# Unit 4 – Lesson 10. Wandering

# – Using Trigonometry to Make Game Controlled Characters Wondering Randomly

**Aim:**

* How do we design a game with the enemy game object wandering around in a certain area?

**Objectives:** After the lesson, students should be able to:

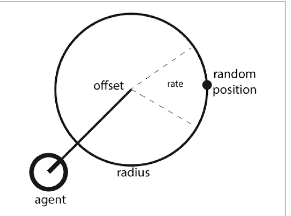
* Obtain better understanding of Unity C# class hierarchies
* Design scripts to make a game object wandering around automatically

**CLASS PROCEDURE:**

***Do Now:***  In your Garden Defender game, how do you make the scarecrow, rabbits and slugs wandering around in the garden?

***Class Discussion:***

In addition to the animation clips, we also want to write a C# script to make the game characters staying inside of the garden area. What is the most efficient way writing such a script for the game?



***Class Discussion / Presentation:***.

1. How do we set the rotation so that the game controlled character stays in the range?

public float MapToRange (float rotation) {

rotation %= 360.0f;

if (Mathf.Abs(rotation) > 180.0f) {

if (rotation < 0.0f)

rotation += 360.0f;

else

rotation -= 360.0f;

}

return rotation;

}

1. How do we convert orientation to vector using trigonometry?

public Vector3 OriToVec (float orientation) {

Vector3 vector = Vector3.zero;

vector.x = Mathf.Sin(orientation \* Mathf.Deg2Rad) \* 1.0f;

vector.z = Mathf.Cos(orientation \* Mathf.Deg2Rad) \* 1.0f;

return vector.normalized;

}

1. How do we define the area where the game character shall stay inside when wandering?

* Well, one of the many ideas is to define a radius as a field, like:

public float radius;

* Then, we can put a constrain to the target position like:

targetPosition = targetPosition + (OriToVec(targetOrientation) \* radius);

1. In which method shall we place the transform to start the game controlled character (gameobject) “wandering” movement?

public override void Awake()

{

target = new GameObject();

target.transform.position = transform.position;

base.Awake();

}

1. How do we make the game character wandering “randomly”?

* We can do something to change the orientation and put it into a random range, like:

float wanderOrientation = Random.Range(-1.0f, 1.0f) \* rate;

* Remember the Steering() and GetSteering() we discussed in the intelligent seek and pursue? We shall certainly put that into use!

public override Steering GetSteering()

{

Steering steering = new Steering();

float wanderOrientation = Random.Range(-1.0f, 1.0f) \* rate;

float targetOrientation = wanderOrientation + agent.orientation;

Vector3 orientationVec = OriToVec(agent.orientation);

Vector3 targetPosition = (offset \* orientationVec) + transform.position;

targetPosition = targetPosition + (OriToVec(targetOrientation) \* radius);

targetAux.transform.position = targetPosition;

steering = base.GetSteering();

steering.linear = targetAux.transform.position - transform.position;

steering.linear.Normalize();

steering.linear \*= agent.maxAccel;

return steering;

}

***Pair – sharing Activity:***

Continue working on your Maze game project. How do you add monsters and make some of them wandering around inside of the maze??

***Example for Wondering AI:***

1. ***Wandering Script:***

using UnityEngine;

using System.Collections;

public class Wander : Face {

public float offset;

public float radius;

public float rate;

public override void Awake()

{

target = new GameObject();

target.transform.position = transform.position;

base.Awake();

}

public override Steering GetSteering()

{

Steering steering = new Steering();

float wanderOrientation = Random.Range(-1.0f, 1.0f) \* rate;

float targetOrientation = wanderOrientation + agent.orientation;

Vector3 orientationVec = OriToVec(agent.orientation);

Vector3 targetPosition = (offset \* orientationVec) + transform.position;

targetPosition = targetPosition + (OriToVec(targetOrientation) \* radius);

targetAux.transform.position = targetPosition;

steering = base.GetSteering();

steering.linear = targetAux.transform.position - transform.position;

steering.linear.Normalize();

steering.linear \*= agent.maxAccel;

return steering;

}}

1. ***Agent Script:***

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

public class Agent : MonoBehaviour

{

public bool blendWeight = false;

public bool blendPriority = false;

public float priorityThreshold = 0.2f;

public bool blendPipeline = false;

public float maxSpeed;

public float maxAccel;

public float maxRotation;

public float maxAngularAccel;

public float orientation;

public float rotation;

public Vector3 velocity;

protected Steering steering;

private Dictionary<int, List<Steering>> groups;

void Start ()

{

velocity = Vector3.zero;

steering = new Steering();

groups = new Dictionary<int, List<Steering>>();

}

public virtual void Update ()

{

Vector3 displacement = velocity \* Time.deltaTime;

orientation += rotation \* Time.deltaTime;

if (orientation < 0.0f)

orientation += 360.0f;

else if (orientation > 360.0f)

orientation -= 360.0f;

transform.Translate(displacement, Space.World);

transform.rotation = new Quaternion();

transform.Rotate(Vector3.up, orientation);

}

public virtual void LateUpdate ()

{

if (blendPriority)

{

steering = GetPrioritySteering();

groups.Clear();

}

velocity += steering.linear \* Time.deltaTime;

rotation += steering.angular \* Time.deltaTime;

if (velocity.magnitude > maxSpeed)

{

velocity.Normalize();

velocity = velocity \* maxSpeed;

}

if (rotation > maxRotation)

{

rotation = maxRotation;

}

if (steering.angular == 0.0f)

{

rotation = 0.0f;

}

if (steering.linear.sqrMagnitude == 0.0f)

{

velocity = Vector3.zero;

}

steering = new Steering();

}

public void SetSteering (Steering steering)

{

this.steering = steering;

}

public void SetSteering (Steering steering, float weight)

{

this.steering.linear += (weight \* steering.linear);

this.steering.angular += (weight \* steering.angular);

}

public void SetSteering (Steering steering, int priority)

{

if (!groups.ContainsKey(priority))

{

groups.Add(priority, new List<Steering>());

}

groups[priority].Add(steering);

}

public void SetSteering (Steering steering, bool pipeline)

{

if (!pipeline)

{

this.steering = steering;

return;

}

}

private Steering GetPrioritySteering ()

{

Steering steering = new Steering();

float sqrThreshold = priorityThreshold \* priorityThreshold;

List<int> gIdList = new List<int>(groups.Keys);

gIdList.Sort();

foreach (int gid in gIdList)

{

steering = new Steering();

foreach (Steering singleSteering in groups[gid])

{

steering.linear += singleSteering.linear;

steering.angular += singleSteering.angular;

}

if (steering.linear.magnitude > priorityThreshold ||

Mathf.Abs(steering.angular) > priorityThreshold)

{

return steering;

}

}

return steering;

}

}

1. ***AgentBehaviour script:***

using UnityEngine;

using System.Collections;

public class AgentBehaviour : MonoBehaviour

{

public float weight = 1.0f;

public int priority = 1;

public GameObject target;

protected Agent agent;

public virtual void Awake ()

{

agent = gameObject.GetComponent<Agent>();

}

public virtual void Update ()

{

if (agent.blendWeight)

agent.SetSteering(GetSteering(), weight);

else if (agent.blendPriority)

agent.SetSteering(GetSteering(), priority);

else if (agent.blendPipeline)

agent.SetSteering(GetSteering(), true);

else

agent.SetSteering(GetSteering());

}

public virtual Steering GetSteering ()

{

return new Steering();

}

public float MapToRange (float rotation) {

rotation %= 360.0f;

if (Mathf.Abs(rotation) > 180.0f) {

if (rotation < 0.0f)

rotation += 360.0f;

else

rotation -= 360.0f;

}

return rotation;

}

public Vector3 OriToVec (float orientation) {

Vector3 vector = Vector3.zero;

vector.x = Mathf.Sin(orientation \* Mathf.Deg2Rad) \* 1.0f;

vector.z = Mathf.Cos(orientation \* Mathf.Deg2Rad) \* 1.0f;

return vector.normalized;

}

}