Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

___________ Едуард ЖАРІКОВ

“___” _________________ 2024 р.

ІГРОВИЙ ЗАСТОСУНОК МОДЕЛЮВАННЯ ПОВЕДІНКИ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ У 3D RPG З ВИКОРИСТАННЯМ ІГРОВОГО РУШІЯ UNITY.

Текст програми

КПІ.ІТ-0223.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____________Олексій ФІНОГЕНОВ

Нормоконтроль:

____________Максим ГОЛОВЧЕНКО

Виконавець:

___________Максим ТЕРЕШКОВИЧ

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду: https://github.com/mcmcmax437/3D-RPG-Diploma-Thesis

1.1 Реалізація інтелекту ворогів зі своїми особливими спектрами поведінки.

1.1.1 Файл EnemyMovement.cs:

```
public class EnemyMovement : MonoBehaviour
{
    public bool Goblin_Warrior = false;
    public bool Piglins = false;
    public bool Skeleton = false;
    public bool temp_Priority = false; // temp to check
    private bool can_call_support = false;
    private Vector3 buffed_Skeleton = new Vector3(15.0f, 15.0f, 15.0f);
    private int buffed_probability = 10;
    private bool sup_skill_used = false;
    private bool change_position = false;
    private float sup_skill_CD = 10f;
    private int amount_of_reinforcment = 2;
    public GameObject support_enemy;



    public GameObject Loot_from_Enemy;

    public GameObject current_enemy;
    private bool is_outliner_active = false;

    private AnimatorStateInfo enemy_information;
    private NavMeshAgent nav;
    private Animator anim;
    private float x;
    private float z;
    private float velocitySpeed;
    public GameObject player;
    private float distance_to_player;
    private bool is_attacking;
    public float attack_Range = 2.0f;
    public float chasing_Range = 12.0f;   //range in which enemy will run after character
    public float rotation_speed = 500.0f; //perfect
    private float stop_distance = 2f;
    private float group_brain_radius = 10f;
```

```
public Transform patrol_main_obj;
public float patrol_radius = 15.0f;
public float wait_time_at_point = 2.0f;

private Vector3 targetPoint;
private bool is_waiting;
private float wait_timer;
private bool is_patroling = true;

private int maxHP;

public int full_HP = 100;
private int curr_HP;

private int fear_lvl = 100;
private int fear_lvl_curr;

private bool enemy_is_alive = true;

private AudioSource audio_Player;
public AudioClip[] get_Hit_SFX;

public GameObject bar_Container;
public Image HP_bar;
private float fillHealth;
public GameObject main_camera;

private bool destination_run = false;

private Vector3 escape_point;
public Transform[] escape_target_point;

private bool roll_out = false;
private bool roll_is_active = false;
public float dodgeDistance = 5f;
public float aggression_lvl = 0.5f; // 0 (passive) to 1 (aggressive)
private bool playerNearby = false;
private float aggression_increase = 0.05f;
private float aggression_decrease = 0.025f;
public float max_aggression = 1.0f;
public float min_aggression = 0.0f;

public bool piglin_was_hit = false;
private bool player_is_armorless = true;
private bool should_reset_armor_trigger = true;

public float distance_of_ray = 12f;
public float time_for_search = 3f;
private Vector3 last_seen_position;
private float search_Timer;
private bool player_is_inSight;
private bool look_for_player;
private bool reset_piglins_chase_range = false;

// Start is called before the first frame update
void Start()
{
    audio_Player = GetComponent<AudioSource>();
    current_enemy.GetComponent<Outline>().enabled = false;
    nav = GetComponent<NavMeshAgent>();
    anim = GetComponent<Animator>();
    nav.avoidancePriority = UnityEngine.Random.Range(5, 75);
    curr_HP = full_HP;
    maxHP = full_HP;

    if (Goblin_Warrior == true)
    {
        Set_Petrol_Destination();
    }
    if (Goblin_Warrior == true && patrol_main_obj == null)
    {
        is_patroling = false;
    }
    if (GetComponent<Enemy_Type>().enemyType == Enemy_Type.EnemyType.Piglin)
    {
        chasing_Range = 0;
    }

    if (Skeleton == true)
    {
        int random = UnityEngine.Random.Range(1, 101);
        if (random <= buffed_probability || temp_Priority == true) //10 per cent to be able to call support
        {
            can_call_support = true;
            transform.localScale = buffed_Skeleton;
        }
        else
        {
            can_call_support = false;
        }
        if (SaveScript.weapon_index == -1 && Piglins == true)
        {
            reset_piglins_chase_range = true;
        }
    }

}

// Update is called once per frame
void Update()
{

    if (main_camera == null)
    {
        main_camera = GameObject.Find("Main Camera");
    }
    if (patrol_main_obj == null)
    {
        new WaitForSeconds(1);
    }
    if(SaveScript.weapon_index != -1)
    {
        reset_piglins_chase_range = false;
    }
    else
    {
        reset_piglins_chase_range = true;
        piglin_was_hit = true;
    }
    if(reset_piglins_chase_range == true)
    {
        chasing_Range = 12.0f;

    }
    if (reset_piglins_chase_range == false && Piglins == true && chasing_Range != 60)
    {
        chasing_Range = 0.0f;
    }

    bar_Container.transform.LookAt(main_camera.transform.position);



    if (Input.GetKeyDown(KeyCode.Z) && distance_to_player < 5f && SaveScript.stamina > 0.2)
    {
        roll_out = true;
    }

    if (enemy_is_alive == true)
    {
        Enemy_Outline();
        if (player == null)
```

```
{
    player = GameObject.FindGameObjectWithTag("Player");
}
Enemy_Running();

enemy_information = anim.GetCurrentAnimatorStateInfo(0);
distance_to_player = Vector3.Distance(transform.position, player.transform.position);

if (destination_run == true && Piglins == true)
{
    chasing_Range = 0;
}

if (distance_to_player <= chasing_Range && destination_run == false)
{
    Check_If_Player_is_InSight();

    if (player_is_inSight == true)
    {
        //last_seen_position = player.transform.position;
        search_Timer = 0f;
        nav.destination = player.transform.position;
        Main_Attack_System();
    }
    else if (!player_is_inSight && last_seen_position != Vector3.zero)
    {
        NavMeshPath path = new NavMeshPath();
        nav.CalculatePath(last_seen_position, path);
        if (path.status != NavMeshPathStatus.PathComplete)
        {
            Look_Aroun_Yourself();
        }
        else if (look_for_player == true)
        {
            search_Timer += Time.deltaTime;
            if (search_Timer >= time_for_search)
            {
                look_for_player = false;
                search_Timer = 0f;
            }
            //Debug.Log(search_Timer);
            Look_Aroun_Yourself();

        }
    }

}
```

```
if (Goblin_Warrior == true && look_for_player == false)
{
    if(patrol_main_obj != null)
    {
        Patrol();
    }
    Correct_Aggression();
}

if (distance_to_player <= chasing_Range)
{
    is_patroling = false;
}

if (roll_out == true && roll_is_active == false)
{
    roll_is_active = true;
    Roll();
    StartCoroutine(Reset_Roll_Triger());
}

//Debug.Log(Skeleton + " " + can_call_support + " " + sup_skill_used);
if (Skeleton == true && can_call_support == true && sup_skill_used == false)
{
    bool enemy_is_near_skeleton = Search_Enemy_Near_Skeleton();
    if (enemy_is_near_skeleton == false && distance_to_player <= 9f && SaveScript.agression_lvl > 0.7f)
    {
        sup_skill_used = true;
        SaveScript.agression_lvl -= 0.5f;
        anim.SetTrigger("skill");
        Spawn_Reinforcment();
        StartCoroutine(Reset_Sup_Skill());
        change_position = false;
    }
}

//curr_HP = was
//full_hp - are
if (curr_HP > full_HP)
{
    anim.SetTrigger("hit");
    curr_HP = full_HP;
    RandomAudio_Hit();
    fillHealth = Convert.ToSingle(full_HP) / Convert.ToSingle(maxHP);
    Debug.Log(fillHealth);
    HP_bar.fillAmount = fillHealth;
```

```
                if (GetComponent<Enemy_Type>().enemyType == Enemy_Type.EnemyType.Piglin)
                {
                    piglin_was_hit = true;
                    chasing_Range = 60f;
                    StartCoroutine(Reset_Piglin_Renge());
                }

            }

            if (full_HP < maxHP / 2 && Piglins == true && destination_run == false)
            {
                destination_run = true;
                chasing_Range = 0;
                //Debug.Log("RUN AWAy");
                Run_Away();
            }

        }



        Vector3 dest = nav.destination;
        if (Vector3.Distance(current_enemy.transform.position, dest) <= 1.0f)
        {
            StartCoroutine(Reset_RunAwayTrigger());
        }

        //Debug.Log(nav.isStopped);
        //Debug.Log(Vector3.Distance(current_enemy.transform.position, dest));

        if (full_HP <= 1 && enemy_is_alive == true)
        {
            Enemy_is_Dead();
        }

    }

    public void Main_Attack_System()
    {
        if (is_patroling == false && Goblin_Warrior == true || Piglins == true && piglin_was_hit == true || Skeleton ==
true && change_position == false)
        {
            if (distance_to_player < attack_Range || distance_to_player > chasing_Range && destination_run != true) //if
character is out of view  range or attack range - than enemy stop
            {
                if (destination_run != true)
                {
                    nav.isStopped = true;
                }

                //if(distance_to_player < chasing_Range)
                // {
                //Look_At_Player_Spherical_LERP();        //can be claimed as self-directed attack
                // }


                if (distance_to_player < attack_Range && enemy_information.IsTag("nonAttack") &&
SaveScript.is_invisible != true && destination_run != true)
                {

                    if (is_attacking == false)
                    {

                        is_attacking = true;
                        anim.SetTrigger("attack");
                        Look_At_Player_Spherical_LERP();   //little bit chunky
                    }
                }

                if (distance_to_player < attack_Range && enemy_information.IsTag("attack"))
                {

                    if (is_attacking == true)
                    {
                        is_attacking = false;
                    }
                }
            }
            else if (distance_to_player > attack_Range && enemy_information.IsTag("nonAttack") &&
!anim.IsInTransition(0))
            {
                if (SaveScript.is_invisible == false && destination_run == false)
                {
                    Go_To_Player();
                }

            }

        }
    }

    public void Go_To_Player()
    {
        NavMeshPath path = new NavMeshPath();
        if (NavMesh.CalculatePath(transform.position, player.transform.position, NavMesh.AllAreas, path))
        {
            if (path.status == NavMeshPathStatus.PathComplete)
            {
                nav.destination = player.transform.position;
                nav.isStopped = false;
            }
            else if (path.status == NavMeshPathStatus.PathPartial)
            {
                nav.destination = path.corners[path.corners.Length - 1];
                nav.isStopped = false;
            }
            else
            {
                nav.isStopped = true;
            }
        }
        else
        {
            nav.isStopped = true;
        }
        if (nav.isStopped && nav.velocity.sqrMagnitude < 0.1f)
        {
            nav.speed = 0;
        }
        else
        {
            nav.speed = 3.5f;
        }
        if(Piglins == true)
        {
            nav.stoppingDistance = 2f;
        }
        else
        {
            nav.stoppingDistance = stop_distance;
        }

    }


    public void Look_At_Player_Spherical_LERP()
    {
        Vector3 Pos = (player.transform.position - transform.position).normalized;
        Quaternion PosRotation = Quaternion.LookRotation(new Vector3(Pos.x, 0, Pos.z));
        transform.rotation = Quaternion.Slerp(transform.rotation, PosRotation, Time.deltaTime * rotation_speed);
    }
```

```
public void Enemy_is_Dead()
{
    SaveScript.agression_lvl = SaveScript.agression_lvl + 0.2f;
    enemy_is_alive = false;
    nav.isStopped = true;
    anim.SetTrigger("death");
    SaveScript.amount_of_chasing_enemies--;
    current_enemy.GetComponent<Outline>().enabled = false;
    is_outliner_active = false;
    nav.avoidancePriority = 1;
    StartCoroutine(Loot_Spawn());
}

public void Enemy_Outline()
{
    //outline
    if (is_outliner_active == false)
    {
        is_outliner_active = true;
        if (SaveScript.spell_target == current_enemy)
        {
            current_enemy.GetComponent<Outline>().enabled = true;
        }
    }
    if (is_outliner_active == true)
    {
        if (SaveScript.spell_target != current_enemy)
        {
            current_enemy.GetComponent<Outline>().enabled = false;
            is_outliner_active = false;
        }
    }
    //
}

public void Enemy_Running()
{
    x = nav.velocity.x;
    z = nav.velocity.z;
    velocitySpeed = new Vector2(x, z).magnitude;
    // velocitySpeed = x+z;
    if (velocitySpeed == 0)
    {
        anim.SetBool("running", false);
        // Debug.Log("RUN = " + check);
    }
    else if (velocitySpeed != 0)
    {

        anim.SetBool("running", true);
        // check = anim.GetBool("running");
        is_attacking = false;
        //Debug.Log("running = " + check);

    }
}

public void RandomAudio_Hit()
{
    int randomNumber = UnityEngine.Random.Range(1, 101);
    if (randomNumber > 0 && randomNumber < 33)
    {
        audio_Player.clip = get_Hit_SFX[0];
    }
    else if (randomNumber >= 33 && randomNumber < 66)
    {
        audio_Player.clip = get_Hit_SFX[1];
    }
    else if (randomNumber >= 66 && randomNumber < 101)
    {
        audio_Player.clip = get_Hit_SFX[2];
    }
    audio_Player.Play();
}

IEnumerator Loot_Spawn()
{
    Enemy_Type enemy_type = GetComponent<Enemy_Type>();
    if (enemy_type.enemyType == Enemy_Type.EnemyType.Skelet)
    {
        yield return new WaitForSeconds(2);
    }
    else
    {
        yield return new WaitForSeconds(1);
    }
    Instantiate(Loot_from_Enemy, transform.position, transform.rotation);
    SaveScript.killed_enemy++;
    Destroy(gameObject, 0.2f);
}

public void Run_Away()
{
    anim.SetBool("running", true);
    nav.isStopped = false;

    //int pos = Random.Range(0, 3);
    //nav.destination = escape_target_point[pos].transform.position;
    Calculate_Escape_Point();
    nav.speed = 1.8f;
    nav.destination = escape_point;

}

IEnumerator Reset_RunAwayTrigger()
{
    yield return new WaitForSeconds(5);
    destination_run = false;
}

IEnumerator Reset_Roll_Triger()
{
    yield return new WaitForSeconds(3f);
    roll_out = false;
    roll_is_active = false;
}

IEnumerator Reset_Piglin_Renge()
{
    yield return new WaitForSeconds(7f);
    Look_At_Player_Spherical_LERP();
    piglin_was_hit = false;
    if(SaveScript.weapon_index != -1)
    {
        chasing_Range = 3f;
    }
    else
    {
        chasing_Range = 12f;
    }

}
IEnumerator Reset_Sup_Skill()
{
    yield return new WaitForSeconds(sup_skill_CD);
    sup_skill_used = false;
}

IEnumerator Wait_and_Attack()
{
    yield return new WaitForSeconds(10f);
```

```
        Main_Attack_System();
    }
    public void Calculate_Escape_Point()
    {

        Vector3 escape_dir = Vector3.zero;
        float max_escape_distance = 0f;
        Vector3 player_dir = (player.transform.position - transform.position).normalized;

        for (int i = 0; i < 360; i += 5)
        {
            Vector3 new_direction = Quaternion.Euler(0, i, 0) * transform.forward;
            if (Vector3.Dot(new_direction.normalized, player_dir) < 0)
            {
                NavMeshHit hit;
                if (NavMesh.Raycast(transform.position, transform.position + new_direction * 100f, out hit,
NavMesh.AllAreas))
                {
                    float distance = Vector3.Distance(transform.position, hit.position);
                    if (distance > max_escape_distance)
                    {
                        max_escape_distance = distance;
                        escape_dir = new_direction;
                    }
                }
            }
        }
        if (max_escape_distance > 0f && escape_dir != Vector3.zero)
        {
            NavMeshHit ray_hit_for_escape;
            if (NavMesh.SamplePosition(transform.position + escape_dir * max_escape_distance, out ray_hit_for_escape,
max_escape_distance, NavMesh.AllAreas))
            {
                escape_point = ray_hit_for_escape.position;
            }
            else
            {
                escape_point = transform.position;
            }
        }

    }

    public void Set_Petrol_Destination()
    {
        Vector3 rand_dirrection = UnityEngine.Random.insideUnitSphere * patrol_radius;
        rand_dirrection += patrol_main_obj.position;
    NavMeshHit navHit;
    NavMesh.SamplePosition(rand_dirrection, out navHit, patrol_radius, -1);

    anim.SetBool("running", true);
    nav.isStopped = false;
    nav.destination = navHit.position;
}

public void Patrol()
{
    is_patroling = true;
    if (!is_waiting && nav.remainingDistance <= 2.0f)
    {

        is_waiting = true;
        wait_timer = wait_time_at_point;
        is_patroling = false;
    }
    if (is_waiting)
    {
        wait_timer -= Time.deltaTime;

        if (wait_timer <= 0 || SaveScript.is_invisible == true)
        {

            is_waiting = false;
            Set_Petrol_Destination();
            nav.isStopped = false;
        }
    }

    if (SaveScript.is_invisible == true)
    {
        is_waiting = false;
        Set_Petrol_Destination();
        nav.isStopped = false;
    }
}

public void Roll()
{
    Vector3 playerDirection = player.transform.position - transform.position;
    playerDirection.Normalize();

    Vector3[] roll_dirrections = {
        -transform.forward,   // roll back
        transform.forward,    // roll forward
        -transform.right,     // roll left
        transform.right       // roll right
    };

    string[] anim_Roll_triggers = {
        "roll_F",
        "roll_B",
        "roll_L",
        "roll_R"
    };
    float[] weights = new float[roll_dirrections.Length];
    for (int i = 0; i < roll_dirrections.Length; i++)
    {
        Vector3 roll_pos = transform.position + roll_dirrections[i] * dodgeDistance;
        if (NavMesh.SamplePosition(roll_pos, out NavMeshHit hit, 1.0f, NavMesh.AllAreas))
        {
            // Calculate weight based on direction, distance to player, and aggression level
            float weight_of_dirrection = Vector3.Dot(playerDirection, roll_dirrections[i]);
            weight_of_dirrection = (1 - Mathf.Abs(weight_of_dirrection)) * (1 - aggression_lvl);
            weights[i] = weight_of_dirrection;
        }
        else
        {
            weights[i] = -1; // Invalid direction
        }
    }
    int the_best_dirrection = -1;
    float the_best_weight = -1;
    for (int i = 0; i < weights.Length; i++)
    {
        if (weights[i] > the_best_weight)
        {
            the_best_weight = weights[i];
            the_best_dirrection = i;
        }
    }

    if (the_best_dirrection != -1)
    {
        anim.SetTrigger(anim_Roll_triggers[the_best_dirrection]);
    }
}
public void Correct_Aggression()
{
    if (curr_HP < 0.5f)
    {
        aggression_lvl -= aggression_increase * Time.deltaTime;
    }
    else
    {
        aggression_lvl += aggression_decrease * Time.deltaTime;
    }

    float distanceToPlayer = Vector3.Distance(transform.position, player.transform.position);
    if (distanceToPlayer < 10f)
    {
        aggression_lvl += aggression_increase * Time.deltaTime;
        playerNearby = true;
    }
    else
    {
        playerNearby = false;
    }

    aggression_lvl = Mathf.Clamp(aggression_lvl, min_aggression, max_aggression);

    if(aggression_lvl == 1)
    {
        StartCoroutine(Reset_Aggression_Lvl());
    }

    Debug.Log("Aggression Level: " + aggression_lvl);
}

IEnumerator Reset_Aggression_Lvl()
{
    yield return new WaitForSeconds(3f);
    aggression_lvl = 0.2f;
}
public bool Search_Enemy_Near_Skeleton()
{
    Collider[] all_colliders = Physics.OverlapSphere(transform.position, 10f);
    foreach (Collider collider in all_colliders)
    {
        if (collider.CompareTag("enemy") && collider.gameObject != gameObject)
        {
            return true;
        }
    }
    return false;
}

public void Spawn_Reinforcment()
```

```
{
    for (int i = 0; i < amount_of_reinforcment; i++)
    {

        Instantiate(support_enemy, GetRandom_Point_Around(), Quaternion.identity);
        support_enemy.GetComponent<EnemyMovement>().Goblin_Warrior = true;
        support_enemy.GetComponent<EnemyMovement>().patrol_main_obj = current_enemy.transform;
        SaveScript.amount_of_chasing_enemies++;
    }
}

public Vector3 GetRandom_Point_Around()
{
    float angle = UnityEngine.Random.Range(0f, Mathf.PI * 2);
    float x = Mathf.Cos(angle) * 8f;
    float z = Mathf.Sin(angle) * 8f;
    Vector3 point_for_spawn = new Vector3(transform.position.x + x, transform.position.y, transform.position.z + z);
    return point_for_spawn;
}


void Check_If_Player_is_InSight()
{
    Vector3 player_dir = player.transform.position - transform.position;
    float angle = Vector3.Angle(player_dir, transform.forward);

    if (angle < 90f && player_dir.magnitude < distance_of_ray)
    {
        RaycastHit hit;

        if (Physics.Raycast(transform.position + transform.up, player_dir.normalized, out hit, distance_of_ray))
        {
            Debug.DrawRay(transform.position, player_dir * 10f, Color.red);
            if (hit.transform == player.transform)
            {
                Debug.DrawRay(transform.position, player_dir * 10f, Color.green);

                Nearby_Enemy_Will_Know();
                look_for_player = false;
                player_is_inSight = true;
                last_seen_position = player.transform.position;
            }
        }
    }
    else if (player_is_inSight)
    {
        Debug.DrawRay(transform.position, player_dir * 10f, Color.red);
```

```
                player_is_inSight = false;
                nav.SetDestination(last_seen_position);
                look_for_player = true;
            }
        }

        public void Nearby_Enemy_Will_Know()
        {
            try
            {
                Vector3 player_dir = player.transform.position - transform.position;
                Collider[] all_colliders = Physics.OverlapSphere(transform.position, group_brain_radius);
                foreach (var collider in all_colliders)
                {
                    EnemyMovement raycast_system = collider.GetComponent<EnemyMovement>();
                    if (raycast_system != null && collider.gameObject != gameObject)
                    {
                        Debug.Log(raycast_system + " KNOW");
                        Debug.DrawRay(transform.position, player_dir * 10f, Color.green);
                        raycast_system.player_is_inSight = true;
                        raycast_system.look_for_player = false;
                        raycast_system.last_seen_position = player.transform.position;
                    }
                }
            }
            catch (Exception e)
            {
                Debug.Log(e);
            }

        }

        public void Look_Aroun_Yourself()
        {
            transform.Rotate(0, 120 * Time.deltaTime, 0);
        }
    }
```

1.1.2 Файл EnemyAttack.cs

```
public class Enemy_Attack : MonoBehaviour
{
    private AudioSource audio_Player;
    private bool enemy_can_attack = true;
    public float damage_enemy = 0.1f;
    private WaitForSeconds wait_before_attack = new WaitForSeconds(1);

    private float correct_dmg_reduce_by_Skill;
    private float correct_dmg_reduce_by_armor;
    void Start()
    {
        audio_Player = GetComponent<AudioSource>();
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
           // Debug.Log("Attack = true");
            float dmg_check;
            if (enemy_can_attack == true && SaveScript.is_Immmortal_object != true)
            {
                Deal_DMG_to_Character();
                SaveScript.time_of_last_damage_recive = Time.time;
                audio_Player.Play();
                StartCoroutine(DMG_Delay_Restart());
            }
        }
    }

    IEnumerator DMG_Delay_Restart()
    {
        yield return wait_before_attack;
        enemy_can_attack = true;
    }

    public void Deal_DMG_to_Character()
    {
        correct_dmg_reduce_by_armor = 1.0f - SaveScript.armora_decrease;
        enemy_can_attack = false;
        if(SaveScript.is_shielf_active == true)
        {
            SaveScript.agression_lvl = SaveScript.agression_lvl + 0.05f;
            correct_dmg_reduce_by_Skill = 1.0f - SaveScript.damage_reduce_by_Guardianship;
            SaveScript.health -= (damage_enemy * correct_dmg_reduce_by_armor * correct_dmg_reduce_by_Skill);
        }
        else
        {
            SaveScript.agression_lvl = SaveScript.agression_lvl + 0.1f;
            SaveScript.health -= damage_enemy * correct_dmg_reduce_by_armor;
        }
    }
}
```

1.1.3 Файл Golem_Movement.cs

```
public class Golem_Movement : MonoBehaviour
{
    public GameObject Loot_from_Enemy;
    public bool Golem = true;

    public GameObject current_enemy;
    private bool is_outliner_active = false;

    private AnimatorStateInfo enemy_information;
    private NavMeshAgent nav;
    private Animator anim;
    private float x;
    private float z;
    private float velocitySpeed;
    public GameObject player;
    private float distance_to_player;
    private bool is_attacking;
    public float attack_Range = 2.0f;
    public float chasing_Range;   //range in which enemy will run after character
    public float rotation_speed = 500.0f; //perfect
    public float dmg_block_probability = 0.15f;

    private bool is_reset = false;
    private bool stun = false;
    private int maxHP;

    public float golem_stamina_MAX = 1.0f;
    public float golem_stamina;
    public float golem_stamina_regeneration = 0.05f;

    public int full_HP = 100;
    private int curr_HP;
    private int fear_lvl = 100;
    private int fear_lvl_curr;
    public bool enemy_is_alive = true;

    private bool skill_was_used = false;

    public AudioSource audio_Player;
    public AudioClip[] get_Hit_SFX;

    public AudioClip block_SFX;

    public GameObject bar_Container;
    public Image HP_bar;
    private float fillHealth;
    public GameObject main_camera;

    // Start is called before the first frame update
    void Start()
    {
        audio_Player = GetComponent<AudioSource>();
        current_enemy.GetComponent<Outline>().enabled = false;
        nav = GetComponent<NavMeshAgent>();
        anim = GetComponent<Animator>();
        nav.avoidancePriority = UnityEngine.Random.Range(1, 1);
        curr_HP = full_HP;
        maxHP = full_HP;
        golem_stamina = golem_stamina_MAX;


    }

    // Update is called once per frame
    void Update()
    {
        bar_Container.transform.LookAt(main_camera.transform.position);
        //HP_bar.transform.LookAt(main_camera.transform.position);



        if (enemy_is_alive == true)
        {
            //outline
            if (is_outliner_active == false)
            {
                is_outliner_active = true;
                if (SaveScript.spell_target == current_enemy)
                {
                    current_enemy.GetComponent<Outline>().enabled = true;
                }
            }
            if (is_outliner_active == true)
            {
                if (SaveScript.spell_target != current_enemy)
                {
                    current_enemy.GetComponent<Outline>().enabled = false;
                    is_outliner_active = false;
                }
            }
```

```
//

Golem_Stamina_Regeneration();

if (player == null)
{
   player = GameObject.FindGameObjectWithTag("Player");
}
x = nav.velocity.x;
z = nav.velocity.z;
velocitySpeed = new Vector2(x, z).magnitude;
if (velocitySpeed == 0)
{
   anim.SetBool("running", false);
}
else if (velocitySpeed != 0)
{

   anim.SetBool("running", true); ;
   is_attacking = false;

}



enemy_information = anim.GetCurrentAnimatorStateInfo(0);
distance_to_player = Vector3.Distance(transform.position, player.transform.position);

//Debug.Log(distance_to_player);

if (enemy_information.IsName("atk_dash") == true && skill_was_used == false)
{
   //golem_stamina -= 0.6f;
   skill_was_used = true;
}

if (skill_was_used == true)
{
   StartCoroutine(Reset_Dash());
}


if (distance_to_player >= 10.0f )//&& golem_stamina > 0.61f)
{

   anim.SetBool("player_too_far", true);
}
else
{
   anim.SetBool("player_too_far", false);
}






if (golem_stamina > 0.01f)
{

   if (distance_to_player < attack_Range || distance_to_player > chasing_Range)
   {
      nav.isStopped = true;


      if (distance_to_player < attack_Range && enemy_information.IsTag("nonAttack") && SaveScript.is_invisible !=
true)  //&& golem_stamina > 0.1f)
      {


         if (is_attacking == false)
```

```
            {
                Look_At_Player_Spherical_LERP();

                int randomNumber = UnityEngine.Random.Range(1, 101);
                if (randomNumber > 0 && randomNumber < 51)
                {
                    if (distance_to_player <= 2.0f)
                    {
                        is_attacking = true;
                        //golem_stamina -= 0.1f;
                        anim.SetTrigger("player_too_close");

                    }

                }
                else
                {
                    int randomNumber2 = UnityEngine.Random.Range(1, 101);
                    is_attacking = true;
                    //golem_stamina -= 0.1f;
                    anim.SetInteger("random", randomNumber2);
                    anim.SetTrigger("attack");

                }

            }
        }

        if (distance_to_player < attack_Range && enemy_information.IsTag("attack"))
        {
            if (is_attacking == true)
            {
                is_attacking = false;
            }
        }
    }
    else if (distance_to_player > attack_Range && enemy_information.IsTag("nonAttack") && !anim.IsInTransition(0))
    {

        if (SaveScript.is_invisible == false)
        {
            nav.isStopped = false;
            nav.destination = player.transform.position;
        }
    }
}


//curr_HP = was
//full_hp - are
if (curr_HP > full_HP)
{
    golem_stamina -= 0.05f;
    anim.SetTrigger("hit");
    curr_HP = full_HP;
    RandomAudio_Hit();
    fillHealth = Convert.ToSingle(full_HP) / Convert.ToSingle(maxHP);
    Debug.Log(fillHealth);
    HP_bar.fillAmount = fillHealth;

}


if (nav.isStopped == false || distance_to_player > 6.0f && enemy_information.IsTag("attack"))
{
    anim.ResetTrigger("player_near");
    anim.ResetTrigger("player_too_close");
    anim.ResetTrigger("attack");
    if (is_attacking == true)
    {
        is_attacking = false;
```

```
            }
        }


        if (full_HP < maxHP / 2 && stun == false)
        {
            stun = true;
            StartCoroutine(Stun_Duration());

        }


        if (full_HP <= 1 && enemy_is_alive == true)
        {
            enemy_is_alive = false;
            nav.isStopped = true;
            anim.SetTrigger("death");
            current_enemy.GetComponent<Outline>().enabled = false;
            is_outliner_active = false;
            nav.avoidancePriority = 1;
            StartCoroutine(Loot_Spawn());
        }


    }
}

public void Look_At_Player_Spherical_LERP()
{
    Vector3 Pos = (player.transform.position - transform.position).normalized;
    Quaternion PosRotation = Quaternion.LookRotation(new Vector3(Pos.x, 0, Pos.z));
    transform.rotation = Quaternion.Slerp(transform.rotation, PosRotation, Time.deltaTime * rotation_speed);
}



public void RandomAudio_Hit()
{
    int randomNumber = UnityEngine.Random.Range(1, 101);
    if (randomNumber > 0 && randomNumber < 33)
    {
        audio_Player.clip = get_Hit_SFX[0];
    }
    else if (randomNumber >= 33 && randomNumber < 66)
    {
        audio_Player.clip = get_Hit_SFX[1];
    }
    else if (randomNumber >= 66 && randomNumber < 101)
    {
        audio_Player.clip = get_Hit_SFX[2];
    }
    audio_Player.Play();
}

public void Golem_Stamina_Regeneration()
{
    golem_stamina += golem_stamina_regeneration * Time.deltaTime;
    golem_stamina = Mathf.Clamp(golem_stamina, 0, golem_stamina_MAX);

}

IEnumerator Loot_Spawn()
{
    yield return new WaitForSeconds(2.5f);
    Instantiate(Loot_from_Enemy, transform.position, transform.rotation);
    SaveScript.killed_enemy++;
    Destroy(gameObject, 0.2f);
}

IEnumerator Stun_Duration()
{
    anim.SetTrigger("stun_start");
    nav.isStopped = true;
    yield return new WaitForSeconds(5);
    anim.SetTrigger("stun_end");
    nav.isStopped = false;
}

IEnumerator Reset_Dash()
{
    yield return new WaitForSeconds(5);
    skill_was_used = false;
}
        }
```

1.2 Реалізація ігрового інтерфейсу.

1.2.1 Файл Lvl_Up_Stats.cs

```
public class Lvl_Up_Stats : MonoBehaviour
{
    public AudioClip selection;
    public AudioSource Inventory_Canvas;




    public void Lvl_UP_Strength()
    {
        if (SaveScript.points_to_upgrade > 0)
        {
            // SaveScript.strength_basic += SaveScript.player_lvl_character;
            SaveScript.strength_basic += 0.05f;
            SaveScript.points_to_upgrade--;
        }
    }

    public void Lvl_UP_Intelligence()
    {
        if (SaveScript.points_to_upgrade > 0)
        {
            // SaveScript.intelligence_basic += SaveScript.player_lvl_character;
            SaveScript.intelligence_basic += 0.05f;
            SaveScript.points_to_upgrade--;
        }
    }

    public void Lvl_UP_Stamina()
    {
        if (SaveScript.points_to_upgrade > 0)
        {
            //  SaveScript.stamina_basic += SaveScript.player_lvl_character;
            SaveScript.stamina_basic += 0.05f;
            SaveScript.points_to_upgrade--;
        }
    }
}
```

1.2.2 Файл Main_Menu.cs

```
public class Main_Menu : MonoBehaviour
{
    public GameObject continue_;
    public GameObject load_;
    public GameObject save_;
    void Start()
    {
        TurnOn_Continue_If_Exists();
        Cursor.visible = true;
    }



    public void Start_New_Game()
    {
        SceneManager.LoadScene(1);
    }

    public void Continue_Button()
    {
        load_.SetActive(true);
        save_.SetActive(true);
        SaveScript.take_data_to_load = true;
        StartCoroutine(LoadGame());
    }

    public void Exit()
    {
        Application.Quit();
    }
    public void Settings()
    {

    }
    IEnumerator LoadGame()
    {
        yield return new WaitForSeconds(1);
        SceneManager.LoadScene(2);
    }
    public void TurnOn_Continue_If_Exists()
    {
        if (Application.persistentDataPath + "/preservation.data" != null)
        {
            continue_.SetActive(true);
        }
        else
        {
            continue_.SetActive(false);
        }
    }

}
```

1.3 Реалізація механік бою.

1.3.1 PlayerMovement.cs

```
public class PlayerMovement : MonoBehaviour
{

    private UnityEngine.AI.NavMeshAgent nav;
    private Animator anim;
    private Ray ray;
    private RaycastHit hit;

    private float x;
    private float z;
    private float velocitySpeed;
    public static int ray_numbers = 6;

    //For Camera
    CinemachineTransposer cinemachineTransposer;
    //public CinemachineVirtualCamera playerCamera;  //free
    CinemachineOrbitalTransposer cinemachine_orbital_Transposer;

    private Vector3 mouse_pos;
    private Vector3 current_pos;
    private string axis_named = "Mouse X";

    private bool isPlayerSelectScene;
    public static bool canMove = true;
    public static bool isPlayerMoving = false;

    public GameObject camera_1_static;
    public GameObject camera_2_free;
    private bool is_camera1_active = true;

    private float previous_health = 1.0f;
```

```
        public GameObject get_hit_VFX_Place;
        private WaitForSeconds life_time_hit_effect = new WaitForSeconds(0.1f);



        //for roof box colider
        public LayerMask boxLayer;

        public GameObject vfx_spawm_point;
        private WaitForSeconds nearEnemy = new WaitForSeconds(0.4f);

        public GameObject[] player_mesh_parts;
        public GameObject[] weapons_props;
        public GameObject[] armor_parts_Torso;
        public GameObject[] armor_parts_Legs;
        public string[] attacks_tags;
        public AudioClip[] weapon_SFX;
        public AudioSource audio_Player;

        private AnimatorStateInfo player_information;

        private GameObject trail_mesh;
        private WaitForSeconds traill_time = new WaitForSeconds(0.1f);
        public bool critical_attack_is_active = false;

        public float[] stamina_cost_for_weapon;
        void Start()
        {
            nav = GetComponent<UnityEngine.AI.NavMeshAgent>();
            anim = GetComponent<Animator>();



            camera_1_static.SetActive(false);
            camera_2_free.SetActive(true);
            SaveScript.vfx_spawn_point = vfx_spawm_point;
            //cinemachineTransposer = playerCamera.GetCinemachineComponent<CinemachineTransposer>();
            //current_pos = cinemachineTransposer.m_FollowOffset;
            cinemachineTransposer                                                          =
camera_1_static.gameObject.GetComponent<CinemachineVirtualCamera>().GetCinemachineComponent<CinemachineTranspo
ser>();
            cinemachine_orbital_Transposer                                                 =
camera_2_free.gameObject.GetComponent<CinemachineVirtualCamera>().GetCinemachineComponent<CinemachineOrbitalTr
ansposer>();

            for (int i = 0; i < weapons_props.Length; i++)
            {
                weapons_props[i].SetActive(false);
            }
```

```
    if (SceneManager.GetActiveScene().name == "PlayerSelect")
    {
        isPlayerSelectScene = true;

    }

    if(SceneManager.GetActiveScene().buildIndex == 2)
    {
        Display_Correct_ArmorInShop();
    }
    Check_Class_Info();
    get_hit_VFX_Place.SetActive(false);

}

void Update()
{
    if (SceneManager.GetActiveScene().buildIndex == 2)
    {
        Display_Correct_ArmorInShop();
    }
    //Debug.Log("can mpve " + canMove);
    player_information = anim.GetCurrentAnimatorStateInfo(0); //listen to Animator

    //change correct weapon
    if (SaveScript.should_change_weapon == true)
    {
        SaveScript.should_change_weapon = false;
        for (int i = 0; i < weapons_props.Length; i++)
        {
            weapons_props[i].SetActive(false);
        }
        weapons_props[SaveScript.weapon_index].SetActive(true);
        StartCoroutine(WaitForTrail());
    }


    if (isPlayerSelectScene == false)
    {
        x = nav.velocity.x;
        z = nav.velocity.z;
        velocitySpeed = new Vector2(x, z).magnitude;

        Ray[] rays = new Ray[ray_numbers];

        if (Input.GetMouseButtonDown(0) && player_information.IsTag("nonAttack") && !anim.IsInTransition(0))
```

```
{
    if (canMove == true)
    {
        for (int i = 0; i < ray_numbers; i++)
        {
            rays[i] = Camera.main.ScreenPointToRay(Input.mousePosition);
        }

        Vector3 averageHitPoint = Vector3.zero;

        foreach (Ray ray in rays)
        {
            RaycastHit hit;

            if (Physics.Raycast(ray, out hit, 300, boxLayer))
            {
                if (hit.transform.gameObject.CompareTag("enemy"))
                {
                    nav.isStopped = false;
                    SaveScript.spell_target = hit.transform.gameObject;
                    averageHitPoint += hit.point;
                    transform.LookAt(SaveScript.spell_target.transform);
                    StartCoroutine(MoveTo()); //wait 3 sec and than isStopped == true

                }
                else
                {
                    SaveScript.spell_target = null;
                    averageHitPoint += hit.point;
                    nav.isStopped = false;
                }
            }
        }
        averageHitPoint /= rays.Length;
        nav.destination = averageHitPoint;
    }
}


if (Input.GetMouseButton(1))
{
    cinemachine_orbital_Transposer.m_XAxis.m_InputAxisName = axis_named;   //we put "Mouse X" into field
of orbital camera to be able to rotate it
}

if (Input.GetMouseButtonUp(1))
{
        cinemachine_orbital_Transposer.m_XAxis.m_InputAxisName = null;
        cinemachine_orbital_Transposer.m_XAxis.m_InputAxisValue = 0;
    }



    // Check if the character is moving (forward or backward)
    anim.SetBool("sprinting", velocitySpeed > 0.1f);
    if(velocitySpeed != 0)
    {
        if(SaveScript.is_character_equip_a_weapon == false)
        {
            anim.SetBool("sprinting", true);
            anim.SetBool("equip_a_weapon", false);
        }
        if (SaveScript.is_character_equip_a_weapon == true)
        {
            anim.SetBool("sprinting", true);
            anim.SetBool("equip_a_weapon", true);
        }
        isPlayerMoving = true;
    }
    if (velocitySpeed == 0)
    {
        anim.SetBool("sprinting", false);
        isPlayerMoving = false;
    }



    if (Input.GetKeyDown(KeyCode.S))
    {
        anim.SetBool("sprinting", false);
        nav.destination = transform.position;
    }
}

if (Input.GetKeyDown(KeyCode.C))
{
    if(is_camera1_active == true)
    {
        camera_1_static.SetActive(false);
        camera_2_free.SetActive(true);

        is_camera1_active = false;
    }
    else if (is_camera1_active == false)
    {
        camera_1_static.SetActive(true);
        camera_2_free.SetActive(false);

        is_camera1_active = true;
    }
}

//make player invisible
if (player_mesh_parts[0].activeSelf == true)
{
    if(SaveScript.is_invisible == true)
    {
        SaveScript.agression_lvl = SaveScript.agression_lvl - 0.15f;
        for (int i = 0; i < player_mesh_parts.Length; i++)
        {
            player_mesh_parts[i].SetActive(false);
        }
    }
}
//make player visible
if (SaveScript.mana <= 0.05)
{
    if (SaveScript.is_invisible == false)
    {
        for (int i = 0; i < player_mesh_parts.Length; i++)
        {
            player_mesh_parts[i].SetActive(true);

        }
        SaveScript.should_change_armor = true;
    }
}



if(SaveScript.should_change_armor == true)
{
    for(int i = 0; i < armor_parts_Torso.Length; i++)
    {
        armor_parts_Torso[i].SetActive(false);
        armor_parts_Legs[i].SetActive(false);
    }
    armor_parts_Torso[SaveScript.index_of_equiped_armor].SetActive(true);
    armor_parts_Legs[SaveScript.index_of_equiped_armor].SetActive(true);
    SaveScript.should_change_armor = false;
}
```

```
            if (Input.GetKeyDown(KeyCode.Z))
            {
                if (SaveScript.is_character_equip_a_weapon == true && SaveScript.stamina > 0.2)
                {
                    Basic_or_Critical_Attack();
                }
            }

            if(SaveScript.health <= 0.0f)
            {
                    if (SaveScript.uniqe_features_index == 3 && Time.time - SaveScript.time_of_uniqe_feature_activasion > SaveScript.uniqe_features_index_CD)
                {
                    SaveScript.time_of_uniqe_feature_activasion = Time.time;
                    SaveScript.health = 0.5f;
                }
                else
                {
                    SceneManager.LoadScene(0);   // 0 - Player Select  1 - Terrain1 (More can check in File -> Build Settings)
                    SaveScript.health = 1.0f;
                }

            }

            if(previous_health > SaveScript.health)
            {
                CharacterGetHit();
            }





        }

        public void Basic_or_Critical_Attack()
        {
                float randomNumber = Random.value;
                if (randomNumber <= SaveScript.critical_hit_chance)
                {

                critical_attack_is_active = true;
                anim.SetTrigger(attacks_tags[6]);
                audio_Player.volume = 0.4f;
                audio_Player.clip = weapon_SFX[6];
                audio_Player.Play();
                SaveScript.stamina -= stamina_cost_for_weapon[6];
```

```
    }
    else
        {
        critical_attack_is_active = false;
        anim.SetTrigger(attacks_tags[SaveScript.weapon_index]);
        audio_Player.volume = 0.3f;
        audio_Player.clip = weapon_SFX[SaveScript.weapon_index];
        //audio_Player.Play();
        SaveScript.stamina -= stamina_cost_for_weapon[SaveScript.weapon_index];
    }
}

IEnumerator TurnOff_Hit_VFX()
{
    yield return life_time_hit_effect;
    get_hit_VFX_Place.SetActive(false);
}

public void CharacterGetHit()
{
    get_hit_VFX_Place.SetActive(true);
    previous_health = SaveScript.health;
    StartCoroutine(TurnOff_Hit_VFX());
}
public void Weapon_SFX_Play()
{

    audio_Player.Play();
}

public void TurnOn_Trail()
{
    trail_mesh.GetComponent<Renderer>().enabled = true;
}

public void TurnOff_Trail()
{
    trail_mesh.GetComponent<Renderer>().enabled = false;
}
IEnumerator MoveTo()
{
    yield return nearEnemy;
    nav.isStopped = true;
}

IEnumerator WaitForTrail()
```

```
{
   yield return traill_time;
   trail_mesh = GameObject.Find("Trail");

   trail_mesh.GetComponent<Renderer>().enabled = false;

}

public void Check_Class_Info()
{
   if (SaveScript.uniqe_features_index == 0)
   {
      Debug.Log(SaveScript.class_Avarage + "" + SaveScript.class_Mage + "" + SaveScript.class_Seller + "" +
SaveScript.class_Warrior);
      Debug.Log("None Ability");
   }
   else if (SaveScript.uniqe_features_index == 1)
   {
      Debug.Log(SaveScript.class_Avarage + "" + SaveScript.class_Mage + "" + SaveScript.class_Seller + "" +
SaveScript.class_Warrior);
      Debug.Log("More Mana Regeneration and +20% spell/magic damage");
   }
   else if (SaveScript.uniqe_features_index == 2)
   {
      Debug.Log(SaveScript.class_Avarage + "" + SaveScript.class_Mage + "" + SaveScript.class_Seller + "" +
SaveScript.class_Warrior);
      Debug.Log("Price in shop is -20% lower");
   }
   else if (SaveScript.uniqe_features_index == 3)
   {
      Debug.Log(SaveScript.class_Avarage + "" + SaveScript.class_Mage + "" + SaveScript.class_Seller + "" +
SaveScript.class_Warrior);
      Debug.Log("You can survive lethal damage and regain 50% HP (500 sec CD)");
   }

}

public void Display_Correct_ArmorInShop()
{
   if(isPlayerSelectScene == true)
   {
      if (SaveScript.player_index_character == 1 || SaveScript.player_index_character == 2 ||
SaveScript.player_index_character == 0)
      {
         GetComponent<Stats_Info>().armor_in_shop[0].SetActive(true);
         GetComponent<Stats_Info>().armor_in_shop[1].SetActive(false);
      }
```

```
            else
            {
                GetComponent<Stats_Info>().armor_in_shop[0].SetActive(false);
                GetComponent<Stats_Info>().armor_in_shop[1].SetActive(true);
            }
        }
    }

}
```

1.3.2 Файл Character_Attack.cs

```
public class Character_Attack : MonoBehaviour
{
    private GameObject mesh_to_Destroy;
    public int basic_weapon_damage;

    private GameObject player;

    private bool can_deal_dmg = true;
    private WaitForSeconds dmg_Pause = new WaitForSeconds(0.1f);
    // Start is called before the first frame update
    void Start()
    {
        if (player == null)
        {
            player = GameObject.FindGameObjectWithTag("Player");
        }
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void OnTriggerEnter(Collider other)
    {

        //if we are attacking crate
        if (other.CompareTag("Crate"))
        {
            other.transform.gameObject.GetComponentInParent<Chest>().VFX_crate_text();
            mesh_to_Destroy = other.transform.parent.gameObject;

            Destroy(other.transform.gameObject);
            StartCoroutine(Wait_before_Destroy());
            }

            if (other.CompareTag("enemy") && can_deal_dmg == true )
            {
                SaveScript.agression_lvl = SaveScript.agression_lvl + 0.2f;

                Enemy_Type enemy_type = other.GetComponent<Enemy_Type>();
                int dmg_check = 0;
                if(player.GetComponent<PlayerMovement>().critical_attack_is_active == true)
                {
                    dmg_check = (basic_weapon_damage + SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase) *
SaveScript.critical_dmg_multiply;
                    if (enemy_type.enemyType == Enemy_Type.EnemyType.Golem)
                    {
                        if (Random.Range(0f, 1f) >= other.GetComponent<Golem_Movement>().dmg_block_probability) //15 per
cent to block dmg
                        {
                            other.transform.gameObject.GetComponent<Golem_Movement>().full_HP -= ((basic_weapon_damage
+ SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase) * SaveScript.critical_dmg_multiply);
                        }
                        else
                        {
                            other.GetComponent<Golem_Movement>().audio_Player.clip =
other.GetComponent<Golem_Movement>().block_SFX;
                            other.GetComponent<Golem_Movement>().audio_Player.Play();
                        }
                    }
                    else
                    {
                        other.transform.gameObject.GetComponent<EnemyMovement>().full_HP -= ((basic_weapon_damage +
SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase) * SaveScript.critical_dmg_multiply);

                    }
                    can_deal_dmg = false;
                }
                else
                {
                    dmg_check = (basic_weapon_damage + SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase);

                    if(enemy_type.enemyType == Enemy_Type.EnemyType.Golem)
                    {
                        if (Random.Range(0f, 1f) >= other.GetComponent<Golem_Movement>().dmg_block_probability) //15 per
cent to block dmg
                        {
                            other.transform.gameObject.GetComponent<Golem_Movement>().full_HP -= (basic_weapon_damage
+ SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase);
                        }
                        else
                        {
                            other.GetComponent<Golem_Movement>().audio_Player.clip = other.GetComponent<Golem_Movement>().block_SFX;
                            other.GetComponent<Golem_Movement>().audio_Player.Play();
                        }
                    }
                    else
                    {
                        other.transform.gameObject.GetComponent<EnemyMovement>().full_HP -= (basic_weapon_damage + SaveScript.weapon_dmg_scaleUP + SaveScript.strength_increase);
                    }


                    can_deal_dmg = false;
                }



                Debug.Log(basic_weapon_damage + " " + SaveScript.weapon_dmg_scaleUP + " " + SaveScript.strength_increase);
                if (enemy_type.enemyType == Enemy_Type.EnemyType.Golem)
                {
                    Debug.Log("Monster = " + other.name + " HP = " + other.transform.gameObject.GetComponent<Golem_Movement>().full_HP + " DMG = " + dmg_check);
                }
                else
                {
                    Debug.Log("Monster = " + other.name + " HP = " + other.transform.gameObject.GetComponent<EnemyMovement>().full_HP + " DMG = " + dmg_check);

                }
                StartCoroutine(ResetDMG());
            }


        }

        IEnumerator Wait_before_Destroy()
        {
            yield return new WaitForSeconds(2);
            Destroy(mesh_to_Destroy);
        }

        IEnumerator ResetDMG()
        {
            yield return dmg_Pause;
```

```
        can_deal_dmg = true;
    }

}
```

1.4 Реалізація функціоналу інвентарю, магазинів та ігрової економіки.

1.4.1 Файл Buying.cs

```
public class Buying : MonoBehaviour
{

    public GameObject shop;
    public GameObject Inventory_Canvas;

    public AudioSource audio_Player;

    public int[] amount_of_stuff_in_shop;
    public int[] cost_of_stuff_in_shop;
    public int[] element_number;

    public int[] inventory_items;

    public Text[] text_amount_of_stuff_in_shop;
    public Text[] text_finance;

    private Text compare;

    public bool isPub;
    public bool isWizzardShop;
    public bool isCraftsmenWorkshop;

    private int max = 0;
    private bool canClick;

    public Text[] price_per_obj;

    void Start()
    {
        shop.SetActive(false);
        max = text_amount_of_stuff_in_shop.Length;
        text_finance[0].text = Inventory.gold.ToString();
        text_finance[1].text = Inventory.diamond.ToString();

        for(int i=0; i < max; i++)
        {
            text_amount_of_stuff_in_shop[i].text = amount_of_stuff_in_shop[i].ToString();
        }

        audio_Player = Inventory_Canvas.GetComponent<AudioSource>();

        if(SaveScript.class_Seller == true)
        {
            SellerClassFeature();
        }
    }




    public void Close()
    {
        shop.SetActive(false);
        PlayerMovement.canMove = true;
    }

    public void BuyButton()
    {
        if (canClick == true)
        {
            for (int i = 0; i < max; i++)
            {
                if (text_amount_of_stuff_in_shop[i] == compare)
                {
                    max = i;
                    if (amount_of_stuff_in_shop[i] > 0)
                    {

                        if (isPub == true)
                        {
                            RefreshShopAmount();
                        }else if(isWizzardShop == true)
                        {
                            RefreshWizardShopAmount();
                        }
                        else if(isCraftsmenWorkshop == true)
                        {
                            RefreshCraftsMenShopAmount();
                        }

                        if (Inventory.gold >= cost_of_stuff_in_shop[i])
                        {
                            if (inventory_items[i] == 0)
                            {
                                Inventory.newIcon = element_number[i];
                                Inventory.iconUpdated = true;
                            }
                            Inventory.gold -= cost_of_stuff_in_shop[i];

                            //RANDOM SFX COIN
                            int randomNumber = UnityEngine.Random.Range(1, 101);
                            if (randomNumber > 0 && randomNumber < 33)
                            {
                                audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin_buy_SFX;
                            }else if (randomNumber >= 33 && randomNumber < 66)
                            {
                                audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin2_buy_SFX;
                            } else if (randomNumber >= 66 && randomNumber < 101)
                            {
                                audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin3_buy_SFX;
                            }
                                audio_Player.Play();
                            //RANDOM SFX COIN

                            if (isPub == true)
                            {
                                SetShopAmount(i);
                            }
                            else if (isWizzardShop == true)
                            {
                                SetWizzardShopAmount(i);
                            }
                            else if (isCraftsmenWorkshop == true)
                            {
                                SetCraftsMenShopAmount(i);
                            }
                        }
                    }
                }
            }
        }
    }

    void RefreshShopAmount()
    {
        inventory_items[0] = Inventory.amount_of_bread;
        inventory_items[1] = Inventory.amount_of_cheese;
        inventory_items[2] = Inventory.amount_of_meat;
    }
    void RefreshWizardShopAmount()
    {
        inventory_items[0] = Inventory.amount_of_redPotion;
        inventory_items[1] = Inventory.amount_of_bluePotion;
        inventory_items[2] = Inventory.amount_of_lazurePotion;
        inventory_items[3] = Inventory.amount_of_greenPotion;

        inventory_items[4] = Inventory.amount_of_monsterEye;
        inventory_items[5] = Inventory.amount_of_roots;
        inventory_items[6] = Inventory.amount_of_leaf;

    }
    void RefreshCraftsMenShopAmount()
    {

    }

    public void UpdateFinance()
    {
        text_finance[0].text = Inventory.gold.ToString();
        text_finance[1].text = Inventory.diamond.ToString();
    }

    void SetShopAmount(int item)
    {

        switch (item)
        {
            case 0:
                Inventory.amount_of_bread++;
                break;
            case 1:
                Inventory.amount_of_cheese++;
                break;
            case 2:
                Inventory.amount_of_meat++;
                break;

            default:
                break;
        }

        amount_of_stuff_in_shop[item]--;
        text_amount_of_stuff_in_shop[item].text = amount_of_stuff_in_shop[item].ToString();
        UpdateFinance();
        max = amount_of_stuff_in_shop.Length;
    }
    void SetWizzardShopAmount(int item)
    {
        switch (item)
        {
            case 0:
                Inventory.amount_of_redPotion++;
                break;
            case 1:
                Inventory.amount_of_bluePotion++;
                break;
            case 2:
                Inventory.amount_of_lazurePotion++;
                break;
            case 3:
                Inventory.amount_of_greenPotion++;
                break;
            case 4:
                Inventory.amount_of_monsterEye++;
                break;
            case 5:
                Inventory.amount_of_roots++;
                break;
            case 6:
                Inventory.amount_of_leaf++;
                break;



            default:
                break;
        }

        amount_of_stuff_in_shop[item]--;
        text_amount_of_stuff_in_shop[item].text = amount_of_stuff_in_shop[item].ToString();
        UpdateFinance();
        max = amount_of_stuff_in_shop.Length;
    }
    void SetCraftsMenShopAmount(int item)
    {

    }



    void CheckAmount(int items_number_general)
    {
        if (amount_of_stuff_in_shop[items_number_general] > 0)
        {
            canClick = true;
        }
        else
        {
            canClick = false;
        }

    }
    void CheckAmount_for_WizzardShop(int items_number_general_v2)
    {
        if (amount_of_stuff_in_shop[items_number_general_v2] > 0)
        {
            canClick = true;
        }
        else
        {
            canClick = false;
        }

    }

    //for Shop basic
    public void bread()
    {
        compare = text_amount_of_stuff_in_shop[0];
        CheckAmount(0);
    }
    public void cheese()
    {
        compare = text_amount_of_stuff_in_shop[1];
        CheckAmount(1);
    }
    public void meat()
    {
        compare = text_amount_of_stuff_in_shop[2];
        CheckAmount(2);
    }

    //for Wizzard Shop
    public void red_Potion()
    {
        compare = text_amount_of_stuff_in_shop[0];
        CheckAmount_for_WizzardShop(0);
    }
    public void blue_Potion()
    {
        compare = text_amount_of_stuff_in_shop[1];
        CheckAmount_for_WizzardShop(1);
    }
    public void lazure_Potion()
```

```
    {
        compare = text_amount_of_stuff_in_shop[2];
        CheckAmount_for_WizzardShop(2);
    }

    public void green_Potion()
    {
        compare = text_amount_of_stuff_in_shop[3];
        CheckAmount_for_WizzardShop(3);
    }
    public void monster_Eye()
    {
        compare = text_amount_of_stuff_in_shop[4];
        CheckAmount_for_WizzardShop(4);
    }
    public void roots()
    {
        compare = text_amount_of_stuff_in_shop[5];
        CheckAmount_for_WizzardShop(5);
    }
    public void leaf()
    {
        compare = text_amount_of_stuff_in_shop[6];
        CheckAmount_for_WizzardShop(6);
    }

    public void SellerClassFeature()
    {
        for(int i =0; i < cost_of_stuff_in_shop.Length; i++)
        {
            cost_of_stuff_in_shop[i] = (cost_of_stuff_in_shop[i] * 4) / 5; // 20 per cent lower price
            price_per_obj[i].text = cost_of_stuff_in_shop[i] + " coins";
        }
    }
}
```

1.4.2 Файл Buying_Weapons.cs

```
public class Buying_Weapons : MonoBehaviour
{
    public Text finance_text_gold;
    public Text finance_text_diamond;

    public int weapon_index;
    public int armor__index;
    public int price;
    public GameObject Inventory_Canvas;
public AudioSource audio_Player;
public Text text_price;




// Start is called before the first frame update
void Start()
{
    finance_text_diamond.text = Inventory.diamond.ToString();
    finance_text_gold.text = Inventory.gold.ToString();
    audio_Player = Inventory_Canvas.GetComponent<AudioSource>();

    text_price.text = price.ToString() +" coins";

    if (SaveScript.class_Seller == true)
    {
        SellerClassFeature();
    }
}

public void BuyButton_Weapon()
{
    if(Inventory.gold >= price)
    {
        Inventory.gold -= price;
        Inventory_Canvas.GetComponent<Inventory>().weapons[weapon_index] = true;

        //RANDOM SFX COIN
        RandomAudio();
        //
        finance_text_diamond.text = Inventory.diamond.ToString();
        finance_text_gold.text = Inventory.gold.ToString();

    }
}

public void BuyButton_Armor()
{
    if (Inventory.gold >= price)
    {
        Inventory.gold -= price;
        SaveScript.index_of_equiped_armor = armor__index;
        SaveScript.should_change_armor = true;
        //RANDOM SFX COIN
        RandomAudio();
        //
        finance_text_diamond.text = Inventory.diamond.ToString();
```

```
            finance_text_gold.text = Inventory.gold.ToString();
        }
    }




    public void RandomAudio()
    {
        int randomNumber = UnityEngine.Random.Range(1, 101);
        if (randomNumber > 0 && randomNumber < 33)
        {
            audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin_buy_SFX;
        }
        else if (randomNumber >= 33 && randomNumber < 66)
        {
            audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin2_buy_SFX;
        }
        else if (randomNumber >= 66 && randomNumber < 101)
        {
            audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().coin3_buy_SFX;
        }
        audio_Player.Play();
    }

    public void SellerClassFeature()
    {
        price = price * 4 / 5;
        text_price.text = price.ToString() + " coins";
    }
}
```

1.5 Реалізація функціонали створення та використання магії.

1.5.1 Файл Particle_Destroyer.cs

```
public class Particle_Destroyer : MonoBehaviour
{

    public float life_time_for_chest = 2.0f;
    // Start is called before the first frame update
    void Start()
    {
        Destroy(gameObject, life_time_for_chest);
    }
}
```

1.5.2 Файл Particle_Point.cs

```
public class Particle_Point : MonoBehaviour
{
    public int damage = 30;
    public float speed = 1.0f;
    public bool should_rotate = false;
    public bool move_to_target = true;

    public GameObject object_triggered;



    // Update is called once per frame
    void Update()
    {
        if (should_rotate == true)
        {
            transform.Rotate(0, speed * Time.deltaTime, 0);
        }
        if(move_to_target == true)
        {
            transform.Translate(Vector3.forward * speed * Time.deltaTime);
        }
    }



    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("enemy") && other.transform.gameObject != object_triggered)
        {
            Enemy_Type enemy_type = other.GetComponent<Enemy_Type>();
            if (SaveScript.class_Mage == true)
            {
                damage = damage * 6 / 5;
            }
            if (enemy_type.enemyType == Enemy_Type.EnemyType.Golem)
            {
                other.transform.gameObject.GetComponent<Golem_Movement>().full_HP -= (damage * 4)/5;   // 20 peer
cent magic decrease
                object_triggered = other.transform.gameObject;
            }
            else
            {
                other.transform.gameObject.GetComponent<EnemyMovement>().full_HP -= damage;
                object_triggered = other.transform.gameObject;
            }
        }

    }
}
```

1.5.3 Файл Particle_Transform.cs

```
public class Particle_Transform : MonoBehaviour
{
    //flame nova/twist
    public GameObject target_point;
    public GameObject vfx_object_container;
    public float speed = 5.0f;
    public float duration_of_life = 1.5f;
    public float spell_mana_cost = 0.06f;



    private GameObject vfx_target_save;
    public GameObject player;
    //
    public bool enemy_search = false ;
    public bool non_moving = false;
    public bool support_spell_follow_player = false;
    public bool shield_spell = false;
    public bool power_stats_up_spell = false;
    public bool heal_magic = false;



    public bool invisibility_spell_is_active = false;

    public GameObject object_triggered;
    public int damage = 30;




    private void Start()
    {

        vfx_target_save = SaveScript.spell_target;
        player = GameObject.FindGameObjectWithTag("Player") ;
        if(invisibility_spell_is_active == true)
        {
            SaveScript.is_invisible = true;
        }

        if (shield_spell == true)
```

```
            {
                SaveScript.is_shielf_active = true;
            }
            if(power_stats_up_spell == true)
            {
                SaveScript.strength_increase = 100;
            }

        }
        // Update is called once per frame
        void Update()
        {



            if (target_point != null) //avarage target spel position - worl*
            {
                transform.position = Vector3.LerpUnclamped(transform.position/*current pos*/,
target_point.transform.position/*target pos*/, speed * Time.deltaTime);   //fuction to move between object a and b with speed c
(from curtrent position to target with speed multiplied by delta time
            }
            if(enemy_search == true) //enemy search spell attack
            {
                if (vfx_target_save != null)
                {
                    transform.position = Vector3.LerpUnclamped(transform.position, vfx_target_save.transform.position, speed
* Time.deltaTime);
                }
                else
                {
                    transform.Translate(Vector3.forward * speed * Time.deltaTime);

                }
            }
            if(non_moving == true) //click on enemy magic
            {
                if (vfx_target_save != null)
                {
                    transform.position = vfx_target_save.transform.position;
                }
                else
                {
                    Destroy(vfx_object_container);
                }
            }
            if(support_spell_follow_player == true)
            {
                transform.position = player.transform.position;
                duration_of_life = 100;
                if(SaveScript.mana <= 0.02)
                {
                    Destroy(vfx_object_container);
                }
            }

            if (heal_magic == true)
            {
                SaveScript.health += SaveScript.health_regeneration_skill * Time.deltaTime;
            }

            SaveScript.mana -= spell_mana_cost * Time.deltaTime;

            Destroy(vfx_object_container, duration_of_life);
        }

        private void OnTriggerEnter(Collider other)
        {
            if (other.CompareTag("enemy") && other.transform.gameObject != object_triggered)
            {
                Enemy_Type enemy_type = other.GetComponent<Enemy_Type>();
                if (SaveScript.class_Mage == true)
                {
                    damage = damage * 6 / 5;
                }
                if(enemy_type.enemyType == Enemy_Type.EnemyType.Golem)
                {
                    other.transform.gameObject.GetComponent<Golem_Movement>().full_HP -= (damage * 4) / 5;  // 20 peer
cent magic decrease
                    object_triggered = other.transform.gameObject;
                }
                else
                {
                    other.transform.gameObject.GetComponent<EnemyMovement>().full_HP -= damage;
                    object_triggered = other.transform.gameObject;
                }
            }

        }
    }
```

1.6 Реалізація функціоналу взаємодії з ігровими об’єктами.

1.6.1 Файл ItemPickUp.cs

```
public class ItemPickUp : MonoBehaviour
{
    private bool can_pick_up = true;
    private WaitForSeconds pickUp_Pause = new WaitForSeconds(0.0001f);

    public int number_of_pickedUp_items;

    public bool is_redMushroom = false;
    public bool is_blueFlower = false;
    public bool is_whiteFlower = false;
    public bool is_purpleFlower = false;
    public bool is_redFlower = false;

    public bool is_roots = false;
    public bool is_leaf = false;
    public bool is_keySimp = false;
    public bool is_keyGold = false;
    public bool is_monsterEye = false;

    public bool is_bluePotion = false;
    public bool is_greenPotion = false;
    public bool is_lazurePotion = false;
    public bool is_redPotion = false;

    public bool is_bread = false;
    public bool is_cheese = false;
    public bool is_meat = false;

    public bool is_purpleMushroom = false;
    public bool is_orangeMushroom = false;

    public bool is_loot_coin = false;

    public static bool is_keySimp_exist = false;
    public static bool is_keyGold_exist = false;

    public GameObject Inventory_Canvas;
    public AudioSource audio_Player;


    private void Start()
    {
        Inventory_Canvas = GameObject.Find("Inventory");
        audio_Player = Inventory_Canvas.GetComponent<AudioSource>();
        if(is_loot_coin == true) // only 10 sec to pick up loot coins from enemy
        {
            Destroy(gameObject, 10);
        }
    }




    private void OnTriggerEnter(Collider other)
    {

        if (other.CompareTag("Player") && can_pick_up == true)
        {
            can_pick_up = false;

            audio_Player.clip = Inventory_Canvas.GetComponent<Inventory>().pick_UP_SFX;
            audio_Player.Play();
            if (is_redMushroom == true)
            {
                if (Inventory.amount_of_redMushrooms == 0)
                {
                    DisplayIcons();
                }
                Inventory.amount_of_redMushrooms++;
                Destroy(gameObject);
            }
            else if (is_blueFlower == true)
            {
                if (Inventory.amount_of_blueFlowers == 0)
                {
                    DisplayIcons();
                }
                Inventory.amount_of_blueFlowers++;
                Destroy(gameObject);
            }
            else if (is_whiteFlower == true)
            {
                if (Inventory.amount_of_whiteFlowers == 0)
                {
                    DisplayIcons();
                }
                Inventory.amount_of_whiteFlowers++;
                Destroy(gameObject);
            }
            else if (is_purpleFlower == true)
```

```
{
    if (Inventory.amount_of_purpleFlowers == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_purpleFlowers++;
    Destroy(gameObject);
}
else if (is_redFlower == true)
{
    if (Inventory.amount_of_redFlowers == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_redFlowers++;
    Destroy(gameObject);
}

else if (is_roots == true)
{
    if (Inventory.amount_of_roots == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_roots++;
    Destroy(gameObject);
}
else if (is_leaf == true)
{
    if (Inventory.amount_of_leaf == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_leaf++;
    Destroy(gameObject);
}
else if (is_keySimp == true)
{
    if (Inventory.amount_of_keySimp == 0 && is_keySimp_exist == false)
    {
        DisplayIcons();
        is_keySimp_exist = true;
    }
    Inventory.amount_of_keySimp++;
    Inventory.player_has_a_common_key = true;
    Destroy(gameObject);
}
```

```
else if (is_keyGold == true)
{
    if (Inventory.amount_of_keyGold == 0 && is_keyGold_exist == false)
    {
        DisplayIcons();
        is_keyGold_exist = true;
    }
    Inventory.amount_of_keyGold++;
    Inventory.player_has_a_gold_key = true;
    Destroy(gameObject);
}
else if (is_monsterEye == true)
{
    if (Inventory.amount_of_monsterEye == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_monsterEye++;
    Destroy(gameObject);
}

else if (is_bluePotion == true)
{
    if (Inventory.amount_of_bluePotion == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_bluePotion++;
    Destroy(gameObject);
}
else if (is_greenPotion == true)
{
    if (Inventory.amount_of_greenPotion == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_greenPotion++;
    Destroy(gameObject);
}
else if (is_lazurePotion == true)
{
    if (Inventory.amount_of_lazurePotion == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_lazurePotion++;
    Destroy(gameObject);
```

```
}
else if (is_redPotion == true)
{
    if (Inventory.amount_of_redPotion == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_redPotion++;
    Destroy(gameObject);
}

else if (is_bread == true)
{
    if (Inventory.amount_of_bread == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_bread++;
    Destroy(gameObject);
}
else if (is_cheese == true)
{
    if (Inventory.amount_of_cheese == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_cheese++;
    Destroy(gameObject);
}
else if (is_meat == true)
{
    if (Inventory.amount_of_meat == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_meat++;
    Destroy(gameObject);
}

else if (is_purpleMushroom == true)
{
    if (Inventory.amount_of_purpleMushroom == 0)
    {
        DisplayIcons();
    }
    Inventory.amount_of_purpleMushroom++;
    Destroy(gameObject);
```

```
            }
            else if (is_orangeMushroom == true)
            {
                if (Inventory.amount_of_orangeMushroom == 0)
                {
                    DisplayIcons();
                }
                Inventory.amount_of_orangeMushroom++;
                Destroy(gameObject);
            }
            else if (is_loot_coin == true)
            {
                Inventory.gold += Random.Range(10, 50);
                Destroy(gameObject);
            }

            else
            {
                DisplayIcons();
                Destroy(gameObject);
            }

            // Destroy(gameObject);
            StartCoroutine(Reset_PickUp());
        }
    }

    void DisplayIcons()
    {
        Inventory.newIcon = number_of_pickedUp_items;
        Inventory.iconUpdated = true;
    }



    public static void DestroyIcon()
    {
        Inventory.newIcon = 0;
        Inventory.iconUpdated = true;
    }

    IEnumerator Reset_PickUp()
    {
        yield return pickUp_Pause;
        can_pick_up = true;
    }
}
```