

# Dokumentacja projektu Gry w UNITY i C#

Kamil Kryus, grupa

12 stycznia 2019

# Część I

## Opis programu

W ramach przedmiotu został stworzony program według następującego polecenia (umieszczonego na platformie polsl):

Projekt należy zaproponować na laboratorium, a następnie wrzucić na platformę wersję do oceny (pełna solucja, wersja skompilowana, dokumentacja). Ocena będzie wystawiona w trakcie lab w styczniu.

Gra 3D powinna zawierać:

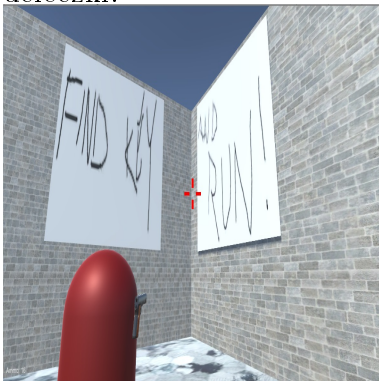
- a) menu
- b) linię fabularną
- c) obsługę ruchu myszy/klawiatury
- d) efekty dźwiękowe
- e) muzykę
- f) przeciwny itd.

## Linia fabularna

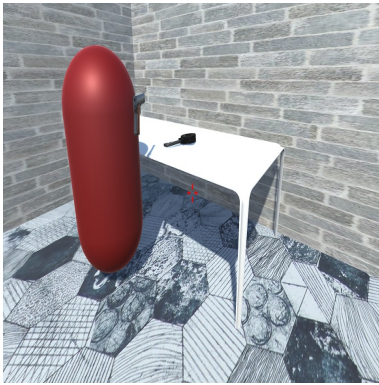
Budzimy się w środku pewnego pomieszczenia posiadając broń w ręku. Rozglądając się jesteśmy w stanie dostrzec plakaty, które budzą w nas niepokój.



Zastanawiając się jak możemy uciec stąd, zauważamy plakaty informujące nas o sposobie ucieczki.



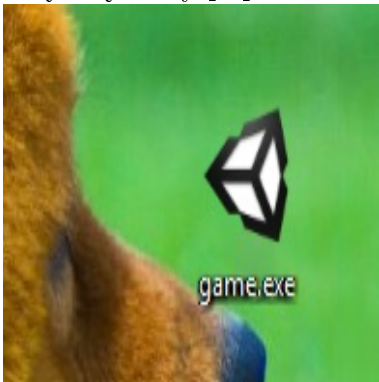
Wychodzimy więc przez pobliską bramę i modlimy się, by zauważony stwór nas nie zauważył. Przechodząc z pokoju do pokoju i zabijając utrudniające nam ucieczkę zombie, natrafiamy w końcu na pokój, w którym znajduje się klucz.



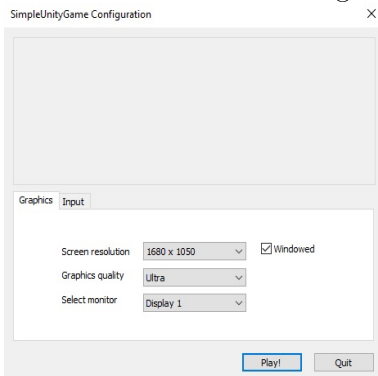
Zabieramy go ze sobą i idziemy dalej. Docieramy aż do bramy, którą można otworzyć za pomocą tego klucza. Po wyjściu na zewnątrz okazuje się, że otacza nas pustka... po czym się budzimy.

## Instrukcja obsługi

Grę włączamy poprzez dwukrotne przyciśnięcie lewym przyciskiem myszy na ikonce gry:



Po ustawieniu ustawień graficznych gry, przyciskamy przycisk Play!



Po załadowaniu gry, aby rozpocząć grę, należy nacisnąć lewym przyciskiem myszy na przycisku Start:



Od tej chwili możemy grać. W każdej chwili jesteśmy w stanie zakończyć grę poprzez wciśnięcie odpowiedniego przycisku (opisanego w sekcji Sterowanie).

## **Sterowanie**

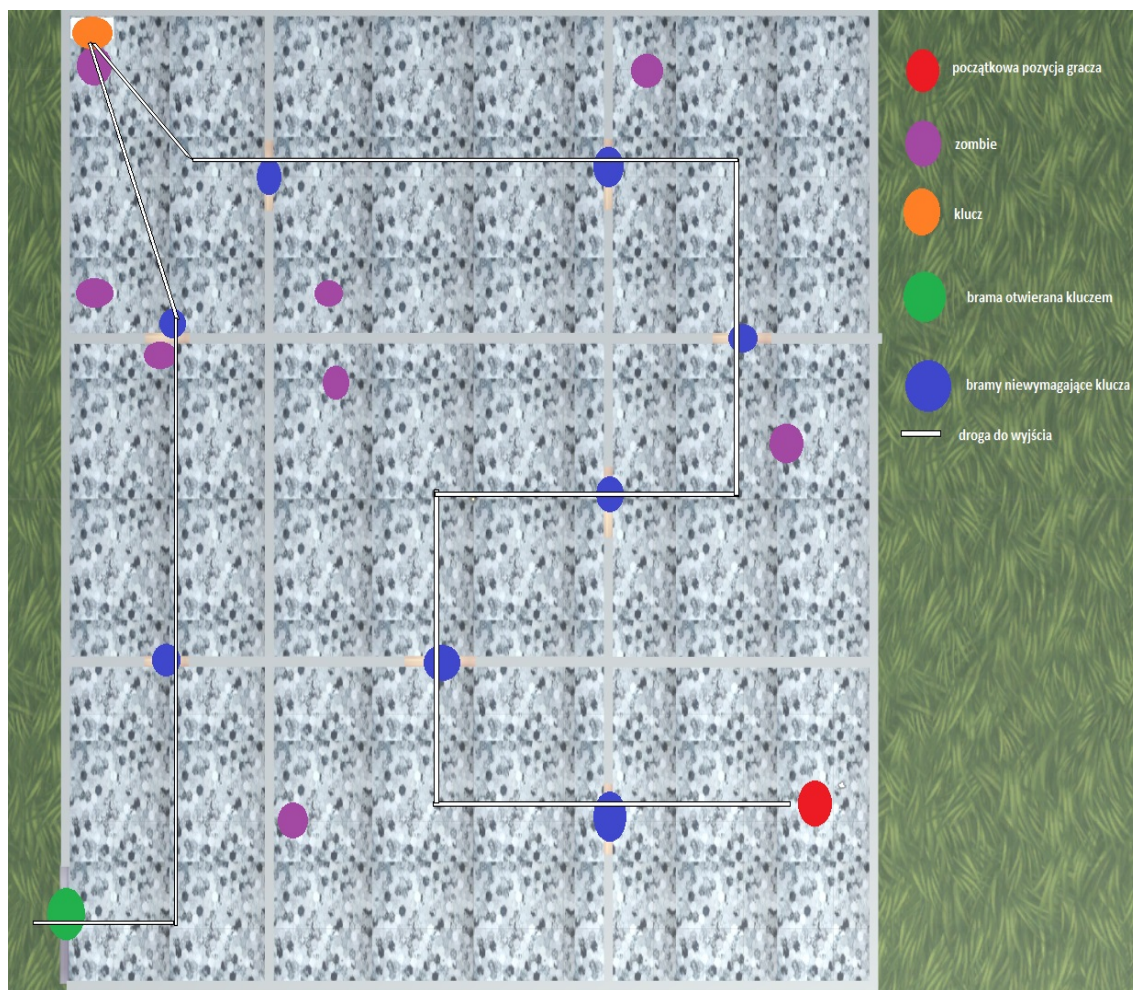
Oto lista przycisków oraz opis ich działania:

- Esc - wyjście z gry,
- Lewy Przycisk Myszy/Ctrl - strzał
- w - ruch do przodu
- s - ruch do tyłu
- a - ruch w lewo
- d - ruch w prawo
- spacja - skok
- e - wykonanie akcji (np. otwarcie bramy, wzięcie klucza).

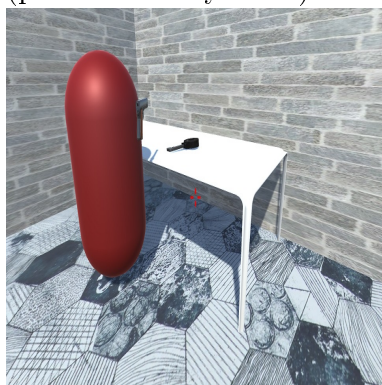
Ponadto w grze jesteśmy w stanie rozglądać się za pomocą ruchu myszy.

## **Solucja do przejścia gry**

Najprostszą drogę prowadzącą od początkowej pozycji gracza do wyjścia przedstawia następujący obrazek wraz z legendą:



Po drodze należy zabijać (lub uciekać) przeciwników za pomocą broni i otwierać bramy klawiszem akcji. Ostatnią bramę otworzymy (na obrazku zaznaczona kolorem zielonym) tylko posiadając przy sobie klucz, który możemy zdobyć (przyciskiem akcji) w trakcie eksploracji (pomarańczowy kolor):



Aby ukończyć grę, należy wyjść z budynku, a następnie odczekać 4 sekundy.

## Część II

### Część techniczna

#### Otwieranie bram

Do każdej bramy podpięty jest skrypt, który wykrywa sytuację, iż obiekt gracza znalazł się blisko i wcisnął przycisk akcji. Gdy zostaną spełnione te warunki, rozpoczyna się proces w pętli obniżania pozycji bramy w osi Y, aż osiągnie pewien punkt, w którym nie jest już widoczna dla gracza.

#### Otwieranie bramy za pomocą klucza

Skrypt ten działa w bardzo podobny sposób do zwykłego otwierania bramy. Mechanizm ten dodatkowo sprawdza czy klucz został zebrany przez gracza - jeżeli gracz nie wykonał tej akcji, brama nie obniży się.

#### Zebranie klucza

Ten mechanizm również działa w bardzo podobny sposób do otwierania się bram. Gdy obiekt klucza wykryje bliskość gracza i wciśnięcie klawisza akcji, modyfikuje zmienną odpowiedzialną za stan zebrania klucza i usuwa się, dzięki czemu nie widać klucza w grze po jego zebraniu.

#### Strzelanie i ranienie przeciwników

Wciśnięcie odpowiedniego przycisku powoduje szereg animacji symulujących strzał z pistoletu. Ranienie przeciwników działa na zasadzie raycastingu - przeprowadzana jest prosta pomiędzy środkiem celownika, a napotkanym obiektem. Jeżeli napotkanym obiektem jest nasz przeciwnik, to system wywołuje podpiętą metodę informując tym samym komponent o zranieniu go. Jeżeli poziom życia przeciwnika osiągnie zero, funkcje związane z poruszaniem się, atakowaniem bohatera itp. ustają.

#### Zachowanie przeciwników

Projekt posiada prostą sztuczną inteligencję przeciwników. Jeżeli gracz znajdzie się w pewnej odległości od przeciwnika, to ten zaczyna poruszać się i obracać w stronę gracza. Przeciwnik może również zacząć ścigać gracza w przypadku otrzymania obrażeń. Przeciwnik jednak musi posiadać przynajmniej 1 punkt życia, aby te funkcje działały.

#### Animacje przeciwników

Animacjami przeciwników zarządza kontroler, który w zależności od parametrów, wykonuje konkretną animację. Parametry natomiast są zmieniane za pomocą skryptów w zależności od wystąpienia pewnych zdarzeń.

## Opis działania

### Ściganie gracza

Klasa Vector3 posiada metodę pozwalającą nam obliczyć odległość pomiędzy obiektami nazwaną Distance(). Jeżeli dystans ten jest większy niż 1.72 jednostki i jednocześnie mniejszy niż 30 jednostek, przeciwnik zaczyna ścigać gracza. Jeżeli przeciwnik jest bliżej niż 2 jednostki od gracza, rozpoczyna atakować bohatera odejmując mu 1 punkt życia na klatkę, zmieniając jest również wtedy animacja.

### Otwieranie bram

Jeżeli dystans (obliczany za pomocą metody Distance() z klasy Vector3) pomiędzy graczem, a bramą jest mniejszy niż 5 jednostek, brama rozpoczyna proces otwierania się. W każdej klatce pozycja bramy na osi Y jest zmniejszana przez iloczyn czasu (w sekundach) od ostatniej klatki i własnej liczby (równej 2.5) manipulującej szybkością animacji. Jeżeli pozycja bramy na osi Y osiągnie pewną liczbę, która jest wystarczająca, aby nie była widoczna dla gracza, proces się kończy.

### Obliczanie obrażeń

Zarówno przeciwnik, jak i gracz zadają obrażenia o wartości 1. Każdy przeciwnik ma 10 punktów życia (a tym samym potrzeba 10 strzałów by zabić go), natomiast gracz posiada aż 100 punktów życia. Jednak przeciwnicy atakują co klatkę i w przypadku ataku, gracz może bardzo szybko zginąć.

## Implementacja

### Sztuczna inteligencja

Zachowanie (sprawdzone w każdej klatce) przeciwników zostało zaimplementowane w następujący sposób:

```
if ((Vector3.Distance(playerTrans.position, this.transform.position) < 30f &&
    life > 0) || (life < 20 && life > 0) &&
    Vector3.Distance(playerTrans.position, this.transform.position) > 1.72f)
{
    float step = 2.0f * Time.deltaTime;
    Vector3 targetDir = playerTrans.position - transform.position;
    targetDir.y = 0.3f;

    Vector3 newDir = Vector3.RotateTowards(transform.forward, targetDir, step,

    transform.rotation = Quaternion.LookRotation(newDir);
    var tmpPos = playerTrans.position;
    tmpPos.y = 0.2f;

    this.transform.position = Vector3.MoveTowards(this.transform.position, tmpP
```

```

        transform.position = new Vector3(transform.position.x, 0.2f, transform.position.z);
    }
    if (Vector3.Distance(playerTrans.position, this.transform.position) < 2f &&
        life > 0)
    {
        thisAnim.SetBool("isAttacking", true);
        Player.life -= 1;
        if (Player.life <= 0)
        {
            SceneManager.LoadScene(2);
        }
    }
    else
    {
        thisAnim.SetBool("isAttacking", false);
    }
}

```

## Raycasting

System strzelania został porany ze sklepu i jedynie przystosowany do potrzeb projektu. W momencie strzału skrypt informuje odpowiedni obiekt o wydarzeniu poprzez wywołanie metody `ChangeHealth()`.

```

Ray ray = new Ray(raycastStartSpot.position, direction);
RaycastHit hit;

if (Physics.Raycast(ray, out hit, range))
{
    // Warmup heat
    float damage = power;
    if (warmup)
    {
        damage *= heat * powerMultiplier;
        heat = 0.0f;
    }
    // Damage
    hit.collider.gameObject.SendMessageUpwards("ChangeHealth", -damage, SendMessageOptions.DontRequireReceiver);

    if (bloodyMessEnabled)
    {
        //call the ApplyDamage() function on the enemy CharacterSetup script
        if (hit.collider.gameObject.layer == LayerMask.NameToLayer("Limb"))
        {
            Vector3 directionShot = hit.collider.transform.position - transform.position;
        }
    }
}

```



```
}
```

```
}
```

Powyższy fragment kodu wywołuje metodę powiązaną z samym przeciwnikiem. Wartość dmg wynosi zawsze -1.

```
public void ChangeHealth(int dmg)
{
    life += dmg;
    if (life <= 0)
    {
        thisAnim.SetBool("isDead", true);
    }
}
```

## Część III

### 0.1 Blender

Tworząc projekt nie korzystałem z oprogramowania Blender.

### 0.2 UNITY

#### Pobrane ze sklepu

##### Zombie

Pobraný asset udostępniał gotowy prefabrykat, animacje, kontroler animacji oraz skrypty. Dla potrzeb projektu, skorzystałem jedynie z niektórych animacji oraz samego modelu zombie.

##### Bronie i system strzelania

Udostępnione materiały zawierały bardzo dużo funkcjonalności, jednak do finalnego projektu zostało użyte tylko kilka. Model (jednej) broni, skrypty związane ze strzelaniem i raycastingiem oraz animacje.

##### Menu

Chociaż pobrane ze sklepu menu zapewniało pełną funkcjonalność, skorzystałem głównie z pojawienia się przycisków w widocznym miejscu kamery oraz skryptów powiązanych z reakcjami na kliknięcie przycisków.

##### Model klucza i stołu

Dla potrzeb projektu skorzystałem z modelu klucza i stołu. Funkcjonalność oraz inne modele nie zostały wykorzystane.

#### Własne

W czasie pracy nad projektem następujące elementy zostały stworzone przeze mnie:

- Linia fabularna
- Otwieranie bram
- Sztuczna inteligencja przeciwników
- Budowla

## Pełen kod programu

- BackToMenu.cs

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

```

public class BackToMenu : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Cursor.lockState = CursorLockMode.None;
            SceneManager.LoadScene(0);
        }
    }
}

```

- CollectKey.cs

```

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

public class CollectKey : MonoBehaviour {

    public GameObject key;
    private Transform playerTrans = null;
    void Start()
    {
        playerTrans = GameObject.Find("Player").transform;
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            if (Vector3.Distance(playerTrans.position, this.transform.position) < 1)
            {
                Player.hasKey = true;
                Destroy(key);
            }
        }
    }
}

```

- FinishGame.cs

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

public class FinishGame : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }

    private void OnTriggerEnter(Collider other)
    {
        StartCoroutine(Wait(4.0f));
    }

    IEnumerator Wait(float seconds)
    {
        yield return new WaitForSeconds(seconds);
        SceneManager.LoadScene(3);
    }
}
```

- MouseLook.cs

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

public class MouseLook : MonoBehaviour {

    // Use this for initialization
    Vector2 _mouseAbsolute;
    Vector2 _smoothMouse;

    public Vector2 clampInDegrees = new Vector2(360, 180);
```

```

public bool lockCursor;
public Vector2 sensitivity = new Vector2(2, 2);
public Vector2 smoothing = new Vector2(3, 3);
public Vector2 targetDirection;
public Vector2 targetCharacterDirection;

// Assign this if there's a parent object controlling motion, such as a Character
// Yaw rotation will affect this object instead of the camera if set.
public GameObject characterBody;

void Start()
{
    // Set target direction to the camera's initial orientation.
    targetDirection = transform.localRotation.eulerAngles;

    // Set target direction for the character body to its initial state.
    if (characterBody)
        targetCharacterDirection = characterBody.transform.localRotation.eulerAngles;
}

void Update()
{
    // Ensure the cursor is always locked when set
    if (lockCursor)
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Allow the script to clamp based on a desired target value.
    var targetOrientation = Quaternion.Euler(targetDirection);
    var targetCharacterOrientation = Quaternion.Euler(targetCharacterDirection);

    // Get raw mouse input for a cleaner reading on more sensitive mice.
    var mouseDelta = new Vector2(Input.GetAxisRaw("Mouse X"), Input.GetAxisRaw("Mouse Y"));

    // Scale input against the sensitivity setting and multiply that against the current movement.
    mouseDelta = Vector2.Scale(mouseDelta, new Vector2(sensitivity.x * smoothing.x, sensitivity.y * smoothing.y));

    // Interpolate mouse movement over time to apply smoothing delta.
    _smoothMouse.x = Mathf.Lerp(_smoothMouse.x, mouseDelta.x, 1f / smoothing.x);
    _smoothMouse.y = Mathf.Lerp(_smoothMouse.y, mouseDelta.y, 1f / smoothing.y);

    // Find the absolute mouse movement value from point zero.
    _mouseAbsolute += _smoothMouse;
}

```

```

// Clamp and apply the local x value first, so as not to be affected by world rotation
if (clampInDegrees.x < 360)
    _mouseAbsolute.x = Mathf.Clamp(_mouseAbsolute.x, -clampInDegrees.x * 0.5f, clampInDegrees.x * 0.5f);

// Then clamp and apply the global y value.
if (clampInDegrees.y < 360)
    _mouseAbsolute.y = Mathf.Clamp(_mouseAbsolute.y, -clampInDegrees.y * 0.5f, clampInDegrees.y * 0.5f);

transform.localRotation = Quaternion.AngleAxis(-_mouseAbsolute.y, targetCharacterOri);

// If there's a character body that acts as a parent to the camera
if (characterBody)
{
    var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x, Vector3.up);
    characterBody.transform.localRotation = yRotation * targetCharacterOri.localRotation;
}
else
{
    var yRotation = Quaternion.AngleAxis(_mouseAbsolute.x, transform.InverseTransformDirection(Vector3.up));
    transform.localRotation *= yRotation;
}
}
}

```

- Movement.cs

```

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

public class Movement : MonoBehaviour {

    //Variables
    public float speed = 6.0f;
    public float jumpSpeed = 8.0f;
    public float gravity = 20.0f;
    private Vector3 moveDirection = Vector3.zero;

    void Update()
    {
        CharacterController controller = GetComponent<CharacterController>();
        // is the controller on the ground?
        if (controller.isGrounded)
        {
            //Feed moveDirection with input.

```

```

        moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
        moveDirection = transform.TransformDirection(moveDirection);
        //Multiply it by speed.
        moveDirection *= speed;
        //Jumping
        if (Input.GetButton("Jump"))
            moveDirection.y = jumpSpeed;
    }
    //Applying gravity to the controller
    moveDirection.y -= gravity * Time.deltaTime;
    //Making the character move
    controller.Move(moveDirection * Time.deltaTime);
}
}

```

- OpenGate.cs

```

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

public class OpenGate : MonoBehaviour
{
    private float animationSpeed = 2.5f;
    public GameObject player;
    private Transform playerTransform;

    public AudioSource audioSource;
    public AudioClip gateOpeningClip;

    void Start()
    {
        playerTransform = player.transform;
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            if (Vector3.Distance(playerTransform.position, this.transform.position) < 10)
            {
                StartCoroutine(MoveGate());
            }
        }
    }
}

```

```

public IEnumerator MoveGate()
{
    AudioSource.PlayOneShot(gateOpeningClip);
    while (transform.localPosition.y > 16.35)
    {
        transform.localPosition = new Vector3(transform.localPosition.x, transform.localPosition.y - 0.1f, transform.localPosition.z);
        yield return null;
    }
}
}

```

- OpenGateWithkey.cs

```

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

public class OpenGateWithKey : MonoBehaviour
{
    private float animationSpeed = 2.5f;
    public GameObject player;
    private Transform playerTransform;
    public AudioSource audioSource;
    public AudioClip gateOpeningClip;

    void Start()
    {
        playerTransform = player.transform;
    }
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            if (Vector3.Distance(playerTransform.position, this.transform.position) < 5)
            {
                StartCoroutine(MoveGate());
            }
        }
    }

    public IEnumerator MoveGate()
    {

```



```

        audioSource.PlayOneShot(gateOpeningClip);
        while (transform.localPosition.y > -7.37)
        {
            transform.localPosition = new Vector3(transform.localPosition.x, transform.localPosition.y, transform.localPosition.z);
            yield return null;
        }
    }
}

```

- Player.cs

```

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

public class Player : MonoBehaviour {

    public static bool hasKey;
    public static int life = 100;

    // Use this for initialization
    void Start () {

    }

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

    }
}

```

- ZombieBehaviour.cs

```

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

public class ZombieBehaviour : MonoBehaviour
{
    public int life = 10;
    private Animator thisAnim;
}

```

```

private Transform playerTrans = null;
void Start()
{
    playerTrans = GameObject.Find("Player").transform;
    thisAnim = GetComponent<Animator>();
}
void Update()
{
    if ((Vector3.Distance(playerTrans.position, this.transform.position) < 30f)
    {
        float step = 2.0f * Time.deltaTime;
        Vector3 targetDir = playerTrans.position - transform.position;
        targetDir.y = 0.3f;

        Vector3 newDir = Vector3.RotateTowards(transform.forward, targetDir, step, 0.0f);

        transform.rotation = Quaternion.LookRotation(newDir);
        var tmpPos = playerTrans.position;
        tmpPos.y = 0.2f;

        this.transform.position = Vector3.MoveTowards(this.transform.position,
        transform.position = new Vector3(transform.position.x, 0.2f, transform.position.z);
    }
    if (Vector3.Distance(playerTrans.position, this.transform.position) < 2f && !thisAnim.GetBool("isAttacking"))
    {
        thisAnim.SetBool("isAttacking", true);
        Player.life -= 1;
        if (Player.life <= 0)
        {
            SceneManager.LoadScene(2);
        }
    }
    else
    {
        thisAnim.SetBool("isAttacking", false);
    }
}

public void ChangeHealth(int dmg)
{
    life += dmg;
    if (life <= 0)
    {
        thisAnim.SetBool("isDead", true);
    }
}

```

} }