Code Parts

Core

AI Move

```
using System.Diagnostics.Contracts;
using System.Security.Cryptography.X509Certificates;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
public class AiMove : MonoBehaviour
{
    public float m_MoveSpeed;
    public NavMeshAgent m_Agent;

    public Animator m_Animator;

    public List<Transform> m_Paths = new List<Transform>();

    public Transform m_PathRoot;

    public int m_PathIndex;

    void Start()
    {
        m_Agent = GetComponent<NavMeshAgent>();
        m_Animator = GetComponent<Animator>();
        m_Paths.Clear();
        for(int i=0;i<m_PathRoot.childCount;i++)
        {
            m_Paths.Add(m_PathRoot.GetChild(i));
        }
        m_PathIndex = 0;
        m_Agent.SetDestination(m_Paths[m_PathIndex].position);
        transform.position = m_Paths[m_PathIndex].position;
        m_Animator.SetBool("Run",true);
        m_Agent.speed = m_MoveSpeed;

    }

    //Character died
    public void Die()
    {
        m_Animator.SetBool("Die",true);
        m_Agent.isStopped = true;
        m_Agent.enabled = false;
        GetComponent<Collider>().enabled = false;
        StartCoroutine("DieAffter");
    }

    private float m_DieTime;
    IEnumerator DieAffter()
    {
        yield return new WaitForSeconds(2);
        while(true)
        {
            yield return 0;
            m_DieTime += Time.deltaTime;
            if(m_DieTime>=3)
                    break;
            transform.Translate(-Vector3.up * Time.deltaTime);
        }

        StopCoroutine("DieAffter");
    }

    void Update()
    {
```

```
       // Debug.Log("distance:" +Vector3.Distance(transform.position, m_Paths[m_PathIndex].position) );

        if(Vector3.Distance(transform.position, m_Paths[m_PathIndex].position)<= m_Agent.stoppingDistance)
        {
            m_PathIndex ++;
            if(m_PathIndex>=m_Paths.Count)
            m_PathIndex = m_PathIndex =  0;
             m_Agent.SetDestination(m_Paths[m_PathIndex].position);

        }
    }

    private void OnTriggerEnter(Collider other) {
        if(other.CompareTag("Player"))
        {
            if(other.GetComponent<PlayerAction>().m_IsGod)
                return;
            //death
            other.GetComponent<PlayerAction>().PlayerDie();
        }
    }
}
```

Camera

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

public class CamerFollow : MonoBehaviour
{
    public Vector3 m_Camera;

    public Transform target;

    public float targetHeight = 1.8f;

    public float targetSide = 0.1f;

    public float distance = 4;

    public float maxDistance = 8;

    public float minDistance = 2.2f;

    public float xSpeed = 250;

    public float ySpeed = 125;

    public float yMinLimit = -10;

    public float yMaxLimit = 72;

    public float zoomRate = 80;

    private float x = 20;

    private float y = 0;

    void Update()
    {
        if(target!=null)
        {
            x = m_Camera.x;
            y = m_Camera.y;
            y = clampAngle(y,yMinLimit,yMaxLimit);
            Quaternion rotation = Quaternion.Euler(y,x,0);
            transform.rotation = rotation;

            distance -= (m_Camera.z * Time.deltaTime) * zoomRate * Mathf.Abs(distance);
            distance = Mathf.Clamp(distance,minDistance,maxDistance);
            transform.position = target.position + new Vector3(0,targetHeight,0) + rotation * (new Vector3(targetSide,0,-1) * distance);
```

```
        }
    }

    float clampAngle(float angle,float min, float max)
    {
        if(angle < -360)
        {
            angle += 360;
        }

        if(angle > 360)
        {
            angle -= 360;
        }

        return Mathf.Clamp(angle,min,max);
    }
}
```

Player Action

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

//character control
public class PlayerAction : MonoBehaviour
{
    private Animator m_Animator;

    //render
    public Renderer m_Render;

    //Character died or not
    public bool m_IsDie;

    //character invincible
    public bool m_IsGod;

    //weapon resource
    public GameObject m_WeaponRes;
    //attack position
    public Transform m_AttactPos;

    //shopping page UI
    public TipFramePanel m_TipFramePanel;

    public GameOverPanel m_GameOverPanel;

    public ShopPanel m_ShopPanel;

    private void Start() {
        m_Animator = GetComponent<Animator>();
    }

    //Character died
    public void PlayerDie()
    {
        m_Animator.SetBool("Die", true);
        m_IsDie = true;
        GetComponent<Rigidbody>().velocity = Vector3.zero;
        GetComponent<PlayerMove>().enabled = false;
        //UI for play again
        Invoke("ReStartGame", 3.0f);
    }

    //restart
    public void ReStartGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
```

```
//player replay
public void PlayerRevive()
{
    m_IsDie = false;
    m_IsGod = true;
    GetComponent<PlayerMove>().enabled = true;
    m_Animator.SetBool("Die",false);
    //character invincible state
    Invoke("NoGod",3);
}

public void NoGod()
{
    m_IsGod = false;
}

//attack
public void AttackToTarget()
{
    if(m_IsDie)
        return;
    m_Animator.SetTrigger("Attack");
}

//attack animation event
public void Event_Attack()
{
    GameObject.Instantiate(m_WeaponRes,m_AttactPos.position,m_AttactPos.rotation);
}

//clean all DEBUFF
public void ClearAllDeBuffer()
{
    GetComponent<BuffMgr>().ClearAllDeBuffer();
}

//speed up
public void AddSpeedBuff()
{
    //If the player is in the deceleration DEBUFF, it cannot be used.
    if(GetComponent<BuffMgr>().m_SlowSpeedDeBuff.gameObject.activeSelf)
        return;
    GetComponent<BuffMgr>().m_AddSpeedBuff.Init(0.3f,5);
}

//frozen
public void Freezi()
{
    for(int i=0;i<m_Render.materials.Length;i++)
    {
         m_Render.materials[i].color = Color.blue;
    }

    GetComponent<Rigidbody>().velocity = Vector3.zero;
    GetComponent<PlayerMove>().enabled =false;
    m_Animator.SetBool("Run",false);
    m_Animator.speed = 0;
}

//clean frozen
public void UFreezi()
{
     for(int i=0;i<m_Render.materials.Length;i++)
    {
         m_Render.materials[i].color = Color.white;
    }
    GetComponent<PlayerMove>().enabled =true;
    m_Animator.speed = 1;
}

private void Update() {
    //test
    if(Input.GetKeyDown(KeyCode.P))
    {
        PlayerRevive();
```

```
        }
        if(Input.GetKeyDown(KeyCode.F))
        {
            AttackToTarget();
        }
    }
}
```

Player Move

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

//Character move
public class PlayerMove : MonoBehaviour
{
    //speed
    public float speed;
    //animition speed
    [HideInInspector]
    public float m_ASpeed = 1;
    private Animator m_Animator;
    private Transform cam;
    private float turnSmoothVelocity;
    private float horizontal;
    private float vertical;
    private float turnSmoothTime = 0.1f;
    private Rigidbody rb;
    public FixedJoystick m_Joystick;

    void Start()
    {
        cam = Camera.main.transform;
        rb = GetComponent<Rigidbody>();
        m_Animator = GetComponent<Animator>();
        m_ASpeed = 1;
    }

    private void FixedUpdate()
    {
        if(m_Joystick==null)
            return;

        //horizontal = m_Joystick.Horizontal;
        //vertical =  m_Joystick.Vertical;

        horizontal = Input.GetAxisRaw("Horizontal");
        vertical = Input.GetAxisRaw("Vertical");

        Vector3 dir = new Vector3(horizontal, 0f, vertical).normalized;

        if (dir.magnitude >= 0.1f)
        {
            float targetAngle = Mathf.Atan2(dir.x, dir.z) * Mathf.Rad2Deg + cam.eulerAngles.y;

            float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref turnSmoothVelocity, turnSmoothTime);

            transform.rotation = Quaternion.Euler(0f, angle, 0f);

            Vector3 moveDir = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;

            rb.velocity = rb.velocity.y * Vector3.up + moveDir * speed ;
        }
        else
        {
            rb.velocity = new Vector3(0, rb.velocity.y, 0);
        }

        if (horizontal != 0 || vertical != 0)
            m_Animator.SetBool("Run", true);
        else
            m_Animator.SetBool("Run", false);
```

```
    }

    private void LateUpdate()
    {
        transform.position = rb.transform.position;
    }

    private void Update() {
          //Here control the attack speed. The attack speed is controlled by the animation speed.
          AnimatorStateInfo state = GetComponent<Animator>().GetCurrentAnimatorStateInfo(0);
          if(state.IsName("Player_Run"))
          {
               GetComponent<Animator>().speed = m_ASpeed;
          }
          else
          {
             GetComponent<Animator>().speed = 1;
          }
    }
}
```

Player use items

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

//player use items
public class PlayerUseItem : MonoBehaviour
{
    public PlayerAction m_PlayerAction;

    //three items
    public int m_ShoeItemNums;
    public int m_YaoShuiNums;
    public int m_WeaponNums;
    //UI
    public Button m_UseWeaponBtn;
    public Button m_UseShoeBtn;
    public Button m_UseYaoShuiBtn;

    public Text m_ShoeItemNumsText;
    public Text m_YaoShuiNumsText;
    public Text m_WeaponNumsText;

    //UI panel
    public TipFramePanel m_TipFranmePanel;


    private void Start() {
      //  LoadData();
        m_PlayerAction = GameObject.FindWithTag("Player").GetComponent<PlayerAction>();
        m_UseWeaponBtn.onClick.AddListener(UseWeapon);
        m_UseShoeBtn.onClick.AddListener(UseShoe);
        m_UseYaoShuiBtn.onClick.AddListener(UseYaoShui);
        UpdataUi();
    }
    //use item-shoes to speed up
    public void UseShoe()
    {
        if(m_ShoeItemNums<=0)
        {
            m_ShoeItemNums = 0;
            m_TipFranmePanel.SetPanel(LanguageDataMgr.GetInstance().GetData()["XieZiStr"],
                                      LanguageDataMgr.GetInstance().GetData()["FanHui"],
                                      LanguageDataMgr.GetInstance().GetData()["GouMai"],()=>{
                    m_ShoeItemNums++;
                    if(m_ShoeItemNums>=999)
                        m_ShoeItemNums = 999;
                    PlayerDataMgr.GetInstance().JianGold(200);
                    UpdataUi();
               });
```

```
            UpdataUi();
            return;
        }
        m_PlayerAction.AddSpeedBuff();
        m_ShoeItemNums--;

        //want to determine whether the character has a deceleration BUFF or not
        UpdataUi();
    }

    //using item no.2
    public void UseYaoShui()
    {
        if(m_YaoShuiNums<=0)
        {
            m_YaoShuiNums = 0;

            m_TipFranmePanel.SetPanel(LanguageDataMgr.GetInstance().GetData()["YaoShui"],
                                      LanguageDataMgr.GetInstance().GetData()["FanHui"],
                                      LanguageDataMgr.GetInstance().GetData()["GouMai"],()=>{
                m_YaoShuiNums++;
                if(m_YaoShuiNums>=999)
                    m_YaoShuiNums = 999;
                 PlayerDataMgr.GetInstance().JianGold(200);
                UpdataUi();
            });

            UpdataUi();
            return;
        }
        m_PlayerAction.ClearAllDeBuffer();
        m_YaoShuiNums--;

        UpdataUi();
    }

    //use weapon
    public void UseWeapon()
    {
        if(m_WeaponNums<=0)
        {
            m_WeaponNums = 0;

            UpdataUi();
            return;
        }

        m_PlayerAction.AttackToTarget();
        m_WeaponNums--;

        UpdataUi();
    }

    //UI refresh
    public void UpdataUi()
    {
        m_ShoeItemNumsText.text = m_ShoeItemNums.ToString();
        m_YaoShuiNumsText.text = m_YaoShuiNums.ToString();
        m_WeaponNumsText.text = m_WeaponNums.ToString();
        SaveData();
    }

    //save data
    public void SaveData()
    {
        PlayerPrefs.SetInt("Shoe",m_ShoeItemNums);
        PlayerPrefs.SetInt("YaoShui",m_YaoShuiNums);
        PlayerPrefs.SetInt("Weapon",m_WeaponNums);
    }

    //read data
    public void LoadData()
    {
        m_ShoeItemNums = PlayerPrefs.GetInt("Shoe");
        m_YaoShuiNums = PlayerPrefs.GetInt("YaoShui");
        m_WeaponNums = PlayerPrefs.GetInt("Weapon");
```

```
    }
}
```

Items

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

//creat items
public class RushItem : MonoBehaviour
{
    //resourses
    public List<GameObject> m_ResList = new List<GameObject>();
    public List<Transform> m_ItemsList = new List<Transform>();

    void Start()
    {
        m_ItemsList.Clear();

        for(int i=0;i<transform.childCount;i++)
        {
            int rate = Random.Range(0,101);
            if(rate>=0 && rate<20)
            {
                //?
                GameObject.Instantiate(m_ResList[1],transform.GetChild(i).position,transform.GetChild(i).rotation);
                continue;
            }

            if(rate>=20 && rate<=100)
            {
                //reel
                GameObject.Instantiate(m_ResList[0],transform.GetChild(i).position,transform.GetChild(i).rotation);
                continue;
            }
        }
    }
}
```

Debuff

Add speed buff

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

public class AddSpeedBuff : BuffBase
{
     //speed percentage
    public float m_SpeedM;

    //changing of speed
    public float m_ChangeSpeed;
    void Update()
    {
        Logic();
    }

    public void Init(float _speedm,float _lasttime)
    {
        m_MaxLastTime = _lasttime;
        m_CurTime = 0;
        if(!gameObject.activeSelf)
        {
            m_SpeedM = _speedm;
```

```
            m_ChangeSpeed = 12 * (1+ m_SpeedM);
        }

        gameObject.SetActive(true);
    }

    public override void Enter()
    {
        base.Enter();
        transform.parent.GetComponent<PlayerMove>().speed += m_ChangeSpeed;
        transform.parent.GetComponent<PlayerMove>().m_ASpeed = 1.5f+m_SpeedM;
    }

    public override void Exit()
    {
        base.Exit();
        // if( transform.parent.GetComponent<BuffMgr>().m_SlowSpeedDeBuff.gameObject.activeSelf)
        // {
        //     return;
        // }
        transform.parent.GetComponent<PlayerMove>().speed -= m_ChangeSpeed;
        gameObject.SetActive(false);
        transform.parent.GetComponent<PlayerMove>().m_ASpeed = 1;
    }

    private void OnDisable() {
        Exit();
    }

}
```

Buff base

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

//Buff effect base class
public class BuffBase : MonoBehaviour
{
    //Record the current time spent
    public float m_CurTime;

    //maximum duration
    public float m_MaxLastTime;

    //Action target TAG
    public string m_TargetTag;

    public virtual void Enter() {}

    public virtual void Exit(){}

    public virtual void Logic()
    {
        m_CurTime += Time.deltaTime;
        if(m_CurTime >= m_MaxLastTime)
        {
            gameObject.SetActive(false);
        }
    }

    private void OnEnable() {
        Enter();
    }
}
```

Buff manager

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

//BUFF managment
public class BuffMgr : MonoBehaviour
{
    //speed down
    public SlowSpeedDebuff m_SlowSpeedDeBuff;
    public FreeziDeBuff m_FreeziDeBuff;
    //speed up
    public AddSpeedBuff m_AddSpeedBuff;

    //clean all DEBUFF
    public void ClearAllDeBuffer()
    {
        m_SlowSpeedDeBuff.gameObject.SetActive(false);
        m_AddSpeedBuff.gameObject.SetActive(false);
        m_FreeziDeBuff.gameObject.SetActive(false);
    }
}
```

Freeze buff

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

//freeze BUFF
public class FreeziDeBuff : BuffBase
{
    void Update()
    {
        Logic();
    }

    public void Init(float _lasttime)
    {
        m_MaxLastTime = _lasttime;
        m_CurTime = 0;
        gameObject.SetActive(true);
    }

    public override void Enter()
    {
        base.Enter();
        transform.parent.GetComponent<PlayerAction>().Freezi();
    }

    public override void Exit()
    {
        base.Exit();
        transform.parent.GetComponent<PlayerAction>().UFreezi();
    }

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

Slow speed debuff

```
using System.Net.Http.Headers;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//speed down
```

```
public class SlowSpeedDebuff : BuffBase
{
    //speed percentage
    public float m_SpeedM;

    //changing speed
    public float m_ChangeSpeed;
    void Update()
    {
        Logic();
    }

    public void Init(float _speedm,float _lasttime)
    {
        m_MaxLastTime = _lasttime;
        m_CurTime = 0;
        if(!gameObject.activeSelf)
        {
            m_SpeedM = _speedm;
            m_ChangeSpeed =  12 * m_SpeedM;
        }
        else
        {
// Found that the current BUFF is still in progress - refresh directly
             //It is judged that the deceleration effect is strong, it should replace the current one
             //1. Restore the current changes
            if(_speedm > m_SpeedM)
            {
                m_SpeedM = _speedm;
                transform.parent.GetComponent<PlayerMove>().speed += m_ChangeSpeed;
                //2.replace
                m_ChangeSpeed =   12 * m_SpeedM;
                Enter();
            }
        }

        gameObject.SetActive(true);
    }

    public override void Enter()
    {
        Debug.Log("enter BUFF");
        base.Enter();
        transform.parent.GetComponent<PlayerMove>().speed -= m_ChangeSpeed;
        transform.parent.GetComponent<PlayerMove>().m_ASpeed = 1.5f-m_SpeedM;
    }

    public override void Exit()
    {
        Debug.Log("exit BUFF");
        base.Exit();
        transform.parent.GetComponent<PlayerMove>().speed += m_ChangeSpeed;
        gameObject.SetActive(false);
        transform.parent.GetComponent<PlayerMove>().m_ASpeed = 1;
    }

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

Head top view

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

//Overhead display - 2D display is used here
public class HeadTopView : MonoBehaviour
{
    //overhead display canvas
    public Transform m_Canvas;
```

```
    //The top of the head shows the ownership - which OBJ it belongs
    //to. Specifically, which OBJ is mounted on the top of the head.
    public Transform m_Owner;


    private Vector2 m_UiPos;

    public Camera m_UiCamera;

    private void Start() {
         transform.SetParent(m_Canvas,false);
    }
    //initialization
    public void Init(Transform _canvas, Transform _owner)
    {
        m_Canvas = _canvas;
        m_Owner = _owner;

        transform.SetParent(_canvas,false);
    }

    private void LateUpdate() {

        Vector2  screenPos =  RectTransformUtility.WorldToScreenPoint(Camera.main,m_Owner.transform.position);

        RectTransformUtility.ScreenPointToLocalPointInRectangle(m_Canvas as RectTransform,screenPos,
                                                    m_UiCamera,out m_UiPos);

        transform.localPosition = m_UiPos;
    }
}
```

UI

Game over page

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class GameOverPanel : MonoBehaviour
{
    public Button m_ReTurnMainMenuBtn;
    public Button m_ReStartBtn;

    //text
    public Text m_StrT;
    public Text m_ReStartBtnT;
    public Text m_ReTurnMainMenuBtnT;

    void Start()
    {
        m_StrT.text = LanguageDataMgr.GetInstance().GetData()["GameOver"];
        m_ReStartBtnT.text = LanguageDataMgr.GetInstance().GetData()["ChongXinKaiShi"];
        m_ReTurnMainMenuBtnT.text = LanguageDataMgr.GetInstance().GetData()["FanHuiCaiDan"];

        //back to menu
        m_ReTurnMainMenuBtn.onClick.AddListener(()=>{
            SceneManager.LoadScene("Menu");
        });

        m_ReStartBtn.onClick.AddListener(()=>{
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        });
    }
}
```

Game win page

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class GameWinPanel : MonoBehaviour
{
    public Button m_OkBtn;

    public string m_SceneName;

    public Text m_Str;
    public Text m_OkBtnT;

    void Start()
    {
        m_Str.text = LanguageDataMgr.GetInstance().GetData()["GameWin"];
        m_OkBtnT.text = LanguageDataMgr.GetInstance().GetData()["NextLevel"];
        m_OkBtn.onClick.AddListener(()=>{
            SceneManager.LoadScene(m_SceneName);
        });
    }
}
```

Weapon

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


public class WuChang : MonoBehaviour
{
    public float m_Speed;

    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime *m_Speed);
    }

    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Enemy"))
        {
            Debug.Log("Attacked");
            other.GetComponent<AiMove>().Die();
            GameObject.Destroy(gameObject);
        }
    }
}
```