\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PLAYER CLASS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Node current\_node;

Node next\_node;

Public int exit\_dir; //direction of player when exiting a node (0 = up, 1 = right, 2 = down, 3 = left)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NODE CLASS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Arraylist<node> neighbors

Ontriggerenter() {

Player.current\_node = this.gameObject;

//update player.next\_node to be the next node in the pathfinding algorithm

}

Ontriggerexit() {

If(player.y – node.y >= node.y\_length/2)

//player is moving up

Else if(player.y – node.y <= node.y\_length/-2)

//player is moving down

Else {

If(player.x – node.x >= node.x\_length/2)

//player is moving right

Else

//player is moving left

}

//Now that we have the player’s “Exiting” direction, the passenger can react appropriately

//We can use the same (or a similar) system to find the “direction” of the player.next\_node object and compare that direction to the player’s exiting direction

//i.e. if(player.exit dir == next\_node.dir) -> player is moving in correct direction

//else -> have passenger “yell” at player and update the pathfinding algorithm

}

-On trigger exit, check the player’s transform.direction

-The “desired” angle of exit is based on the angle between the current node and “next” node

--Get the dot product between the player’s transform.direction and the desired angle

--Have a “tolerance” area (say, if(dotproduct > 0.8) and THEN respond appropriately

<http://tutorial.math.lamar.edu/Classes/CalcII/DotProduct.aspx>