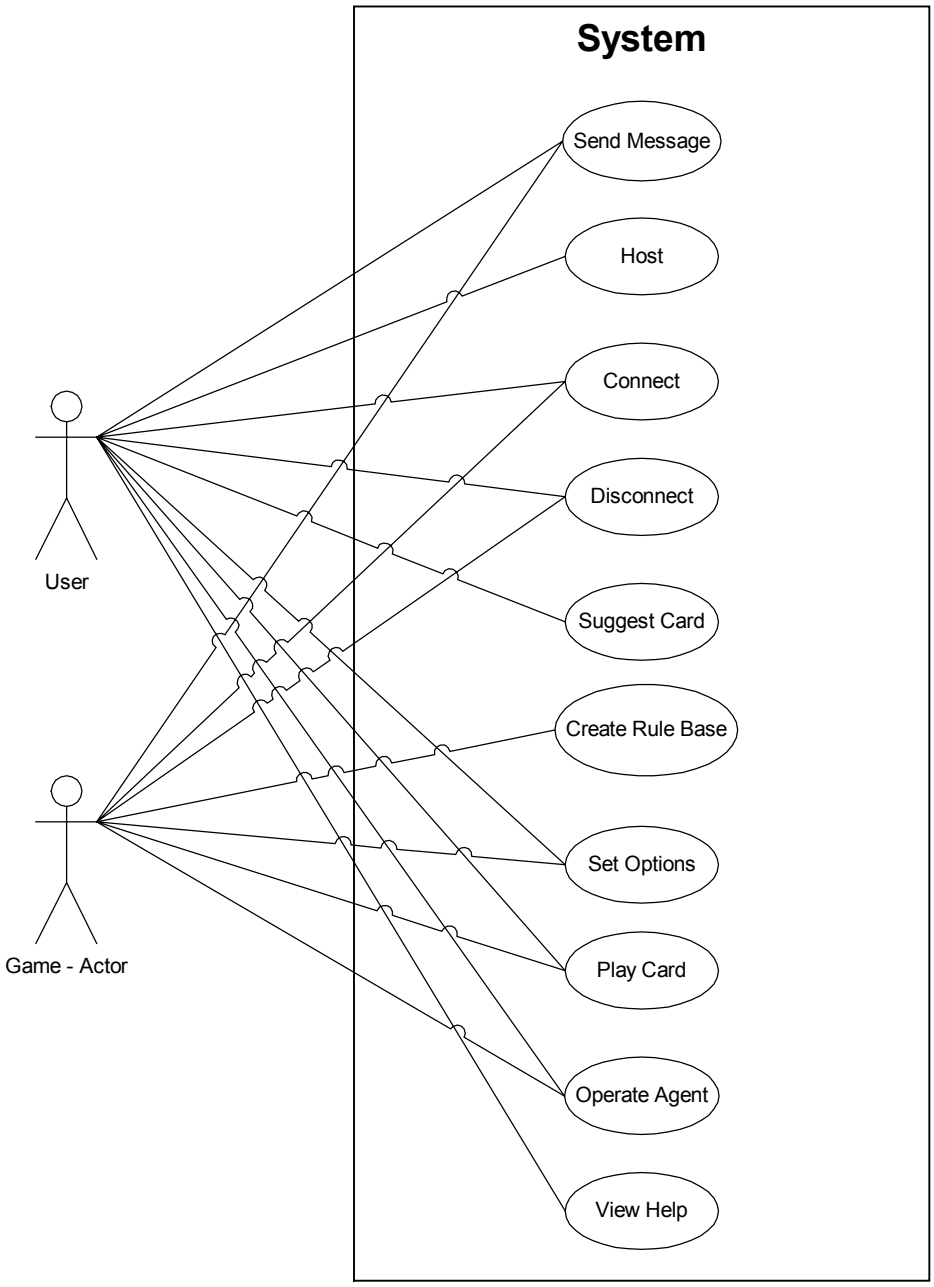
GUI – The Graphical User Interface presented to the user.

Comm – The communications object.

Help – The online help system.

System – The game object itself.

4.2.4 Expanded Use Case Diagram



4.2.5 Expanded Use Cases

4.2.5.1 Send Message

4.2.5.1.1 Purpose/Overview

Sends a message to the current user or users (during a network game).

4.2.5.1.2 Actors

User and Game Actor

4.2.5.1.3 Type
Primary and Essential

4.2.5.1.4 Cross Reference

4.2.5.2 Host

4.2.5.2.1 Purpose/Overview
Current user and machine will be the host of the networked game.

4.2.5.2.2 Actors
User

4.2.5.2.3 Type
Primary and Essential

4.2.5.2.4 Cross Reference

4.2.5.3 Connect

4.2.5.4 Purpose/Overview
Connects the current user and machine to a networked game.

4.2.5.4.1 Actors
User and Game Actor

4.2.5.4.2 Type
Primary and Essential

4.2.5.4.3 Cross Reference

4.2.5.5 Disconnect

4.2.5.5.1 Purpose/Overview
Disconnects the user from a pending network game.

4.2.5.5.2 Actors
User and Game Actor

4.2.5.5.3 Type
Primary and Essential

4.2.5.5.4 Cross Reference

4.2.5.6 Suggest Card

4.2.5.6.1 Purpose/Overview

User or game can get a suggestion to what card should be played.

4.2.5.6.2 Actors

User and Game Actor

4.2.5.6.3 Type

Primary

4.2.5.6.4 Cross Reference

4.2.5.7 Create Rule Base

4.2.5.7.1 Purpose/Overview

Sets up the rules/parameters of an eminent game, given the user's selection of available rules.

4.2.5.7.2 Actors

Game Actor

4.2.5.7.3 Type

Primary

4.2.5.7.4 Cross Reference

4.2.5.8 Set Options

4.2.5.8.1 Purpose/Overview

Sets any mutable options for the game application, given input from the user.

4.2.5.8.2 Actors

Game Actor

4.2.5.8.3 Type

Secondary

4.2.5.8.4 Cross Reference

4.2.5.9 Play Card

4.2.5.9.1 Purpose/Overview

Delegates the necessary tasks (e.g. updating visible cards on table, notifying other users involved in network game of the played card) in response to a player playing a card.

4.2.5.9.2 Actors
Game Actor

4.2.5.9.3 Type
Primary and Essential

4.2.5.9.4 Cross Reference

4.2.5.10 Operate Agent

4.2.5.10.1 Purpose/Overview
Operates the visual Agent character that interfaces with the user.

4.2.5.10.2 Actors
Game Actor

4.2.5.10.3 Type
Secondary

4.2.5.10.4 Cross Reference

4.2.5.11 View Help

4.2.5.11.1 Purpose/Overview
Allows user to navigate the available help system (implemented using HTML Help).

4.2.5.11.2 Actors
User

4.2.5.11.3 Type
Secondary

4.2.5.11.4 Cross Reference

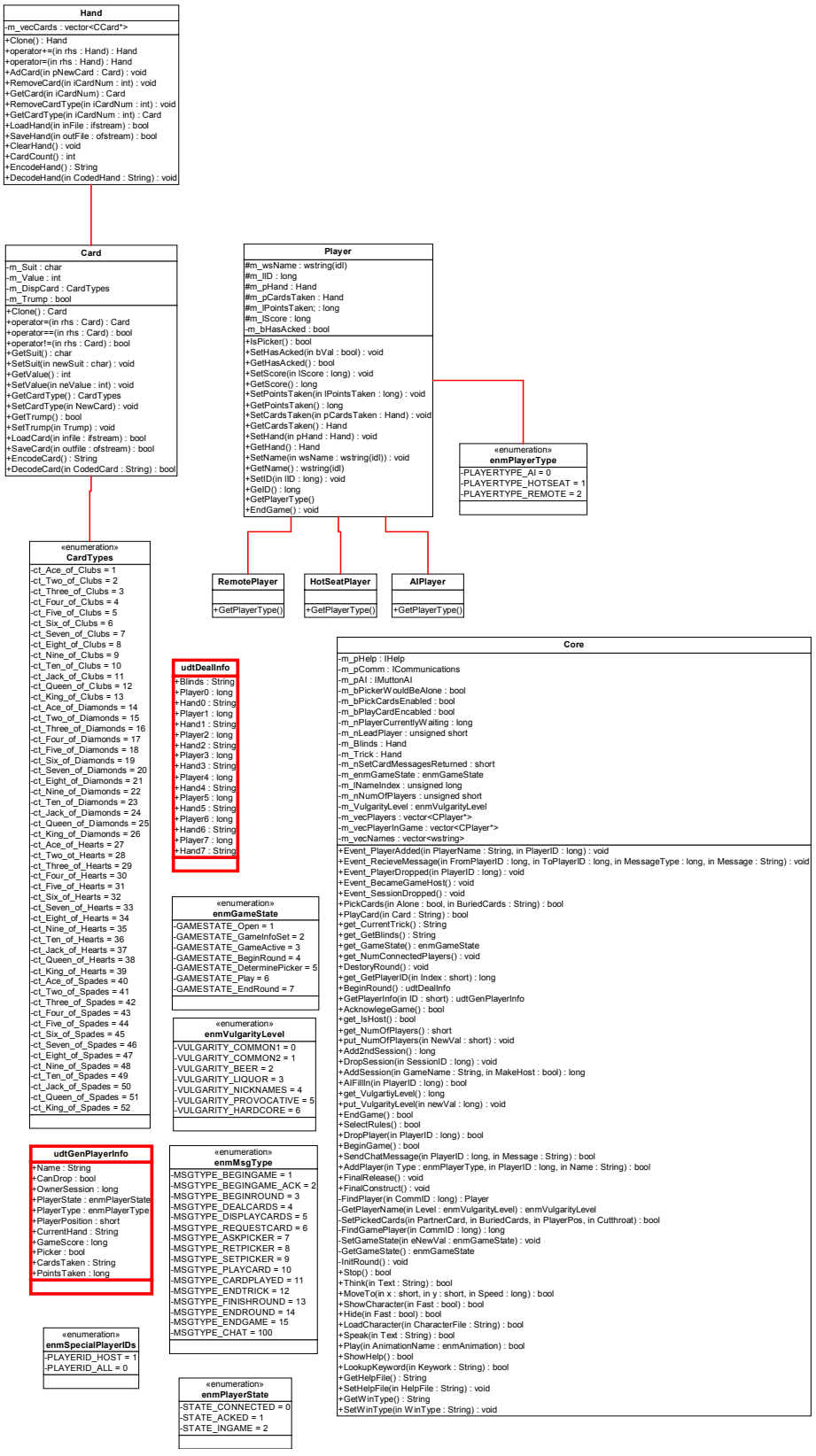
4.2.5.11.5 Typical Course of Events

Actor Action	System Response
Send Message	Message is cascaded to a single user or all pertinent users—depending on message source (user or game).
Host	Sets up current machine as the host machine to which all other machines communication.
Connect	Connects current machine to an existing

	network game (set up by the host machine).
Disconnect	Disconnects current machine from the network game.
Suggest Card	AI object is queried for an appropriate card to play.
Create Rule Base	Parameters of the pending game are defined.
Set Options	Extraneous options (i.e. not the rule base) are set for the game application.
Play Card	Game notifies all relevant players about the newly played card and GUI is updated to reflect the played card.
Operate Agent	Agent character is shown and allowed to interface with the user.
View Help	Help system is shown to the user.

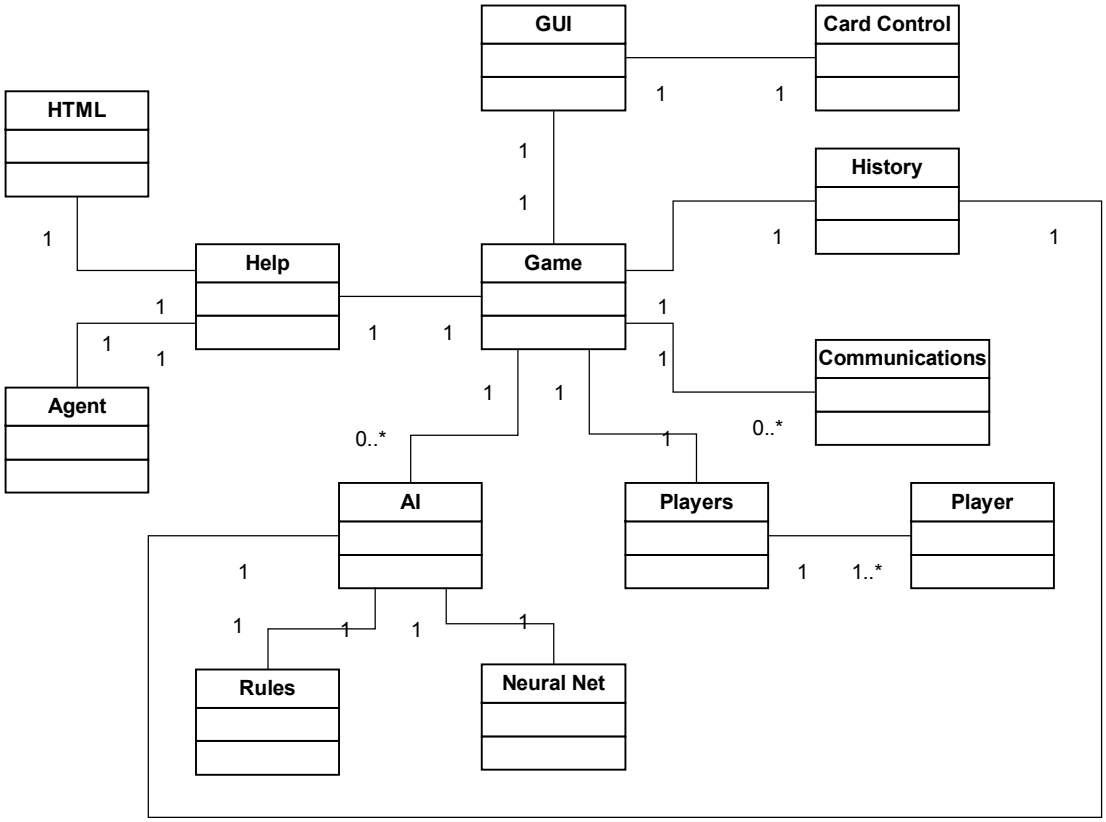
4.2.5.11.6 Alternative Courses

4.2.6 Class Diagram



5 Visio

5.1 Global Object Model



5.2 Visio Class Documentation

UML Static Structure Report

AIDesign

Model: Static Model

FullPath: UML System 1::Static Model
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
Contains Package(s): UML System 1::Static Model::Top Package

Model Element Statistic Summary

Number of classes:	9
Number of datatypes:	3
Number of attributes:	70
Number of parameters:	474
Number of operations:	148
Number of methods:	148
Number of associations:	7
Number of association ends:	14
Number of tagged values:	1150
Number of links:	3
Number of link ends:	6
Number of packages:	1

Package: Top Package

Contains Package(s): <None>

Contains Subsystem(s): <None>

FullPath: UML System 1::Static Model::Top Package
Visibility: public
Stereotype: topLevelPackage
IsRoot: No
IsLeaf: No
IsAbstract: No

Classifier: AI

FullPath: UML System 1::Static Model::Top Package::AI
Visibility: public
Stereotype: implementation class
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_GameRules
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Rules
TaggedValues
documentation All of the game rules created by the player, or any rules created by other players in a network game that have been loaded onto this player's computer.

2. m_NeuralNetwork
Visibility: protected
InitialValue:

Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Network
TaggedValues
documentation The neural network used if the rules are unable to provide a solution.

3. m_GameHistory
Visibility: protected
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Hand
TaggedValues
documentation A vector containing Hand objects representing the history of the game.

4. m_PlayerCount
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The number of players in the game. Used for determining partners and the architecture of the neural network.

5. m_GameType
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The type of game being played (regular or lester)

6. m_CardValues
Visibility: private
InitialValue:
Multiplicity: 32
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Values
TaggedValues
documentation Contains information on the point and rank value of each card.

7. m_FuzzyGameHistory
Visibility: private
InitialValue:
Multiplicity: *

Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The history of the game, fuzzified.

8. m_TrickWinner
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation A vector containing the ID number of the player that won each trick.

9. m_Partners
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation A vector of vectors containing parter information on a trick by trick basis.

10. m_Picker
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The player ID of the picker. -1 if there was no blind.

11. m_FirstPlayer
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The ID number of the player that lead the first trick.

12. m_CutthroatGame
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool

TaggedValues
documentation True if this is a cutthroat game (picker stands alone)

13. m_PartnerFound
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if the rules have determined which players are partnered.

14. m_PlayerCards
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The number of cards each player has in their hand.

Operations

1. UpdateHistory
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
1.1. strTrick
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The trick, coded as a string.

TaggedValues
documentation Given a trick, the AI will update the locally stored history, as well as check for any players that have become partners.

2. SetActiveRules
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void

Parameters

2.1. strActiveRules

Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The coded set of active rules.

TaggedValues
documentation Given a BSTR containing the set of rules that should be active, this fucntion will decode and update the Rules class.

3. GetActiveRules

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Returns a BSTR containing all of the active rules.

4. AnalyzeGame

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Tells the AI that the current game has ended, and to search the stored history for canidates for updating the neural net.

5. UpdateNetwork

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Tells the neural net to attempt to update its weight matricies with data gathered from the last games.

6. SelectCard

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: Card
Parameters

6.1. strPlayerHand
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards in the player's hand, coded as a string

6.2. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player requesting help.

6.3. TrickNumber
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation Not used, but kept in the interface for compatability.

6.4. strPlayedCards
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards that have been played so far, coded as a string.

TaggedValues
documentation Given the current player's ID and hand, the system will attempt to select the best card to play. First, the system will check for any rules that may apply to the situation. If that fails, the system will attempt to use the neural net to select the best card.

7. IsLegal
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

7.1. strPlayedCard
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The card the player is attempting to play, coded as a string.

7.2. strPlayerHand

Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards in the player's hand, coded as a string.

7.3. strPlayedCards
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards played in the trick so far, coded as a string.

7.4. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player in question.

TaggedValues
documentation Given the card a player is attempting to play, the AI will check the game's history to see if that is a legal move.

8. DetermineWin
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters
8.1. strTrick
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The trick, coded as a string.

TaggedValues
documentation Based on the game history, returns the ID of the player that has won the hand.

9. DeterminePoints
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters
9.1. strTrick
Type Expression: String

Kind: in
DefaultValue:
TaggedValues
documentation The trick, coded as a string.

TaggedValues
documentation Returns the number of points in a trick.

10. CheckPartners
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters
10.1. CheckPlayer
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player the system is supposed to check for partnering.

TaggedValues
documentation Given a player ID, the AI will check to see if that player is partnered with another one based on the active rules.

11. UpdatePartnerCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
11.1. strNewCard
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The card to update to, coded as a string.

TaggedValues
documentation In a non-cutthroat game that has a picker, this will update the card that causes a partnering. Used in case the default card is in the picker's hand.

12. AutoUpdatePartnerCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation In a non-cutthroat game that has a picker, this will automatically update the card that causes a partnering to the next highest trump card. Used in case the default card is in the picker's hand. Returns the updated card.

13. GetPartnerCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Returns the card that causes a partnering, coded as a string.

14. SetCutthroat
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
14.1. Cutthroat
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation The new cutthroat value.

TaggedValues
documentation Determines if this is a cutthroat game (picker stands alone)

15. GetCutthroat
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns a bool indicating if this is a cutthroat game.

16. ClearHistory
Visibility: public
OwnerScope: instance

IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears all of the stored game data.

17. SetFirstPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
17.1. FirstPlayer
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the first player in the game

TaggedValues
documentation Sets the ID number of the first player in the game.

18. GetFirstPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the ID number of the first player in the game.

19. SetPicker
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
19.1. Picker
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the Picker.

TaggedValues
documentation Sets the ID number of the picker. (-1 if there is no picker)

20. GetPicker
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the ID number of the picker (-1 if there is no picker)

21. BuryCards
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
Parameters
21.1. strPlayerHand
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards in the player's hand, and the blind, coded as a string.

21.2. BlindSize
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The number of cards in the blind.

TaggedValues
documentation Given the cards in a player's hand and the blind, and the number of cards in the blind, this function will return a Hand object coded as a string, that contains which cards to bury. Note that this function does not remove those cards from the hand.

22. PickerCheck
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:

Return Type Expression: bool
Parameters

22.1. BlindSize
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The number of cards in the blind.

22.2. strPlayerHand
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The player's hand, coded as a string.

TaggedValues
documentation Given the number of cards in the blind, and the cards in the player's hand, this function will return a boolean indicating if the player should take the blind.

23. SetGameType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

23.1. GameType
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The type of game being played.

TaggedValues
documentation Sets the type of game.

24. GetGameType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the type of game being played.

25. SetPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential

Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

25.1. Players
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The number of players in the game.

TaggedValues
documentation Sets the number of players in the game.

26. GetPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the number of players in the game.

27. Fuzzify
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters

27.1. BaseCard
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The card to find a fuzzy value for.

TaggedValues
documentation Given a card object, this function will convert it to a fuzzy value based on the point and power values of the cards.

28. DeFuzzify
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
Parameters

28.1. BaseValue
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The fuzzy value to translate back into a card.

TaggedValues
documentation Given a fuzzy value, returns the card that most closely matches it.

29. ResequenceHand
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
29.1. BaseHand
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation A reference to the hand that will be reseqenced.

TaggedValues
documentation Give a reference to a Hand object, this function will reorder the cards, starting with the player zero, based on the stored values of which player won the last trick.

30. InitPartners
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Creates the intial parters array before a game starts.

31. LoadValues
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Fills the m_CardValues vector with values from a data file.

32. SaveValues
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Saves the data in m_GameHistory to a file.

33. ClearValues
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Frees all of the memory allocated in the m_CardValues vector.

34. UpdateTrump
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
34.1. UpdateHand
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation A reference to the Hand object that is to be updated,

TaggedValues
documentation Given a reference to a Hand object, this function will update the Trump data member in the Card class. The data member in the Card class is not valid until this function is called.

35. PlayerCards
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:

Return Type Expression: int
TaggedValues
documentation Returns the number of cards each player has in their hand, based on the m_PlayerCount and m_CutthroatGame parameters.

36. TrickPoints
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
Parameters
36.1. Trick
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The set of cards to determine points for.

TaggedValues
documentation Determines the number of points in a hand.

37. FindValues
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Values
Parameters
37.1. Parameter1
Type Expression: Card
Kind: in
DefaultValue:

TaggedValues
documentation Given a card, this function finds that card's corresponding Value class.

38. SequenceHistory
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters
38.1. Player
Type Expression: long
Kind: in
DefaultValue:

TaggedValues
documentation The player number to sequence.

38.2. Trick
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The trick number to sequence

TaggedValues
documentation Given a trick and a player, this function will create the proper sequence for input to the neural net. Used to create training data files.

39. SequenceCurrent
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters
39.1. PlayedCards
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The cards that have been played in the trick.

39.2. PlayerHand
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The cards in the player's hand.

39.3. PlayerID
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The ID of the player to sequence.

TaggedValues
documentation This function will create a sequence for the neural network based on a game in progress. Used to create the sequence for SelectCard.

Associations
1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier

En En Network

AssociationEnds

1.1. End7
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End8
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

En En Values

AssociationEnds

2.1. End15
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2.2. End16
IsOrdered: No

Navigable: No
Aggregation: none
Multiplicity: 32
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

3. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Rules

AssociationEnds

3.1. End9
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

3.2. End10
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues
documentation The main AI class. This class will interface with the game engine.

Classifier: Condition

FullPath: UML System 1::Static Model::Top Package::Condition
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No

IsActive: No

Attributes

1. m_CheckType

Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: CheckType
TaggedValues
documentation Defines what the condition is to check: a card, suit, value, or trump

2. m_CheckWhere

Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: CheckWhere
TaggedValues
documentation Defines where the condition is to check: the player's hand, the current trick, or the game history.

3. m_CheckTrick

Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The trick to check. A positive number represents an absolute trick (i.e. 1 = the first trick in the game). A negative number represents relative tricks (i.e. -2 = the previous two tricks). Zero represents all tricks in the history Only used if m_CheckWhere specifies the history.

4. m_CheckPlayer

Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The player to check. A positive number represents an absolute player (i.e. 1 = the first player in the trick). A negative number represents relative players (i.e. -2 = the previous two players). Zero represents all players. Only used if m_CheckWhere specifies the current trick.

5. m_CheckPartners

Visibility: private
InitialValue:
Multiplicity: 1

Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if the condition is to apply to the player's partners. Not used if partnering has not been determined.

6. m_CheckOpponents
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if the condition is to apply to the player's opponents. Not used if partnering has not been determined.

7. m_MatchCard
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Card
TaggedValues
documentation A specific card to match in the area defined by the other data members.

8. m_MatchSuit
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: char
TaggedValues
documentation A specific suit to match in the area defined by the other data members.

9. m_MatchValue
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation A specific value to match in the area defined by the other data members.

10. m_MatchTrump
Visibility: private
InitialValue:
Multiplicity: 1

Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation A specific trump value to match in the area defined by the other data members.

Operations

1. GetCheckType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: CheckType
TaggedValues
documentation Returns the what is being searched for by the condition: a specific card, a card value, a card suit, or a trump type

2. SetCheckType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
2.1. CheckType
Type Expression: CheckType
Kind: in
DefaultValue:
TaggedValues
documentation The new check type value for the condition.

TaggedValues
documentation Sets what the condition searches for to determine if the condition applies.

3. GetCheckWhere
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Returns where the condition checks: the player's hand, the cards in the trick, or the game history.

4. SetCheckWhere

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: CheckWhere
Parameters

4.1. NewCheckWhere
Type Expression: CheckWhere
Kind: in
DefaultValue:
TaggedValues
documentation The new CheckWhere value for the condition.

TaggedValues
documentation Sets where the condition checks.

5. GetCheckTrick
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the trick being searched. Only valid if CheckWhere specifies the history.

6. SetCheckTrick
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

6.1. CheckTrick
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The trick to be checked.

TaggedValues
documentation Sets the trick to be searched.

7. GetCheckPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential

Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the player to check. Only applies if CheckWhere specifies the trick.

8. SetCheckPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
8.1. CheckPlayer
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The player to match the condition criteria with.

TaggedValues
documentation Returns the player to be searched.

9. GetCheckPartners
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns true if only the player's partners are to be checked by the condition.

10. SetCheckPartners
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
10.1. CheckPartners
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation Sets whether the condtion should only check the player's partners for fulfillment of the

condition.

TaggedValues
documentation Sets if the condition only searches the player's partners.

11. GetCheckOpponents
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns truie if the condition is to check the player's opponents.

12. SetCheckOppoents
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
12.1. CheckOpponents
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation Sets whether the condtion should only check the player's opponents for fulfillment of the condition.

TaggedValues
documentation Sets if the condition checks only the player's opponents.

13. GetMatchCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
TaggedValues
documentation Returns the card to match if CheckType specifies a card.

14. SetMatchCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No

IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

14.1. MatchCard
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The card that must be matched for the criteria to be fulfilled.

TaggedValues
documentation Sets the card to check for if CheckType specifies a card to search for.

15. GetMatchSuit
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: char
TaggedValues
documentation Returns the suit to match if CheckType specifies a suit.

16. SetMatchSuit
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

16.1. MatchSuit
Type Expression: char
Kind: in
DefaultValue:
TaggedValues
documentation Sets the suit that must be matched for fulfillment of the condition.

TaggedValues
documentation Sets the suit to check for if CheckType specifies that the condition is fulfilled by a suit.

17. GetMatchValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential

Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the valiue to search for if CheckType specifies a value.

18. SetMatchValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
18.1. MatchValue
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation Sets the value that must be matched for fulfillment of the condition.

TaggedValues
documentation Sets the value to search for if CheckType specifies matching a value.

19. GetMatchTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns the if the condtion should search for a (non) trump card if CheckType specifies trump.

20. SetMatchTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
20.1. MatchTrump
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation Sets what type of trump value must be found

for fulfillment of the Condition

TaggedValues
documentation Sets the trump type the condtion should search for if CheckType specifies trump.

21. SaveCondition
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
21.1. outfile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output stream to save the condition to.

TaggedValues
documentation Saves the condition to a file stream

22. LoadCondition
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
22.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input stream to read the condition from.

TaggedValues
documentation Loads the conditions from a file stream

23. ConditionApplies
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
23.1. PlayerID

Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the current player.

23.2. LeadPlayer
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player that lead the trick.

23.3. PlayerHand
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The cards in the hand of the current player.

23.4. PlayedCards
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The cards played so far in the trick.

23.5. Partners
Type Expression: vector<int>
Kind: in
DefaultValue:
TaggedValues
documentation Contains the partner matches for the game (eg, all 0s are partners, all 1s are partners).

23.6. GameHistory
Type Expression: vector<Hand>
Kind: in
DefaultValue:
TaggedValues
documentation The history of the game so far.

TaggedValues
documentation Returns true if the condition has been satisfied by the current game conditions.

Associations

1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Rule

AssociationEnds

1.1. End6

IsOrdered:	No
Navigable:	No
Aggregation:	none
Multiplicity:	*
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

1.2. End5

IsOrdered:	No
Navigable:	No
Aggregation:	composite
Multiplicity:	1
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

TaggedValues

documentation	The conditions that must be met for a rule to be applied.

Classifier: Event

FullPath: UML System 1::Static Model::Top Package::Event

Visibility:	public
Stereotype:	implementation class
IsRoot:	No
IsLeaf:	No
IsAbstract:	No
IsActive:	No

Attributes

1. m_EventType

Visibility:	protected
InitialValue:	et_SetLegalCard
Multiplicity:	1
Changeable:	none
OwnerScope:	instance
TargetScope:	instance
Type Expression:	EvemtTypes

TaggedValues

documentation	The type of event indicated: Set legal card, suit, value, or trump, or set partner card, suit, value, or trump.

2. m_PlayCard

Visibility:	protected
InitialValue:	NULL
Multiplicity:	1
Changeable:	none
OwnerScope:	instance
TargetScope:	instance
Type Expression:	Card

TaggedValues

documentation	If the event type specifies a card, check this value

3. m_PlayValue
Visibility: protected
InitialValue: 1
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation Contains the value for legality/partners.

4. m_PlaySuit
Visibility: protected
InitialValue: "
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: char
TaggedValues
documentation Contains the suit for legality/partners.

5. m_PlayTrump
Visibility: protected
InitialValue: false
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation Contains the trump (true/false) for legality/partnering

Operations

1. LoadEvent
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
1.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input stream the events are read from.

TaggedValues
documentation Loads the events from a file stream.

2. SaveEvent
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
2.1. outfile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output file stream.

TaggedValues
documentation Saves the event to an output stream.

3. GetEventType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: EvemtTypes
TaggedValues
documentation Returns what type of event this is.

4. SetEventType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
4.1. EventType
Type Expression: EvemtTypes
Kind: in
DefaultValue:
TaggedValues
documentation The new type for the event.

TaggedValues
documentation Sets what type of event this is.

5. GetPlayCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
TaggedValues

documentation Gets the card used for determining legality/partnering.

6. SetPlayCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
6.1. PlayCard
Type Expression: Card
Kind: in
DefaultValue:

TaggedValues
documentation Sets the card used for determining legality/partnering.

7. GetPlayValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Gets the value (A-K) used for determining legality/partnering.

8. SetPlayValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
8.1. PlayValue
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation Sets the value of card to be played when the event is fired.

TaggedValues
documentation Sets the value (A-K) used for determining legality/partnering.

9. GetPlayTrump
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Gets the trump type (true/false) used for determining legality/partnering.

10. SetPlayTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
10.1. PlayTrump
Type Expression: Boolean
Kind: in
DefaultValue:
TaggedValues
documentation Sets the trump value to be played when the event is fired.

TaggedValues
documentation Gets the trump type (true/false) used for determining legality/partnering.

11. SetPlaySuit
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
11.1. PlaySuit
Type Expression: char
Kind: in
DefaultValue:
TaggedValues
documentation Sets the suit to be played when the event is fired.

TaggedValues
documentation Sets the suit (H, S, D, C) used for determining legality/partnering.

12. GetPlaySuit
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: char
TaggedValues
documentation Gets the suit (H, S, D, C) used for determining legality/partnering.

Associations
1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Rule

AssociationEnds
1.1. End4
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End3
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues
documentation The events that occur if a rule is applied.

Classifier: Layer

FullPath: UML System 1::Static Model::Top Package::Layer
Visibility: public
Stereotype:
IsRoot: No

IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. LayerNeurons
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation A list of all neurons in the layer.

2. NeuronCount
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The number of neurons in the layer.

3. Error
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The error in this layer of the network.

Operations

1. GetNeuronCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Gets the number of neurons in this layer.

2. SetNeruonCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters

2.1. LayerSize
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The number of neurons in the layer.

TaggedValues
documentation Sets the number of neurons in the layer.

3. LoadWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: ifstream
Parameters
3.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input stream that contains the weights.

TaggedValues
documentation Loads the weights for the layer in from a file.

4. SaveWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: ofstream
Parameters
4.1. outrile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output stream the values should be saved to.

TaggedValues
documentation Saves the weights in the layer to a file.

5. CalculateError
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:

Return Type Expression:
TaggedValues
documentation Calculates the error for this layer of the network.

6. GetNeuron
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Neuron
Parameters
6.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the neuron.

TaggedValues
documentation Returns a neuron from the layer, give a valid index number.

7. UpdateWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters
7.1. LearnRate
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The learning rate parameter for the network.

TaggedValues
documentation Updates the weights in this layer.

8. GetError
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the error in this layer.

TaggedValues
documentation Defines one layer in the neural network.

Classifier: Network

FullPath: UML System 1::Static Model::Top Package::Network

Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_InputLayer
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation A vector containing all of the neurons in the input layer.

2. m_HiddenLayerOne
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation A vector containing all of the neurons in the first hidden layer.

3. m_HiddenLayerTwo
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation A vector containing all of the neurons in the second hidden layer.

4. m_HiddenLayerThree
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation A vector containing all of the neurons in the third hidden layer of the network.

5. m_OutputLayer
Visibility: private

InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Neuron
TaggedValues
documentation The output layer of the network

6. m_InputNeurons
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The number of neurons in the input layer.

7. m_LearningRate
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The learning rate of the network. The smaller this is, the better the network's learning, but the more time it will take.

8. m_Players
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The number of players in the game. Used to determine the architecture of the network.

9. m_Picker
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if there is a picker in the game. Used to determine the network's architecture.

10. m_NetworkBuild
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none

OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation Boolean flag that determines if the network has been built.

Operations

1. GetLearningRate
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the learning rate of the network

2. SetLearningRate
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
2.1. NewRate
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The new learning rate of the network.

TaggedValues
documentation Sets the learning rate of the network.

3. GetPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the number of players in the game.

4. SetPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: void
Parameters

4.1. PlayerCount
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The new number of players in the game.

TaggedValues
documentation Sets the number of players in the game. Used to determine the structure of the network.

5. FeedForward
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters

5.1. InVector
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation An array of fuzzy values formatted for the neural net.

TaggedValues
documentation Performs feedforward calculations with the network (given an input vector, all of the neurons and weights are evaluated). Since all of the networks will have one output neuron, the return of this function will be the output of that neuron.

6. LoadWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Loads in the weights for a network from a file. The file is determined by GetWeightFile().

7. SaveWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Saves the current weights to a file. the filename is determined by GetWeightFile()

8. Train
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
8.1. DataFile
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation A file containing the training data.

TaggedValues
documentation Tells the network to attempt to update its weight matricies based on stored games.

9. GetPicker
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns true if there was a picker in the game, false if there was not.

10. SetPicker
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
10.1. Picker
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation Sets whether there was a picker in the game or not.

TaggedValues
documentation Sets if there was a picker in the game.

11. NetworkBuilt
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns true/false if the network has been built or not.

12. ClearNetwork
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Frees all memory associated with the neural network.

13. BuildNetwork
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Builds the neural network. Does not create or load any weights.

14. RandomizeNetwork
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Randomizes the connections in the network.

15. FeedForward
Visibility: public
OwnerScope: instance
IsPolymorphic: No

IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters

15.1. InVector
Type Expression: vector<double>
Kind: in
DefaultValue:
TaggedValues
documentation The input values to the network.

TaggedValues
documentation Given an input vector, performs the feedforward calculations.

16. GetInputNeurons
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the number of input neurons the network expects given the number of players and the presence of a picker.

17. GetWeightFile
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Returns the file that contains the weights for this network architecture.

Associations

1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Neuron

AssociationEnds

1.1. End13

IsOrdered:	No
Navigable:	No
Aggregation:	composite
Multiplicity:	1
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

1.2. End14

IsOrdered:	No
Navigable:	No
Aggregation:	none
Multiplicity:	*
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

2. contains

FullPath:	UML System 1::Static Model::Top Package::contains
NameReadingDirection:	<none specified>
EndCount:	2

	Fr	To	Related Classifier
	En	En	AI

AssociationEnds

2.1. End8

IsOrdered:	No
Navigable:	No
Aggregation:	none
Multiplicity:	1
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

2.2. End7

IsOrdered:	No
Navigable:	No
Aggregation:	composite
Multiplicity:	1
Changeable:	none
TargetScope:	instance
Visibility:	private

Classifier List: <None>

TaggedValues
documentation Implementation of a neural network.

Classifier: Neuron

FullPath: UML System 1::Static Model::Top Package::Neuron
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_Output
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The calculated output of the neuron. Note that this value is not adjusted for any weights.

2. m_Connections
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Integer
TaggedValues
documentation The number of connections the neuron has (equal to the number of neurons in the next layer).

3. m_Weights
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The weights of the connections from this neuron to every neuron in the next layer.

4. m_Input
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues

documentation The input into the neuron.

5. m_Error
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: double
TaggedValues
documentation The error in the neuron.

Operations

1. ClearInput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears the stored inputs of the neuron.

2. AddInput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
2.1. NewInput
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The input to add to the neuron.

TaggedValues
documentation Adds a value to the input value stored by the neuron.

3. GetInput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the current intput into the neuron.

4. SetConnectionNumber
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
4.1. NeuronConnections
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The new number of connections.

TaggedValues
documentation Sets the number of connections this neuron has to the next layer.

5. GetConnectionNumber
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the number of connections between this neuron and the next layer.

6. LoadWeights
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
6.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The file stream containing the weights for this neuron.

TaggedValues
documentation Loads the weights for this neuron from an input stream.

7. SaveWeights
Visibility: public
OwnerScope: instance

IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

7.1. outfile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output stream the weights are to be written to.

TaggedValues
documentation Saves the current weights for the neuron ro an output stream.

8. RandomizeConnections
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Used to create random weights for the neural network.

9. ClearOutput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears the output stored by the neuron.

10. CalculateOutput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

10.1. PassThrough
Type Expression: bool
Kind: in
DefaultValue:

TaggedValues
documentation Determines if the input of the neuron is pased through or if the sigmoid function is used to calculate the output.

TaggedValues
documentation Calculates the ouput of the neuron. Determines if the sigmoid function is applied to the calculations.

11. GetOutput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Gets the output of the neuron, without the application of the weights.

12. GetWeightedOutput
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters
12.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The weight to use to calculate the output.

TaggedValues
documentation Returns the output of the neuron weighted with the appropriate connection.

13. GetError
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the error in this neuron.

14. SetError
Visibility: public
OwnerScope: instance
IsPolymorphic: No

IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

14.1. NeuronError
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The error in the neuron.

TaggedValues
documentation Sets the error in this neuron.

15. ClearError
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears the error stored by the neuron.

16. UpdateConnectionValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

16.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the connection to update.

16.2. UpdateValue
Type Expression: double
Kind: in
DefaultValue:
TaggedValues
documentation The value to add to the specified connection.

TaggedValues
documentation Adds a given value to one of the connections of the neuron.

17. GetConnectionValue
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters

17.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the weight to return.

TaggedValues
documentation Returns the value of one of the weights maintained by the neuron.

Associations

1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Network

AssociationEnds

1.1. End14
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End13
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues

documentation Definition of a neuron present in the AI's neural network.

Classifier: Rule

FullPath: UML System 1::Static Model::Top Package::Rule

Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_IsActive
Visibility: protected
InitialValue: true
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation Boolean flag that determines if the rule will be used for this game.

2. m_RuleConditions
Visibility: protected
InitialValue: NULL
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Condition
TaggedValues
documentation Any conditions that must be met for this rule to be applied.

3. m_RuleEvents
Visibility: protected
InitialValue: NULL
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Event
TaggedValues
documentation All events that will take place if the conditions are met.

Operations

1. ActivateRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:
TaggedValues
documentation Makes the rule active for the game (sets IsActive to

true).

2. DeactivateRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:
TaggedValues
documentation Deactivates the rule for the game (sets IsActive to false).

3. IsActive
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns the status of the rule.

4. LoadRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
4.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The file stream to load the rule from.

TaggedValues
documentation Given a file stream, load the next rule into memory. Returns a bool to indicate success.

5. SaveRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

5.1. outfile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output file to save the rule to.

TaggedValues
documentation Given a file stream, save this rule to that stream. Returns a bool to indicate success.

6. ConditionCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the number of conditions for the rule.

7. EventCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the number of events for the rule.

8. AddCondition
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters
8.1. NewCondition
Type Expression: Condition
Kind: in
DefaultValue:
TaggedValues
documentation The condition to be added to the rule.

TaggedValues
documentation Adds a new condition to the rule, and returns the index number of the condition.

9. AddEvent
Visibility: public
OwnerScope: instance

IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
Parameters

9.1. NewEvent
Type Expression: Event
Kind: in
DefaultValue:
TaggedValues
documentation The new event to add to the rule.

TaggedValues
documentation Adds a new event to the rule, and returns an index number to the event.

10. GetCondition
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Condition
Parameters

10.1. Index
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the condition.

TaggedValues
documentation Returns a condition of the rule, given a zero based index number.

11. GetEvent
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Event
TaggedValues
documentation Returns one of the events of the rule given an zero based index number

12. DeleteCondition
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: void
Parameters

12.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the condition to delete.

TaggedValues
documentation Removes a condition from the rule.

13. DeleteEvent
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

13.1. Index
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the event to remove from the rule.

TaggedValues
documentation Removes one of the events from the rule.

14. ClearConditions
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears all of the conditions from the rule.

15. ClearEvents
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears all of the events from the rule.

Associations

1. contains

FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Condition

AssociationEnds

1.1. End5

IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End6

IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2. contains

FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Rules

AssociationEnds

2.1. End2

IsOrdered: No
Navigable: No

Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2.2. End1
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

3. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	Event

AssociationEnds

3.1. End3
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

3.2. End4
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues
documentation A rule used either to determine legal plays or determine which players are partners

Classifier: Rules

FullPath: UML System 1::Static Model::Top Package::Rules

Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No

Attributes

1. m_RuleList
Visibility: private
InitialValue:
Multiplicity: *
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Rule
TaggedValues
documentation A list of all of the rules in the system.

Operations

1. RuleCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
TaggedValues
documentation Returns the number of rules in the system.

2. GetRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Rule
Parameters
2.1. Index
Type Expression: long
Kind: in
DefaultValue:

TaggedValues
documentation Given an index, returns a reference to a rule object.

3. LoadRules
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:

MethodBody:
Return Type Expression: void
TaggedValues
documentation Loads the rules in from a data file.

4. SaveRules
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Saves the rules to a data file.

5. AddRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: double
Parameters
5.1. NewRule
Type Expression: Rule
Kind: in
DefaultValue:
TaggedValues
documentation The rule to add to the system.

TaggedValues
documentation Adds a rule to the collection.

6. DeleteRule
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
6.1. Index
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the rule to remove.

TaggedValues
documentation Removes a rule from the collection.

7. ClearRules
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Clears all of the rules in the system.

Associations

1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	AI

AssociationEnds

1.1. End10
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End9
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier

En En Rule

AssociationEnds

2.1. End1
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2.2. End2
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues
documentation A collection of all of the rules present on this system.

Classifier: Values

FullPath: UML System 1::Static Model::Top Package::Values
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_Points
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The points this card is worth.

2. m_Trump
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool

TaggedValues
documentation Specifies if this is a trump card or not.

3. m_Rank
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation Specifies the rank of this card.

4. m_Card
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Card
TaggedValues
documentation The card these values refer to.

Operations

1. SetCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

1.1. NewCard
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The new card these values refer to.

TaggedValues
documentation Sets the card these values refer to.

2. GetCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
TaggedValues
documentation Returns the card these values refer to.

3. SetPoints
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

3.1. CardPoints
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The number of points this card is worth.

TaggedValues
documentation Sets the number of points the card is worth.

4. GetPoints
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Gets the number of points the card is worth.

5. SetTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

5.1. IsTrump
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation True/false if the card is/is not a trump card.

TaggedValues
documentation Sets whether or not this is a trump card.

6. GetTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:

Return Type Expression: bool
TaggedValues
documentation Returns if the card is a trump card or not.

7. SetRank
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
7.1. CardRank
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The rank of the card.

TaggedValues
documentation Sets the rank (power) of the card.

8. GetRank
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the rank of the card.

9. SaveEvent
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
9.1. outfile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output stream the value should be saved to.

10. LoadEvent
Visibility: public
OwnerScope: instance

IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

10.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input stream the values are read from

Associations

1. contains
FullPath: UML System 1::Static Model::Top Package::contains
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	AI

AssociationEnds

1.1. End16
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: 32
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End15
IsOrdered: No
Navigable: No
Aggregation: composite
Multiplicity: 1
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

TaggedValues
documentation A class that maintains information about the rank and value of a card.

DataType: CheckType

FullPath: UML System 1::Static Model::Top Package::CheckType

Visibility:	public
IsRoot:	No
IsLeaf:	No
IsAbstract:	No
TaggedValues	
documentation	Defines what a condition is to check: a card, suit, value, or trump

DataType: CheckWhere

FullPath: UML System 1::Static Model::Top Package::CheckWhere

Visibility:	public
IsRoot:	No
IsLeaf:	No
IsAbstract:	No
TaggedValues	
documentation	Defines where a condition is to check.

DataType: EvemtTypes

FullPath: UML System 1::Static Model::Top Package::EvemtTypes

Visibility:	public
IsRoot:	No
IsLeaf:	No
IsAbstract:	No
TaggedValues	
documentation	Specifies what an event does: sets a legal card, suit, value, or trump type, or specifies what card, suit, value, or trump type results in a partnering.

UML Static Structure Report

MuttonHelp

Model: Static Model

FullPath: UML System 1::Static Model

Visibility:	public
Stereotype:	
IsRoot:	No
IsLeaf:	No
IsAbstract:	No

Contains Package(s): UML System 1::Static Model::Top Package

Model Element Statistic Summary

Number of interfaces: 2
Number of classes: 3
Number of attributes: 11
Number of parameters: 127
Number of operations: 36
Number of methods: 35
Number of associations: 2
Number of association ends: 4
Number of usages: 1
Number of tagged values: 290
Number of packages: 1

Package: Top Package

Contains Package(s): <None>

Contains Subsystem(s): <None>

FullPath: UML System 1::Static Model::Top Package
Visibility: public
Stereotype: topLevelPackage
IsRoot: No
IsLeaf: No
IsAbstract: No

Classifier: CHelp

FullPath: UML System 1::Static Model::Top Package::CHelp
Visibility: public
Stereotype: metaclass
IsRoot: Yes
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_pCharacter
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: CMuttonCharacter*

2. m_pHTMLHelp
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none

OwnerScope: instance
TargetScope: instance
Type Expression: CMuttonHTMLHelp*

Operations

1. CHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

2. ~CHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

3. GetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

3.1. pHelpFile
Type Expression: BSTR*
Kind: out
DefaultValue:

4. GetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

4.1. pWinType
Type Expression: BSTR*
Kind: out
DefaultValue:

5. Hide

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

5.1. Fast
Type Expression: VARIANT_BOOL
Kind: in
DefaultValue:

5.2. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

6. LoadCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

6.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: in
DefaultValue:

6.2. CharacterFile
Type Expression: BSTR
Kind: in
DefaultValue:

7. LookupKeyword
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

7.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

7.2. Keyword
Type Expression: BSTR
Kind: in

DefaultValue:

8. MoveTo
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

8.1. y
Type Expression: short(idl)
Kind: in
DefaultValue:

8.2. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

8.3. Speed
Type Expression: long(idl)
Kind: in
DefaultValue:

8.4. x
Type Expression: short(idl)
Kind: in
DefaultValue:

9. Play
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

9.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

9.2. AnimationName
Type Expression: enmAnimation
Kind: in
DefaultValue:

10. SetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

10.1. HelpFile
Type Expression: BSTR
Kind: in
DefaultValue:

11. SetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

11.1. WinType
Type Expression: BSTR
Kind: in
DefaultValue:

12. ShowCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

12.1. Fast
Type Expression: VARIANT_BOOL
Kind: in
DefaultValue:

12.2. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

13. ShowHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

13.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

14. Speak
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

14.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

14.2. Text
Type Expression: BSTR
Kind: in
DefaultValue:

15. Stop
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

15.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

16. Think
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: HRESULT
Parameters

16.1. pSuccess
Type Expression: VARIANT_BOOL*
Kind: out
DefaultValue:

16.2. Text

Type Expression: BSTR
Kind: in
DefaultValue:

Associations

1. Association1
FullPath: UML System 1::Static Model::Top Package::Association1
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	CMuttonHTMLHelp

AssociationEnds

1.1. End1
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End2
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

2. Association2
FullPath: UML System 1::Static Model::Top Package::Association2
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	CMuttonCharacter

AssociationEnds

2.1. End3

IsOrdered:	No
Navigable:	No
Aggregation:	none
Multiplicity:	*
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

2.2. End4

IsOrdered:	No
Navigable:	No
Aggregation:	none
Multiplicity:	*
Changeable:	none
TargetScope:	instance
Visibility:	private
Classifier List:	<None>

Classifier: CMuttonCharacter

FullPath: UML System 1::Static Model::Top Package::CMuttonCharacter

Visibility:	public
Stereotype:	implementation class
IsRoot:	No
IsLeaf:	No
IsAbstract:	No
IsActive:	No

Attributes

1. m_lCharID

Visibility:	private
InitialValue:	
Multiplicity:	1
Changeable:	none
OwnerScope:	instance
TargetScope:	instance
Type Expression:	long

2. m_lRequestID

Visibility:	private
InitialValue:	
Multiplicity:	1
Changeable:	none
OwnerScope:	instance
TargetScope:	instance
Type Expression:	long

3. m_pAgent

Visibility:	private
InitialValue:	
Multiplicity:	1
Changeable:	none
OwnerScope:	instance
TargetScope:	instance

Type Expression: IAgentEx*

4. m_pCharacter
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: IAgentCharacterEx*

Operations

1. CMuttonCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

2. ~CMuttonCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

3. GetAnimationName
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: BSTR
Parameters

3.1. enmAnimName
Type Expression: enmAnimation
Kind: in
DefaultValue:

4. Hide
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

4.1. bFast
Type Expression: bool
Kind: in
DefaultValue:

5. LoadCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

5.1. wsCharacterFile
Type Expression: wstring(idl)
Kind: in
DefaultValue:

6. MoveTo
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

6.1. lSpeed
Type Expression: long
Kind: in
DefaultValue:

6.2. y
Type Expression: short
Kind: in
DefaultValue:

6.3. x
Type Expression: short
Kind: in
DefaultValue:

7. Play
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

7.1. AnimationName

Type Expression: enmAnimation
Kind: in
DefaultValue:

8. Show
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

8.1. bFast
Type Expression: bool
Kind: in
DefaultValue:

9. Speak
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

9.1. wsText
Type Expression: wstring(idl)
Kind: in
DefaultValue:

10. Stop
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool

11. Think
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

11.1. wsText

Type Expression: wstring(idl)
Kind: in
DefaultValue:

Dependents

	Rel. Name	Rel. Type	Related Element
	Usage1	Usage	UML System 1::C++ Data Types::enmAnimation

1. Usage1
Description:

Associations

1. Association2
FullPath: UML System 1::Static Model::Top Package::Association2
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	CHelp

AssociationEnds

1.1. End4
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End3
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

Classifier: CMuttonHTMLHelp

FullPath: UML System 1::Static Model::Top Package::CMuttonHTMLHelp

Visibility: public
Stereotype: implementation class
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No

Attributes

1. m_dwCookie
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long

2. m_hInstanceHTMLHelp
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: HINSTANCE

3. m_pfnHTMLHelp
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: PFNHTMLHELP

4. m_wsHelpFile
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: wstring(idl)

5. m_wsWinType
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: wstring(idl)

Operations

1. CMuttonHTMLHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression:

2. ~CMuttonHTMLHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

3. GetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: wstring(idl)

4. GetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: wstring(idl)

5. LookupKeyword
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
5.1. wsKeyword
Type Expression: wstring(idl)
Kind: in
DefaultValue:

6. SetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:

MethodBody:
Return Type Expression: wstring(idl)
Parameters
6.1. wsHelpFile
Type Expression: wstring(idl)
Kind: in
DefaultValue:

7. SetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: wstring(idl)
Parameters
7.1. wsWinType
Type Expression: wstring(idl)
Kind: in
DefaultValue:

8. Show
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:

Associations
1. Association1
FullPath: UML System 1::Static Model::Top Package::Association1
NameReadingDirection: <none specified>
EndCount: 2

	Fr	To	Related Classifier
	En	En	CHelp

AssociationEnds
1.1. End2
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *

Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

1.2. End1
IsOrdered: No
Navigable: No
Aggregation: none
Multiplicity: *
Changeable: none
TargetScope: instance
Visibility: private
Classifier List: <None>

Interface: IHelp
FullPath: UML System 1::Static Model::Top Package::IHelp
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
Operations
1. GetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Return Type Expression: wstring(idl)

Interface: Interface1
FullPath: UML System 1::Static Model::Top Package::Interface1
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No

UML Static Structure Report

Core

Model: Static Model

FullPath: UML System 1::Static Model
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
Contains Package(s): UML System 1::Static Model::Top Package

Model Element Statistic Summary

Number of classes: 9
Number of datatypes: 9
Number of attributes: 148
Number of parameters: 342
Number of operations: 104
Number of methods: 104
Number of tagged values: 1060
Number of links: 6
Number of link ends: 12
Number of packages: 1

Package: Top Package

Contains Package(s): <None>

Contains Subsystem(s): <None>

FullPath: UML System 1::Static Model::Top Package
Visibility: public
Stereotype: topLevelPackage
IsRoot: No
IsLeaf: No
IsAbstract: No

Classifier: AIPlayer

FullPath: UML System 1::Static Model::Top Package::AIPlayer
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Operations

1. GetPlayerType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:

Return Type Expression:
TaggedValues
documentation Overridden from the Player class.

Classifier: Card

FullPath: UML System 1::Static Model::Top Package::Card
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_Suit
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: char
TaggedValues
documentation The suit of the card (C)lubs, (S)pades, (H)earts, or (D)iamonds

2. m_Value
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: int
TaggedValues
documentation The value of the card, from 1 = Ace to 13 = King

3. m_DispCard
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: CardTypes
TaggedValues
documentation The equivalent CardTypes value. Used so this class can match up with the Card control.

4. m_Trump
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if this is a trump card.

Operations

1. Clone
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
TaggedValues
documentation Returns a deep copy of the card.

2. operator=
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
Parameters
2.1. rhs
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The base card to copy.

TaggedValues
documentation Sets two Card object equal to each other.

3. operator==
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
3.1. rhs
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The card to compare

TaggedValues
documentation Returns true if the card's suit and value are equal

4. operator!=
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

4.1. rhs
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The card value to compare

TaggedValues
documentation Returns false if the card's suit and values are equal.

5. GetSuit
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: char
TaggedValues
documentation Returns the suit of the card.

6. SetSuit
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

6.1. newSuit
Type Expression: char
Kind: in
DefaultValue:
TaggedValues
documentation The new suit of the card.

TaggedValues
documentation Sets the suit of the card.

7. GetValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues

documentation Returns the value of the card.

8. SetValue
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
8.1. neValue
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The new value of the card.

TaggedValues
documentation Sets the value of the card.

9. GetCardType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: CardTypes
TaggedValues
documentation Returns the type of the card.

10. SetCardType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
10.1. NewCard
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The new type of the card.

TaggedValues
documentation Sets the type of card.

11. GetTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns the trump value of the card.

12. SetTrump
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
12.1. Trump
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The trump value of the card.

TaggedValues
documentation Sets the trump value of the card.

13. LoadCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
13.1. infile
Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input file stream

TaggedValues
documentation Loads a card in from a filestream

14. SaveCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
14.1. outfile

Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output file stream.

15. EncodeCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Encodes the card into a string.

16. DecodeCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
16.1. CodedCard
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The string to decode.

TaggedValues
documentation Given a coded string, decodes it into a card value

TaggedValues
documentation Implementation

Classifier: Core

FullPath: UML System 1::Static Model::Top Package::Core
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes
1. m_pHelp
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance

Type Expression: IHelp
TaggedValues
documentation A pointer to the Help interface

2. m_pComm
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: ICommunications
TaggedValues
documentation A pointer to the communications interface.

3. m_pAI
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: IMuttonAI
TaggedValues
documentation A pointer to the AI interface

4. m_bPickerWouldBeAlone
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if the picker is partnered with no one else (Cutthroat game)

5. m_bPickCardsEnabled
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation Enables the player cards to be set by the user interface.

6. m_bPlayCardEncabled
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues

documentation Enables the user interface to determine the card to be played.

7. m_nPlayerCurrentlyWaiting
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID of the player waiting for a message from the user interface.

8. m_nLeadPlayer
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: unsigned short
TaggedValues
documentation The player who lead the trick.

9. m_Blinds
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Hand
TaggedValues
documentation The cards in the blind.

10. m_Trick
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Hand
TaggedValues
documentation The cards currently in the trick.

11. m_nSetCardMessagesReturned
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: short

12. m_enmGameState
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none

OwnerScope: instance
TargetScope: instance
Type Expression: enmGameState
TaggedValues
documentation The current state of the game.

13. m_lNameIndex
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: unsigned long
TaggedValues
documentation The last internal name assigned to an AI player.

14. m_nNumOfPlayers
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: unsigned short
TaggedValues
documentation The number of players in the game.

15. m_VulgarityLevel
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: enmVulgarityLevel
TaggedValues
documentation The level of vulgarity the AI can use.

16. m_vecPlayers
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: vector<CPlayer*>
TaggedValues
documentation A pointer to all of the player objects maintained by the core. This is a list of all players connected to the game, they do not have to be playing.

17. m_vecPlayerInGame
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: vector<CPlayer*>
TaggedValues
documentation A vector of pointers to all of the players currently

in the game.

18. m_vecNames
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: vector<wstring>
TaggedValues
documentation A vector of names that can be assigned to AI players.

Operations

1. Event_PlayerAdded
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

1.1. PlayerName
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The name of the new player.

1.2. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number assigned to the player.

TaggedValues
documentation Event raised when a player is added to the game.

2. Event_RecieveMessage
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

2.1. FromPlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID of the player that sent the message

2.2. ToPlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID of the player that is to recieve the message.

2.3. MessageType
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The type of message being sent.

2.4. Message
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The message being sent.

TaggedValues
documentation Recieves a game message from another player.

3. Event_PlayerDropped
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
3.1. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the dropped player.

TaggedValues
documentation An event sink that drops a player from the game.

4. Event_BecameGameHost
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation An event sink that occurs when the game host is determined.

5. Event_SessionDropped
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation An event sink that occurs when a session is dropped.

6. PickCards
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
6.1. BuriedCards
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The cards the player has buried

6.2. Alone
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation True if the picker is alone (cutthroat game)

TaggedValues
documentation Called from the interface to pick the cards.

7. PlayCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
7.1. Card
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The card the player has played.

TaggedValues
documentation Called by the interface to set the cards a human player has played.

8. get_CurrentTrick

Visibility:	public
OwnerScope:	instance
IsPolymorphic:	No
IsQuery:	No
CallConcurrency:	sequential
Specification:	
Language:	
MethodBody:	
Return Type Expression:	String
TaggedValues	
documentation	Returns the cards played as a string.

9. get_GetBlinds

Visibility:	public
OwnerScope:	instance
IsPolymorphic:	No
IsQuery:	No
CallConcurrency:	sequential
Specification:	
Language:	
MethodBody:	
Return Type Expression:	String
TaggedValues	
documentation	Returns the cards in the blind as a string.

10. get_GameState

Visibility:	public
OwnerScope:	instance
IsPolymorphic:	No
IsQuery:	No
CallConcurrency:	sequential
Specification:	
Language:	
MethodBody:	
Return Type Expression:	enmGameState
TaggedValues	
documentation	Returns the current state the game is in.

11. get_NumConnectedPlayers

Visibility:	public
OwnerScope:	instance
IsPolymorphic:	No
IsQuery:	No
CallConcurrency:	sequential
Specification:	
Language:	
MethodBody:	
Return Type Expression:	void
TaggedValues	
documentation	Returns the number of players connected to the game. Note that not all connected players are playing the game.

12. DestoryRound

Visibility:	public
OwnerScope:	instance
IsPolymorphic:	No
IsQuery:	No
CallConcurrency:	sequential
Specification:	

Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Stops the current round.

13. get_GetPlayerID
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
Parameters
13.1. Index
Type Expression: short
Kind: in
DefaultValue:
TaggedValues
documentation The AI player ID

TaggedValues
documentation Given the ID number the AI uses, returns the ID number the communication uses.

14. BeginRound
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: udtDealInfo
TaggedValues
documentation Begins a round in the game.

15. GetPlayerInfo
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: udtGenPlayerInfo
Parameters
15.1. ID
Type Expression: short
Kind: in
DefaultValue:
TaggedValues
documentation The player ID to return information about.

TaggedValues
documentation Given the index of the player, returns information about that player.

16. AcknowlegeGame
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Called by remote playes to indicate that they are ready for the game to begin. Returns a boolean indicating success.

17. get_IsHost
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns true if this instance of the game is the host.

18. get_NumOfPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: short
TaggedValues
documentation Returns the number of players in the game.

19. put_NumOfPlayers
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
19.1. NewVal
Type Expression: short
Kind: in
DefaultValue:
TaggedValues
documentation The number of people playing.

TaggedValues
documentation Sets the number of people playing the game.

20. Add2ndSession
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Adds a secondary communications session to the game. Allows people using different connections to play together. Returns the ID number of the new session.

21. DropSession
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
21.1. SessionID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the session to drop.

TaggedValues
documentation Drops a communications session.

22. AddSession
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
Parameters
22.1. MakeHost
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation Determines if the player is becoming a host or connecting to an existing game.

22.2. GameName
Type Expression: String
Kind: in
DefaultValue:
TaggedValues

documentation The name of the game to create or connect to.

TaggedValues
documentation Adds a primary communication session to the game.

23. AIFillIn
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
23.1. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the dropped player.

TaggedValues
documentation If a player is dropped, and AI player will fill in for them until the round is over.

24. get_VulgartiyLevel
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns how vulgar the AI will be.

25. put_VulgarityLevel
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
25.1. newVal
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The new vulgarity level of the AI.

TaggedValues
documentation Sets how vulgar the AI will be.

26. EndGame
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Ends the current game.

27. SelectRules
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Selects the rules that will be in use for this game.

28. DropPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
28.1. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player to drop.

TaggedValues
documentation Drops a player from the game.

29. BeginGame
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Begins a game

30. SendChatMessage
Visibility: public

OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

30.1. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID of the player sending the message.

30.2. Message
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The chat message to send.

TaggedValues
documentation Sends a chat message to all other players in the game.

31. AddPlayer
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

31.1. Type
Type Expression: enmPlayerType
Kind: in
DefaultValue:
TaggedValues
documentation The type of player: AI, remote, or Hotseat to add.

31.2. PlayerID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player.

31.3. Name
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The name of the player.

TaggedValues
documentation Adds a player to an existing game.

32. FinalRelease
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Performs all initialization for the DLL.

33. FinalConstruct
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Performs all cleanup for the DLL.

34. FindPlayer
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Player
Parameters
34.1. CommID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The communications ID of the player.

TaggedValues
documentation Given the player's communications ID, returns a refernce to the player object.

35. GetPlayerName
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: enmVulgarityLevel
Parameters
35.1. Level
Type Expression: enmVulgarityLevel

Kind: in
DefaultValue:
TaggedValues
documentation The vulgarity level to use in determining the player's name.

TaggedValues
documentation Gets a name for an AI player.

36. SetPickedCards
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
36.1. PartnerCard
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The card that will determine a partnering.

36.2. BuriedCards
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The cards the player is burying.

36.3. PlayerPos
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The position of the player that is the picker.

36.4. Cutthroat
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation Sets if this is a cutthroat game.

TaggedValues
documentation Sets the cards that were buried, and by who.

37. FindGamePlayer
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
Parameters

37.1. CommID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The communications ID.

TaggedValues
documentation Given the communications ID, returns the AI ID.

38. SetGameState
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
38.1. eNewVal
Type Expression: enmGameState
Kind: in
DefaultValue:
TaggedValues
documentation The new state the game is in.

TaggedValues
documentation Sets the state the game is currently in.

39. GetGameState
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: enmGameState
TaggedValues
documentation Returns the state the game is currently in.

40. InitRound
Visibility: private
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Initializes the round.

41. Stop
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Stops the MS Agent character. Returns true if successful.

42. Think
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
42.1. Text
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The text to display.

TaggedValues
documentation Begins the Agent character's 'thought' animation.

43. MoveTo
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
43.1. x
Type Expression: short
Kind: in
DefaultValue:
TaggedValues
documentation The x coordinate to move to.

43.2. y
Type Expression: short
Kind: in
DefaultValue:
TaggedValues
documentation The y coordinate to move to.

43.3. Speed
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The speed the agent moves at.

TaggedValues
documentation Moves the Agent to the specified loction. Returns true if successful.

44. ShowCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
44.1. Fast
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation True if agent is supposed to skip the show animation.

TaggedValues
documentation Shows the selected character.

45. Hide
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
45.1. Fast
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation True if the Agent should skip the Hide animation.

TaggedValues
documentation hides the Agent. Returns true if successful.

46. LoadCharacter
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
46.1. CharacterFile
Type Expression: String
Kind: in

DefaultValue:
TaggedValues
documentation The file the Agent character is stored in.

TaggedValues
documentation Loads a specific Agent character

47. Speak
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
47.1. Text
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The text to speak.

TaggedValues
documentation Has the Agent character speak the text.

48. Play
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
48.1. AnimationName
Type Expression: enmAnimation
Kind: in
DefaultValue:

TaggedValues
documentation Plays an Agent animation.

49. ShowHelp
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Shows the HTML help file.

50. LookupKeyword
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters
50.1. Keywork
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The word to look up.

TaggedValues
documentation Looks up a keyword in the help file.

51. GetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Returns the CHTML help file being used.

52. SetHelpFile
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
52.1. HelpFile
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The help file to use.

TaggedValues
documentation Sets the CHTML help file to use.

53. GetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Gets the window type of the HTML help viewer.

54. SetWinType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
54.1. WinType
Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The new window type for the HTML help viewer.

TaggedValues
documentation Sets the window type for the HTML help viewer.

TaggedValues
documentation The core of the game. Handles all messaging between the communications, AI, help, and interface, and manages communications between multiple instances of the game.

Classifier: Hand

FullPath: UML System 1::Static Model::Top Package::Hand
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes
1. m_vecCards
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: vector<CCard*>
TaggedValues
documentation The cards maintained by the Hand class.

Operations
1. Clone
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No

CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Hand
TaggedValues
documentation Returns a deep copy of the Hand object.

2. operator+=
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Hand
Parameters
2.1. rhs
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The hand to append to the current hand.

TaggedValues
documentation Appends a Hand to the existing Hand object.

3. operator=
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Hand
Parameters
3.1. rhs
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation The hand to copy

TaggedValues
documentation Sets the Hand object equal to another one.

4. AdCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters

4.1. pNewCard
Type Expression: Card
Kind: in
DefaultValue:
TaggedValues
documentation The card to add to the hand.

TaggedValues
documentation Adds a card to the hand.

5. RemoveCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
5.1. iCardNum
Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the card to remove.

TaggedValues
documentation Removes a card from the hand.

6. GetCard
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
Parameters
6.1. iCardNum
Type Expression: <unspecified>
Kind: in
DefaultValue:
TaggedValues
documentation The index number of the card to return.

TaggedValues
documentation Returns a reference to a card in the hand.

7. RemoveCardType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void

Parameters

7.1. iCardNum

Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The CardType to remove from the hand.

TaggedValues
documentation Removes a card given the specific CardType

8. GetCardType

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Card
Parameters

8.1. iCardNum

Type Expression: int
Kind: in
DefaultValue:
TaggedValues
documentation The CardType value to find.

TaggedValues
documentation Returns a card object from the hand given the CardType value.

9. LoadHand

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
Parameters

9.1. inFile

Type Expression: ifstream
Kind: in
DefaultValue:
TaggedValues
documentation The input file stream

TaggedValues
documentation Loads a hand from a data file.

10. SaveHand

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:

MethodBody:
Return Type Expression: bool
Parameters
10.1. outFile
Type Expression: ofstream
Kind: in
DefaultValue:
TaggedValues
documentation The output file stream

TaggedValues
documentation Saves a hand to an output file.

11. ClearHand
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Removes all of the cards from the Hand object.

12. CardCount
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: int
TaggedValues
documentation Returns the number of cards in the hand.

13. EncodeHand
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: String
TaggedValues
documentation Encodes the hand into a string.

14. DecodeHand
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void

Parameters

14.1. CodedHand

Type Expression: String
Kind: in
DefaultValue:
TaggedValues
documentation The hand to decode.

TaggedValues
documentation Decodes a hand from a string.

TaggedValues
documentation Container for a group of cards.

Classifier: HotSeatPlayer

FullPath: UML System 1::Static Model::Top Package::HotSeatPlayer
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Operations

1. GetPlayerType

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:
TaggedValues
documentation Overridden from the Players class

TaggedValues
documentation Maintains data for the player at the computer.

Classifier: Player

FullPath: UML System 1::Static Model::Top Package::Player
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. m_wsName

Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: wstring(idl)

2. m_lID
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long

3. m_pHand
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Hand

4. m_pCardsTaken
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: Hand

5. m_lPointsTaken;
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long

6. m_lScore
Visibility: protected
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long

7. m_bHasAcked
Visibility: private
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool

Operations

1. IsPicker
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns true if the player is the picker.

2. SetHasAcked
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
2.1. bVal
Type Expression: bool
Kind: in
DefaultValue:
TaggedValues
documentation The state of the player's acknowledgement.

TaggedValues
documentation Set if the player has acknowledged the game.

3. GetHasAcked
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: bool
TaggedValues
documentation Returns the status of the player's acknowledgement of the game.

4. SetScore
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
4.1. lScore
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The player's score

TaggedValues

documentation Sets the player's score after the game is over.

5. GetScore
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the player's score.

6. SetPointsTaken
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
6.1. lPointsTaken
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The number of points taken.

TaggedValues
documentation Sets the points the player has taken from a trick.

7. GetPointsTaken
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the number of points the player has taken.

8. SetCardsTaken
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
8.1. pCardsTaken
Type Expression: Hand

Kind: in
DefaultValue:
TaggedValues
documentation The cards the player has taken.

TaggedValues
documentation Sets the cards taken by the player in a trick.

9. GetCardsTaken
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Hand
TaggedValues
documentation Returns the cards the player has taken.

10. SetHand
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
10.1. pHand
Type Expression: Hand
Kind: in
DefaultValue:
TaggedValues
documentation Determines the cards in the player's hand.

TaggedValues
documentation Sets the player's hand.

11. GetHand
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: Hand
TaggedValues
documentation Returns the player's hand.

12. SetName
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:

Language:
MethodBody:
Return Type Expression: void
Parameters
12.1. wsName
Type Expression: wstring(idl)
Kind: in
DefaultValue:
TaggedValues
documentation The name of the player.

TaggedValues
documentation Sets the player's name

13. GetName
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: wstring(idl)
TaggedValues
documentation Returns the name of the player.

14. SetID
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
Parameters
14.1. lID
Type Expression: long
Kind: in
DefaultValue:
TaggedValues
documentation The ID number of the player.

TaggedValues
documentation Sets the ID number of the player.

15. GeID
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: long
TaggedValues
documentation Returns the ID number of the player.

16. GetPlayerType

Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:
TaggedValues
documentation Returns what type of player this is.

17. EndGame
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression: void
TaggedValues
documentation Ends the game and clears the internal storage.

TaggedValues
documentation Class that handles the functionality for a basic Mutton player

Classifier: RemotePlayer

FullPath: UML System 1::Static Model::Top Package::RemotePlayer
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Operations

1. GetPlayerType
Visibility: public
OwnerScope: instance
IsPolymorphic: No
IsQuery: No
CallConcurrency: sequential
Specification:
Language:
MethodBody:
Return Type Expression:
TaggedValues
documentation Overidden from the Player class.

TaggedValues
documentation A class that maintains information about a remote player.

Classifier: udtDealInfo

FullPath: UML System 1::Static Model::Top Package::udtDealInfo
Visibility: public
Stereotype:

IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. Blinds
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in the blind

2. Player0
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 0

3. Hand0
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 0s hand.

4. Player1
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 1

5. Hand1
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 1s hand.

6. Player2

Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 2

7. Hand2
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 2s hand.

8. Player3
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 3

9. Hand3
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 3s hand.

10. Player4
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 4

11. Hand4
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance

Type Expression: String
TaggedValues
documentation The cards in player 4s hand.

12. Player5
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 5

13. Hand5
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 5s hand.

14. Player6
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 6

15. Hand6
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 6s hand.

16. Player7
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The ID number of player 7

17. Hand7
Visibility: public

InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards in player 7s hand.

TaggedValues
documentation Information about the dealt cards.

Classifier: udtGenPlayerInfo

FullPath: UML System 1::Static Model::Top Package::udtGenPlayerInfo
Visibility: public
Stereotype:
IsRoot: No
IsLeaf: No
IsAbstract: No
IsActive: No
Attributes

1. Name
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The name of the player.

2. CanDrop
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation Determines if the player can be dropped.

3. OwnerSession
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The communication session this player is under.

4. PlayerState
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance

TargetScope: instance
Type Expression: enmPlayerState
TaggedValues
documentation The state of the player.

5. PlayerType
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: enmPlayerType
TaggedValues
documentation The type of player.

6. PlayerPosition
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: short
TaggedValues
documentation The position of the player.

7. CurrentHand
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The player's current hand.

8. GameScore
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The player's current score.

9. Picker
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: bool
TaggedValues
documentation True if the player is the picker.

10. CardsTaken
Visibility: public

InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: String
TaggedValues
documentation The cards the player has taken.

11. PointsTaken
Visibility: public
InitialValue:
Multiplicity: 1
Changeable: none
OwnerScope: instance
TargetScope: instance
Type Expression: long
TaggedValues
documentation The amount of points the player has taken.

TaggedValues
documentation A structure conaining player information.

DataType: CardTypes

FullPath: UML System 1::Static Model::Top Package::CardTypes
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation Used for compatibility with the Card control.

DataType: enmGameState

FullPath: UML System 1::Static Model::Top Package::enmGameState
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation The state the game is currently in.

DataType: enmMsgType

FullPath: UML System 1::Static Model::Top Package::enmMsgType
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation Defines the type of message that is being sent.

DataType: enmPlayerState

FullPath: UML System 1::Static Model::Top Package::enmPlayerState
Visibility: public
IsRoot: No

IsLeaf: No
IsAbstract: No
TaggedValues
documentation The status of the player.

DataType: enmPlayerType
FullPath: UML System 1::Static Model::Top Package::enmPlayerType
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation The type of player: HotSeat, AI, or Remote

DataType: enmSpecialPlayerIDs
FullPath: UML System 1::Static Model::Top Package::enmSpecialPlayerIDs
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation An ID that defines host and all computers.

DataType: enmVulgarityLevel
FullPath: UML System 1::Static Model::Top Package::enmVulgarityLevel
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No
TaggedValues
documentation The vulgarity level the AI will use.

DataType: udtDealInfo
FullPath: UML System 1::Static Model::Top Package::udtDealInfo
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No

DataType: udtGenPlayerInfo
FullPath: UML System 1::Static Model::Top Package::udtGenPlayerInfo
Visibility: public
IsRoot: No
IsLeaf: No
IsAbstract: No

6 Test Plans

6.1 Card Control

6.1.1 Plan

Load multiple instances of the Card, Hand, and Deck objects onto a form. Link the hands to the deck. Add message boxes to test when events get fired. Change the properties of the objects from code and test all methods of the controls.

6.1.2 Results

The Card object behaved as was expected. The Hand object had trouble with not being able to use the display ratio to change the size of the control. I also found that a few of the events were not being fired, whether from the code not being implemented or not implemented properly. The deck object was able to deal out to the hands properly which was what was desired. The rest of the functionality of the Deck object worked properly as well.

6.2 GUI

6.2.1 Plan

When the GUI is finished, try using all possible functions to make sure that they work.

6.2.2 Results

Since not all functionality was implemented at this time, the GUI could be used only in a certain way. The human player must be the picker; there can be only one human player. Only a fourhanded game is supported.

6.3 HTML Help

6.3.1 Plan

Create a temporary COM object that wraps around the main CMuttonHTMLHelp class to ensure that the function calls operate correctly when called from outside of a COM object. After verifying that the functionality is correct, transplant the

CMuttonHTMLHelp class into the final COM object, MuttonHelp, as a private data member.

6.3.2 Results

The testing of the CMuttonHTMLHelp class was successful in providing the basic functionality required for this project including the display of the target compiled help file via the HTML Help Viewer and the lookup of specified keywords in the help file's index and search topics.

6.4 MS Agent

6.4.1 Plan

Testing of the CMuttonCharacter class is identical to that for the CMuttonHTMLHelp class—use a temporary COM wrapper to test the class external from the COM object (using Visual Basic as the client). After verification that the class works properly, integrate it into the final component, MuttonHelp, as a subordinate class.
With both classes—CMuttonHTMLHelp and CMuttonCharacter—implemented in the MuttonHelp component, test all functionality from outside of the component.

6.4.2 Results

Testing the class verified that the expected behavior was fully met. This behavior includes the ability to load a character file, manipulate the Agent character (show, hide, move, speak, think, animate), load the compiled help file in the HTML Help viewer, and provide searching abilities for the help file.

Testing the functionality for both CMuttonHTMLHelp and CMuttonCharacter externally from the MuttonHelp component served as a final check to ensure that the entire component works as expected which it did flawlessly.

6.5 Communications

6.5.1 Plan

The plan for testing included creating a subset of the functionality required and testing as many cases as possible to verify functionality. At first the component will be created to communicate with a sample Direct Play application provided in the DirectX SDK, and then a test app will be used (likely a simple VB app) to test each method provided by the component.

6.5.2 Results

The test plan worked well. Many issues were resolved quickly when they arise due to this test plan and the final component was able to be easily tested for a variety of cases.

6.6 Artificial Intelligence

6.6.1 Plan

Test all of the underlying classes to make sure that they compile, test all get/set functions, test all save/load functions, and all functions that add an object, remove and object, or get an object. Test the feedforward and training algorithims for the neural network. Test the ConditionSatisfied function in the Condition class. Test all of the functions in the top level class with a test VB app, and test the integration with the Core.

6.6.2 Results

All of the tests were performed successfully, with a few minor changes to the code. The rules functioned properly, the network performed the feedforward calculations, and successfully trained. All of the functions in the AI class worked properly in the VB test app, and were successfully used in the Core.

6.7 Core

6.7.1 Plan

The test plan for this component was to at first build some of the basic components required for some basic functionality. Once this is done, start writing a test application in Visual Basic to test the functionality of what was written. With this approach, functionality was added and then verified before moving on since much of the functionality requires some other functionality to work properly. First, Begin game and end game were developed, then start and destroy round, then determine picker, then start, play and end trick, and then end round.

6.7.2 Results

The test/implementation plan described above caught many subtle problems before they got out of hand in the testing cycle, providing for easy debugging. The test app also forced the creation of simple and intuitive interface since many of the functions were designed on the fly due to sudden design changes.

7 Time Management

7.1 Updated Schedule

Mike Krautkramer

Week	Objective	Time
1	Work with Nhat to write COM interface for the AI class	10 H
2	Write card value class and fuzzy mapping function	15 H
3	Write neuron, layer, and network class	20 H
4	Write rules, rule, condition, and event classes	25 H
5	Finish AI class	10 H
6	Write rules for the rules class	30 H
7	Test AI	20 H
8	Finish final report and presentation	20 H

Total: 160 Hours

Nhat Nguyen

Week	Objective	Time
1	Show Mike how to write COM object	10 H
2	Modify exist COM object to include HTML Help	5 H
3	Write CPlayer object	15 H
4	Write CPlayer object	15 H
5	Write CCore object	15 H

6	Write CCore object	15 H
7	Test both objects	20 H
8	Write final report and presentation	20 H

Total: 125 Hours

Adam Gritt

Week	Objective	Time
1	Work on Card control.	15 H
2	Work on Card control.	20 H
3	Work on Card control. Show Greg how to do COM programming.	25 H
4	Work on Card control. Create basic Help files.	20 H
5	Have all controls ready for implementation on the GUI Create GUI object with all controls.	20 H
6	Connect GUI to all other objects and start dry run testing Run testing and debug application	20 H
7	Run testing and debug application	20 H
8	Write Final report and presentation	20 H

Total: 175 Hours

Greg Schreiner

Week	Objective	Time
1	Finalize COM Interface	15 H
2	Investigate how to write a COM interface with Adam	15 H
3	Begin writing Communication Interface (host side)	20 H
4	Finish writing Communication Interface (client side)	20 H
5	Integrate Communications with rest of game & test (ATL COM wrapper)	20 H
6	Work with Mike for Rules	20 H
7	Test application	20 H
8	Write Final report and presentation	20 H

Total: 170 Hours

7.2 Activity Information

	Adam	Greg	Mike	Nhat
Analysis	0	42.9	21.5	2
Design	3	11.5	24.5	2
Implementation	120	155.8	111	58.5
Knowledge Exchange	0.28	19	3	1
Testing	5	11	0	9.17

7.3 Statistics

7.3.1 Spring Semester

	Total Hours	Expected	Percent of Expected
Adam	87.6	175	50.1%
Greg	193.6	170	113.9%
Mike	116.0	160	72.5%
Nhat	59.75	125	47.8%

7.3.2 Overall

	Total Hours
Adam	128.2
Greg	240.2
Mike	160.0
Nhat	72.76

8 Press Release

The Mutton Project

This senior design project members include Adam Gritt, Greg Schreiner, Mike Krautkramer, and Nhat Nguyen. This application is a computer version of a popular German card game, Sheepshead. The application consists of several COM objects that are used for wrapping the Direct Play API's for the Communications, an AI (Artificial Intelligence) object which uses a neural network to represent the non-human players, a help object which wraps MS Agent and the HTML Help API's, and a Card Control which the user interacts with visually. This game supports anywhere from three to eight players, multiple connection types simultaneously, and an intelligent AI..

9 User Documentation

See the compiled HTML file MuttonHelp.chm in Mutton Project.zip.

10 Conclusion

This concludes the requirements and modified design for a versatile Sheepshead game. Implementation took a large amount of time, but much of the testing and some of the resesign was included in the time. Overall, the project was a success. A multiplayer Sheepshead game was developed which supports a human player, computer (AI) players,

and players connected over several types of connections to different computers. Although we didn't get all of the requirements originally specified in this version of the game, it wouldn't be difficult to add those features and make this game solid and possibly marketable in the real world—or at least to a greater part of Wisconsin