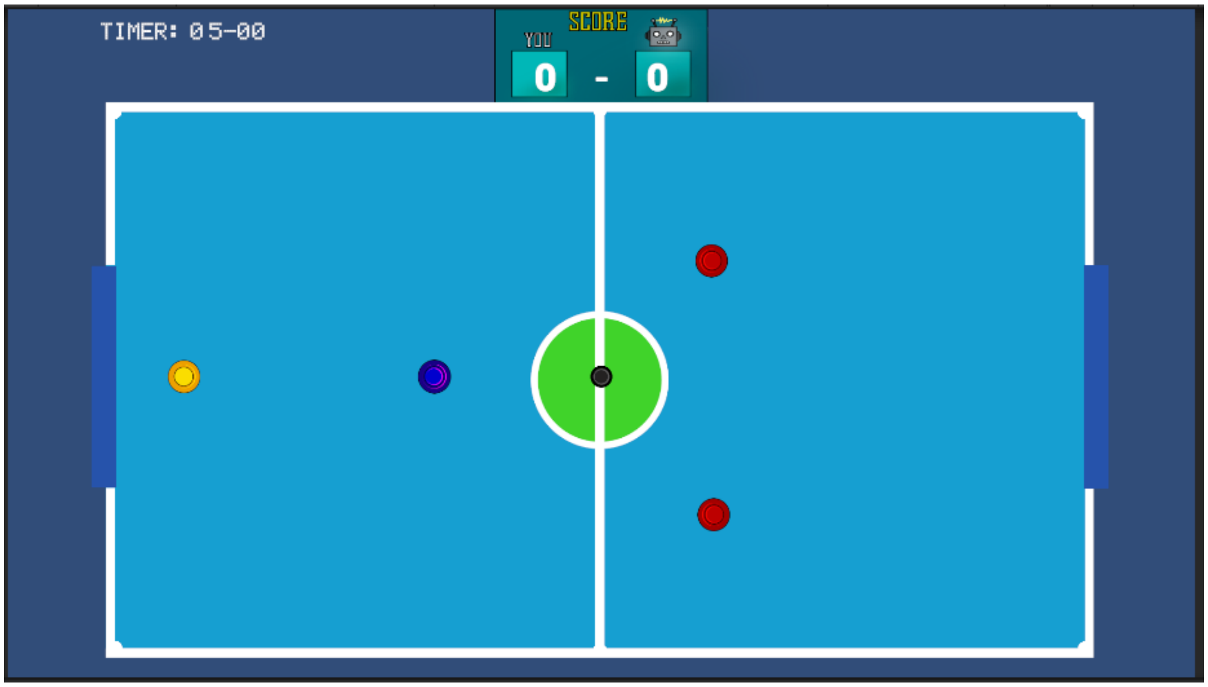
PS-1 GameDev Challenge

Hall-2 Kshatriyas

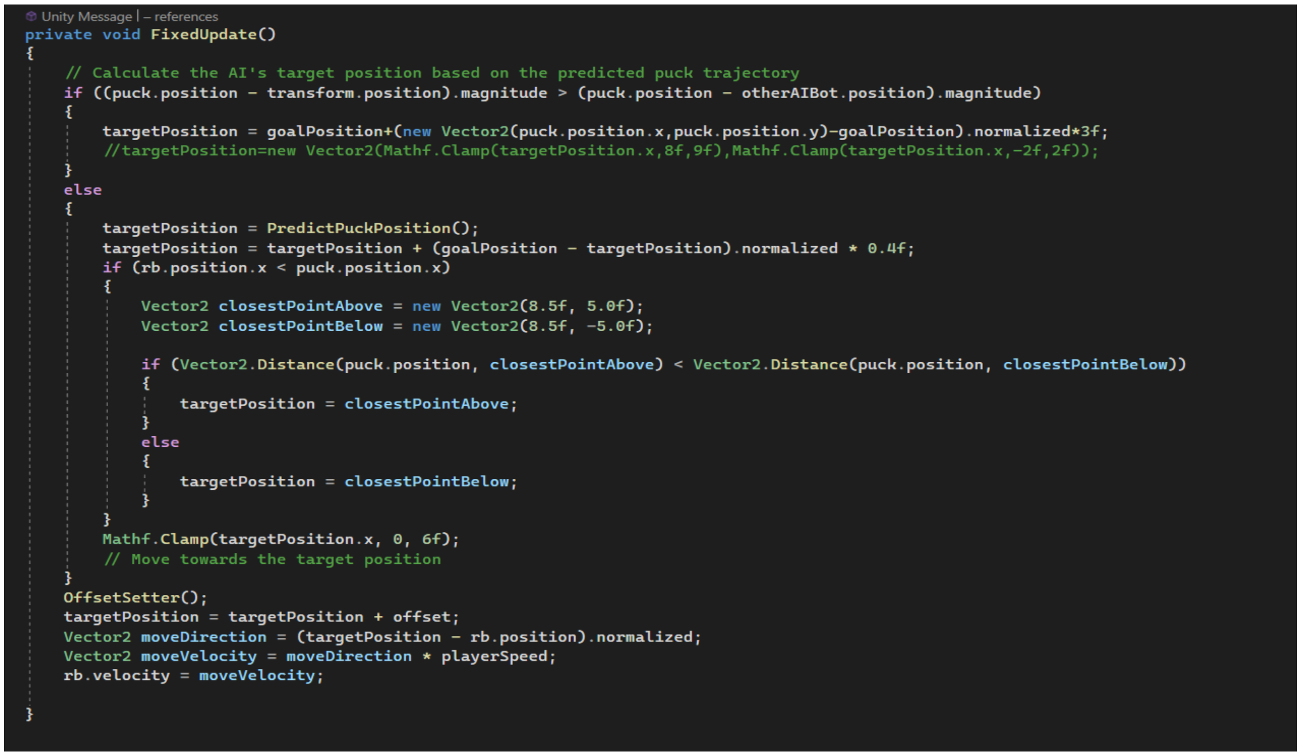
We have made a game based on air hockey, in which we have added AI(artificial intelligence)

As our opponent which will defend our movement and in brief we have two AI as our opponent and one as our companion .



For the AI implementation, the bots calculate the predicted position of the puck in the time taken by the bot to reach the puck.

And the target position is updated with each frame.



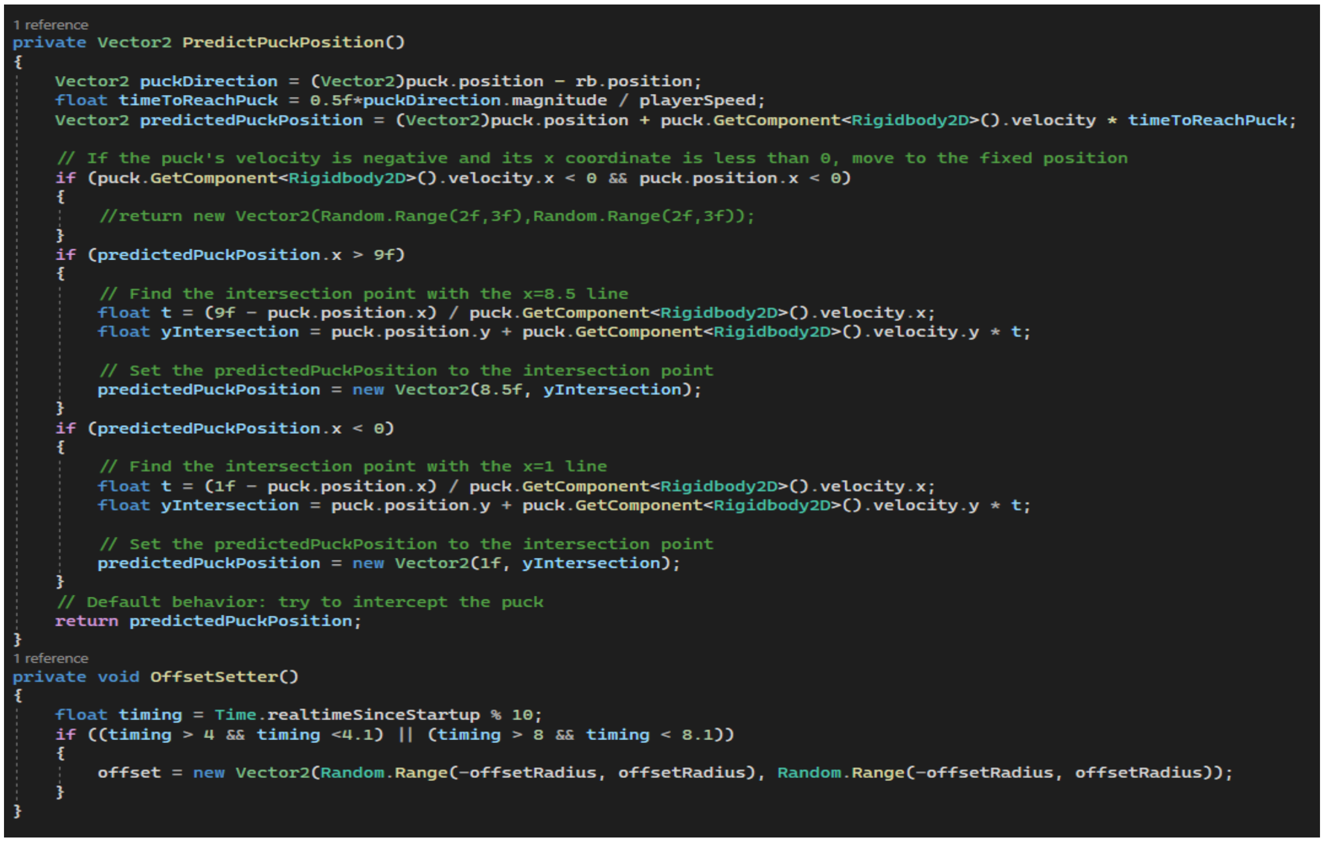
```
Unity Message | – references
private void FixedUpdate()
{
    // Calculate the AI's target position based on the predicted puck trajectory
    if ((puck.position - transform.position).magnitude > (puck.position - otherAIBot.position).magnitude)
    {
        targetPosition = goalPosition+(new Vector2(puck.position.x,puck.position.y)-goalPosition).normalized*3f;
        //targetPosition=new Vector2(Mathf.Clamp(targetPosition.x,8f,9f),Mathf.Clamp(targetPosition.x,-2f,2f));
    }
    else
    {
        targetPosition = PredictPuckPosition();
        targetPosition = targetPosition + (goalPosition - targetPosition).normalized * 0.4f;
        if (rb.position.x < puck.position.x)
        {
            Vector2 closestPointAbove = new Vector2(8.5f, 5.0f);
            Vector2 closestPointBelow = new Vector2(8.5f, -5.0f);

            if (Vector2.Distance(puck.position, closestPointAbove) < Vector2.Distance(puck.position, closestPointBelow))
            {
                targetPosition = closestPointAbove;
            }
            else
            {
                targetPosition = closestPointBelow;
            }
        }
        Mathf.Clamp(targetPosition.x, 0, 6f);
        // Move towards the target position
    }
    OffsetSetter();
    targetPosition = targetPosition + offset;
    Vector2 moveDirection = (targetPosition - rb.position).normalized;
    Vector2 moveVelocity = moveDirection * playerSpeed;
    rb.velocity = moveVelocity;

}
```

Regarding the difficulty levels, we have three levels in the game. The difference is in the accuracy of the bots. We have introduced an offset variable which reduces as the difficulty

increases, so the error in the target position is minimised as the difficulty increases. Also the player and the movement speeds of the player vary adversely as the difficulty increases.



```
1 reference
private Vector2 PredictPuckPosition()
{
    Vector2 puckDirection = (Vector2)puck.position - rb.position;
    float timeToReachPuck = 0.5f*puckDirection.magnitude / playerSpeed;
    Vector2 predictedPuckPosition = (Vector2)puck.position + puck.GetComponent<Rigidbody2D>().velocity * timeToReachPuck;

    // If the puck's velocity is negative and its x coordinate is less than 0, move to the fixed position
    if (puck.GetComponent<Rigidbody2D>().velocity.x < 0 && puck.position.x < 0)
    {
        //return new Vector2(Random.Range(2f,3f),Random.Range(2f,3f));
    }
    if (predictedPuckPosition.x > 9f)
    {
        // Find the intersection point with the x=8.5 line
        float t = (9f - puck.position.x) / puck.GetComponent<Rigidbody2D>().velocity.x;
        float yIntersection = puck.position.y + puck.GetComponent<Rigidbody2D>().velocity.y * t;

        // Set the predictedPuckPosition to the intersection point
        predictedPuckPosition = new Vector2(8.5f, yIntersection);
    }
    if (predictedPuckPosition.x < 0)
    {
        // Find the intersection point with the x=1 line
        float t = (1f - puck.position.x) / puck.GetComponent<Rigidbody2D>().velocity.x;
        float yIntersection = puck.position.y + puck.GetComponent<Rigidbody2D>().velocity.y * t;

        // Set the predictedPuckPosition to the intersection point
        predictedPuckPosition = new Vector2(1f, yIntersection);
    }
    // Default behavior: try to intercept the puck
    return predictedPuckPosition;
}
1 reference
private void OffsetSetter()
{
    float timing = Time.realtimeSinceStartup % 10;
    if ((timing > 4 && timing <4.1) || (timing > 8 && timing < 8.1))
    {
        offset = new Vector2(Random.Range(-offsetRadius, offsetRadius), Random.Range(-offsetRadius, offsetRadius));
    }
}
```