Ahmed Hassan ElRaggal	7794
Bishoy Adel	7843
Youssef Magdy	7825

—---

Connect 4 Game

Pseudocodes for the algorithms Used :

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth == 0 or node is terminal then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            if value > β then
                break (* β cutoff *)
            α := max(α, value)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            if value < α then
                break (* α cutoff *)
            β := min(β, value)
        return value
```

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

```
function expectiminimax(node, depth)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    if the adversary is to play at node
        // Return value of minimum-valued child node
        let α := +∞
        foreach child of node
            α := min(α, expectiminimax(child, depth-1))
    else if we are to play at node
        // Return value of maximum-valued child node
        let α := -∞
        foreach child of node
            α := max(α, expectiminimax(child, depth-1))
    else if random event at node
        // Return weighted average of all child nodes' values
        let α := 0
        foreach child of node
            α := α + (Probability[child] × expectiminimax(child, depth-1))
    return α
```

—---

Data Structures Used :

—----------------------------

Numpy array to represent board
Lists used to get the windows in the array to check the heuristic value
Tuples used in checking winning move
Set used to store winning moves to prevent checking duplicates
Deque used to traverse nodes

—---

Properties of minimax

- Optimal (opponent plays optimally) and complete (finite tree)
- DFS time: $O(b^m)$
- DFS space: $O(bm)$

Alpha Beta Pruning Strategy :

$\alpha - \beta$ pruning

- **Strategy**: Just like minimax, it performs a DFS.
- **Parameters**: Keep track of two bounds
  - α: largest value for Max across seen children (current lower bound on MAX's outcome).
  - β: lowest value for MIN across seen children (current upper bound on MIN's outcome).
- **Initialization**: $\alpha = -\infty$, $\beta = \infty$
- **Propagation**: Send α, β values *down* during the search to be used for pruning.
  - Update α, β values by *propagating upwards* values of terminal nodes.
  - Update α only at Max nodes and update β only at Min nodes.
- **Pruning**: Prune any remaining branches whenever $\alpha \geq \beta$.

- **Worst ordering**: no pruning happens (best moves are on the right of the game tree). Complexity $O(b^m)$.
- **Ideal ordering**: lots of pruning happens (best moves are on the left of the game tree). This solves tree twice as deep as minimax in the same amount of time. Complexity $O(b^{m/2})$ (in practice). The search can go deeper in the game tree.

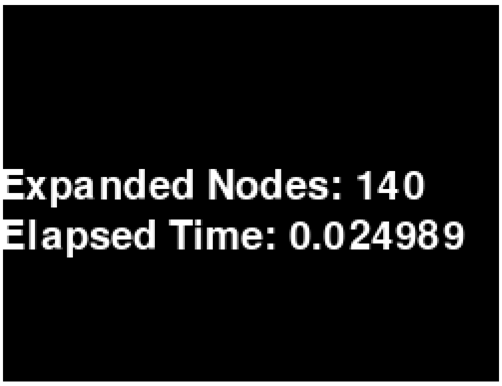
Heuristic Function Used :

Positive Scoring:
- Achieving 4 consecutive AI colors earns 4 points.
- Achieving 3 consecutive candidate AI colors earns 3 points.
- Achieving 2 consecutive candidate AI colors earns 2 points.
- Preventing the opponent from achieving a point earns 1 point.

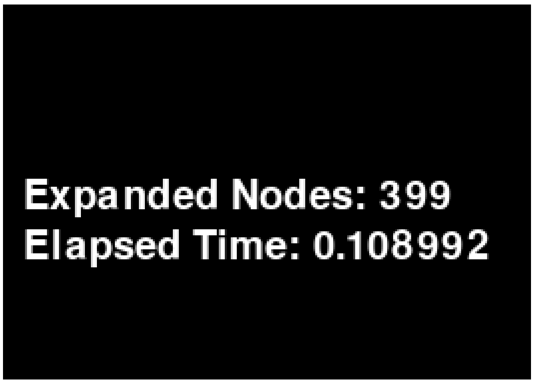
Negative Scoring:
- Allowing the opponent to achieve 4 consecutive Human colors results in a deduction of 4 points.
- Allowing the opponent to achieve 3 consecutive candidate Human colors results in a deduction of 3 points.
- Allowing the opponent to achieve 2 consecutive candidate Human colors results in a deduction of 2 points.
- Failing to prevent opponent   from achieving a point results in a deduction of 1 point.

Number Of Nodes expanded and time for all types and diff k

K = 3

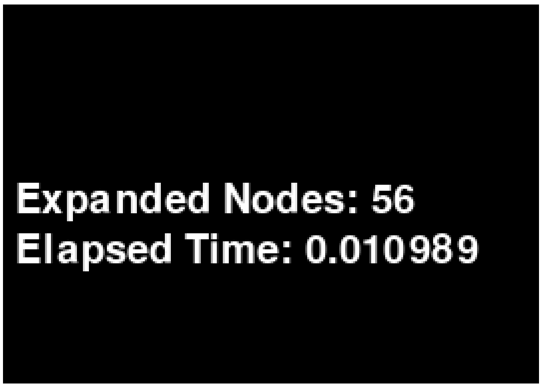


Expect MiniMax   MiniMax with AlphaBeta pruning

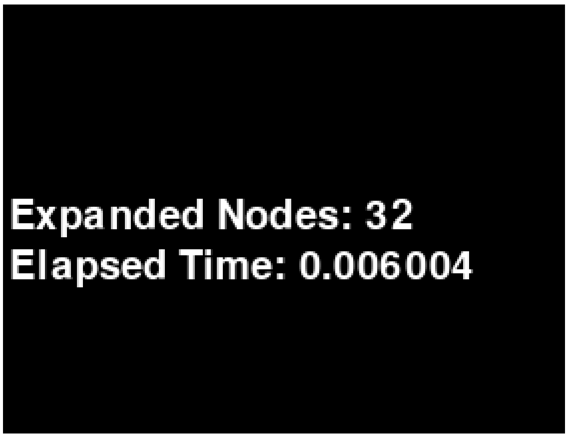


MiniMax

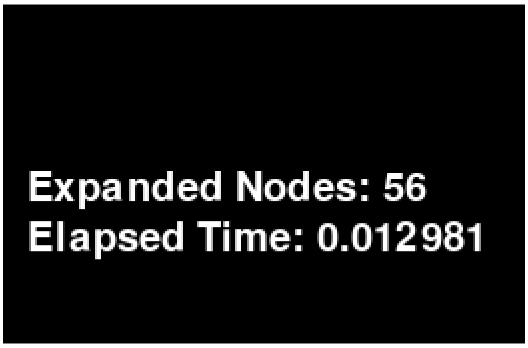
K = 2



MiniMax



MiniMax with AlphaBeta pruning



Expect MiniMax

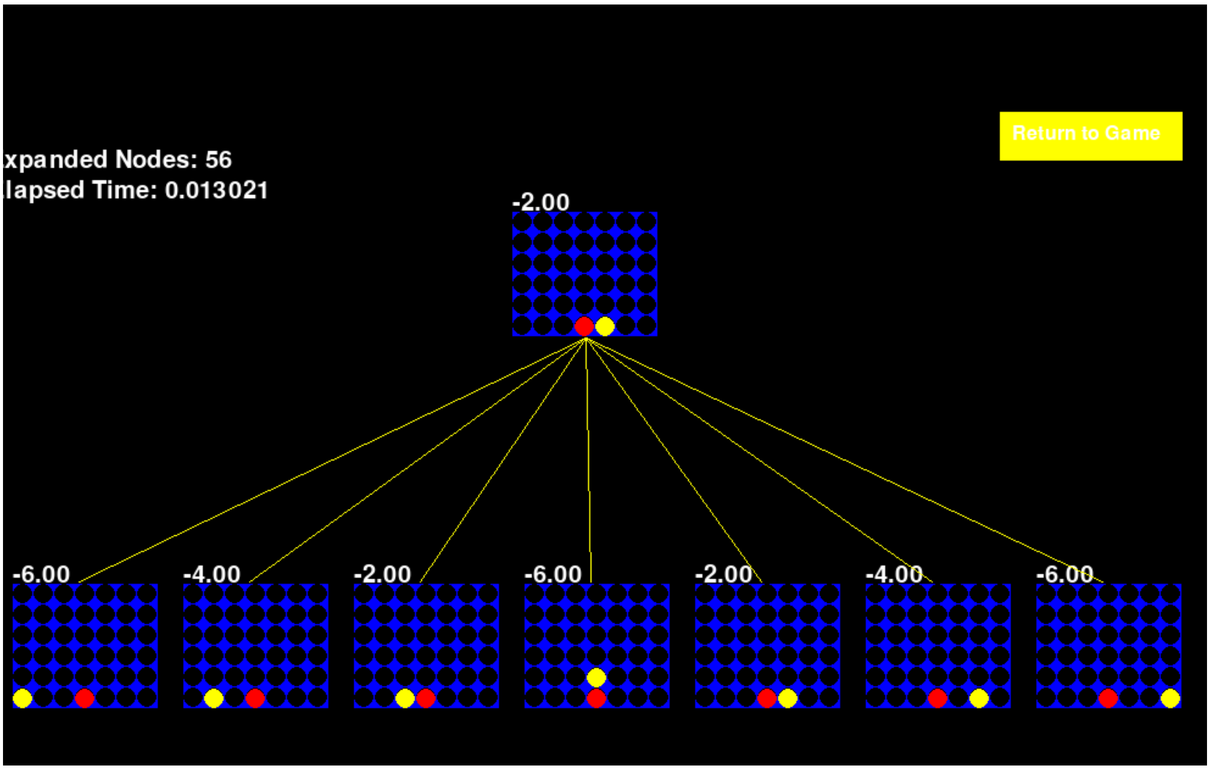
—--

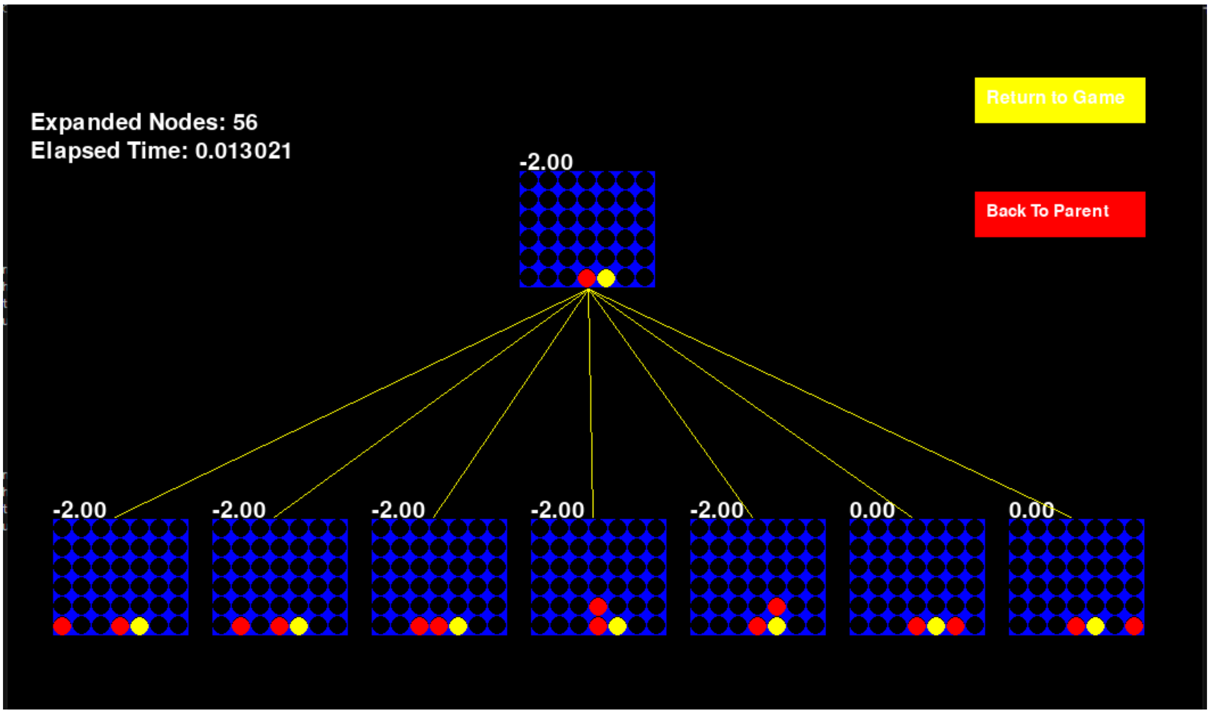
Full Sample Run

—--------------------

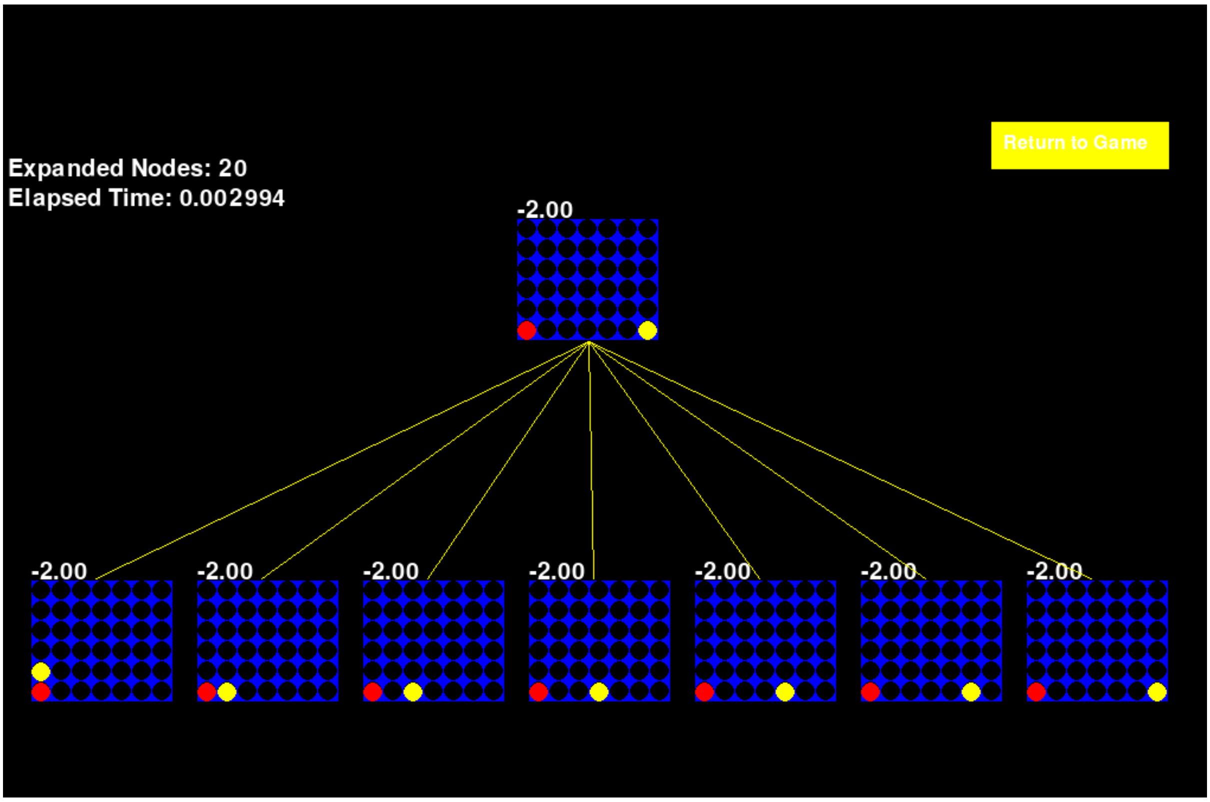
No Pruning

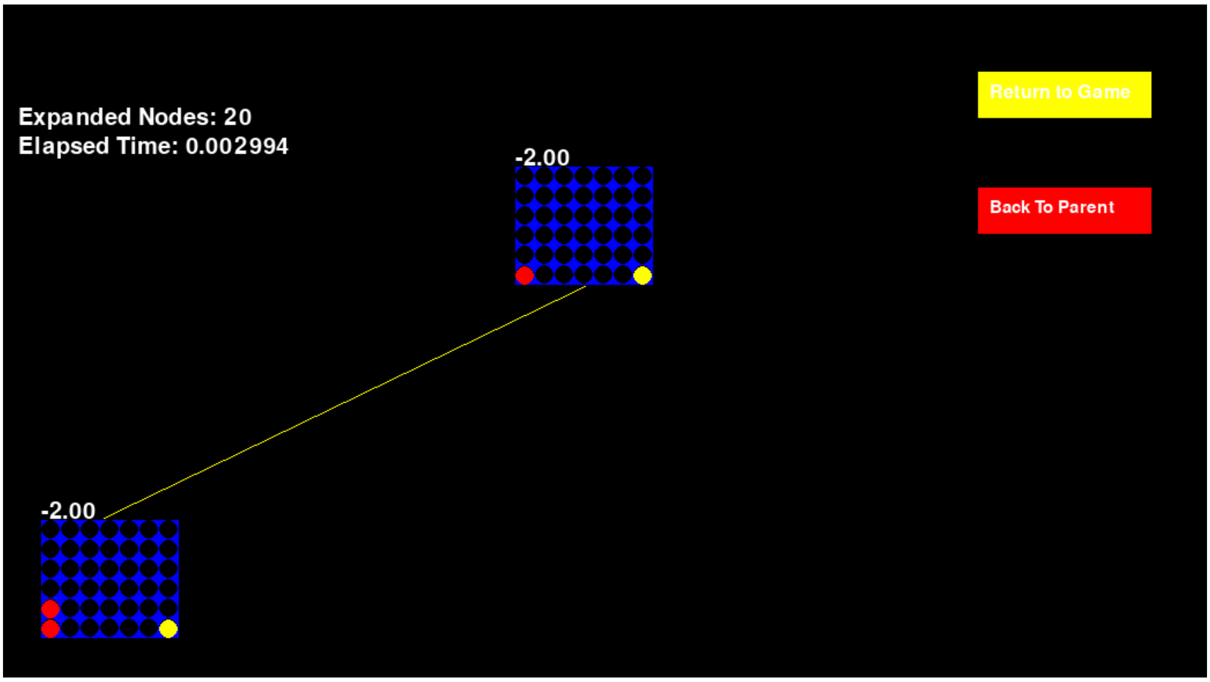
—--------------





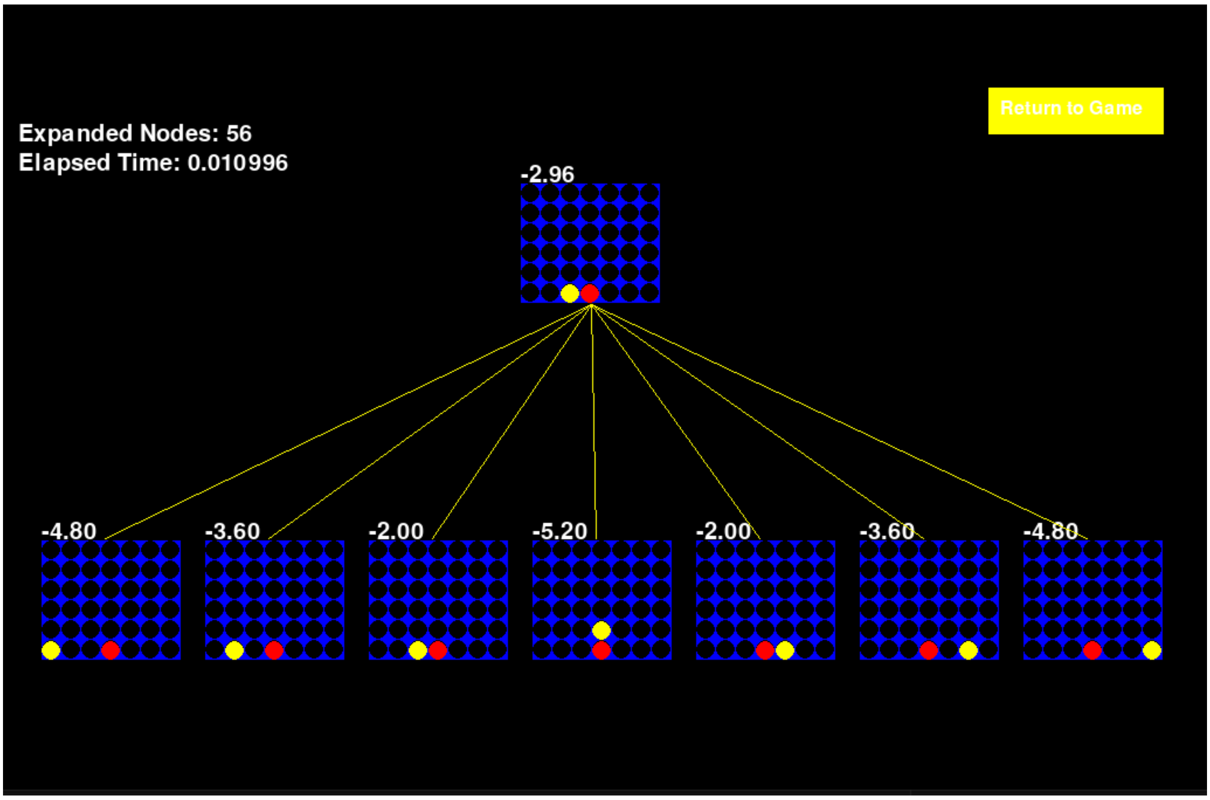
Full Sample Run With Alpha-Beta Pruning

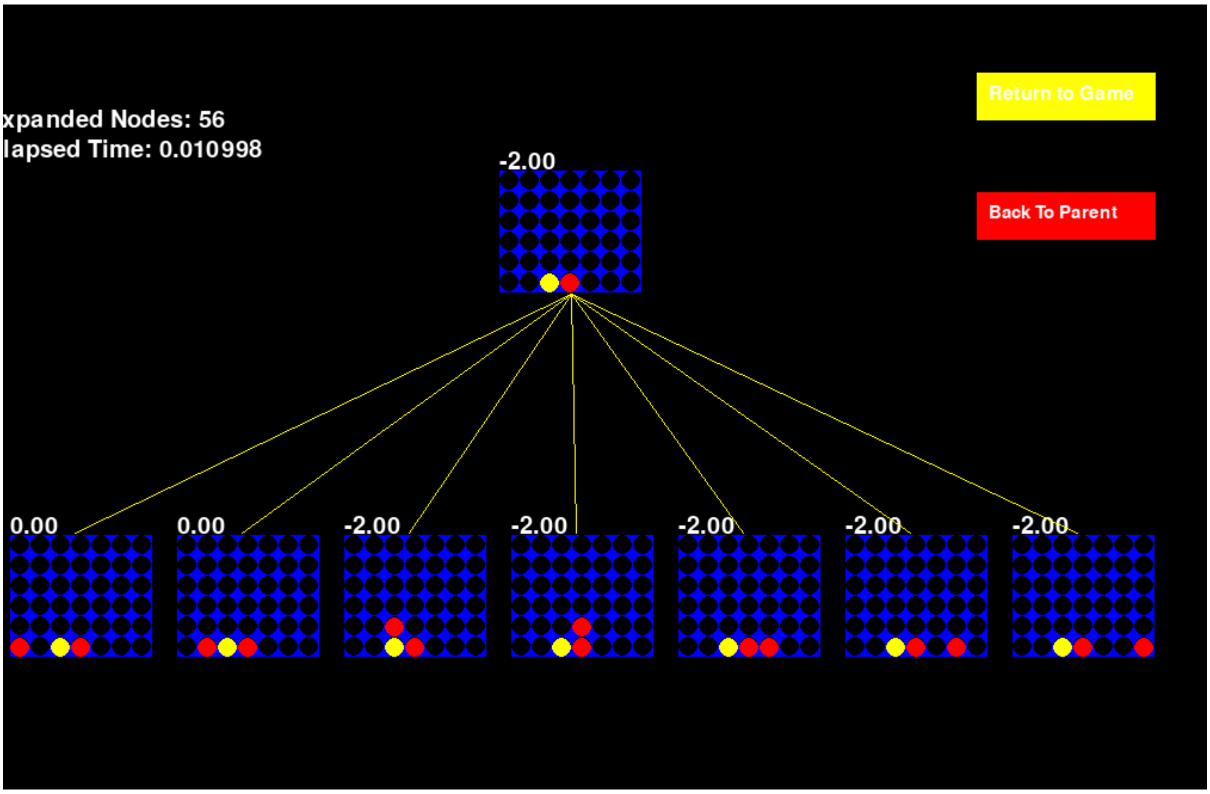




Full Sample Run With Expect-MiniMax

—---




xpanded Nodes: 56
lapsed Time: 0.010998
Return to Game
Back To Parent
-2.00
0.00
0.00
-2.00
-2.00
-2.00
-2.00
-2.00