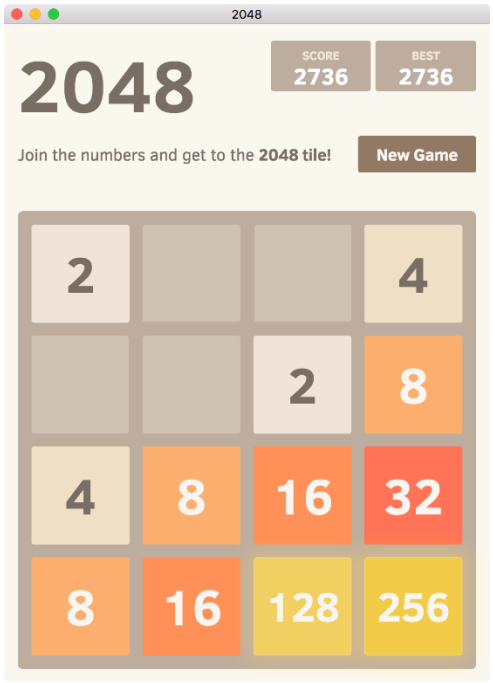
Kelompok 22:

- Eka Novita Sari (Persevere)
- Sella Batania Br. Tobing (Visioner)
- Hidayatul Mustafida (Atlas)
- Maria Emiliana Nogo (Alan Turing)

"MONTE CARLO TREE SEARCH PADA GAME 2048"

Game 2048 dimainkan pada kotak berukuran 4x4, tujuan dari game ini adalah untuk mencapai nilai 2048 dengan memindahkan dan menyatukan tiles yang terdapat pada board 4x4. 2 tiles secara acak berisi angka 2 atau 4. Pemain dapat menggerakkan angka keatas, kebawah, kekanan, dan kekiri. Ketika 2 tiles dengan angka yang sama bertabrakan, maka tiles tersebut akan menambahkan nilainya dan nilai tersebut adalah skor pemain.



Penjelasan Program

```
function moveName(input move : dir)
{menginisialisasi move berdasarkan nilai yang ditentukan}
Deklarasi
Algoritma
return (0 <- 'up', 1 <- 'right', 2 <- 'down' , 3 <- 'left')[move]
```

Fungsi moveName berguna untuk melakukan inisialisasi move berdasarkan nilai yang telah ditentukan.

```
Function AI_getBest(input grid : grid)
{melakukan suatu pencarian fungsi dan return best move}
Deklarasi
runs : integer
Algoritma
runs <- document.getElementById('run-count').value
return getBestMove(grid, runs)
```

Fungsi AI_getBest berguna untuk memulai perhitungan proses algoritma monte carlo dan mendapatkan move terbaik berdasarkan perhitungan yang telah dilakukan.

```
Function MonteSearch(input grid : grid, runs : integer)
{melakukan proses pencarian untuk menemukan bestMove}
Deklarasi
bestMove : integer
bestScore : double
Algoritma
for i <- 0 to 4 do
res <- multiRandomRun(grid, i, runs)
score <- res.score
if score >= bestScore then
bestScore <- score
bestMove <- i
end if
End for
return move <- bestMove, score <- bestScore
```

Fungsi MonteSearch berguna untuk melakukan pencarian bestMove pada tree dan akan terus melakukan looping terhadap seluruh move yang sudah diinisialisasi dan setelah itu menentukan nilai tertinggi yang sudah didapatkan.

```
Function multiRandomRun(input grid : grid, move : integer, runs : integer)
{melakukan proses pencarian untuk mencari nilai bestScore}
Deklarasi
total : integer = 0
avg : double
Algortima
for i<-0 to runs then
res <- randomRun(grid,move)
s <- res.score
if s == -1 then
return -1
end if
total <- total + s
end for
avg <- total / runs
return score<-avg
```

Fungsi multiRandomRun berguna untuk melakukan suatu penambahan cabang tree pada algoritma monte carlo berdasarkan nilai input yang telah ditentukan oleh user yaitu runs, dan masing-masing move akan dilakukan looping sebanyak runs.

```
Function randomRun(input grid : grid, move : integer)
{Melakukan proses random move sampai tidak bisa move dan return score
tertinggi}
Deklarasi
g : grid = grid.clone()
score : integer = 0
moves : integer = 0
Algoritma
res <- moveAndAddRandomTiles(g, move)
if !res.moved then
return -1
end if
score <- score + res.score
while true do
if !g.movesAvailable() then
break
end if
```

```
rand <- g.move(Math.floor(Math.random() * 4 ))
if !rand.moved then
continue
end if
score <- score + res.score
g.addRandomTile()
moves++
end while
return score
```

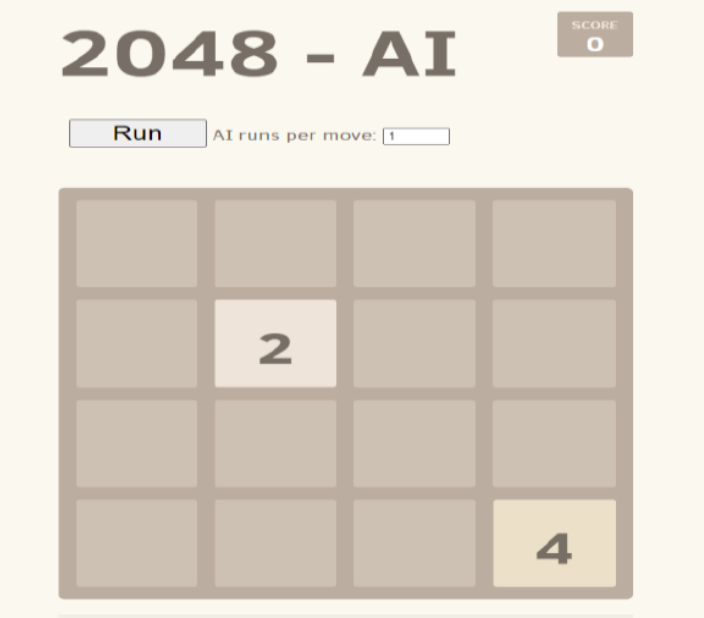
Fungsi randomRun berguna untuk melakukan proses simulasi tree pada algoritma monte carlo, dengan melakukan random move pada move yang sedang dijalankan serta looping pada papan yang telah dilakukan klon sampai tidak dapat melakukan move dan setelah itu akan melakukan return score untuk update nilai bestScore.

```
Function moveAndAddRandomTiles(input grid : grid, direction : dir)
{melakukan proses move dan menambah random tile}
Algoritma
res <- grid.move(direction)
if res.moved then
grid.addRandomTile()
end if
return res
```

Fungsi yang digunakan untuk melakukan proses move dan menambahkan random tile jika memungkinkan untuk melakukan pergerakan.

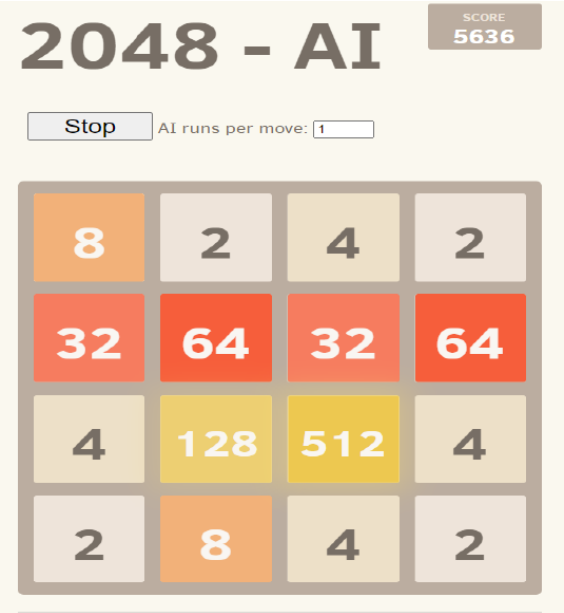
Tampilan Awal

Gambar dibawah ini merupakan tampilan ketika pertama kali program dijalankan.

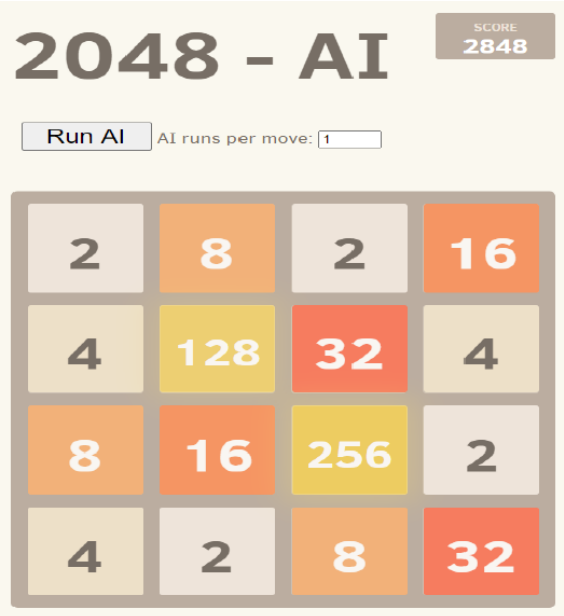


Tampilan Game Over

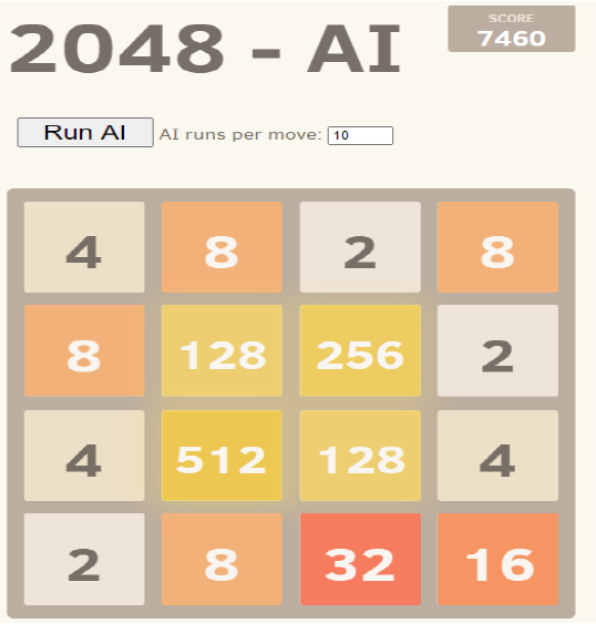
Gambar dibawah ini merupakan tampilan game over, jika tidak terdapat gerakan yang dapat dilakukan maka tampilan hanya seperti ini.



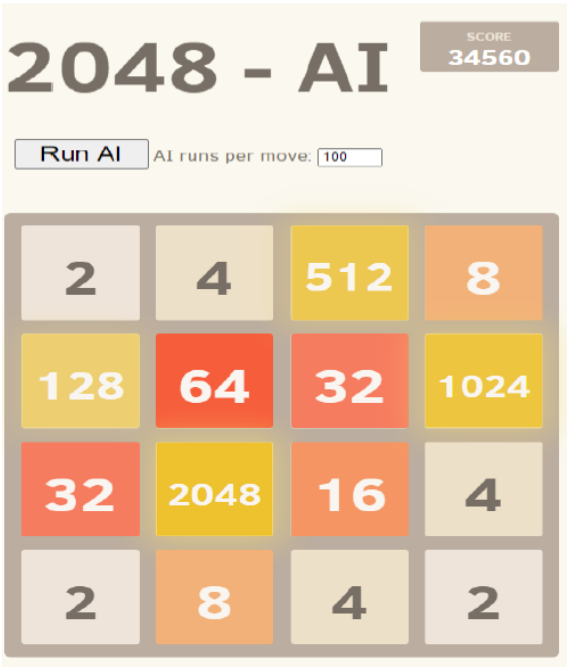
Percobaan 1



Percobaan 6



Percobaan 11



Jadi, cara kerja algoritma monte carlo tree search pada game 2048 adalah dengan memilih move yang telah diinisialisasi dan akan dilakukan penelusuran cabang tree pada masing-masing move tersebut, dan akan dilakukan pengecekan jika move tersebut dapat digerakkan. Jika dapat maka akan dilakukan simulasi yaitu melakukan pergerakan move secara random sampai tidak dapat melakukan move lagi sebanyak nilai yang telah di input oleh user dan setelah itu akan melakukan backpropagation ke cabang teratas dan akan return nilai total dari cabang tersebut. Lalu akan melakukan pengecekan sampai seluruh cabang telah diproses, setelah itu baru akan ditentukan move berdasarkan cabang dengan nilai tertinggi.