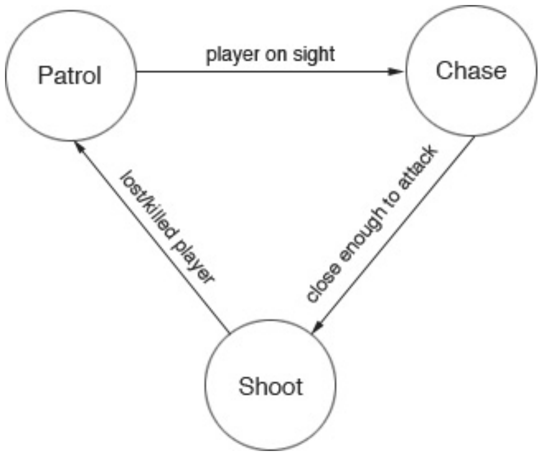
Demo: Basic Finite State Machine AI Project in Unity

Finite State Machines (**FSMs**) consists of a finite number of **states** that are connected by one or more **transitions**, resulting in a data structure known as a **graph**. Each game entity starts with an initial state. Then, environment events trigger specific rules that will make the entity move into another state. Such triggering rules are called **transitions**. A game entity can only be in one state at any given time.



There are four components in a simple FSM:

- **States**: This component defines a set of states that a game entity or an NPC can choose from (**Patrol**, **Chase**, and **Shoot**).
- **Transitions**: This component defines the relationships between different states.
- **Rules**: This component defines when to perform a state transition (**Player in sight**, **Close enough to attack**, and **Lost/killed player**).
- **Events**: This is the component that will trigger to check the rules (the guard's visible area, distance to the player, and so on).

from Unity Artificial Intelligence Programming - Fifth Edition, Dr. Davide Aversa, Packt, 2022

Hands-On

Let's implement a simple FSM in Unity.

1. Make a **new 3D project** in Unity.

2. Import **FPSController.unitypackage** (available on the LMS).

3. In Assets, make a **Resources** folder and in it make **two Materials**:
    a. name one **NormalMaterial** with a dark blue Albedo color,
    b. and name the other **SeePlayerMaterial** with a light red Albedo color, and for added fun check "Emissions" and set the color below THAT to bright red, too.

4. Create a little arena:
    a. A **Plane** at xyz position 0, 0, 0.

   b. From Assets > FirstPersonCharacter, drag an **FPSController** prefab to one corner of the Plane, at xyz position –4, 1, -4.
   c. Create a Cube at xyz position 4, 1, 4. Name it "**NPC**".
   d. Drag the **NormalMaterial** onto the NPC.
   e. Test the game. Be sure the FPS Controller can walk over to the NPC.

5. In the Assets folder, create a new C# script and name it **NPCBehavior.cs**.

6. Drag that script onto the NPC Cube.

7. Open it into your preferred editor and add an FSM setup to the code:

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

public class NPCBehavior : MonoBehaviour
{
    public enum FSMState
    {
        SitLikeALump,
        SeesPlayer
    }

    //Current state that the NPC is reaching
    public FSMState curState = FSMState.SitLikeALump;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        switch (curState)
        {
            case FSMState.SitLikeALump:
                UpdateSitLikeALump();
                break;
            case FSMState.SeesPlayer:
                UpdateSeesPlayer();
                break;
        }
    }

    /// <summary>
```

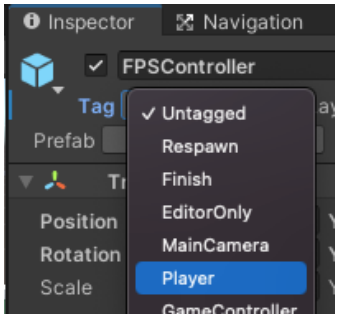
```
        /// SitLikeALump state
        /// </summary>
        protected void UpdateSitLikeALump()
        {
        }

        /// <summary>
        /// SeesPlayer state
        /// </summary>
        protected void UpdateSeesPlayer()
        {
        }

    }
```

Here we only set up two states: SitsLikeALump and SeesPlayer, but obviously you could add a variety of different states. The `Update` methods' switch statement ensures that either the `UpdateSitLikeALump` or `UpdateSeesPlayer` method will run many times per second. Currently there is nothing to change the states from one to another. As an example, we will have this code detect the player, and switch states depending on how close the player is...

8. Back in the Unity Hierarchy, select **FPSController** and in the Inspector, in the top left corner change the "**Tag**" dropdown to say "**Player**" instead of "Untagged." (You may need to use "Add Tag" at the bottom of the dropdown to create that tag first).



9. Add code to **NPCBehavior.cs**:

```
    [--etc--]
        //Current state that the NPC is reaching
        public FSMState curState = FSMState.SitLikeALump;

        //Player Transform
        private Transform playerTransform;

        // Start is called before the first frame update
        void Start()
        {
            //Get the target enemy(Player)
            GameObject objPlayer =
    GameObject.FindGameObjectWithTag("Player");
            playerTransform = objPlayer.transform;
        }

        // Update is called once per frame
         [--etc--]

        /// <summary>
        /// SitLikeALump state
        /// </summary>
        protected void UpdateSitLikeALump()
        {
            //Check the distance with the player
            float dist = Vector3.Distance(transform.position,
    playerTransform.position);
            if (dist < 2)
            {
                Debug.Log("CHANGING TO FSMState.SeesPlayer!!!");
                curState = FSMState.SeesPlayer;
            }
        }

        /// <summary>
        /// SeesPlayer state
        /// </summary>
        protected void UpdateSeesPlayer()
        {
            //Check the distance with the player
            float dist = Vector3.Distance(transform.position,
    playerTransform.position);
            if (dist > 2)
            {
                Debug.Log("CHANGING TO FSMState.SitLikeALump!!!");
                curState = FSMState.SitLikeALump;
            }
        }
    }
```

Test the project. The NPC Cube just sits, but it logs "`CHANGING TO FSMState.SeesPlayer`" messages to the console when the Player comes near, then logs "`CHANGING TO FSMState.SitLikeALump`" messages to the console when the Player moves away. You can and should have the NPC do more than log messages. For example, let's have the NPC change which colorful material it wears:

10. Add code to **NPCBehavior.cs:**

```
[--etc--]
 private Transform playerTransform;

 //Materials
 Material matSit;
 Material matSee;

 // Start is called before the first frame update
 void Start()
 {
     //Get the target enemy(Player)
     GameObject objPlayer = GameObject.FindGameObjectWithTag("Player");
     playerTransform = objPlayer.transform;
```

```
        //Load materials once:
        matSit = Resources.Load<Material>("NormalMaterial");
        matSee = Resources.Load<Material>("SeePlayerMaterial");
    }

   [--etc--]
   protected void UpdateSitLikeALump()
    {
        //Check the distance with the player
        float dist = Vector3.Distance(transform.position, playerTransform.position);
        if (dist < 2)
        {
            Debug.Log("CHANGING TO FSMState.SeesPlayer!!!");
            curState = FSMState.SeesPlayer;
            //change color:
            GetComponent<Renderer>().material = matSee;
        }
    }

    /// <summary>
    /// SeesPlayer state
    /// </summary>
    protected void UpdateSeesPlayer()
    {
        //Check the distance with the player
        float dist = Vector3.Distance(transform.position, playerTransform.position);
        if (dist > 2)
        {
            Debug.Log("CHANGING TO FSMState.SitLikeALump!!!");
            curState = FSMState.SitLikeALump;
            //change color:
            GetComponent<Renderer>().material = matSit;
        }
    }

}
```

11. Test the project. The NPC Cube still just sits, but turns a red material when the Player comes near, then turns a blue material when the Player moves away. You could get the Cube to move toward or away from, the player using the code we reviewed in our Code Warmup activity.

END DEMO