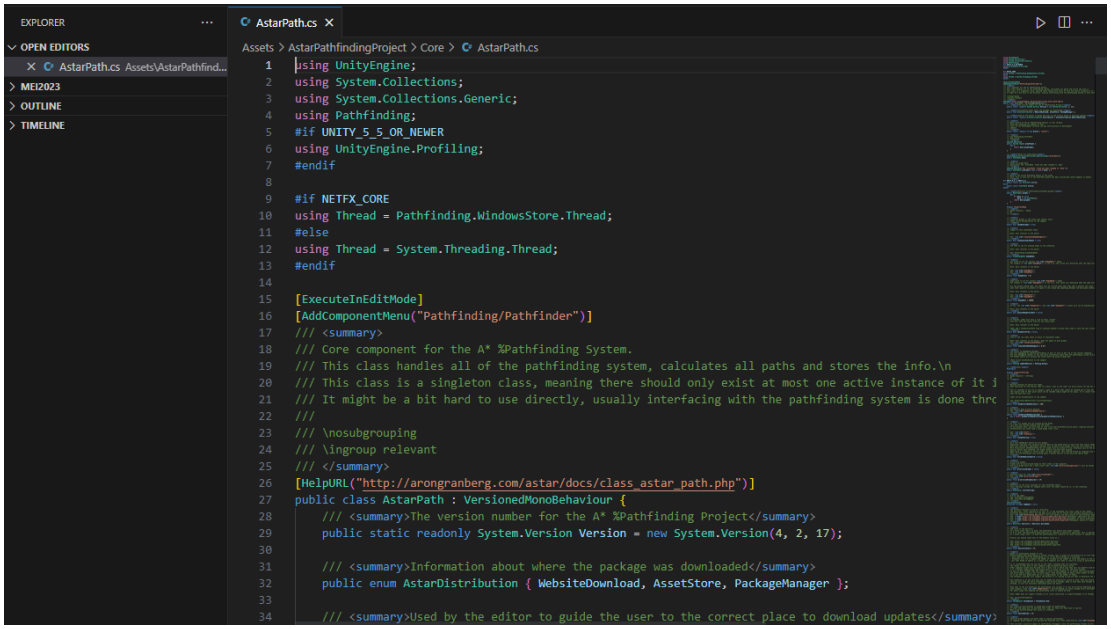
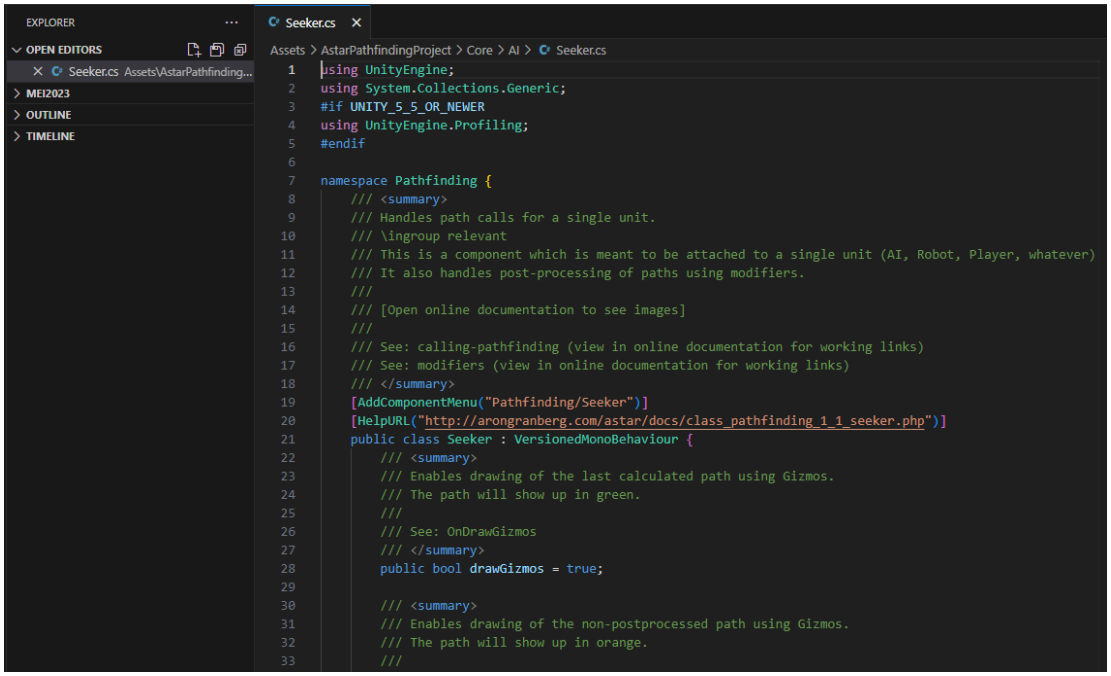
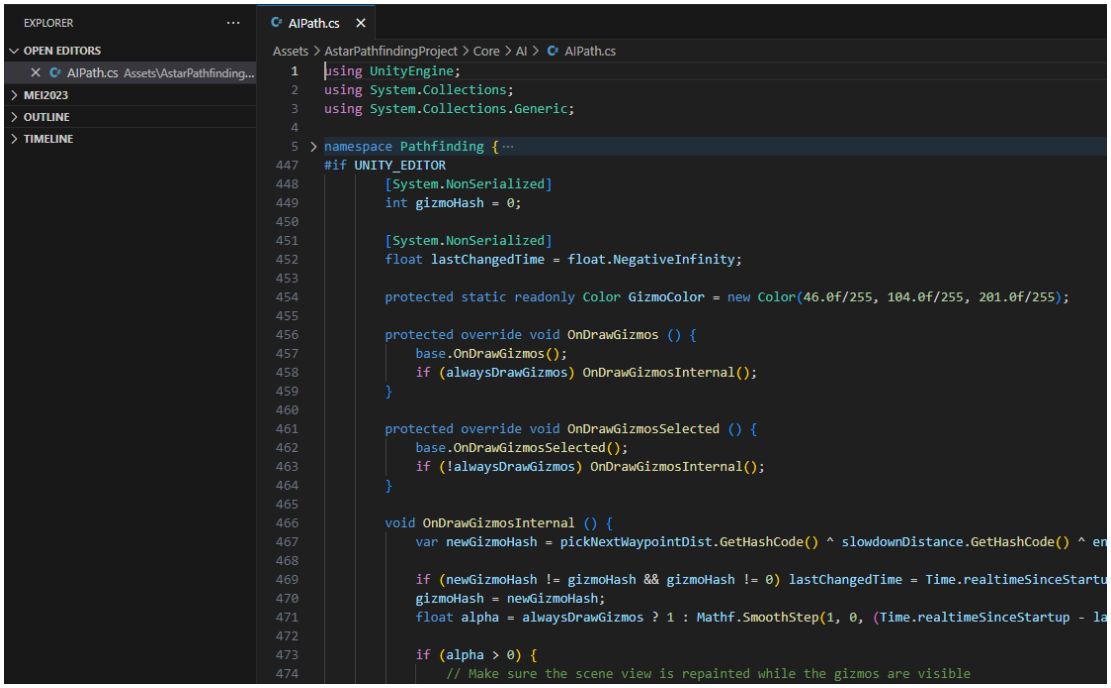
LAMPIRAN

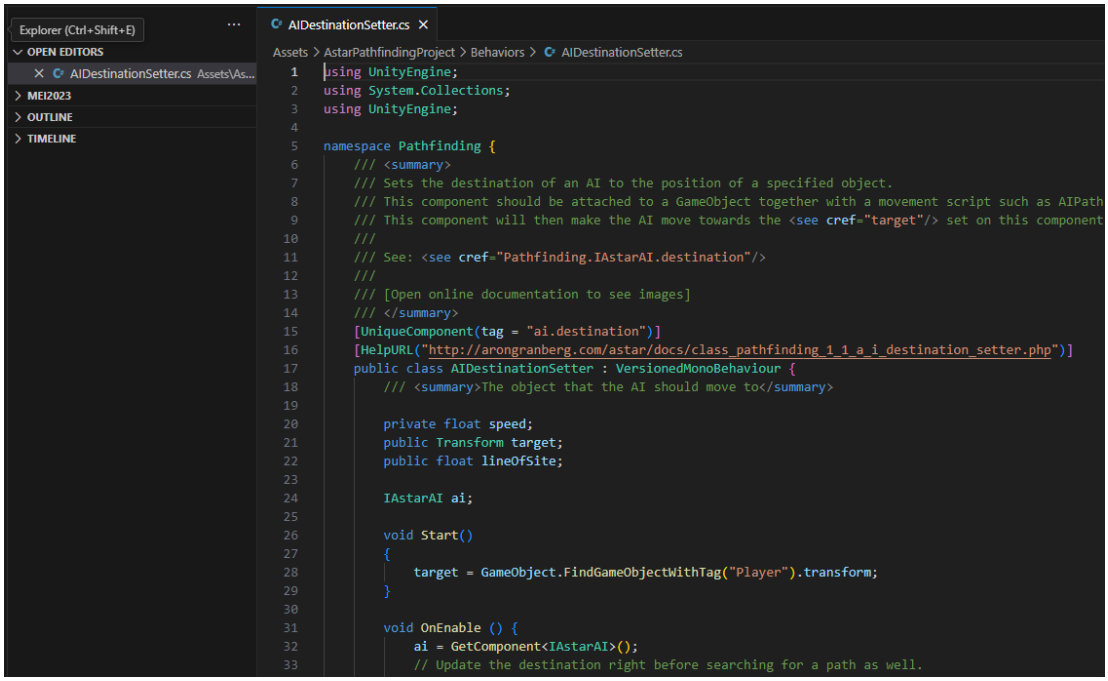


```
EXPLORER
OPEN EDITORS
  AstarPath.cs Assets\AstarPathfind...
MEI2023
OUTLINE
TIMELINE

AstarPath.cs
Assets > AstarPathfindingProject > Core > AstarPath.cs
1  using UnityEngine;
2  using System.Collections;
3  using System.Collections.Generic;
4  using Pathfinding;
5  #if UNITY_5_5_OR_NEWER
6  using UnityEngine.Profiling;
7  #endif
8
9  #if NETFX_CORE
10 using Thread = Pathfinding.WindowsStore.Thread;
11 #else
12 using Thread = System.Threading.Thread;
13 #endif
14
15 [ExecuteInEditMode]
16 [AddComponentMenu("Pathfinding/Pathfinder")]
17 /// <summary>
18 /// Core component for the A* %Pathfinding System.
19 /// This class handles all of the pathfinding system, calculates all paths and stores the info.\n
20 /// This class is a singleton class, meaning there should only exist at most one active instance of it i
21 /// It might be a bit hard to use directly, usually interfacing with the pathfinding system is done thro
22 ///
23 /// \nosubgrouping
24 /// \ingroup relevant
25 /// </summary>
26 [HelpURL("http://arongranberg.com/astar/docs/class_astar_path.php")]
27 public class AstarPath : VersionedMonoBehaviour {
28     /// <summary>The version number for the A* %Pathfinding Project</summary>
29     public static readonly System.Version Version = new System.Version(4, 2, 17);
30
31     /// <summary>Information about where the package was downloaded</summary>
32     public enum AstarDistribution { WebsiteDownload, AssetStore, PackageManager };
33
34     /// <summary>Used by the editor to guide the user to the correct place to download updates</summary>
```



```
EXPLORER
OPEN EDITORS
  Seeker.cs Assets\AstarPathfinding...
MEI2023
OUTLINE
TIMELINE

Seeker.cs
Assets > AstarPathfindingProject > Core > AI > Seeker.cs
1  using UnityEngine;
2  using System.Collections.Generic;
3  #if UNITY_5_5_OR_NEWER
4  using UnityEngine.Profiling;
5  #endif
6
7  namespace Pathfinding {
8      /// <summary>
9      /// Handles path calls for a single unit.
10     /// \ingroup relevant
11     /// This is a component which is meant to be attached to a single unit (AI, Robot, Player, whatever)
12     /// It also handles post-processing of paths using modifiers.
13     ///
14     /// [Open online documentation to see images]
15     ///
16     /// See: calling-pathfinding (view in online documentation for working links)
17     /// See: modifiers (view in online documentation for working links)
18     /// </summary>
19     [AddComponentMenu("Pathfinding/Seeker")]
20     [HelpURL("http://arongranberg.com/astar/docs/class_pathfinding_1_1_seeker.php")]
21     public class Seeker : VersionedMonoBehaviour {
22         /// <summary>
23         /// Enables drawing of the last calculated path using Gizmos.
24         /// The path will show up in green.
25         ///
26         /// See: OnDrawGizmos
27         /// </summary>
28         public bool drawGizmos = true;
29
30         /// <summary>
31         /// Enables drawing of the non-postprocessed path using Gizmos.
32         /// The path will show up in orange.
33         ///
```


EXPLORER
OPEN EDITORS
AIPath.cs Assets\AstarPathfinding...
MEI2023
OUTLINE
TIMELINE
AIPath.cs
Assets > AstarPathfindingProject > Core > AI > AIPath.cs
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
namespace Pathfinding {
#if UNITY_EDITOR
[System.NonSerialized]
int gizmoHash = 0;
[System.NonSerialized]
float lastChangedTime = float.NegativeInfinity;
protected static readonly Color GizmoColor = new Color(46.0f/255, 104.0f/255, 201.0f/255);
protected override void OnDrawGizmos () {
base.OnDrawGizmos();
if (alwaysDrawGizmos) OnDrawGizmosInternal();
}
protected override void OnDrawGizmosSelected () {
base.OnDrawGizmosSelected();
if (!alwaysDrawGizmos) OnDrawGizmosInternal();
}
void OnDrawGizmosInternal () {
var newGizmoHash = pickNextWaypointDist.GetHashCode() ^ slowdownDistance.GetHashCode() ^ en
if (newGizmoHash != gizmoHash && gizmoHash != 0) lastChangedTime = Time.realtimeSinceStartu
gizmoHash = newGizmoHash;
float alpha = alwaysDrawGizmos ? 1 : Mathf.SmoothStep(1, 0, (Time.realtimeSinceStartup - la
if (alpha > 0) {
// Make sure the scene view is repainted while the gizmos are visible


Explorer (Ctrl+Shift+E)
OPEN EDITORS
AIDestinationSetter.cs Assets\As...
MEI2023
OUTLINE
TIMELINE
AIDestinationSetter.cs
Assets > AstarPathfindingProject > Behaviors > AIDestinationSetter.cs
using UnityEngine;
using System.Collections;
using UnityEngine;
namespace Pathfinding {
/// <summary>
/// Sets the destination of an AI to the position of a specified object.
/// This component should be attached to a GameObject together with a movement script such as AIPath
/// This component will then make the AI move towards the <see cref="target"/> set on this component
///
/// See: <see cref="Pathfinding.IAstarAI.destination"/>
///
/// [Open online documentation to see images]
/// </summary>
[UniqueComponent(tag = "ai.destination")]
[HelpURL("http://arongranberg.com/astar/docs/class_pathfinding_1_1_a_i_destination_setter.php")]
public class AIDestinationSetter : VersionedMonoBehaviour {
/// <summary>The object that the AI should move to</summary>
private float speed;
public Transform target;
public float lineOfSite;
IAstarAI ai;
void Start()
{
target = GameObject.FindGameObjectWithTag("Player").transform;
}
void OnEnable () {
ai = GetComponent<IAstarAI>();
// Update the destination right before searching for a path as well.

AIDestinationSetter.cs

Assets > AstarPathfindingProject > Behaviors > AIDestinationSetter.cs

```
            // This is enough in theory, but this script will also update the destination every
            // frame as the destination is used for debugging and may be used for other things by other
            // scripts as well. So it makes sense that it is up to date every frame.
            if (ai != null) ai.onSearchPath += Update;
        }

        void OnDisable () {
            if (ai != null) ai.onSearchPath -= Update;
        }

        /// <summary>Updates the AI's destination every frame</summary>
        void Update () {
            float distanceFromPlayer = Vector2.Distance(target.position, transform.position);
            if (distanceFromPlayer < lineOfSite)
            {
                transform.position = Vector2.MoveTowards(this.transform.position, target.position, speed
                if (target != null && ai != null) ai.destination = target.position;
            }


            //GetTarget();
        }

        private void OnDrawGizmosSelected()
        {
            Gizmos.color = Color.yellow;
            Gizmos.DrawWireSphere(transform.position, lineOfSite);
        }

        //private void GetTarget () {
        //  if (GameObject.FindGameObjectWithTag("Player")) {
        //      target = GameObject.FindGameObjectWithTag("Player").transform;
        //  }
        //}
```

EnemyMovement.cs

EnemyMovement.cs

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

public class EnemyMovement : MonoBehaviour
{

    public AIPath aiPath;

    void Update()
    {
        if(aiPath.desiredVelocity.x >= 0.01f)
        {
            transform.localScale = new Vector3(-1f, 1f, 1f);
        } else if (aiPath.desiredVelocity.x <= -0.01f)
        {
            transform.localScale = new Vector3(1f, 1f, 1f);
        }
    }
}
```

EnemyFire.cs

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

public class EnemyFire : MonoBehaviour
{
    public float shootingRange;
    public float fireRate = 2f;
    private float nextFireTime;

    public GameObject bullet;
    public GameObject bulletParent;

    private Transform player;

    //[SerializeField] private AudioSource bulletSoundEffect;

    //float fireRate;
    //float nextFire;
    // Start is called before the first frame update
    void Start()
    {
        //fireRate = 2f;
        //nextFire = Time.time;
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    // Update is called once per frame
    void Update()
    {
        //CheckIfTimeToFire();
        float distanceFromPlayer = Vector2.Distance(player.position, transform.position);
        if (distanceFromPlayer <= shootingRange && nextFireTime < Time.time)
        {
            Instantiate (bullet, bulletParent.transform.position, Quaternion.identity);
            nextFireTime = Time.time + fireRate;
        }

    }

    //void CheckIfTimeToFire()
    //{
    //    if (Time.time > nextFire) {
    //        bulletSoundEffect.Play();
    //        Instantiate (bullet, transform.position, Quaternion.identity);
    //        nextFire = Time.time + fireRate;
    //    }
    //}

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(transform.position, shootingRange);
    }
}
```

EnemyBehaviour.cs

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

public class EnemyBehaviour : MonoBehaviour
{
    public float Hitpoints;
    public float MaxHitpoints = 5;
    public HealthbarBehaviour Healthbar;

    void Start()
    {
        Hitpoints = MaxHitpoints;
        Healthbar.SetHealth(Hitpoints, MaxHitpoints);
    }

    public void TakeHit(float damage)
    {
        Hitpoints -= damage;
        Healthbar.SetHealth(Hitpoints, MaxHitpoints);

        if (Hitpoints <= 0)
        {
            Die();
        }
    }

    void Die()
    {
        GetComponent<LootBag>().InstantiateLoot(transform.position);
        Destroy(gameObject);
    }

}
```

HealthbarBehaviour.cs

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

public class HealthbarBehaviour : MonoBehaviour
{

    public Slider Slider;
    public Color Low;
    public Color High;
    public Vector3 Offset;


    // Start is called before the first frame update
    public void SetHealth(float health, float maxHealth)
    {
        Slider.gameObject.SetActive(health < maxHealth);
        Slider.value = health;
        Slider.maxValue = maxHealth;

        Slider.fillRect.GetComponentInChildren<Image>().color = Color.Lerp(Low, High, Slider.normalizedVa
    }

    // Update is called once per frame
    void Update()
    {
        Slider.transform.position = Camera.main.WorldToScreenPoint(transform.parent.position + Offset);
    }
}

```

bullet.cs

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

public class bullet : MonoBehaviour
{
    [SerializeField]
    private float speed;

    private Transform player;
    private Vector2 target;
    private Rigidbody2D myRigidBody;

    void Start() {
        myRigidBody=GetComponent<Rigidbody2D>();
        player = GameObject.FindGameObjectWithTag("Player").transform;
        target = new Vector2(player.position.x, player.position.y);
        Vector3 direction = player.transform.position - transform.position;
        float rotation = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg - 90;
        transform.rotation = Quaternion.Euler(0, 0, rotation);

    }

    private void Update(){

        transform.position = Vector2.MoveTowards(transform.position, target, speed * Time.deltaTime);

        if(transform.position.x == target.x && transform.position.y == target.y){
            DestroyBullet();
        }
    }

    private void OnTriggerEnter2D(Collider2D other) {

        if(other.gameObject.CompareTag("obscale")){
            DestroyBullet();
        }

        if(other.gameObject.CompareTag("Player")){
            DestroyBullet();
        }
    }

    //private void OnCollisionEnter2D(Collision2D collision) {

    //    if(collision.gameObject.TryGetComponent<healthPlayer>(out healthPlayer hp)){
    //        DestroyBullet();
    //        hp.TakeDamage(1);
    //    }
    // }

    private void DestroyBullet(){
        Destroy(gameObject);
    }
}
```

bullet3.cs

SpecialBulletNPC > bullet3.cs

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

public class bullet3 : MonoBehaviour
{
    private Vector2 moveDirection;
    private float moveSpeed;

    private void OnTriggerEnter2D(Collider2D collision){
        if(collision.gameObject.CompareTag("obscale")){
            Invoke("Destroy", 2f);
            Destroy();
        }
        //private void OnEnable() {
       // Invoke("Destroy", 3f);
   // }
    }

    void Start()
    {
        moveSpeed = 10f;
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(moveDirection * moveSpeed * Time.deltaTime);
    }

    public void SetMoveDirection(Vector2 dir){
        moveDirection = dir;
    }

    private void Destroy() {
        gameObject.SetActive(false);
    }

    private void OnDisable() {
        CancelInvoke();
    }
}
```

PeluruPool.cs

SpecialBulletNPC > PeluruPool.cs

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

public class PeluruPool : MonoBehaviour
{
    public static PeluruPool bulletPoolInstance;

    [SerializeField]
    private GameObject pooledBullet;
    private bool notEnoughBulletsInPool = true;

    private List<GameObject> bullets;

    private void Awake()
    {
        bulletPoolInstance = this;
    }

    void Start()
    {
        bullets = new List<GameObject>();
    }

    public GameObject GetBullet()
    {
        if (bullets.Count > 0)
        {
            for (int i = 0; i < bullets.Count; i++)
            {
                if (!bullets[i].activeInHierarchy)
                {
                    return bullets[i];
                }
            }
        }

        if (notEnoughBulletsInPool)
        {
            GameObject bul = Instantiate(pooledBullet);
            bul.SetActive(false);
            bullets.Add(bul);
            return bul;
        }

        return null;
    }
}
```

SpecialBullet2.cs

SpecialBulletNPC > SpecialBullet2.cs

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

public class SpecialBullet2 : MonoBehaviour
{
    //[SerializeField] private Cooldown cooldown;
    public float shootingRange;

    private float angle = 0f;
    private Transform player;

    void Start()
    {
        InvokeRepeating("Update", 0f, 0.1f);
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void Update()
    {
        float distanceFromPlayer = Vector2.Distance(player.position, transform.position);
        if (distanceFromPlayer <= shootingRange)
        {
            float bulDirX = transform. position.x + Mathf.Sin((angle * Mathf.PI) / 180f);
            float bulDirY = transform. position.y + Mathf.Cos((angle * Mathf.PI) / 180f);

            Vector3 bulMoveVector = new Vector3(bulDirX, bulDirY, 0f);
            Vector2 bulDir = (bulMoveVector - transform.position).normalized;

            GameObject bul = PeluruPool.bulletPoolInstance.GetBullet();
                bul.transform.position = transform.position;
                bul.transform.rotation = transform.rotation;
                bul.SetActive(true);
                bul.GetComponent<bullet3>().SetMoveDirection(bulDir);
```

```
            angle += 10f;

        }
    }

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(transform.position, shootingRange);
    }

```

SpecialBullet3.cs

SpecialBulletNPC > SpecialBullet3.cs

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

public class SpecialBullet3 : MonoBehaviour
{
    public float shootingRange;

    private float angle = 0f;

    private Vector2 bulletMoveDirection;
    private Transform player;

    void Start()
    {
        InvokeRepeating("Update", 0f, 0.1f);
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void Update()
    {
        float distanceFromPlayer = Vector2.Distance(player.position, transform.position);
        if (distanceFromPlayer <= shootingRange)
        {
            for (int i = 0; i <= 1; i++)
            {
                float bulDirX = transform.position.x + Mathf.Sin(((angle + 180f * i) * Mathf.PI) / 180f);
                float bulDirY = transform.position.y + Mathf.Cos(((angle + 180f * i) * Mathf.PI) / 180f);

                Vector3 bulMoveVector = new Vector3(bulDirX, bulDirY, 0f);
                Vector2 bulDir = (bulMoveVector - transform.position).normalized;

                GameObject bul = PeluruPool.bulletPoolInstance.GetBullet();
                bul.transform.position = transform.position;
                bul.transform.rotation = transform.rotation;
                bul.SetActive(true);
                bul.GetComponent<bullet3>().SetMoveDirection(bulDir);
            }

            angle += 10f;

            if (angle >= 360f)
            {
                angle = 0f;
            }
        }
    }

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(transform.position, shootingRange);
    }
```

SpecialBullet4.cs

SpecialBulletNPC > SpecialBullet4.cs

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

public class SpecialBullet4 : MonoBehaviour
{
    public float shootingRange;

    [SerializeField] private Cooldown cooldown;

    [SerializeField]
    private int bulletAmount = 10;

    [SerializeField]
    private float startAngle = 90f, endAngle = 270f;

    private Vector2 bulletMoveDirection;
    private Transform player;

    void Start(){
        InvokeRepeating("Update", 0f, 1f);
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void Update(){
        float distanceFromPlayer = Vector2.Distance(player.position, transform.position);
        if (distanceFromPlayer <= shootingRange)
        {
            float angleStep = (endAngle - startAngle) / bulletAmount;
            float angle = startAngle;

            if (cooldown.IsCoolingDown) return;

            for (int i = 0; i < bulletAmount + 1; i++)
            {
                float bulDirX = transform.position.x + Mathf.Sin((angle * Mathf.PI) / 180f);
                float bulDirY = transform.position.y + Mathf.Cos((angle * Mathf.PI) / 180f);

                Vector3 bulMoveVector = new Vector3(bulDirX, bulDirY, 0f);
                Vector2 bulDir = (bulMoveVector - transform.position).normalized;

                GameObject bul = PeluruPool.bulletPoolInstance.GetBullet();
                    bul.transform.position = transform.position;
                    bul.transform.rotation = transform.rotation;
                    bul.SetActive(true);
                    bul.GetComponent<bullet3>().SetMoveDirection(bulDir);

                angle += angleStep;
            }

            cooldown.StartCooldown();
        }
    }

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(transform.position, shootingRange);
    }

```

SpecialBullet5.cs

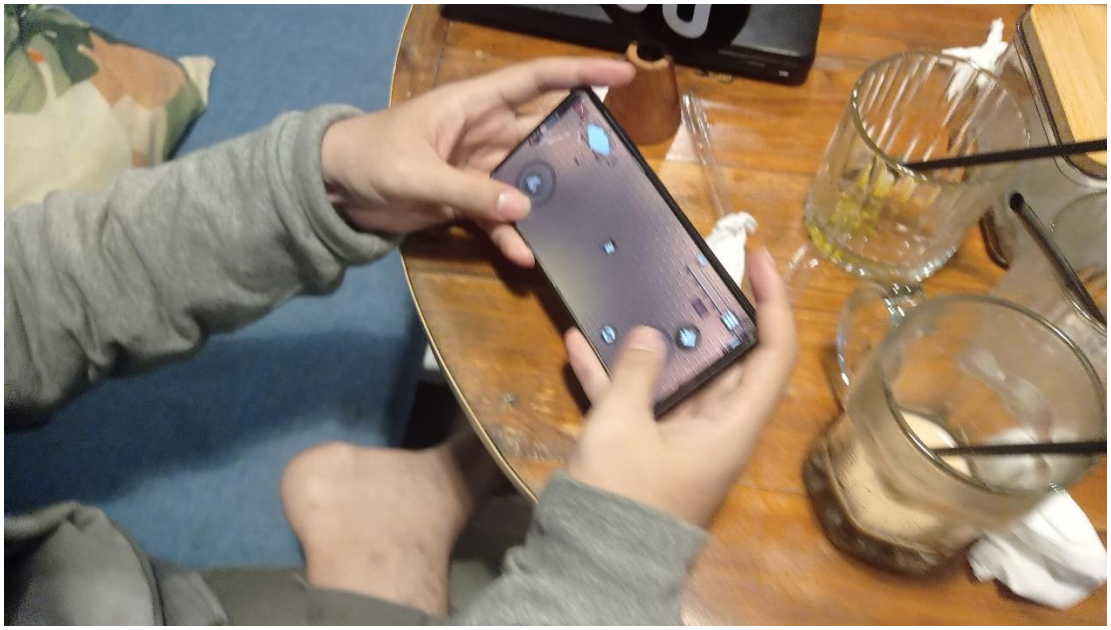
SpecialBulletNPC > SpecialBullet5.cs

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

public class SpecialBullet5 : MonoBehaviour
{
    [SerializeField] private AudioSource exploSoundEffect;

    public GameObject tower;
    public GameObject target;

    public float speed = 10f;

    private float towerX;
    private float targetX;

    private float dist;
    private float nextX;
    private float baseY;
    private float height;

    void Start()
    {
        tower = GameObject.FindGameObjectWithTag("Senjata");
        target = GameObject.FindGameObjectWithTag("Player");
    }

    void Update()
    {
        towerX = tower.transform.position.x;
        targetX = target.transform.position.x;

        dist = targetX - towerX;
        nextX = Mathf.MoveTowards(transform.position.x, targetX, speed * Time.deltaTime);
        baseY = Mathf.Lerp(tower.transform.position.y, target.transform.position.y, (nextX - towerX) / di
        height = 2 * (nextX - towerX) * (nextX - targetX) / (-0.25f * dist * dist);

        Vector3 movePosition = new Vector3(nextX, baseY + height, transform.position.z);
        transform.rotation = LookAtTarget(movePosition - transform.position);
        transform.position = movePosition;

        if(transform.position == target.transform.position)
        {
            exploSoundEffect.Play();
            Destroy(gameObject);
        }
    }

    public static Quaternion LookAtTarget(Vector2 rotation)
    {
        return Quaternion.Euler(0,0, Mathf.Atan2(rotation.y, rotation.x) * Mathf.Rad2Deg);
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("obscale")){
            DestroyBomb();
        }

        if (other.gameObject.CompareTag("Bullet")){
            DestroyBomb();
        }

        if (other.gameObject.CompareTag("Player")){

            DestroyBomb();
        }
    }

    void DestroyBomb(){
        Destroy(gameObject);
    }
}
```

Cooldown.cs

SpecialBulletNPC > Cooldown.cs

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   [System.Serializable]
6   public class Cooldown
7   {
8       [SerializeField] private float cooldownTime;
9       private float _nextFireTime;
10
11      public bool IsCoolingDown=> Time.time <_nextFireTime;
12      public void StartCooldown() => _nextFireTime = Time.time + cooldownTime;
13  }
14
```



UMKT
Program Studi
Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Muhammadiyah Kalimantan Timur
Berkarakter | Berwawasan | Berkemajuan
Telp. 0541-748511 Fax. 0541-766832
Website http://informatika.umkt.ac.id
email: informatika@umkt.ac.id

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

SURAT KETERANGAN
Nomor: 152-004/FST.1/KET/I/2023

Assalamualaikum Warrahmatullahi Wabarrakatuh

Yang bertanda tangan dibawah ini:

Nama : Arbansyah, S.Kom., M.TI
NIDN : 1118019203
Keterangan : Ketua Program Studi

Dengan ini menerangkan bahwa mahasiswa di bawah ini:

Nama : Muhammad Luthfi Setiawan
NIM : 1911102441076
Program Studi : S1 Teknik Informatika
Semester : VIII (Delapan)
Fakultas : Sains dan Teknologi

Merupakan mahasiswa Program Studi S1 Teknik Informatika dan telah menyelesaikan Penelitian Skripsi pada bulan Februari s/d Juni 2023 dengan judul Skripsi "Penerapan Algoritma A* dan Behaviour Trees untuk Perilaku Non-Player Character (NPC) pada Game "The Last Hope" Berbasis Android Menggunakan Unity 2D".

Demikian hal ini disampaikan, atas kejasamanya kami ucapkan terima kasih.

Wassalamu'alaikum Warahmatullahhi Wabarrakatuh

Samarinda, 29 Rabiul Akhir 1445 H
13 November 2023 M

Ketua Prodi S1 Teknik Informatika

Arbansyah, S.Kom., M.TI
NIDN. 1118019203

Kampus 1 : Jl. Ir. H. Juanda, No.15, Samarinda
Kampus 2 : Jl. Pelita, Pesona Mahakam, Samarinda

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR
FAKULTAS SAINS DAN TEKNOLOGI
PROGRAM STUDI TEKNIK INFORMATIKA
Jl. Ir. H. Juanda No 15 Samarinda Telp. 0541-748511

LEMBAR BIMBINGAN SKRIPSI

Nama : MUHAMMAD LUTHFI SETIAWAN
NIM : 1911102441076
Program Studi : TEKNIK INFORMATIKA
Judul Skripsi : PENERAPAN ALGORITMA A* DAN BEHAVIOUR TREES UNTUK PERILAKU NON-PLAYER CHARACTER(NPC) PADA GAME "THE LAST HOPE" BERBASIS ANDROID MENGGUNAKAN UNITY 2D

No.	Tanggal	Keterangan	Tanda Tangan
1	08/03/2023	Bab I 1. Latar Belakang 2. Rumusan Masalah	
2	15/03/2023	Bab I 3. Batasan Masalah 4. Manfaat Penelitian	
3	23/03/2023	Bab II 1. Tinjauan Pustaka 2. Objek Penelitian	
4	25/03/2023	Bab III 1. Tahapan Penelitian 2. Jadwal Penelitian	

5	10/05/2023	Bab IV	1. Pembahasan Tahapan Penelitian	
6	15/05/2023	Bab IV	1. membahas konsep algoritma 2. menentukan Gameplay	
7	22/06/2023	Bab IV	1. membahas Black box 2. membahas Fungsional	
8	10/07/2023	Bab IV	1. Revisi kompleksitas algoritma	
9	18/07/2023	Jurnal	1. Penulisan 2. tata letak	
10	20/07/2023	Jurnal & BAB V	1. Revisi kesimpulan	

Samarinda, 08 November 2023
Dosen Pembimbing

Arbansyah, S.Kom.,M.TI

Skripsi: Penerapan Algoritma A* dan Behaviour Trees Untuk Perilaku Non-Player Character (NPC) Pada Game “The Last Hope” Berbasis Android Menggunakan Unity 2D

by Muhammad Luthfi Setiawan

Submission date: 20-Jul-2023 03:43PM (UTC+0800)
Submission ID: 2133960976
File name: 4_-_1911102441076_Muhammad_Luthfi_Setiawan_Seminar_Hasil.docx (2.2M)
Word count: 7584
Character count: 45061

Skripsi: Penerapan Algoritma A* dan Behaviour Trees Untuk Perilaku Non-Player Character (NPC) Pada Game "The Last Hope" Berbasis Android Menggunakan Unity 2D

ORIGINALITY REPORT

25%	24%	7%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	jurnal.machung.ac.id Internet Source	2%
2	repository.uinsu.ac.id Internet Source	2%
3	123dok.com Internet Source	2%
4	j-ptiik.ub.ac.id Internet Source	2%
5	repositori.unsil.ac.id Internet Source	2%
6	p3m.sinus.ac.id Internet Source	1%
7	Submitted to Universitas Brawijaya Student Paper	1%
8	eprints.itn.ac.id Internet Source	1%
	etheses.uin-malang.ac.id	