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: | Виконавець: |
| \_\_\_\_\_\_\_\_\_\_\_\_Максим ГОЛОВЧЕНКО | \_\_\_\_\_\_\_\_\_\_\_Максим ТЕРЕШКОВИЧ |

Київ – 2024

**Посилання на репозиторій з повним текстом програмного коду:**

<https://github.com/mcmcmax437/3D-RPG-Diploma-Thesis>

## Реалізація інтелекту ворогів зі своїми особливими спектрами поведінки.

### Файл EnemyMovement.сs:

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);

}

}

### 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;

}

}

}

### 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;

}

}

## Реалізація ігрового інтерфейсу.

### 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--;

}

}

}

### 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);

}

}

}

## Реалізація механік бою.

### 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<CinemachineTransposer>();

cinemachine\_orbital\_Transposer = camera\_2\_free.gameObject.GetComponent<CinemachineVirtualCamera>().GetCinemachineComponent<CinemachineOrbitalTransposer>();

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);

}

}

}

}

### 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;

}

}

## Реалізація функціоналу інвентарю, магазинів та ігрової економіки.

### 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";

}

}

}

### 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";

}

}

## Реалізація функціонали створення та використання магії.

### 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);

}

}

### 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;

}

}

}

}

### 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;

}

}

}

}

## Реалізація функціоналу взаємодії з ігровими об’єктами.

### 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;

}

}