|  |  |
| --- | --- |
| Bitcoin Ninja Documentation |  |

# Project Name: Bitcoin Ninja

December 19, 2016

# About

Web-based 2D Game, simulates an AI technique (A\* Algorithm) used to determine the shortest valid path between two points.

# Flow

* Pre-Settings

A road map is generated randomly with a treasure(Bitcoin) in the middle and a ninja thief at the bottom side. The user adds some guards to guard the treasure. Adding a guard includes positioning and range of movement. Then the user starts the game.

* On Running

The ninja thief starts to move towards the treasure stealthy and avoiding the guards, he steals the treasure and gets out of the map again with the same way he entered.

* On End

There are two ends for the game, the first is that the ninja steals the treasure and gets out safely, the second one is that at least one of the guards sees the ninja thief then he would kill him and the game ends.

# Technical Specifications

* Language: JavaScript
* Algorithms: A\* Algorithm

# Functions and Complexity

|  |
| --- |
| Function A\* - O(N) |

|  |
| --- |
| function AStar(test){ |
|  | var visited = CreateVisited(); |
|  | var open\_list = new PriorityQueue({ comparator: function(a, b) { |
|  | if(a.cost+a.heu>b.cost+b.heu||(a.cost+a.heu==b.cost+b.heu&&a.cost>b.cost)) return 1; |
|  | return -1; |
|  | }}); |
|  | visited[Ninja.Top][Ninja.Left]=1; |
|  | open\_list.queue({ Top:Ninja.Top, Left:Ninja.Left, cost:0, heu:Heuristic(Ninja), parent:null }); |
|  | while (open\_list.length){ |
|  | var current = open\_list.dequeue(); |
|  | visited[current.Top][current.Left]=1; |
|  | if(test==undefined) UI.GetCell(current.Top,current.Left).classList.add("visited"); |
|  | if(Heuristic(current)==0) |
|  | return ConstructPath(current); |
|  | var up = JSON.parse(JSON.stringify(current)); |
|  | var down = JSON.parse(JSON.stringify(current)); |
|  | var left = JSON.parse(JSON.stringify(current)); |
|  | var right = JSON.parse(JSON.stringify(current)); |
|  | up.Top--,down.Top++,left.Left--,right.Left++; |
|  | up.heu=Heuristic(up),down.heu=Heuristic(down),left.heu=Heuristic(left),right.heu=Heuristic(right); |
|  | up.parent=down.parent=left.parent=right.parent=current; |
|  | //if(isValid(wait)) open\_list.queue(wait); |
|  | var uf=0, df=0, lf=0, rf=0; |
|  | for(var k=1;k<=10;k++){ |
|  | if(!isValid(current,current.cost+k-1)) break; |
|  | up.cost=down.cost=left.cost=right.cost=current.cost+k; |
|  | if(isValid(up,up.cost)&&!visited[up.Top][up.Left]) uf=1,open\_list.queue(JSON.parse(JSON.stringify(up))); |
|  | if(isValid(down,down.cost)&&!visited[down.Top][down.Left]) df=1,open\_list.queue(JSON.parse(JSON.stringify(down))); |
|  | if(isValid(left,left.cost)&&!visited[left.Top][left.Left]) lf=1,open\_list.queue(JSON.parse(JSON.stringify(left))); |
|  | if(isValid(right,right.cost)&&!visited[right.Top][right.Left]) rf=1,open\_list.queue(JSON.parse(JSON.stringify(right))); |
|  | } |
|  | if(uf) visited[up.Top][up.Left]=1; |
|  | if(df) visited[down.Top][down.Left]=1; |
|  | if(lf) visited[left.Top][left.Left]=1; |
|  | if(rf) visited[right.Top][right.Left]=1; |
|  | } |
|  | return false; // no path exists |
|  | } |

|  |
| --- |
| Function ConstructPath(node); O(N) |

|  |
| --- |
| function ConstructPath(node){ |
|  | var path=[]; |
|  | path.push(JSON.parse(JSON.stringify(node))); |
|  | do{ |
|  | while(node.cost>node.parent.cost) path.push(JSON.parse(JSON.stringify(node.parent))),node.cost--; |
|  | node = node.parent; |
|  | } |
|  | while(node.parent!=null); |
|  | while(node.cost>-1)path.push(JSON.parse(JSON.stringify(node))),node.cost--; |
|  | return path.reverse(); |
|  | } |

|  |
| --- |
| Function Heuristic(node); O(1) |

|  |
| --- |
| function Heuristic(node){ |
|  | return Math.abs(Coin.Top-node.Top)+Math.abs(Coin.Left-node.Left); |
|  | } |

|  |
| --- |
| Function isValid(node,cost); O(N^2) |

|  |
| --- |
| function isValid(node,cost){ |
|  | cost+=UI.lastGuardPos; |
|  | cost=Math.max(0,cost); |
|  | if(node.Top<0||node.Top>=SizeX||node.Left<0||node.Left>=SizeY) return 0; |
|  | var sol=true; |
|  | for (var i = 0; i < Guards.length; i++) { |
|  | var length = Guards[i].length; |
|  | var index = GetMoveIndex(cost,length); |
|  | sol&=!AreEqual(Guards[i][index],node); |
|  | if(index>0)sol&=!AreEqual(Guards[i][index-1],node); |
|  | if(index<Guards[i].length-1)sol&=!AreEqual(Guards[i][index+1],node); |
|  | } |
|  | return sol&&Map[node.Top][node.Left]; |
|  | } |

|  |
| --- |
| Function CreateVisited(); O(N^2) |

|  |
| --- |
| function CreateVisited(){ |
|  | var arr = new Array(SizeY); |
|  | for(var i=0;i<SizeY;i++) { |
|  | arr[i]=new Array(SizeX); |
|  | for(var j=0;j<SizeX;j++) |
|  | arr[i][j]=false; |
|  | } |
|  | return arr; |
|  | } |

|  |
| --- |
| Function GenerateMap(); O(N^2) |

|  |
| --- |
| function GenerateMap(){ |
|  | var arr = new Array(SizeX); |
|  | for(var i=0;i<SizeX;i++) { |
|  | arr[i]=new Array(SizeY); |
|  | for(var j=0;j<SizeY;j++) |
|  | arr[i][j]=1; |
|  | } |
|  | var obstacle=0; |
|  | while(obstacle!=65){ |
|  | var random\_obstacle\_Top = Math.floor(Math.random()\*SizeX); |
|  | var random\_obstacle\_Left = Math.floor(Math.random()\*SizeY); |
|  | if(arr[random\_obstacle\_Top][random\_obstacle\_Left]!=0){ |
|  | arr[random\_obstacle\_Top][random\_obstacle\_Left]=0; |
|  | obstacle++; |
|  | } |
|  | } |
|  | do{ |
|  | Coin.Top = Math.floor(Math.random()\*SizeX); |
|  | Coin.Left = Math.floor(Math.random()\*SizeY); |
|  | }while((Coin.Top > (SizeX/2) || Coin.Left > (SizeX/2)) |
|  | || (arr[Coin.Top][Coin.Left]==0)); |
|  |  |
|  | do{ |
|  | Ninja.Top = Math.floor(Math.random()\*SizeX); |
|  | Ninja.Left = Math.floor(Math.random()\*SizeY); |
|  | }while((Ninja.Top < (SizeX/2) || Ninja.Left < (3\*SizeY/4)) || (arr[Ninja.Top][Ninja.Left]==0)); |
|  | Map = arr; |
|  | var test = AStar(1); |
|  | if(!test) |
|  | return GenerateMap(); |
|  | return arr; |
|  | } |

|  |
| --- |
| Function Move --> O(1) |

|  |
| --- |
| Move: function(cell, char) { |
|  | UI.Rotate(char, cell); |
|  | UI.SetCellPosition(char,cell); |
|  | } |

|  |
| --- |
| Function SimulateNinjaMove --> O(1) |

|  |
| --- |
| SimulateNinjaMove: function(i) { |
|  | if(i<Path.length){ |
|  | UI.Move(Path[i], document.getElementById('ninja')); |
|  | if(i) UI.GetCell(Path[i-1].Top, Path[i-1].Left).classList.remove('ninja-here'); |
|  | //UI.GetCell(Path[i].Top, Path[i].Left).classList.add('ninja-here'); |
|  | var here = UI.GetCell(Path[i].Top, Path[i].Left); |
|  | here.style.opacity=UI.CellHighlight; |
|  | if(here.style.backgroundColor=="red") here.style.backgroundColor="rgb(236, 65, 0)" |
|  | } |
|  | else return; |
|  | UI.SimulateGuardMove(i); |
|  | setTimeout(UI.SimulateNinjaMove.bind(null,i+1),300); |
|  | } |

|  |
| --- |
| Function SimulateGuardMove --> O(N) |

|  |
| --- |
| SimulateGuardMove: function(cnt) { |
|  | cnt--; |
|  | cnt+=UI.lastGuardPos; |
|  | if(cnt<0) return; |
|  | for (var i = 0; i < Guards.length; i++) { |
|  | var index; |
|  | var length = Guards[i].length; |
|  | if(cnt>1) { |
|  | index = GetMoveIndex(cnt-2,length); |
|  | UI.GetCell(Guards[i][index].Top,Guards[i][index].Left).classList.remove('danger'); |
|  | } |
|  |  |
|  | index = GetMoveIndex(cnt,length); |
|  | UI.Move(Guards[i][index], document.getElementsByClassName('guard')[i]); |
|  |  |
|  | UI.GetCell(Guards[i][index].Top,Guards[i][index].Left).classList.add('danger'); |
|  | if(index>0) UI.GetCell(Guards[i][index-1].Top,Guards[i][index-1].Left).classList.add('danger'); |
|  | if(index<Guards[i].length-1) UI.GetCell(Guards[i][index+1].Top,Guards[i][index+1].Left).classList.add('danger'); |
|  |  |
|  | } |
|  | } |

|  |
| --- |
| Function Rotate --> O(1) |

|  |
| --- |
| Rotate: function(char,cell) { |
|  | var cur, rot,tmp; |
|  | cur = {Left:parseInt(char.style.left.split('p')[0],10) , Top:parseInt(char.style.top.split('p')[0],10)}; |
|  | rot = parseInt(char.style.transform.split('(')[1].split('d')[0],10); |
|  | tmp=rot+360; |
|  | if(tmp%360==0){ |
|  | if (cur.Left > cell.Left \* CellSize) char.style.transform = "rotate("+(rot-90)+"deg)"; |
|  | else if (cur.Left < cell.Left \* CellSize) char.style.transform = "rotate("+(rot+90)+"deg)"; |
|  | else if(cur.Top < cell.Top \* CellSize) char.style.transform = "rotate("+(rot+180)+"deg)"; |
|  | } |
|  | else if(tmp%360==270){ |
|  | if (cur.Top < cell.Top \* CellSize) char.style.transform = "rotate("+(rot-90)+"deg)"; |
|  | else if (cur.Top > cell.Top \* CellSize) char.style.transform = "rotate("+(rot+90)+"deg)"; |
|  | else if(cur.Left < cell.Left \* CellSize) char.style.transform = "rotate("+(rot+180)+"deg)"; |
|  | } |
|  | else if(tmp%360==180){ |
|  | if (cur.Left < cell.Left \* CellSize) char.style.transform = "rotate("+(rot-90)+"deg)"; |
|  | else if (cur.Left > cell.Left \* CellSize) char.style.transform = "rotate("+(rot+90)+"deg)"; |
|  | else if(cur.Top > cell.Top \* CellSize) char.style.transform = "rotate("+(rot+180)+"deg)"; |
|  | } |
|  | else{ |
|  | if (cur.Top > cell.Top \* CellSize) char.style.transform = "rotate("+(rot-90)+"deg)"; |
|  | else if (cur.Top < cell.Top \* CellSize) char.style.transform = "rotate("+(rot+90)+"deg)"; |
|  | else if(cur.Left > cell.Left \* CellSize) char.style.transform = "rotate("+(rot+180)+"deg)"; |
|  | } |
|  | } |

|  |
| --- |
| Function GetCell --> O(1) |

|  |
| --- |
| GetCell: function(i,j){ |
|  | return document.getElementById(""+i+"-"+j); |
|  | } |

|  |
| --- |
| Function GetCellPosition --> O(1) |

|  |
| --- |
| GetCellPosition: function(cell){ |
|  | var IdStr = cell.id; |
|  | var SplittedId = IdStr.split('-'); |
|  | var i=parseInt(SplittedId[0],10); |
|  | var j=parseInt(SplittedId[1],10); |
|  | return {Top:i, Left:j}; |
|  | } |

|  |
| --- |
| Function SetCellPosition --> O(1) |

|  |
| --- |
| SetCellPosition: function(cell,position){ |
|  | cell.style.top=""+(position.Top\*CellSize)+"px"; |
|  | cell.style.left=""+(position.Left\*CellSize)+"px"; |
|  | } |

|  |
| --- |
| Function HighlightAdjcent --> O(1) |

|  |
| --- |
| HighlightAdjcent: function(cell){ |
|  | var position = UI.GetCellPosition(cell); |
|  | var adjcent = [ |
|  | UI.GetCell(position.Top-1,position.Left), |
|  | UI.GetCell(position.Top+1,position.Left), |
|  | UI.GetCell(position.Top,position.Left-1), |
|  | UI.GetCell(position.Top,position.Left+1), |
|  | ]; |
|  | adjcent.forEach(function(item){ |
|  | if(item==null||!item.classList.contains('add')) return; |
|  | item.classList.add('add-active'); |
|  | item.setAttribute('onclick',"AddRange(this)"); |
|  | CurrentGuardMoves.push(item); |
|  | } |

# Milestones

|  |  |
| --- | --- |
| Milestone1 19/12/2016 | Control the Ninja’s Movement Timing |
| Milestone2 26/12/2016 | Control the Ninja’s Movement Timing |

# Project Map

|  |  |
| --- | --- |
| Engine | UI |
| BFS | Map Generation |
| Add Heuristic | Characters' Textures |
| Add Timing | Characters' Movement |
|  | Add Guard (Position + Range) |
|  | Main Buttons |

# Project Team

|  |  |  |
| --- | --- | --- |
| 2013170046 | Ahmad Muhammad Mustafa Elsayid | Section 3 |
| 2013170147 | Hassan Raafat Hassan Morsy | Section 7 |
| 2013170176 | Zeyad Alfaris Mohammed | Section 8 |
| 2013170177 | Zeyad Etman Attia | Section 8 |
| 2013170236 | Abdelrahman Shafiq El-Morsi | Section 10 |

# Contact [hassanazzam95@gmail.com](mailto:hassanazzam95@gmail.com)