```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameState
{
    public float idle1;
    public float running1;
    public static bool crouching;
    // common base class for sharing stuff (e.g. static counter variables)
    // also forces people to implement minimal functionality
    public virtual void HandlePlayerInput(PlayerEnemyState thisObject) { }
    public virtual void currentplayerstate(PlayerEnemyState thisObject) { }




}
public class Idle : GameState
{

    public override void HandlePlayerInput(PlayerEnemyState thisObject)
    {
        if (Input.GetKeyDown(thisObject.CrouchKey))
        {
            crouching = true;
                Debug.Log(thisObject.playerName + " starts crouching");
                thisObject.state = "Crouching";
                thisObject.currentState = new Crouching();
        }
       else if (Input.GetKeyDown(thisObject.WKey) || Input.GetKeyDown
         (thisObject.AKey) || Input.GetKeyDown(thisObject.SKey) ||
         Input.GetKeyDown(thisObject.DKey))
        {
            Debug.Log(thisObject.playerName + " Stops being idle");

                crouching = false;
                Debug.Log(thisObject.playerName + " starts running");
                thisObject.state = "Running";
                thisObject.currentState = new Running();
            }
        else
        {
            crouching = false;
            Debug.Log(thisObject.playerName + " is idle");
            thisObject.state = "Idle";
            thisObject.currentState = new Idle();
        }
    }
    public override void currentplayerstate(PlayerEnemyState thisObject)
    {
        Debug.Log(thisObject.playerName + " is currently idle");
```

```
52      }
53  }
54  public class Running : GameState
55  {
56      public override void HandlePlayerInput(PlayerEnemyState thisObject)
57      {
58
59          if (Input.GetKeyDown(thisObject.WKey) || Input.GetKeyDown
              (thisObject.AKey) || Input.GetKeyDown(thisObject.SKey) ||
              Input.GetKeyDown(thisObject.DKey))
60          {
61              if (Input.GetKeyDown(thisObject.CrouchKey))
62              {
63                  crouching = true;
64                  Debug.Log(thisObject.playerName + " starts crouchwalking");
65                  thisObject.state = "CrouchWalking";
66                  thisObject.currentState = new CrouchWalking();
67              }
68              else
69              {
70                  crouching = false;
71                  Debug.Log(thisObject.playerName + " starts running");
72                  thisObject.state = "Running";
73                  thisObject.currentState = new Running();
74              }
75
76          }
77          //When the player stops pressing running key
78          else if (Input.GetKeyUp(thisObject.WKey) || Input.GetKeyUp
              (thisObject.AKey) || Input.GetKeyUp(thisObject.SKey) ||
              Input.GetKeyUp(thisObject.DKey))
79          {
80              crouching = false;
81              Debug.Log(thisObject.playerName + " stops running");
82              thisObject.state = "Idle";
83              thisObject.currentState = new Idle();
84          }
85      }
86      public override void currentplayerstate(PlayerEnemyState thisObject)
87      {
88          Debug.Log(thisObject.playerName + " is currently running");
89
90      }
91  }
92  public class Crouching : GameState
93  {
94      public override void HandlePlayerInput(PlayerEnemyState thisObject)
95      {
96          if (Input.GetKeyUp(thisObject.CrouchKey))
97          {
98                  crouching = false;
99                  Debug.Log(thisObject.playerName + " stops crouching");
100                 thisObject.state = "Idle";
```

```
101                 thisObject.currentState = new Idle();
102         }
103
104         if (Input.GetKeyDown(thisObject.WKey) || Input.GetKeyDown
              (thisObject.AKey) || Input.GetKeyDown(thisObject.SKey) ||
              Input.GetKeyDown(thisObject.DKey))
105         {
106                 crouching = true;
107                 Debug.Log(thisObject.playerName + " crouchwalking");
108                 thisObject.state = "CrouchWalking";
109                 thisObject.currentState = new CrouchWalking();
110         }
111
112
113     }
114     public override void currentplayerstate(PlayerEnemyState thisObject)
115     {
116         Debug.Log(thisObject.playerName + " is currently crouching");
117
118     }
119 }
120 public class CrouchWalking : GameState
121 {
122     public override void HandlePlayerInput(PlayerEnemyState thisObject)
123     {
124         if (Input.GetKeyDown(thisObject.WKey) || Input.GetKeyDown
              (thisObject.AKey) || Input.GetKeyDown(thisObject.SKey) ||
              Input.GetKeyDown(thisObject.DKey))
125         {
126             if (Input.GetKey(thisObject.CrouchKey))
127             {
128                 crouching = true;
129                 Debug.Log(thisObject.playerName + " crouchwalking");
130                 thisObject.state = "CrouchWalking";
131                 thisObject.currentState = new CrouchWalking();
132             }
133             else
134             {
135                 crouching = false;
136                 Debug.Log(thisObject.playerName + " stops crouching");
137                 Debug.Log(thisObject.playerName + " starts running");
138                 thisObject.state = "Running";
139                 thisObject.currentState = new Running();
140             }
141
142         }
143         else if (Input.GetKeyUp(thisObject.WKey) || Input.GetKeyUp
              (thisObject.AKey) || Input.GetKeyUp(thisObject.SKey) ||
              Input.GetKeyUp(thisObject.DKey))
144         {
145             if (Input.GetKey(thisObject.CrouchKey))
146             {
147                 crouching = true;
```

```
148                 thisObject.currentState = new Crouching();
149                 thisObject.state = "Crouching";
150                 Debug.Log(thisObject.playerName + " is crouching");
151             }
152             else
153             {
154                 crouching = false;
155                 Debug.Log(thisObject.playerName + " stops crouching");
156                 thisObject.state = "Idle";
157                 thisObject.currentState = new Idle();
158             }
159             }
160     }
161     public override void currentplayerstate(PlayerEnemyState thisObject)
162     {
163 
164         Debug.Log(thisObject.playerName + " is currently crouchwalking");
165     }
166 }
167 //AI Game States
168 public class AIGameState  : GameState
169 {
170     public static bool AIPatrolling;
171     public static bool AISuspects;
172   //  public static float AIhypot;
173     // common base class for sharing stuff (e.g. static counter variables)
174     // also forces people to implement minimal functionality
175     public virtual void HandleAIEnemyInput(PlayerEnemyState thisAIObject)
        { }
176     public virtual void CurrentEnemyAIstate(PlayerEnemyState thisAIObject)
        { }
177 }
178 public class AIPatrol : AIGameState
179 {
180     public override void HandleAIEnemyInput(PlayerEnemyState thisAIObject)
181     {
182         if (PlayerEnemyState.hypotenuse < 10.0f &&
            PlayerEnemyState.hypotenuse > -10.0f)
183         {
184 
185             if (GameState.crouching == true)
186             {
187                 AIPatrolling = true;
188                 AISuspects = false;
189                 thisAIObject.AIstate = "Patrol";
190                 thisAIObject.currentAIState = new AIPatrol();
191             }
192             else
193             {
194                 AIPatrolling = false;
195                 AISuspects = true;
196                 thisAIObject.AIstate = "Suspicious";
197                 thisAIObject.currentAIState = new AISuspicious();
```

```
198                 thisAIObject.AIstate = "Suspicious";
199             }
200             }
201         else
202         {
203             AIPatrolling = true;
204             AISuspects = false;
205             thisAIObject.AIstate = "Patrol";
206             thisAIObject.currentAIState = new AIPatrol();
207         }
208     }
209     public override void CurrentEnemyAIstate(PlayerEnemyState thisAIObject)
210     {
211         Debug.Log(thisAIObject.AIName + "Enemy patrolling");
212     }
213 }
214 public class AISuspicious : AIGameState
215 {
216     public override void HandleAIEnemyInput(PlayerEnemyState thisAIObject)
217     {
218         if (PlayerEnemyState.hypotenuse > 10.0f ||
              PlayerEnemyState.hypotenuse < -10.0f)
219         {
220                 AIPatrolling = true;
221                 AISuspects = false;
222                 thisAIObject.AIstate = "Patrolling";
223                 thisAIObject.currentAIState = new AIPatrol();
224                 thisAIObject.AIstate = "Patrolling";
225         }
226        else if (GameState.crouching == true)
227         {
228             AIPatrolling = true;
229             AISuspects = false;
230             thisAIObject.AIstate = "Patrolling";
231             thisAIObject.currentAIState = new AIPatrol();
232             thisAIObject.AIstate = "Patrolling";
233         }
234         else
235         {
236             AIPatrolling = false;
237             AISuspects = true;
238             thisAIObject.currentAIState = new AISuspicious();
239             thisAIObject.AIstate = "Suspicious";
240         }
241     }
242     public override void CurrentEnemyAIstate(PlayerEnemyState thisAIObject)
243     {
244         Debug.Log(thisAIObject.AIName + "Enemy suspicious");
245     }
246 }
247 public class PlayerEnemyState : MonoBehaviour
248 {    //Animation Variables
249      //This is a patrol script for moving enemies/objects from point to
```

```
        point
250
251     //Animation Variables
252     public Animator animatonset;
253     //GameObject Variables to store patrol positions
254     public GameObject patrolpoint1,patrolpoint2,patrolpoint3,patrolpoint4;
255     public GameObject enemy;
256     public GameObject respawnpoint;
257     //speed variable is used for the MoveTowards function and controls how
          fast the object will move between vectors
258     public float speed;
259     //Vector variables are used to sotre the position of objects in this
          instance
260     private Vector3
          currentpos,patrolpos1,patrolpos2,patrolpos3,patrolpos4,enemyrespawnpos
          ;
261     //Bool positions check to see if the enemy has reached a patrol point
262     private bool pos1;
263     private bool pos2;
264     private bool pos3;
265     private bool pos4;
266     //Respawn variables
267     public float falllimit;
268     //GameState Variables
269     public GameState currentState;
270     public AIGameState currentAIState;
271     //KeyCode Variables
272     public KeyCode WKey,AKey,SKey,DKey,CrouchKey,ShootKey;
273     public string playerName;
274     public string AIName;
275     public string state;
276     public string AIstate;
277
278     //Hypotenuse variables
279     [SerializeField]
280     public static float hypotenuse;
281     public float adjacent;
282     public float opposite;
283     private float enemyadjacent;
284     private float enemyopposite;
285
286     public GameObject player;
287
288     public Vector3 playerpos1;
289     public Vector3 enemypos1;
290
291     //PlayerGameState
292     // Start is called before the first frame update
293     void Start()
294     {
295
296         //Retrieves animator component and sets it equal to variable in
            scripts
```

```
297         animatonset = GetComponent<Animator>();
298         //Variables are set to make sure code runs properly
299         //Positions are set to false as they have not been reached by enemy
              yet
300         pos1 = false;
301         pos2 = false;
302         pos3 = false;
303         pos4 = false;
304         speed = 5.0f;
305         patrolpos1 = patrolpoint1.transform.position;
306         patrolpos2 = patrolpoint2.transform.position;
307         patrolpos3 = patrolpoint3.transform.position;
308         patrolpos4 = patrolpoint4.transform.position;
309         //  enemy.transform.position = new Vector3(patrolpos1.x,
              patrolpos1.y, patrolpos1.z);
310
311         //Sets enemy to spawn at position 1
312         currentpos = patrolpos1;
313         enemy.transform.position = currentpos;
314         WKey = KeyCode.W;
315         AKey = KeyCode.A;
316         SKey = KeyCode.S;
317         DKey = KeyCode.D;
318         CrouchKey = KeyCode.C;
319         ShootKey = KeyCode.Mouse1;
320         playerName = "Jeff";
321         AIName = "Enemy";
322         currentState = new Idle();
323         currentAIState = new AIPatrol();
324         InvokeRepeating("PlayerReport", 0.0f, 3.0f);
325
326         AIGameState.AISuspects = false;
327         AIGameState.AIPatrolling = true;
328     }
329
330     // Update is called once per frame
331     void Update()
332     {
333         playerpos1 = player.transform.position;
334         enemypos1 = enemy.transform.position;
335         CalculateHypotenuse();
336         PatrolArea();
337         currentState.HandlePlayerInput(this);
338         currentAIState.HandleAIEnemyInput(this);
339     }
340     //Used instead of seperate respawn script for ease of use
341     void EnemyRespawn()
342     {
343         if (RoomAttributes.enemyalive == false)
344         {
345             Object.Instantiate(enemy);
346             enemyrespawnpos = respawnpoint.transform.position;
347             enemy.transform.position = enemyrespawnpos;
```

```
348             RoomAttributes.enemyalive = true;
349         }
350     }
351     //Rotates the enemy
352     void SetAIRotation(int x, int y, int z)
353     {
354         animatonset.rootRotation = Quaternion.Euler(new Vector3(x, y, z));
355         transform.rotation = Quaternion.Euler(new Vector3(x, y, z));
356         animatonset.SetBool("IsWalking", true);
357     }
358     void StopAI()
359     {
360         //animatonset.rootRotation = Quaternion.Euler(new Vector3(0, 0, 0));
361         //transform.rotation = Quaternion.Euler(new Vector3(0, 0, 0));
362         animatonset.SetBool("IsWalking", false);
363     }
364     void PatrolArea()
365     {
366     if (AIPatrol.AIPatrolling == true)
367     {
368
369         //This if statement checks if the enemy has been at a patrol point
              and set the enemy to move to the first point
370         if (pos1 == false && pos2 == false && pos3 == false && pos4 ==
              false)
371         {
372
373             //Moves the vector current position towards a specfic vector
374             currentpos = Vector3.MoveTowards(currentpos, patrolpos1, speed *
                  Time.deltaTime);
375             //Moves the enemy to the position of the current pos vector
376             enemy.transform.position = currentpos;
377             //Debug.Log("Moving");
378         }
379
380         //Once the enemy reachs the first patrol point the code above --^ is
                no longer used due to pos1 being set to true. It then moves the
              enemy to the next patrol point
381         if (currentpos == patrolpos1 || pos1 == true)
382         {
383             pos4 = false;
384             pos1 = true;
385             currentpos = Vector3.MoveTowards(currentpos, patrolpos2, speed *
                  Time.deltaTime);
386             enemy.transform.position = currentpos;
387             //Debug.Log("Moving to pos 2");
388             SetAIRotation(0, 0, 0);
389         }
390
391         //The second patrol point moves the enemy to the third patrol point
392         if (currentpos == patrolpos2 || pos2 == true)
393         {
394             pos1 = false;
```

```
395             pos2 = true;
396             currentpos = Vector3.MoveTowards(currentpos, patrolpos3, speed * ↪
                    Time.deltaTime);
397             enemy.transform.position = currentpos;
398             //Debug.Log("Moving to pos 3");
399             SetAIRotation(0, 270, 0);
400         }
401         //The third patrol point moves the enemy to the fourth patrol point
402         if (currentpos == patrolpos3 || pos3 == true)
403         {
404             pos2 = false;
405             pos3 = true;
406             currentpos = Vector3.MoveTowards(currentpos, patrolpos4, speed * ↪
                    Time.deltaTime);
407             enemy.transform.position = currentpos;
408             // Debug.Log("Moving to pos 4");
409             SetAIRotation(0, 180, 0);
410         }
411         //The second patrol point moves the enemy back to the first patrol   ↪
              point
412         if (currentpos == patrolpos4 || pos4 == true)
413         {
414             pos3 = false;
415             pos4 = true;
416             currentpos = Vector3.MoveTowards(currentpos, patrolpos1, speed * ↪
                    Time.deltaTime);
417             enemy.transform.position = currentpos;
418             //   Debug.Log("Moving to pos 1");
419             SetAIRotation(0, 90, 0);
420         }
421         EnemyRespawn();
422     }
423     else if (AIGameState.AISuspects == true)
424     {
425             currentpos = enemy.transform.position;
426               //  currentpos = Vector3.MoveTowards(currentpos, playerpos1,   ↪
                    speed * Time.deltaTime);
427                enemy.transform.position = currentpos;
428             StopAI();
429             }
430     }
431
432
433     void PlayerReport()
434     {
435         currentState.currentplayerstate(this);
436         currentAIState.HandleAIEnemyInput(this);
437     }
438     //Used instead of seperate respawn script for ease of use
439
440     void CalculateHypotenuse()
441     {
442         //Calculates vector from enemy to player
```

```
443         //Enemy and player positions
444         opposite = playerpos1.x;
445         adjacent = playerpos1.z;
446         enemyadjacent = enemypos1.x;
447         enemyopposite = enemypos1.z;
448         //  opposite = 10.0f;
449         //     adjacent = 10.0f;
450         //Calculates Hypotenuse between enemy and player
451         hypotenuse = (Mathf.Sqrt(Mathf.Pow(opposite,2) + Mathf.Pow
              (adjacent,2))) - (Mathf.Sqrt(Mathf.Pow(enemyopposite, 2) +
              Mathf.Pow(enemyadjacent, 2)));
452         Debug.Log("Hypotenuse"+ hypotenuse);
453        // AIGameState.AIhypot = hypotenuse;
454     }
455
456
457 }
458
```