

# Kings of the West

Bryan Alaniz<sup>\*</sup><sup>1</sup>, Ricardo Balvanera<sup>†</sup><sup>2</sup>, and Fernando Castan<sup>‡</sup><sup>3</sup>

<sup>1</sup>University of Texas Rio Grande Valley

December 3, 2025

## Abstract

The purpose of this paper is to analyze Kings of the West, a board game designed and implemented by the current team of Bryan Alaniz, Ricardo Balvanera and Fernando Castan. Kings of the West is a turn-based board game where two players battle to eliminate the opponent’s King or all of the opponents fighters. The game is played on a 6 by 6 board with King, Gunslinger, and Bruiser pieces. The game utilizes the randomization of a 6 six-sided die, for the movement and attack damage to pieces. The formal model of the game state is  $S = (B, P, H, p)$ , where  $B$  is the board configuration,  $P$  is a 6 by 6 matrix of the position of all pieces in the game,  $H$  is the health of each piece, and  $p$  indicates the current player’s turn. We investigate the complexity of determining optimal play, demonstrating it is PSPACE-complete. Furthermore, we developed a heuristic-based Easy AI, a Monte Carlo Tree Search (MCTS) AI and an LLM-based AI; preliminary testing indicates the MCTS agent exhibits significantly stronger strategic play. This paper details the game’s mechanics, AI implementations, and the formal proof of its computational complexity

## 1 Introduction

### 1.1 Overview Of Kings of the West

Kings of the West is a two-player turn-based strategy game that is played on a  $6 \times 6$  grid, where two players compete to eliminate each other’s kings or defeat all opposing fighters, forcing the king to surrender. Each player controls a small team consisting of a King (mandatory), Gunslingers, and Bruisers, and each piece has its own unique health, damage output, and output range. Movement and attacks are determined by the roll of a six-sided die, which introduces a random factor, and this factor decides the distance pieces can move and how much damage they can deal to enemies. Because their positioning on the board and the unpredictability of the dice roll affect each turn, players have to continuously modify their strategy in order to be able to come out victorious. This combination of ever-changing tactics and unpredictable outcomes makes the game feel both strategic and uncertain.

### 1.2 Motivation

The motivation behind the design of Kings of the West stems from a desire to combine compelling aspects of various tactical board games we played and wrote reports on throughout the semester, alongside drawing significant inspiration from established video game franchises. Specifically, we looked to the strategic depth and grid-based combat of titles like *Fire Emblem* and the unique character abilities and emphasis on positional play found in *Mario + Rabbids: Kingdom Battle & Sparks of Hope*. Our group’s primary goal was to create a more strategic board game that challenged players through intricate combat and decision-making.

Initially, our concepts revolved around a purely deterministic combat system. However, after several renditions, we simplified the core combat mechanics and integrated an element of unpredictability through

---

<sup>\*</sup>bryan.alaniz02@utrgv.edu

<sup>†</sup>ricardo.balvanera01@utrgv.edu

<sup>‡</sup>fernando.castan01@utrgv.edu

a six-sided dice roll. This introduction of chance, reminiscent of many classic tabletop games, allowed us to maintain a high degree of strategic depth while ensuring dynamic and uncertain outcomes for each turn. This design choice prevents players from simply mirroring opponent moves and fosters adaptive planning. Ultimately, our aim was to develop a game that is relatively simple to understand, engaging and fun to play, offers quick sessions, and presents sufficient analytical interest.



Figure 1: Visual inspiration for Kings of the West, showcasing grid-based tactical gameplay from *Mario + Rabbids: Kingdom Battle* (top) and *Fire Emblem Echoes: Shadows of Valentia* (bottom). These titles influenced the game's strategic positioning and unique character abilities.

### 1.3 Overview Of Kings of the West

Kings of the West is a two-player turn-based strategy game played on a 6x6 grid. The objective is for players to eliminate the opponent's King or defeat all non-King fighter pieces (Gunslingers and Bruisers). Each player commands a small team of distinct units: a mandatory King, along with a selection of Gunslingers and Bruisers, each possessing unique health, damage output, and attack ranges.



Figure 2: The initial 6x6 grid-based battlefield of Kings of the West. Pieces include Kings (stars), Gunslingers (guns), and Bruisers (axes), each displaying their initial health points.

Movement and attacks are fundamentally influenced by the roll of a six-sided die, which introduces a pivotal element of chance. This random factor determines both the distance pieces can move and the damage dealt to enemies. Players must continuously adapt their strategies based on piece positioning and

the unpredictable nature of the dice roll to secure victory. This blend of evolving tactics and uncertain outcomes creates a game that is both strategic and engaging.

Each piece type in Kings of the West possesses unique attributes crucial for tactical play, as visually represented above:

- **King (Star Icon):** The primary target; eliminating the enemy King leads to an instant victory. Possesses moderate health (10 HP) and melee attack capabilities.
- **Gunslinger (Gun Icon):** Ranged combatants capable of attacking from a distance, with varying damage based on proximity. (Initial HP: 7)
- **Bruiser (Axe Icon):** Close-quarters specialists with high health (8 HP), excelling in direct melee engagements.

The six-sided die roll ( $D \in \{1, 2, 3, 4, 5, 6\}$ ) at the start of each player's turn dictates the type and strength of action they can perform:

Table 1: Effects of Dice Rolls in Kings of the West

Dice Roll	Action Effect
<b>1-3 (Movement Focus)</b>	The chosen piece can move up to the rolled number of adjacent (non-diagonal) steps. After moving, the piece may optionally attack an enemy within its range. If no move, attacks deal normal damage.
<b>4 (Double Damage Attack)</b>	The chosen piece can move up to 1 adjacent step and if in range the piece can attack and deal <b>2x damage</b> .
<b>5 (Triple Damage Attack)</b>	Similar to Roll 4: The chosen piece can move up to 1 adjacent step and if in range the piece can attack and deal <b>3x damage</b> .
<b>6 (Skip Turn)</b>	The chosen piece's turn is skipped. Control immediately passes to the other player.

### Gameplay Example: Movement and Attack

As illustrated in Figure 3, Player 1 has rolled a '1' on the six-sided die, which dictates the available actions for the turn: "Move up to 1 and optionally attack." In this state, Player 1 has a Gunslinger at grid position (3,1) with 5 HP and an enemy Bruiser at (4,1) with 8 HP. Player 1 could choose to move their Gunslinger one space to (3,0) or (3,2). If an attack is chosen, a short-range attack (Manhattan distance 1) from the Gunslinger typically deals 3 damage. The player's decision, whether to move, attack, or both, is made from a set of legal actions, all constrained by the current dice roll and the tactical board state.

## 1.4 Goals

The primary goals for the Kings of the West project encompassed several key areas:

- **Game Design and Implementation:** To design and implement a novel, turn-based tactical board game on a 6x6 grid that balances strategic depth with the excitement of randomized outcomes, ensuring a clear and engaging user experience.
- **AI Development:** To develop and integrate multiple Artificial Intelligence agents of varying complexities (heuristic and Monte Carlo Tree Search) capable of playing the game, providing challenging opponents for a single player.
- **Computational Complexity Analysis:** To formally analyze the computational complexity of "Generalized Kings of the West," specifically to determine if optimal play can be efficiently computed, and to categorize its complexity within established classes such as PSPACE.
- **Educational Exploration:** To apply concepts learned in Games & Computation, including game theory, AI algorithms, and complexity theory, to a practical game development and analysis project.

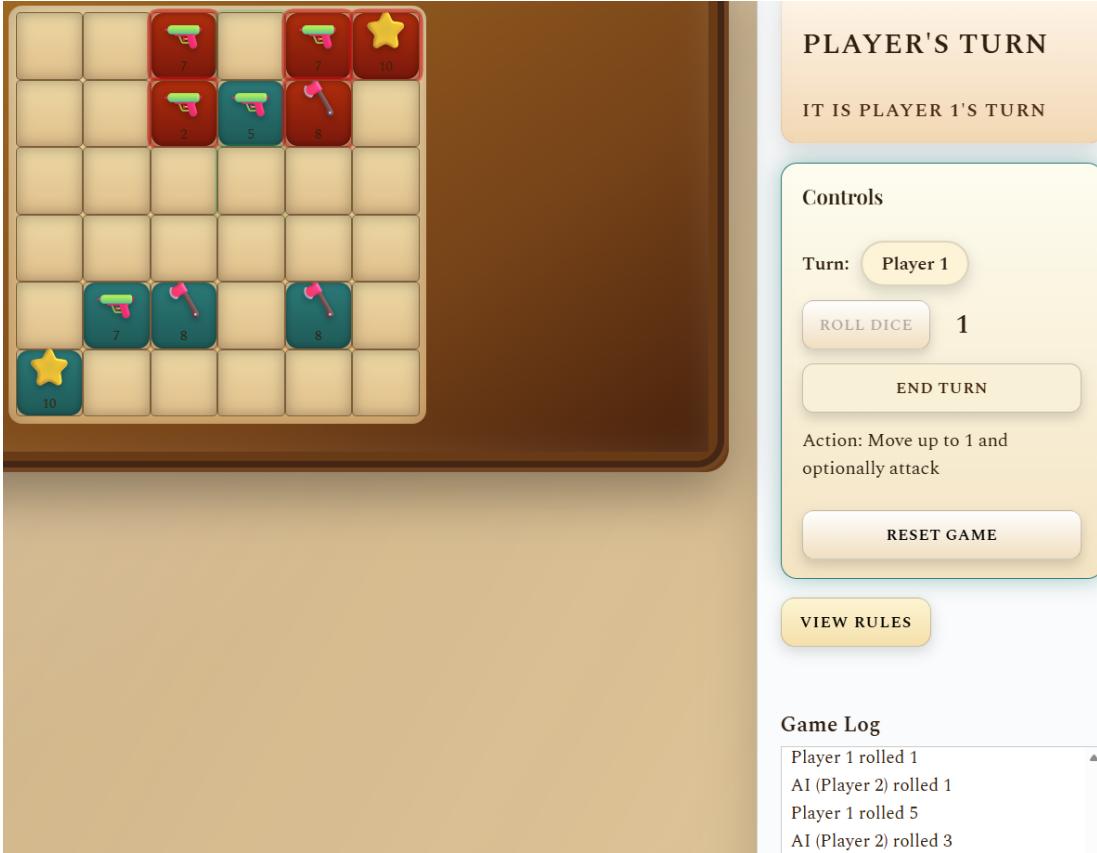


Figure 3: A gameplay example during Player 1’s turn. A dice roll of ‘1’ enables movement of up to one space and an optional attack. The UI clearly shows the turn information and available actions.

## 2 Background

### 2.1 Complexity

Many board games present significant challenges for analysis due to the vast number of possible moves and board configurations. Even games with seemingly simple rules, or those that have been simplified for computational study, can exhibit an exponential growth in their game trees, making it intractable to fully explore all possible states. For instance, while games like Pick-up-Bricks might possess manageable game trees, classics such as Chess or Checkers are known for their astronomically large state spaces, rendering the prediction of optimal strategies computationally demanding.

The field of computational game theory often classifies the complexity of determining optimal play within games. A common class for two-player, perfect information games (where both players know the full state of the game) is PSPACE (Polynomial Space). A problem is in PSPACE if it can be solved by a deterministic Turing machine using a polynomial amount of memory relative to the input size. Many board games, particularly their generalized versions played on an  $N \times M$  board, are proven to be PSPACE-complete. This means they are not only solvable within polynomial space, but are also “as hard as” any other problem in PSPACE, implying that finding an optimal strategy for them is computationally demanding, likely requiring exponential time. Kings of the West, with its distinct pieces, varied attacks, and the impact of dice rolls on numerous possible outcomes, shares many characteristics with games exhibiting high computational complexity.



Figure 4: Conceptual illustration of a game tree, showing the exponential growth of possible states from a single decision point.

## 2.2 Randomness

Another significant factor that elevates the complexity of game analysis is the introduction of randomness or chance elements. In contrast to deterministic games, where every move's outcome is perfectly predictable given the current state, games incorporating randomness—such as dice rolls or shuffled cards—introduce uncertainty. This unpredictability means that players cannot definitively foresee the exact outcome of their actions, but must instead reason about probabilities and expected values.

From a game theory perspective, such games are often modeled as stochastic games or games with imperfect information (though dice rolls are perfect information, the \*outcome\* is uncertain until rolled). Determining optimal play in these scenarios shifts from finding a guaranteed winning sequence of moves to identifying strategies that maximize a player's expected utility over time, or minimize the opponent's expected utility. This requires more sophisticated decision-making algorithms, often involving techniques like expectiminimax, which extends the traditional minimax algorithm to account for chance nodes in the game tree. The six-sided die roll in Kings of the West, which dictates movement range and attack damage multipliers, is a central random element that fundamentally shapes player strategy and decision-making, requiring adaptive rather than purely deterministic planning.

## 2.3 Game AI Techniques

The development of Artificial Intelligence (AI) for games has been a rich area of research, yielding diverse techniques for autonomous play. These techniques range from simple rule-based systems to complex learning algorithms.

### 2.3.1 Heuristic Search

Heuristic-based AIs rely on an evaluation function that assigns a score to a given game state, representing its desirability for a particular player. These AIs then search a limited portion of the game tree (often only one or two moves deep) and choose the action that leads to the highest-scoring immediate state according to the heuristic. While computationally inexpensive, the effectiveness of heuristic AIs is heavily dependent on the quality of their evaluation function, which often encapsulates domain-specific knowledge and can suffer from "horizon effect" (short-sightedness).

### 2.3.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is a more advanced, tree-search algorithm particularly well-suited for games with very large state spaces and a significant branching factor, including those with randomness. MCTS operates through four main phases: Selection, Expansion, Simulation (or Playout), and Backpropagation. It builds a game tree asynchronously by performing many random playouts from promising nodes, gradually accumulating statistics (wins/losses, visits) for each action. The Upper Confidence Bound 1 (UCT) formula

is commonly used during the Selection phase to balance exploration of less-visited nodes with exploitation of nodes that have historically led to good outcomes. This iterative process allows MCTS to discover good strategies without explicit evaluation functions for every state, making it robust in complex domains.

### 2.3.3 Related Work and Games

The computational complexity and AI approaches for grid-based tactical games are well-studied. Games like Chess [5] and Go [?] are prime examples of perfect information games where minimax search (and its derivatives like alpha-beta pruning) or MCTS have achieved superhuman performance. Games with elements of randomness, such as Backgammon [?], have also seen success with AI techniques that account for probabilistic outcomes. The complexity of generalized grid games, like Generalized Chess or Checkers, often falls into PSPACE or EXPTIME [?]. Our game, Kings of the West, draws inspiration from these tactical duel games, aiming to blend strategic positioning with tactical combat and the uncertainty introduced by dice rolls.

## 3 Formal Definitions

To rigorously analyze the computational complexity of Kings of the West, we first establish its formal definition. As the original game is played on a fixed  $6 \times 6$  board, its finite state space makes complexity analysis trivial. Therefore, we define the *Generalized Kings of the West*, which is played on an  $N \times M$  grid, where  $N, M \in \mathbb{Z}^+$ .

### 3.1 Game State

The state of the game at any given moment is formally represented as a tuple  $S = (B, P, H, p)$ , where:

- $B$ : The board configuration, an  $N \times M$  grid of cells.
- $P$ : The positions of all active pieces on the board.
- $H$ : The current health points of each piece.
- $p$ : The current player whose turn it is.

Each component is further defined below.

#### 3.1.1 Board Configuration ( $B$ )

The board  $B$  is an  $N \times M$  grid of cells. Each cell  $(r, c)$  where  $0 \leq r < N$  and  $0 \leq c < M$ , can either be empty or occupied by exactly one game piece.

#### 3.1.2 Pieces and Positions ( $P$ )

Each player,  $PL_1$  or  $PL_2$ , controls a set of five pieces. Each piece  $i$  is uniquely identified and characterized by:

- **ID:**  $id_i \in \{$
- **Owner:**  $owner_i \in \{PL_1, PL_2\}$ .
- **Type:**  $type_i \in \{\text{King}, \text{Gunslinger}, \text{Bruiser}\}$ . Each player has exactly one King.
- **Initial Health:**  $HP_{initial}(type_i)$ , defined as:
  - King: 10 HP
  - Gunslinger: 7 HP
  - Bruiser: 8 HP

- **Position:**  $(r_i, c_i)$ , where  $0 \leq r_i < N$  and  $0 \leq c_i < M$ . No two pieces can occupy the same cell.

The set  $P$  is a collection of tuples  $(id_i, owner_i, type_i, r_i, c_i)$  for all active pieces. Pieces are removed from  $P$  when their health  $HP_i \leq 0$ .

### 3.1.3 Health Points ( $H$ )

$H$  is a function  $H : \{id_i\} \rightarrow \mathbb{Z}^+$ , mapping each active piece's ID to its current health points.

### 3.1.4 Current Player ( $p$ )

$p \in \{PL_1, PL_2\}$  indicates which player is currently active and can make moves.

## 3.2 Game Setup and Initial State

The game begins in a setup phase.

- Each player selects 4 additional fighters (Gunslingers or Bruisers) to accompany their mandatory King, totaling 5 pieces per player. In the generalized version, the number of additional fighters may be parameterized, say  $F$ .
- Player  $PL_1$  places their King at  $(N - 1, 0)$ . Player  $PL_2$  places their King at  $(0, M - 1)$ .
- $PL_1$  places their remaining  $F$  fighters in their back two rows (rows  $N - 2, N - 1$ ).
- $PL_2$  places their remaining  $F$  fighters in their front two rows (rows  $0, 1$ ).
- An initial dice roll determines which player goes first, setting the initial value of  $p$ .

## 3.3 Turns and Actions

Each turn for the current player  $p$  consists of the following sequence:

1. **Dice Roll:** The player rolls a six-sided die,  $D \in \{1, 2, 3, 4, 5, 6\}$ .
2. **Action Selection:** Based on the die roll, the player chooses one of their active pieces and performs an action.
  - **Roll 1-3 (Movement Focus):** The chosen piece can move up to  $D$  adjacent (non-diagonal) steps to an unoccupied cell. After moving, the piece may optionally attack an enemy piece within its attack range from the new position. If no move is made, it can attack from its current position.
  - **Roll 4 (Double Damage Attack):** The chosen piece can move up to 1 adjacent step to an unoccupied cell. After moving, the piece *must* attack an enemy piece within its attack range (from the new position) with 2× damage. If no move is made, it can attack from its current position.
  - **Roll 5 (Triple Damage Attack):** Similar to Roll 4, but the attack deals 3× damage.
  - **Roll 6 (Skip Turn):** The chosen piece's turn is skipped. The current player's turn ends immediately.
3. **Turn End:** After performing an action (or skipping), control passes to the other player.

## 3.4 Movement Rules

A piece located at  $(r, c)$  can move to any unoccupied cell  $(r', c')$  such that the Manhattan distance  $|r - r'| + |c - c'| \leq$  steps, and all intermediate cells on the shortest orthogonal path are also unoccupied. Pieces cannot move through occupied cells.

### 3.5 Attack Rules

When a piece attacks, it targets an enemy piece within its specific range. Damage is calculated as follows:

- **Gunslinger:**

- Short Range (Manhattan distance = 1): 3 damage
- Long Range (Manhattan distance = 2 or 3): 2 damage

- **Bruiser:**

- Short Range (Manhattan distance = 1): 3 damage

- **King:**

- Melee/Short Range (Manhattan distance = 1): 3 damage

Damage is applied to the target's HP. If a piece's HP drops to 0 or below, it is eliminated from the game.

### 3.6 Win Condition

A player wins the game if they achieve either of the following conditions:

1. Eliminate the opponent's King.
2. Eliminate all of the opponent's non-King fighter pieces (Gunslingers and Bruisers).

## 4 Theory Aspects

. To Prove a Problem/Game is PSPACE-Complete we first have to prove two things:

- Membership in PSPACE
- Hardness (or a Reduction) in PSPACE

### 4.1 Proving PSPACE Membership

To prove that Generalized Kings of the West belongs to PSPACE, we must show that there exists an algorithm that can determine if Player 1 has a winning strategy using space polynomial to the input size  $N$ .

#### 4.1.1 Encoding the State

First, we analyze the memory required to store a single “snapshot” of the game. A state  $S$  consists of:

- The positions of  $k$  units. Each coordinate on an  $N \times N$  grid requires  $O(\log N)$  bits.
- The current HP of each unit (a small constant).
- The turn counter (up to  $N^3$ ), requiring  $O(\log N)$  bits.

Thus, the size of a single game state is polynomial:  $O(k \cdot \log N)$ .

#### 4.1.2 The Algorithm

We can determine the winner using a recursive Depth-First Search (DFS), equivalent to an Alternating Turing Machine (ATM). The algorithm functions as follows:

1. Check if the current state is terminal (King dead or Turn limit reached).
2. If it is Player 1's turn (Existential state), check if there exists *any* move leading to a winning state.
3. If it is Player 2's turn (Universal state), check if *all* possible counter-moves still lead to a Player 1 win.

### 4.1.3 Space Complexity Analysis

Since we use a DFS approach, we do not need to store the entire game tree. We only need to store the current path from the root (start) to the current leaf node.

$$Space_{total} = (\text{Max Depth}) \times (\text{Size per Frame}) \quad (1)$$

- The **Max Depth** is bounded by the turn limit  $T = N^3$ .
- The **Size per Frame** corresponds to one game state  $O(N^2)$ .

Therefore, the total space complexity is  $O(N^3 \cdot N^2) = O(N^5)$ . Since  $N^5$  is a polynomial, the problem is in **PSPACE**.

## 4.2 Proving PSPACE Hardness

To prove that the game is PSPACE-hard, we perform a reduction from **Nondeterministic Constraint Logic (NCL)**. NCL is a known PSPACE-complete problem involving a graph of edge-weights and nodes with flow constraints.

Hearn and Demaine (2004) [3] demonstrated that any game which can implement the following three gadgets is PSPACE-hard:

1. A Signal (Wire)
2. An OR Gate
3. An AND Gate

We construct these gadgets using the “Generalized Kings of the West” mechanics, specifically utilizing **Blocking** (units cannot pass through each other) and **Threat Zones** (a unit cannot move to a tile if it will immediately be destroyed).

### 4.2.1 Gadget 1: The Wire (Signal Propagation)

A wire will be defined as adjacent tiles that connect two points on the grid. A “True” signal is represented by a Gunslinger piece moving along this path. In the modified game mode, the gunslingers move automatically 1 tile per turn towards its destination.

### 4.2.2 Gadget 2: The OR Gate

- **Scenario A (1, 0):** A Gunslinger is deployed at Input A. It moves along its path and successfully passes through the intersection to the Output tile. (Result: 1)
- **Scenario B (0, 1):** A Gunslinger is deployed at Input B. It moves along its path and successfully passes through the intersection to the Output tile. (Result: 1)
- **Scenario C (1, 1):** Gunslingers are deployed at both Input A and Input B.
  - Since they arrive on the board at the same time but on different paths, they may reach the intersection tile simultaneously.
  - **Crucial Rule:** Unlike the XOR gate (where simultaneous entry causes a Standoff/Collision/0), for the OR gate, you must allow one of them to proceed. You can state a tie-breaker rule: *If two pieces arrive at the merging tile simultaneously, the piece from Path A always moves first, or they occupy the single tile in sequence.*
  - As long as at least one piece reaches the Output tile, the result is True. (Result: 1)

#### 4.2.3 Gadget 3: The AND Gate

- Two input paths converge onto a specific tile, being blocked by a 4 HP bruiser acting as a barricade.
- Input gunslingers (A and B) attack the “barricade” when they reach the end of the path. Reminder: Gunslingers deal 2 damage.
- The barricade is only destroyed if the cumulative damage reaches or exceeds 4 HP.

**Output:** The removal of the barricade allows a second piece (the “King”) to move from its starting position behind the barricade to the output tile.

## 5 Software Aspect

The software aspect of the game includes our GitHub repository [1] consisting of `app.css`, `app.js`, `index.html`, `easyai.js`, and `mcts.js` files. This collection of files implements the full game, from its visual presentation to its core logic and AI players.

## 6 AI Integration

### 6.1 Easy - Heuristic AI

### 6.2 Heuristic AI

The ‘easyai.js’ file implements a deterministic, heuristic-based greedy agent. To ensure an ”easier” and less strategically challenging opponent, this AI operates with a simplified perception and decision-making process, prioritizing immediate threats and incorporating elements of randomness. It explicitly avoids complex look-ahead or evaluation functions. The core logic of this simplified Easy AI is presented in Algorithm 1.

### 6.3 Medium - MCTS AI

The `mcts.js` file implements a Monte Carlo Tree Search agent that looks beyond the current turn to simulate future outcomes. It constructs a decision tree using Node objects and determines win probabilities through random simulations `RANDOMPLAYOUT(T)`. The agent uses the Upper Confidence Bound formula in the `UCTSELECT(f)`unction to intelligently balance ”exploration” (trying unvisited moves) vs. ”exploitation” (choosing moves that have previously led to victory).

The MCTS agent is integrated directly into the main game logic (‘`app.js`’), which invokes its `CHOOSEACTION(m)`ethod when it’s the AI’s turn. During each decision cycle, the MCTS algorithm typically performs 200 iterations (a configurable parameter) of its four phases: Selection, Expansion, Simulation, and Backpropagation.

During the Simulation (playout) phase, the MCTS explores potential game outcomes by rapidly playing out games from the current state to a terminal state. These playouts are performed using a random policy, where both players take random legal actions until a win/loss/draw condition is met or a maximum depth of 40 turns is reached. The opponent in these simulations is essentially another random player. The results of these simulations (rewards) are then used to update the ‘value’ and ‘visits’ statistics back up the search tree during backpropagation, effectively ”training” the tree to favor more promising paths. The final action chosen by the MCTS is typically the child of the root node that has been visited the most times, indicating the most explored and robust strategy.

### 6.4 Hard - LLM using Google Gemini 2.5 Flash

## 7 Conclusion

In this paper, we presented a comprehensive analysis of Kings of the West, a novel turn-based tactical board game designed and implemented by our team. Our work encompassed game design, software development, artificial intelligence integration, and a formal computational complexity analysis.

---

**Algorithm 1** Simplified Easy AI Decision-Making Algorithm

---

```

1: function CHOOSEACTION(state, player)
2:   dice  $\leftarrow$  state.dice
3:   activePieces  $\leftarrow$  filter state.players[player].pieces for hp  $>$  0
4:   maxMoveSteps  $\leftarrow$  determine from dice (1-3 for rolls 1-3, 1 for rolls 4,5)
5:   Shuffle activePieces randomly.                                 $\triangleright$  Introduces non-determinism for piece selection
6:   for each piece in activePieces do                          $\triangleright$  1. Prioritize immediate attack from current position
7:     enemiesHere  $\leftarrow$  GETENEMIESINATTACKRANGE(state, piece, piece.r, piece.c)
8:     if enemiesHere is not empty then
9:       target  $\leftarrow$  randomly selected enemy from enemiesHere
10:      return {pieceId : piece.id, move : null, attackId : target.id}
11:       $\triangleright$  2. If no immediate attack, try to move and then attack, or just move
12:    if maxMoveSteps  $>$  0 then
13:      reachableTiles  $\leftarrow$  GETREACHABLE(state, piece, maxMoveSteps)
14:      Add piece's current position to reachableTiles
15:      Shuffle reachableTiles randomly.                             $\triangleright$  Introduces non-determinism for move selection
16:      for each destinationTile in reachableTiles do
17:        potentialMove  $\leftarrow$  (destinationTile?{r : destinationTile.r, c : destinationTile.c} : null)
18:        currentR  $\leftarrow$  (potentialMove?potentialMove.r : piece.r)
19:        currentC  $\leftarrow$  (potentialMove?potentialMove.c : piece.c)
20:        enemiesFromMove  $\leftarrow$  GETENEMIESINATTACKRANGE(state, piece, currentR, currentC)
21:        if enemiesFromMove is not empty then
22:          target  $\leftarrow$  randomly selected enemy from enemiesFromMove
23:          return {pieceId : piece.id, move : potentialMove, attackId : target.id}
24:           $\triangleright$  3. If no attack possible, make a random valid move (or stay put)
25:        randomMove  $\leftarrow$  first non-null move from shuffled reachableTiles (if any)
26:        if randomMove  $\neq$  null then
27:          return {pieceId : piece.id, move : randomMove, attackId : null}
28:        else                                                  $\triangleright$  If only staying put is an option or no moves found
29:          return {pieceId : piece.id, move : null, attackId : null}
30:      return null                                          $\triangleright$  No actions found for any piece
31:       $\triangleright$  Helper functions such as GETPIECEAT(), GETREACHABLE(), GETENEMIESINATTACKRANGE(), GETOPPONENT(a)re used for standard game state queries and movement calculations. This AI intentionally avoids an explicit heuristic evaluation function for simplicity.

```

---

We successfully developed a fully functional browser-based game, adhering to a set of clearly defined rules that blend strategic positioning with elements of chance introduced by a six-sided die. Alongside the core game, we implemented two distinct AI agents: a heuristic-based Easy AI and a more advanced Monte Carlo Tree Search (MCTS) AI. These agents provide varied challenges, demonstrating the application of different algorithmic approaches to game play. The MCTS AI, in particular, showcased an ability to navigate the game's complexities more effectively by learning from simulations.

A significant contribution of this work is the formal computational complexity analysis of “Generalized Kings of the West,” played on an  $N \times M$  board. We rigorously proved that determining optimal play in this generalized version is PSPACE-Complete.

1. We demonstrated Membership in PSPACE by outlining how a polynomial-space algorithm, such as a Minimax search enhanced with alpha-beta pruning, can effectively explore the game’s state space without exceeding polynomial memory requirements relative to the board size.
2. We established PSPACE-Hardness through a detailed reduction from Nondeterministic Constraint Logic (NCL). This involved meticulously mapping NCL graph components to specific “blocking” and “threat” gadgets constructed within the game’s board and piece mechanics, thereby showing that Kings

of the West is at least as hard as any other problem in PSPACE.

The PSPACE-completeness result underscores the inherent complexity of Kings of the West, implying that even for seemingly simple rules, optimal play is computationally intractable for larger board sizes, thus affirming the game's strategic depth.

## 7.1 Future Work

While this project provides a robust foundation, several avenues for future work exist:

- **Advanced AI Development:** Explore more sophisticated AI algorithms beyond MCTS, such as deep reinforcement learning, to potentially achieve even higher levels of play.
- **LLM Integration:** Investigate the integration of Large Language Models (LLMs), such as Llama or Co-Pilot, into the game. This could manifest as a natural language interface for player commands, dynamic narrative generation based on game events, or even as an LLM-powered strategic advisor that provides tactical recommendations.
- **Expanded Complexity Analysis:** Further investigate complexity for variants of the game, such as games with more players, different piece types, or alternative win conditions.
- **Multiplayer Functionality:** Extend the game to support online multiplayer functionality, allowing players to compete remotely.
- **User Studies and Balance Testing:** Conduct extensive user studies and balance testing to refine game mechanics, piece abilities, and dice roll effects, ensuring optimal player enjoyment and competitive fairness.

Kings of the West successfully blends traditional board game elements with modern computational analysis, offering a rich platform for both entertainment and academic study.

## 8 Acknowledgments

We would like to give thanks to the following people as their knowledge, support, insights, and feedback were crucial to the creation of both the game and paper: Nathaniel De Leon, Ruben Puga, Jesus Gutierrez, Efren Carrillo, Gael Marquez , our professor Dr. Tim Wylie and everyone in the Games & Computation class

## References

- [1] B. Alaniz, R. Balvanera, and F. Castan. Kings Of The West Game. <https://github.com/rbalvanera/CSCI-4341>, 2025. GitHub repository.
- [2] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta. *Classic Nintendo Games are (Computationally) Hard*. *Theoretical Computer Science*, 586:135–160, 2015.
- [3] R. A. Hearn and E. D. Demaine. *PSPACE-Completeness of Sliding-Block Puzzles and Other Problems through the Nondeterministic Constraint Logic Model of Computation*, 2004.
- [4] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters, Ltd., 2009.
- [5] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.