

COMP/IT- 491: Capstone Preparation

Proposal Report

Fall 2023

**Artificial Intelligence Player**

**for Ninety-Nine Card Game**

Department of Computer Science

California State University, Channel Islands

Prepared by

Skyler Atchison

Adviser

Dr. Reza Abdolee

[1. Introduction and Background 4](#_Toc152666816)

[1.1: Problem Statement 4](#_Toc152666817)

[1.2: Previous Works 5](#_Toc152666818)

[1.3: Background 6](#_Toc152666819)

[1.3.1 Monte Carlo Simulation: 6](#_Toc152666820)

[1.3.2 Reinforcement Learning: 7](#_Toc152666821)

[2. Project Description 7](#_Toc152666822)

[2.1 Functional Specification 7](#_Toc152666823)

[2.1.1 Functions Performed 7](#_Toc152666824)

[2.1.2 Limitations and Restrictions 8](#_Toc152666825)

[2.1.3 User Inputs 8](#_Toc152666826)

[2.1.4 Outputs 8](#_Toc152666827)

[2.1.5 System Data Files 9](#_Toc152666828)

[2.2 Design Specification 9](#_Toc152666829)

[2.2.1 System Data Flow Diagrams 9](#_Toc152666830)

[2.2.2 System Data Dictionary 9](#_Toc152666831)

[2.2.3 Implementation Languages 10](#_Toc152666832)

[2.3 Implementation Plan 10](#_Toc152666833)

[2.3.1 Deliverable Items 10](#_Toc152666834)

[2.3.2 Milestone Identification and Completion Criteria 11](#_Toc152666835)

[2.3.3 Schedule 12](#_Toc152666836)

[References 13](#_Toc152666837)

# 1. Introduction and Background

Artificial Intelligence (AI) has captured the imagination of the public for well over a century, and the field has become a source of research and innovation over the past decade. AI has been harnessed for great practical use in many domains, including image processing, natural language generation, scientific analysis, and more. Many of these practical uses are built on techniques which were first applied to game-playing systems. There is a long history of AI developers using game-playing to study decision-making between competitive agents (Amit et al., 2006). The first game engines attempted by AI researchers played games of perfect information, such as Chess and Checkers. Games of partial information, such as card games, add another layer of complexity. This capstone project seeks to develop a game-playing engine for the card game Ninety-Nine, contributing to the speed and accuracy of decision-making systems under limited information environments.

## 1.1: Problem Statement

The specific problem that this capstone project seeks to address is how to achieve human-level play from an Artificial Intelligence at the game of Ninety-Nine. Ninety-Nine is a trick taking game created by the card game historian and inventor David Parlett and recognized as a complex and interesting game by the card-playing world (Parlett).

As a trick-taking game, it is similar to the games of Bridge, Hearts, and Skat which have been played by game engines in the past. However, it is sufficiently different in that it is an exact trick game, where extra points are earned for correctly predicting the number of tricks that will be taken. Another innovation in the gameplay is that the bid is made by discarding three cards from each player’s hand, which is another difficult decision the computer player will have to make.

This capstone project will involve the development of different variations of engines in competition against each other and human players, with experimentation to determine which approach results in the best engine, the engine that can win the majority of games against a variety of opponents. The results may be interesting for future developers of game-playing AI.

## 1.2: Previous Works

Game-playing is a unique field for Artificial Intelligence (AI) research, providing a variety of opportunities in a discrete, simplified environment. While many engines have been developed to play perfect-information games over the past several decades, high-level AIs in the domain of partial-information games have been a more recent phenomenon. This section reviews previous milestones in the history of game-playing AI.

Perfect-information games: Chess was one of the first games to be studied by programmers, culminating in the engine Deep Blue that was able to beat a grandmaster, Garry Kasparov, in a full chess match. Several decades later, a neural network-based system called AlphaZero defeated the top conventional chess engines at the time, and another AI by the same company also defeated the human champion at the game Go, which previous engines had been unsuccessful at.

Partial-information games: One of the first card-playing programs to rival human opponents was GIB, created to play the game Bridge (Ginsberg, 1999). This program used a Monte Carlo approach to decide which cards to play. Another team developed a similar AI to play the European game Skat (Kupferschmid et al., 2006). A more recent program developed by NukkAI incorporated reinforcement learning techniques to achieve super-human play as the declarer in Bridge (Cazenave et al., 2020).

## 1.3: Background

Artificial Intelligence game-playing engines have tended to fall into two broad categories, statistical analysis and self-play. Of the first category, Monte Carlo Simulation is a promising technique for games of partial information such as Ninety-Nine. Of the second category, Reinforcement Learning has seen much success at many different types of problems.

### 1.3.1 Monte Carlo Simulation:

Many card-playing systems utilize Monte Carlo Simulation during card play (Ginsberg, 1999). This ingenious technique involves two steps: First, a solving algorithm is able to quickly analyze the best play for a *perfect information* scenario, a version of the game where all players’ hands are known. To determine the best move for the *partial information* scenario, the computer randomly samples possible game states based on known information such as the cards in its own hand and the cards played so far, using the perfect information solver to determine what the best card to play would be for this distribution. It then uses these possible game states to vote on the best card to play in the partial information scenario, and plays that card.

### 1.3.2 Reinforcement Learning:

Another approach game engines take (especially more recent engines) is called Reinforcement Learning. This approach involves neural networks which are trained by attempting the process over and over. The AI will update its parameters on an unsuccessful attempt through gradient descent and backpropogation, resulting in the fine-tuning of the connection between its inputs (the environment of the program, such as the current state of the player’s hand and the cards played so far) and its chosen output (the action to take, such as playing a card). One specific version of this approach is the Actor-Critic method (Wang et al., 2017).

# 2. Project Description

## 2.1 Functional Specification

### 2.1.1 Functions Performed

Ninety-Nine AI will play the card game Ninety-Nine against two opponents. It will carry out this task as one of the players in a game interface. The two opponents may be human-controlled or simulated by a simple algorithm. If there is at least one human player, the human player(s) will be able to use an ASCII interface to select cards to play and see the current state of the board, such as the cards played by the opponents and the cards still held in hand.

In the case of a game using only simulated opponents, the user will play the role of a tester and will be able to instruct the program to run any number of games against the opponents with various settings. For example, the user will be able to select the specific search algorithms used by each computer player to compare their performance and merits.

### 2.1.2 Limitations and Restrictions

This primary purpose of this project is to test Artificial Intelligence algorithms and explore their strategies. However, as human testing is an important part of this process, a simple UI will be implemented.

Neural Networks and Machine Learning are also outside of the main goals of this project. Instead, the decisions of the AI will be purely based on algorithmic search.

### 2.1.3 User Inputs

The program can take user input from the command line or graphical interface. When the program is launched, the user (a tester) will be prompted to select whether each opponent will be controlled by human input or one of a few algorithms. If there are no human players, The user will also be prompted to select how many games will be played. If there are human players, the user will be prompted to select a card to play on each of their turns.

### 2.1.4 Outputs

In a game involving human players, the system will display the user’s current hand and opponent’s cards played after each turn to allow the human player to make a move. In a simulated game, the computer will display the entire state of the game after every turn for analysis. If the computer has been instructed to play several games, it will only write summary statistics to the result files such as the percentage of games won by each opponent.

### 2.1.5 System Data Files

The program will have no need for a database or input files, although if the analysis setting is enabled it will write data to text or JSON files to allow human testers to review its performance against various opponents after the games have concluded.

## 2.2 Design Specification

### 2.2.1 System Data Flow Diagrams

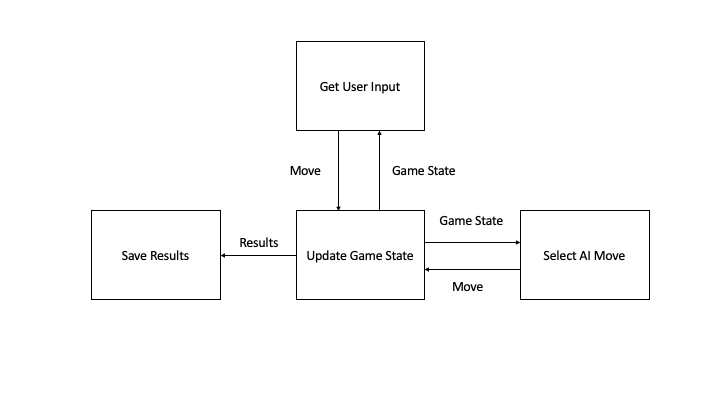


Figure 1: System Data Flow

The above data flow diagram shows the flow of data between the four key segments of the software.

### 2.2.2 System Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Data Item** | **Type** | **Description** |
| Game State | Python Object | An object representing the current state of the game, such as the cards in the player’s hands and the cards played so far. |
| Current Trick | Python Dictionary | An object representing the card chosen to play by each player |
| Results | JSON, TXT | Files with the results of the game for further analysis. |

### 2.2.3 Implementation Languages

The program will predominantly use Python as it is a standard language for AI research and comes with useful object-oriented functionality. Visuals will use the Pygame library as it is built for Python and is good for simple interfaces, or a higher-level Python library. The Pytest library will be used to build up a comprehensive test suite during the development process.

## 2.3 Implementation Plan

### 2.3.1 Deliverable Items

* User Interface: The repository will include a graphical interface for a user to play the game against the AI or run tests.
* AI: There will be at least three AI options for the computer players: An AI that only chooses random moves, an AI that uses a Monte Carlo Tree Search to choose moves, and an AI that uses a partial rules-based approach in its move selection. I will also include variations on these basic types to allow the user to experiment with different combinations of AI playing against each other and test hypotheses.
* System files: There will of course be modules and scripts for playing the game itself, along with a clean, comprehensive test suite.
* Documentation: This will be minimal, as the program and code will be clean and clear enough for programmers to understand without much effort. I will include a readme file for setup and dependency information.
* Sample runs: There will be tables and charts containing statistical information about how the various AI perform in trials against each other and against human players.

### 2.3.2 Milestone Identification and Completion Criteria

These are the milestones and criteria by which they will be judged complete:

* The Monte Carlo Tree Search algorithm is integrated with Ninety-Nine, allowing the computer to use it in play against human or random opponents. Criteria for completion is when the AI using this algorithm earns a better average score than computer players selecting random moves.
* The User Interface looks professional and is easy to use, both for individual games and simulation testing. Criteria for completion is when a human can play through several games without noticing visual bugs.
* All functionality has been thoroughly tested (this will be accomplished at the same time as the other milestones through a Test Driven Development process).
* Additional Artificial Intelligences have been implemented (this could include a neural-network based AI built on PyTorch if that proves to be achievable). This one is more flexible, but criteria for completion is when at least one additional AI using a different algorithm for choosing moves has been added to the possible AI players for the system.

### 2.3.3 Schedule

|  |  |  |
| --- | --- | --- |
| Item | Start | End |
| PyTorch game research | Jan 15 | Feb 5 |
| PyTorch game experimentation | Feb 5 | Feb 26 |
| Implementation of Additional AI | Feb 26 | Mar 18 |
| User Interface Improvement | Mar 18 | Apr 8 |
| Refactoring / Improvement of AI | Apr 8 | Apr 29 |
| Tests and presentation preparation | Apr 29 | May 13 |

## References

1. Amit, A., Markovitch, S. Learning to bid in bridge. *Mach Learn* **63**, 287–327 (2006). <https://doi-org.ezproxy.csuci.edu/10.1007/s10994-006-6225-2>
2. Cazenave, T., Ventos, V. (2021). The 𝛼𝜇 Search Algorithm for the Game of Bridge. In: Cazenave, T., Teytaud, O., Winands, M.H.M. (eds) Monte Carlo Search. MCS 2020. Communications in Computer and Information Science, vol 1379. Springer, Cham. <https://doi.org/10.1007/978-3-030-89453-5_1>
3. Ginsberg, Matthew L.. “GIB: Steps Toward an Expert-Level Bridge-Playing Program.” International Joint Conference on Artificial Intelligence (1999). <https://www.ijcai.org/Proceedings/99-1/Papers/084.pdf>
4. Kupferschmid, Sebastian and Malte Helmert. “A Skat Player Based on Monte-Carlo Simulation.” Computers and Games (2006). <https://ai.dmi.unibas.ch/papers/kupferschmid-helmert-cg2006.pdf>
5. Parlett, David. “Ninety-Nine.” *Parlett Games*, *www.parlettgames.uk/oricards/ninety9.html*. Accessed 19 Sept. 2023.
6. Wang, Ziyun et al. “Sample Efficient Actor-Critic with Experience Replay.” ArXiv abs/1611.01224 (2016): n. pag.