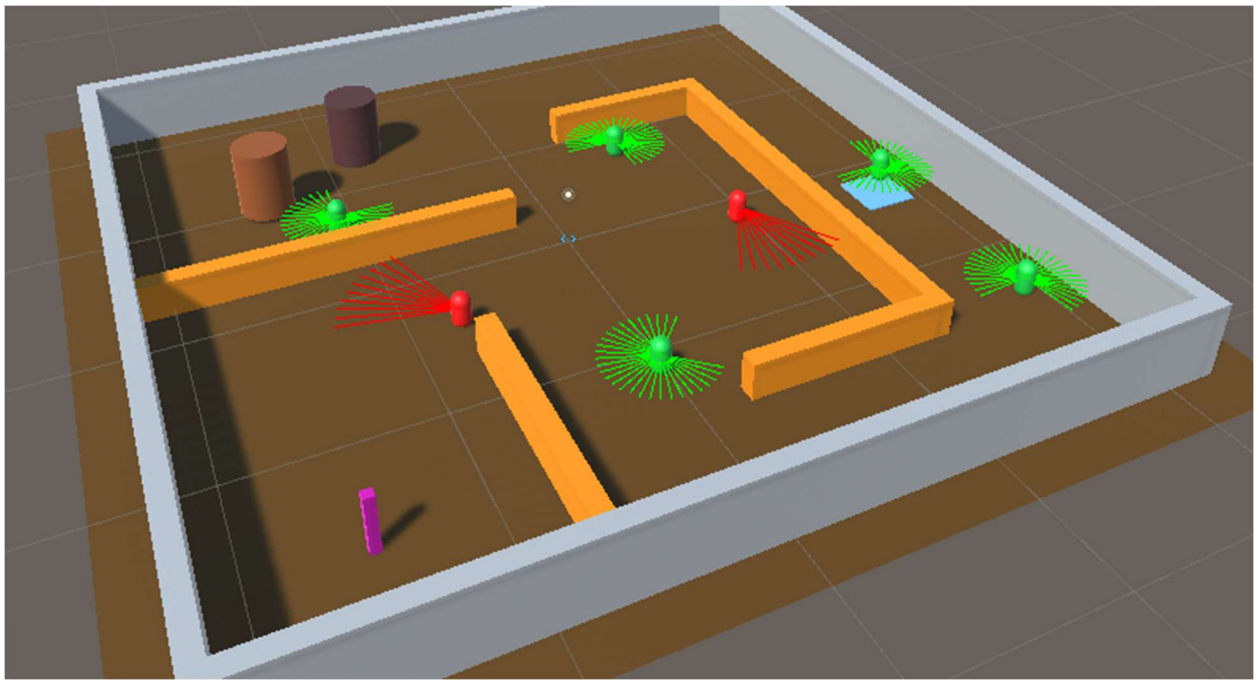
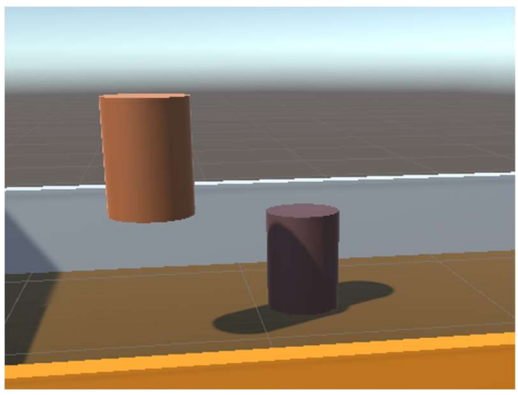
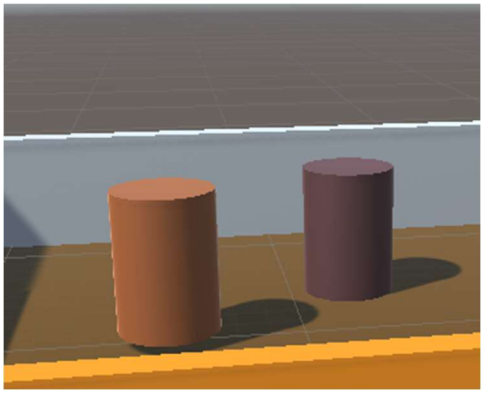
Patrick Cheng

For this lab, the overall difficulty was between medium and high. There were a lot of ways to do the prey and predator behavior. I used ray casting for my scripts. The predator use ray cast to find prey and chases the prey if the ray hits one, the predator can also avoid obstacles with the ray. The prey use ray cast to avoid obstacles and the predator. The prey has a wider range of view than the predator.



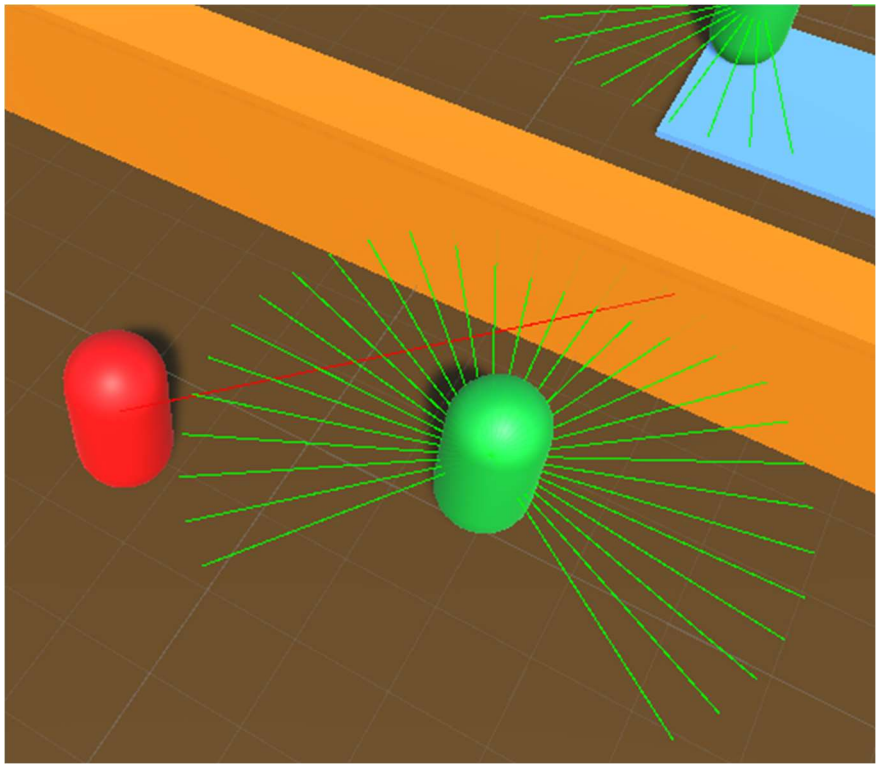
For the scene, I have obstacles that creatures cannot move through. There are open areas and dead ends (around the pink pole). The blue square is a choke point that is also the respawn point of prey. The brown cylinder will move up and down to interfere with the prey and predator.



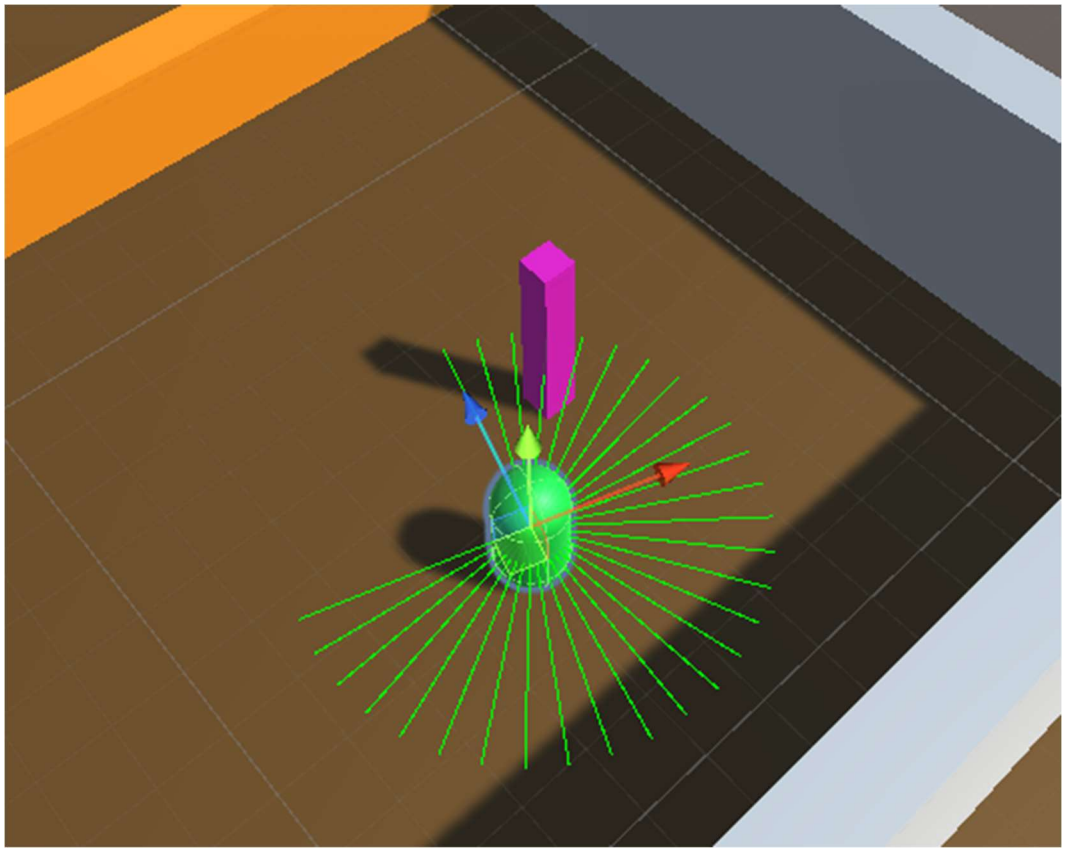


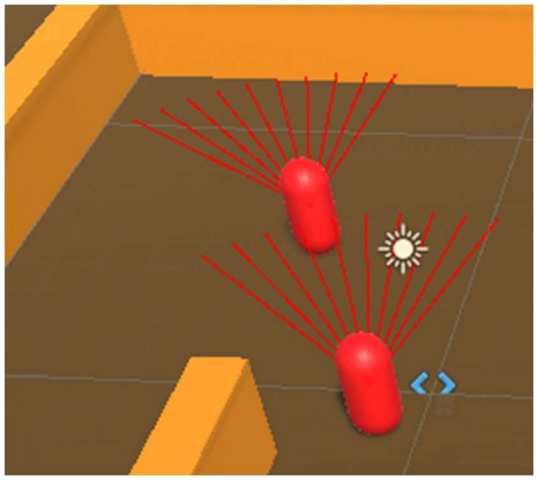
The predator will chase the prey when the prey is within the predators range, the prey will try and avoid the predator. The prey's vision for predator is slightly shorter than the vision for detecting wall.

The predator has a cone field of vision with a range of 6, the prey has 270 degrees of vision with range of 1.5. When the predator sees the prey, the vision will combine into one ray and start chasing the prey. If the predator catches up with the prey it will then eat the prey and the prey will go back to respawn.



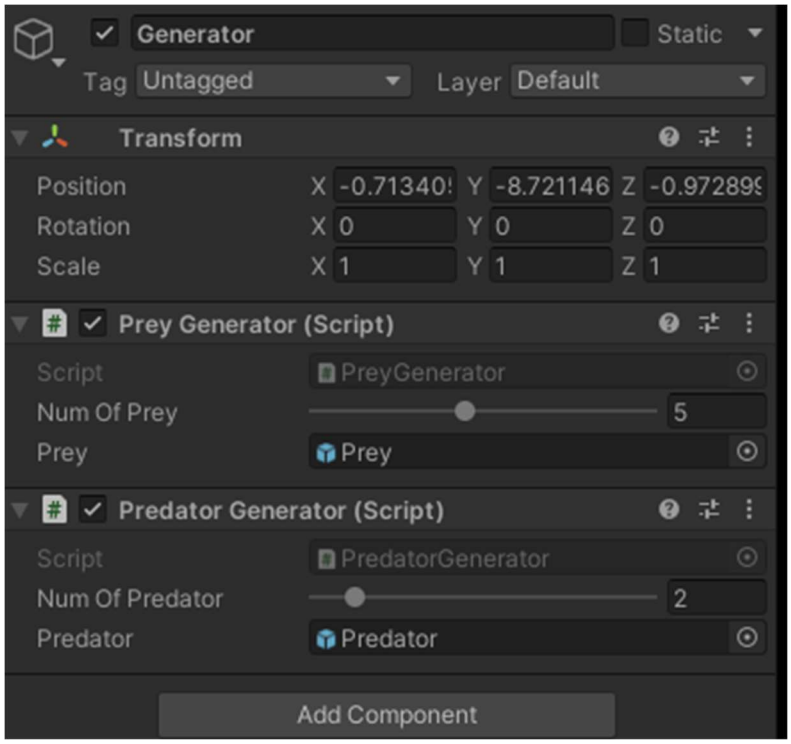
For additional features, I did visible field of vision for both agents using ray casting draw ray in the debugger. I added a goal which is the pink pole, when the prey gets near the pole the prey will wander around the goal pole. I also have flocking, but it's for the predator not the prey.



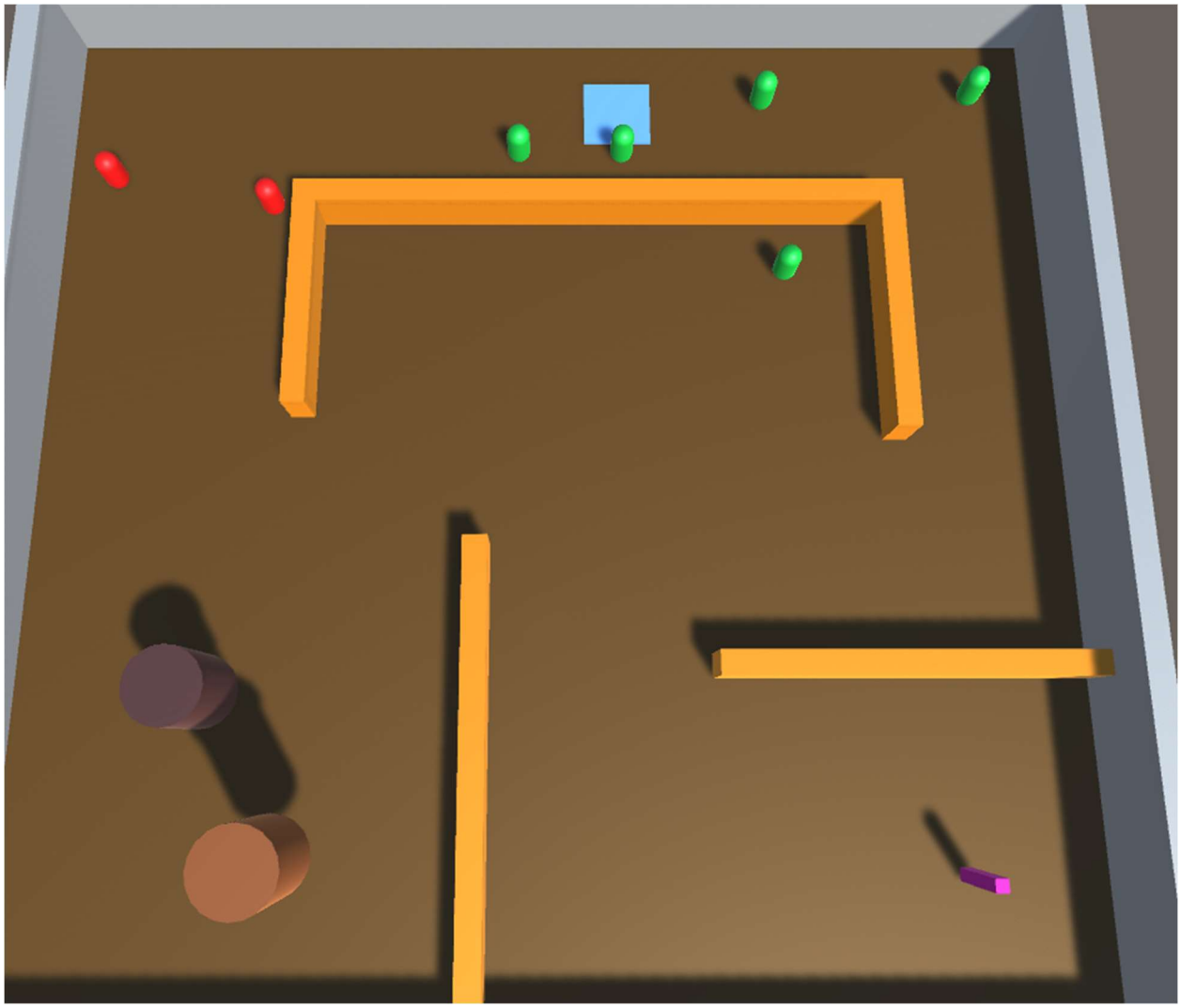


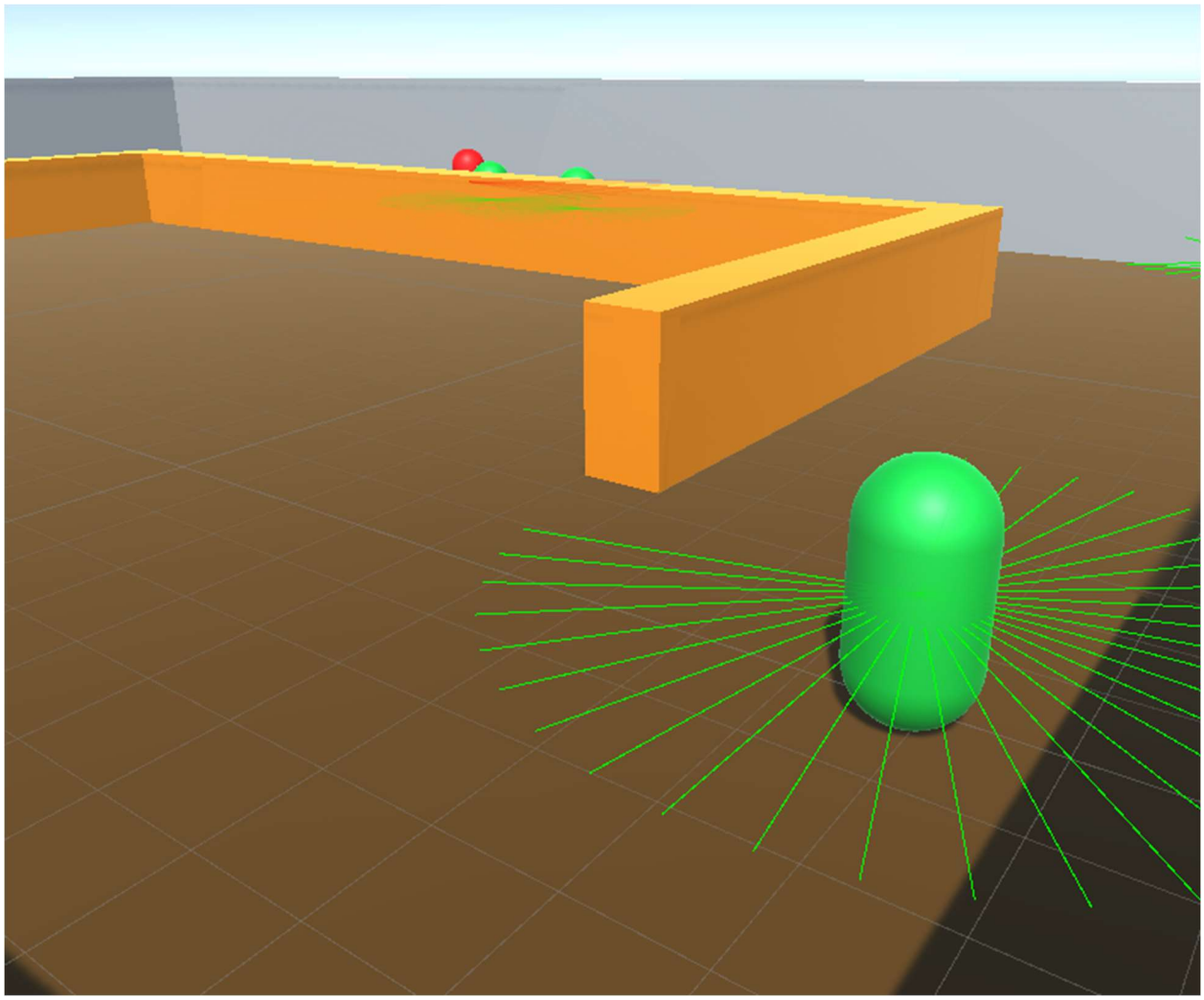
Additional:

There's an object called generator that can control how many preys and predators to generate, default is 5 preys and 2 predators.



Screen Shots:





Source Code:

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

namespace ChengPatrick.lab5
{
    public class PredatorBehavior : MonoBehaviour
    {
        public float speed = 15f;
        public float rotateSpeed = 1f;
        public float wanderRotate = 30f;
        public int numOfRays = 10;
        private float angle = 60;
        public float rayRange = 1.5f;

        private bool isWandering = false;
        private bool startWandering = false;
        private bool isTurning = false;
        private int rotateDirection = 1;
        private float rotateStep;
        private float org_ang;

        // Start is called before the first frame update
        void Start()
        {
            // use this so speed is not affected
            org_ang = angle;
        }

        // Update is called once per frame
        void Update()
        {
            var deltaPosition = Vector3.zero;
            angle = org_ang;

            this.transform.eulerAngles = new Vector3(0, this.transform.eulerAngles.y, 0);

            if (this.transform.position.y > 2)
            {
                this.transform.position = new Vector3(0, 1, 0);
                this.transform.rotation = Quaternion.Euler(0, 0, 0);
            }

            for (int i = 0; i < numOfRays; ++i)
            {
                var rotation = this.transform.rotation;
                var rotationMod = Quaternion.AngleAxis(i / ((float)numOfRays - 1) *
org_ang, this.transform.up);
                var direction = rotation * rotationMod * Vector3.forward;
                // for focusing onto one prey
                var temp_mod = rotationMod;
                var temp_dir = direction;

                Vector3 agentToVertext;
                Quaternion look;
```

```
                var ray = new Ray(this.transform.position, direction);
                RaycastHit hitInfo;
                if (Physics.Raycast(ray, out hitInfo, rayRange))
                {
                    // check if is prey, if not move away
                    if (hitInfo.collider.gameObject.transform.parent.gameObject.name !=
"Prey")
                    {
                        // rotate away from collide object
                        look = Quaternion.LookRotation((hitInfo.point -
this.transform.position).normalized);
                        this.transform.rotation = Quaternion.Slerp(transform.rotation,
look, Time.deltaTime * rotateSpeed);
                    }
                    // if is, chase
                    else
                    {
                        // combine ray into one
                        GameObject target = hitInfo.transform.gameObject;
                        angle = Mathf.SmoothStep(org_ang, 0, 1);
                        StartCoroutine(chaseTarget(target));

                        // rotate towards prey
                        agentToVertext = hitInfo.transform.position -
this.transform.position;

                        //this.transform.rotation= Quaternion.Slerp(transform.rotation,
Quaternion.LookRotation(temp_dir), Time.deltaTime * rotateSpeed/5);
                        //Quaternion targetRotation =
Quaternion.LookRotation(agentToVertext);
                        //this.transform.rotation =
Quaternion.RotateTowards(this.transform.rotation, targetRotation, Time.deltaTime);
                        Debug.Log("Found prey");
                        deltaPosition = new Vector3(0, 0, 0);
                        // eat prey ( or check if prey is flying)
                        if (Vector3.Distance(this.transform.position,
hitInfo.collider.transform.position) < 2 || hitInfo.collider.transform.position.y > 2)
                        {

hitInfo.collider.gameObject.transform.parent.transform.position = new Vector3(3, 1, 19);
                        }
                    }
                }
                else
                {
                    // move
                    deltaPosition += (1f / numOfRays) * speed * direction;
                }
            }

            if (!isWandering)
            {
                StartCoroutine(WalkForAWhile());
            }
            if (startWandering)
            {
                if (isTurning)
                {
```

```
                    // random wander turn
                    if (rotateDirection == 1)
                        this.transform.Rotate(0, wanderRotate * Time.deltaTime, 0);
                    else
                        this.transform.Rotate(0, -wanderRotate * Time.deltaTime, 0);
                }
            }

            this.transform.position += deltaPosition * Time.deltaTime;
        }

        IEnumerator WalkForAWhile()
        {
            // Coroutine for wandering

            int walkTime = Random.Range(1, 5);

            isWandering = true;
            startWandering = true;
            rotateDirection = Random.Range(1, 3);
            isTurning = true;
            yield return new WaitForSeconds(walkTime);

            isTurning = false;
            isWandering = false;
            startWandering = false;
            yield return new WaitForSeconds(3);
        }

        IEnumerator chaseTarget(GameObject target)
        {
            // Coroutine for chasing

            Vector3 agentToVertext = this.transform.position - target.transform.position;
            Quaternion targetRotation = Quaternion.LookRotation(agentToVertext);
            this.transform.rotation = Quaternion.RotateTowards(this.transform.rotation,
targetRotation, Time.deltaTime);
            this.transform.position += this.transform.forward * Time.deltaTime;

            yield return new WaitForSeconds(1);
        }

        private void OnDrawGizmos()
        {
            for (int i = 0; i < numOfRays; ++i)
            {
                var rotation = this.transform.rotation;
                var rotationMod = Quaternion.AngleAxis(i / ((float)numOfRays - 1) *
angle, this.transform.up);
                var direction = rotation * rotationMod * Vector3.forward;
                // draw rays
                Gizmos.color = Color.red;
                Gizmos.DrawRay(this.transform.position, direction * rayRange);
            }
        }
    }
}
```

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

namespace ChengPatrick.lab5
{
    public class PreyBehavior : MonoBehaviour
    {
        public float speed = 15f;
        public float rotateSpeed = 1f;
        public float wanderRotate = 30f;
        public int numOfRays = 30;
        public float angle = 270;
        public float rayRange = 3f;

        private bool isWandering = false;
        private bool startWandering = false;
        private bool isTurning = false;
        private int rotateDirection = 1;
        private float rotateStep;

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

        }

        // Update is called once per frame
        void Update()
        {
            var deltaPosition = Vector3.zero;

            if (this.transform.position.y > 2)
            {
                this.transform.position = new Vector3(0, 1, 0);
                this.transform.rotation = Quaternion.Euler(0, 0, 0);
            }

            for (int i = 0; i < numOfRays; ++i)
            {
                var rotation = this.transform.rotation;
                var rotationMod = Quaternion.AngleAxis(i / ((float)numOfRays - 1) *
angle, this.transform.up);
                var direction = rotation * rotationMod * Vector3.forward;
                Vector3 agentToVertext;
                Quaternion look;

                var ray = new Ray(this.transform.position, direction);
                RaycastHit hitInfo;
                if (Physics.Raycast(ray, out hitInfo, rayRange))
                {
                    if (hitInfo.collider.gameObject.name != "Goal")
                    {
                        // rotate away from collide object
```

```
                        look = Quaternion.LookRotation((hitInfo.point -
this.transform.position).normalized);
                        this.transform.rotation = Quaternion.Slerp(transform.rotation,
look, Time.deltaTime * rotateSpeed);
                    }
                }
                else if (Physics.Raycast(ray, out hitInfo, rayRange * 2))
                {
                    if (hitInfo.collider.gameObject.transform.parent.gameObject.name ==
"Predator")
                    {
                        // avoid predator
                        look = Quaternion.LookRotation((hitInfo.point -
this.transform.position).normalized);
                        this.transform.rotation = Quaternion.Slerp(transform.rotation,
look, Time.deltaTime * rotateSpeed);
                    }
                }
                else
                {
                    deltaPosition += (1f / numOfRays) * speed * direction;
                }
            }

            if (!isWandering)
            {
                StartCoroutine(WalkForAWhile());
            }
            if (startWandering)
            {
                if (isTurning)
                {
                    if (rotateDirection == 1)
                        this.transform.Rotate(0, wanderRotate * Time.deltaTime, 0);
                    else
                        this.transform.Rotate(0, -wanderRotate * Time.deltaTime, 0);
                }
            }

            this.transform.position += deltaPosition * Time.deltaTime;
        }

        IEnumerator WalkForAWhile()
        {
            // Coroutine for wandering

            int walkTime = Random.Range(1, 5);

            isWandering = true;
            startWandering = true;
            rotateDirection = Random.Range(1, 3);
            isTurning = true;
            yield return new WaitForSeconds(walkTime);

            isTurning = false;
            isWandering = false;
            startWandering = false;
            yield return new WaitForSeconds(3);
```

```
        }

        private void OnDrawGizmos()
        {
            for (int i = 0; i < numOfRays; ++i)
            {
                var rotation = this.transform.rotation;
                var rotationMod = Quaternion.AngleAxis(i / ((float)numOfRays - 1) *
angle, this.transform.up);
                var direction = rotation * rotationMod * Vector3.forward;
                // draw rays
                Gizmos.color = Color.green;
                Gizmos.DrawRay(this.transform.position, direction * rayRange * 2);
            }
        }
    }
}



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

namespace ChengPatrick.lab5
{
    public class PredatorGenerator : MonoBehaviour
{
    [SerializeField] [Range(1, 10)] int NumOfPredator;
    [SerializeField] GameObject predator;
    float spawnTime = 0f; // time to separate spawn
    int counter = 0;

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

    }

    private void Update()
    {
        // clone prefabs
        if (counter < NumOfPredator)
        {
            if (Time.time > spawnTime)
            {
                GameObject newClone = GameObject.Instantiate(predator);
                newClone.name = "Predator";
                newClone.transform.position = new Vector3(0, 1, 0);
                Debug.Log("Predator " + " position at " + newClone.transform.position);
                counter++;
                // wait for spawn time
                spawnTime = Time.time + 2f;
            }
        }
    }
}
}
```

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

namespace ChengPatrick.lab5
{
    public class PreyGenerator : MonoBehaviour
    {
        [SerializeField] [Range(1, 10)] int NumOfPrey;
        [SerializeField] GameObject prey;
        float spawnTime = 0f; // time to separate spawn
        int counter = 0;

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

        }

        private void Update()
        {
            // clone prefabs
            if (counter < NumOfPrey)
            {
                if (Time.time > spawnTime)
                {
                    GameObject newClone = GameObject.Instantiate(prey);
                    newClone.name = "Prey";
                    newClone.transform.position = new Vector3(3, 1, 19);
                    Debug.Log("Prey " + " position at " + newClone.transform.position);
                    counter++;
                    // wait for spawn time
                    spawnTime = Time.time + 2f;
                }
            }
        }
    }
}


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

namespace ChengPatrick.lab5
{
    public class ObsticleMovement : MonoBehaviour
    {
        bool high = false;
        // Start is called before the first frame update
        void Start()
        {

        }
```

```
        // Update is called once per frame
        void Update()
        {
            if (this.transform.position.y < 10 && !high)
            {
                this.transform.position += new Vector3(0, 5, 0) * Time.deltaTime;
                if (this.transform.position.y > 10)
                    high = true;
            }
            else if (this.transform.position.y > 2 && high)
            {
                this.transform.position -= new Vector3(0, 5, 0) * Time.deltaTime;
                if (this.transform.position.y < 2)
                    high = false;
            }
        }
    }
}
```