

```
1 //Telling what Libraries we can get Components from.
2 using System.Collections;
3 using System.Collections.Generic;
4 using UnityEngine;
5
6 public class VillagerMovement : MonoBehaviour {
7     //Define a bunch of variables.
8     public float moveSpeed;
9     private Vector2 minWalkPoint;
10    private Vector2 maxWalkPoint;
11
12    private Rigidbody2D myRigidBody;
13
14    public bool isWalking;
15
16    public float walkTime;
17    private float walkCounter;
18    public float waitTime;
19    private float waitCounter;
20
21    private int WalkDirection;
22
23    public Collider2D walkZone;
24    private bool hasWalkZone;
25
26    public bool canMove;
27
28
29    // Use this for initialization
30    void Start () {
31        //villager bodies are now defined as Rigidbody2D
32        myRigidBody = GetComponent<Rigidbody2D>();
33
34
35        walkCounter = walkTime;
36        waitCounter = waitTime;
37
38        //once game chooses the direction
39        ChooseDirection();
40
41        if (walkZone != null)
42        {
43            //setting bounds for where the villagers can move in with a collider ➤
44            //show where they can go.
45            minWalkPoint = walkZone.bounds.min;
46            maxWalkPoint = walkZone.bounds.max;
47            hasWalkZone = true;
48        }
49
50        canMove = true;
51    }
```

```
52
53     // Update is called once per frame
54     void Update () {
55
56
57         {
58             canMove = true;
59         }
60
61         if(!canMove)
62         {
63             myRigidBody.velocity = Vector2.zero;
64             return;
65         }
66         if(isWalking)
67         {
68             walkCounter -= Time.deltaTime;
69
70             //What this script below is doing is in the random cases decribed
71             //below, each one defines a specific
72             //movement for the villager. When the Villager hits the BoxCollider2D
73             //set as a trigger, checks where it
74             //is. if the villager is outside of the box it moves it in, and if it
75             //is inside it stays inside. It is
76             //continuously checking on whether or not the villager is in, so it
77             //the villager will always stop before
78             //the villager goes outside. This is helpful for making sure villager
79             //accidently don't trip the warp
80             //feature described below.
81
82             switch(WalkDirection)
83             {
84                 case 0:
85                     myRigidBody.velocity = new Vector2(0, moveSpeed);
86                     if(hasWalkZone && transform.position.y > maxWalkPoint.y)
87                     {
88                         isWalking = false;
89                         waitCounter = waitTime;
90                     }
91                     break;
92
93                 case 1:
94                     myRigidBody.velocity = new Vector2(moveSpeed, 0);
95                     if (hasWalkZone && transform.position.x > maxWalkPoint.x)
96                     {
97                         isWalking = false;
98                         waitCounter = waitTime;
99                     }
100                     break;
101                 case 2:
102                     myRigidBody.velocity = new Vector2(0, -moveSpeed);
```

```
99         if (hasWalkZone && transform.position.y < minWalkPoint.y)
100         {
101             isWalking = false;
102             waitCounter = waitTime;
103         }
104         break;
105
106     case 3:
107         myRigidBody.velocity = new Vector2(-moveSpeed, 0);
108         if (hasWalkZone && transform.position.x < minWalkPoint.x)
109         {
110             isWalking = false;
111             waitCounter = waitTime;
112         }
113         break;
114     }
115
116     if (walkCounter < 0)
117     {
118         isWalking = false;
119         waitCounter = waitTime;
120     }
121
122 }
123 else
124 {
125     waitCounter -= Time.deltaTime;
126
127     myRigidBody.velocity = Vector2.zero;
128
129     if(waitCounter < 0)
130     {
131         ChooseDirection();
132     }
133 }
134 }
135
136 public void ChooseDirection()
137 {
138     //Essentially RNG with directions 0=up 1=right 2=down 3=left; range
139     //doesn't include 4
140     WalkDirection = Random.Range(0, 4);
141     isWalking = true;
142     walkCounter = walkTime;
143 }
144
145 }
```