Project Summary

- Ability at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\Ability.cs:
Summary of Ability:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\Ability.csÐ
This file named Ability.cs defines an abstract class called Ability that inherits from ScriptableObject in Unity. It has various attributes such as abilityName, baseDamage, strengthScaling, and intelligenceScaling which determine the characteristics of the ability. Additionally, it has abstract methods named OnAbilityObjectHit and Activate that specific abilities must implement in order to work. It also has protected virtual methods named StartActivation, UpdateActivation, and EndActivation that subclasses may override to add functionality to an ability's activation process. Ð
Ð
The class also declares several delegate types and events for use within the ability system. For instance, the AbilityObjectEvent is a delegate for methods that receive an instance of AbilityObject and a GameObject as arguments. The class contains two public event fields: OnAbilityObjectSpawnedEvent and OnAbilityObjectHitEvent, both of which are of type AbilityObjectEvent and can be subscribed to by other methods. The OnAbilityActivated event is also a public field of type AbilityEvent and is raised when an ability is activated. Finally, the file also contains a public class called AbilityData, which carries information related to an ability's use, such as the target, CasterStats, CasterController, damage, projectileSpeed, and stunDuration.
Code of file Ability:

```
þÿusing UnityEngine;Ð
public abstract class Ability : ScriptableObjectÐ
{Ð
    public string abilityName;Ð
    public string abilityDescription;Ð
    public Sprite icon;Ð
    public float baseDamage;Ð
    public float strengthScaling;Ð
    public float intelligenceScaling;Ð
    Ð
    public float cooldown;Ð
    public string animationName;Ð
    public float lastTimeUsed = 0;Ð
    public float ActivateDelayTime = 0;Ð
Ð
    Ð
Ð
    public abstract void OnAbilityObjectHit(AbilityObject abilityObject,
```

```
GameObject target);Đ
   public abstract void Activate(AbilityData abilityData);Đ
Đ
   public virtual void PreActivateAbility(AbilityData abilityData){Đ
Đ
   }Đ
   protected virtual void StartActivation(AbilityData abilityData) { }Đ
   protected virtual void UpdateActivation(AbilityData abilityData) { }Đ
   protected virtual void EndActivation(AbilityData abilityData) { }Đ
Đ
    public delegate void AbilityEvent(Ability ability);Đ
    public delegate void AbilityObjectEvent(AbilityObject abilityObject,
GameObject target);Đ
Đ
    public event AbilityObjectEvent OnAbilityObjectSpawnedEvent;Đ
    public event AbilityObjectEvent OnAbilityObjectHitEvent;Đ
 Đ
   public event AbilityEvent OnAbilityActivated;Đ
   public float getLastTimeUsed()Đ
   {Đ
      return lastTimeUsed;Đ
   }Đ
   public float setLastTimeUsed(float time)Đ
   {Đ
      return lastTimeUsed = time;Đ
   }Đ
   protected void RaiseOnObjectSpawned(AbilityObject
abilityObject,GameObject target)Đ
   {Đ
      OnAbilityObjectSpawnedEvent?.Invoke(abilityObject,null);Đ
   }Đ
  Đ
   protected void RaiseOnObjectHit(AbilityObject abilityObject, GameObject
target)Đ
   {Đ
      OnAbilityObjectHitEvent?.Invoke(abilityObject, target);Đ
   }Đ
   protected void RaiseOnAbilityActivated()Đ
   {Đ
      OnAbilityActivated?.Invoke(this);Đ
   }Đ
   Đ
Đ
}Đ
Đ
```

public class AbilityDataÐ
{Ð
    public GameObject Target;Ð
    public CharacterStats CasterStats;Ð
    public AbilityController CasterController;Ð
    public CharacterCombatController CasterCombatController;Ð
    public float damage;Ð
    public float projectileSpeed;Ð
    public float stunDuration;Ð
Ð
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\Ability.csÐ
Ability.cs:Ð
- abilityName: stringÐ
- abilityDescription: stringÐ
- icon: SpriteÐ
- baseDamage: floatÐ
- strengthScaling: floatÐ
- intelligenceScaling: floatÐ
- cooldown: floatÐ
- animationName: stringÐ
- lastTimeUsed: float = 0Ð
- ActivateDelayTime: float = 0Ð
- OnAbilityObjectSpawnedEvent: event (AbilityObject, GameObject)Ð
- OnAbilityObjectHitEvent: event (AbilityObject, GameObject)Ð
- OnAbilityActivated: event (Ability)Ð
- OnAbilityObjectHit: abstract void (AbilityObject, GameObject)Ð
- Activate: abstract void (AbilityData)Ð
- PreActivateAbility: virtual void (AbilityData)Ð
- StartActivation: protected virtual void (AbilityData)Ð
- UpdateActivation: protected virtual void (AbilityData)Ð
- EndActivation: protected virtual void (AbilityData)Ð
- getLastTimeUsed: floatÐ
- setLastTimeUsed: floatÐ
- RaiseOnObjectSpawned: protected void (AbilityObject, GameObject)Ð
- RaiseOnObjectHit: protected void (AbilityObject, GameObject)Ð
- RaiseOnAbilityActivated: protected void ()Ð
  Ð
AbilityData:Ð
- Target: GameObjectÐ
- CasterStats: CharacterStatsÐ
- CasterController: AbilityControllerÐ
- CasterCombatController: CharacterCombatControllerÐ

- damage: floatÐ
- projectileSpeed: floatÐ
- stunDuration: float

- AbilityControllData at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityControllData.cs:
Summary of AbilityControllData:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityControllData.csÐ
"AbilityControllData.cs" is a script that contains class "AbilityControllData", which is used to control abilities in the game. The class has four public variables, including "type" which stores the type of ability, "direction" which stores the direction of the ability, "target" which stores the target game object, and "targetPosition" which stores the target position of the ability. This script is essential for managing abilities in the game and provides easy access to information related to the abilities."
Code of file AbilityControllData:
þÿusing UnityEngine;Ð
class AbilityControllDataÐ
{Ð
    public string type;Ð
    public Vector3 direction;Ð
    public GameObject target;Ð
    public Vector3 targetPosition;Ð
}Ð
Ð

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityControllData.csÐ
AbilityControllData:Ð
- type: stringÐ
- direction: Vector3Ð
- target: GameObjectÐ
- targetPosition: Vector3

- AbilityObject at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityObject.cs:
Summary of AbilityObject:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityObject.csÐ
The file named AbilityObject.cs contains a class that defines an ability object. It has properties for events OnHit, OnUpdate, OnSpawn, and OnDelete, which are triggered during various stages of the ability's lifecycle. The object has an

AbilityData property and a boolean shouldDestroy, deleteOnCollision, and deleteOnTimer. It also has a ParentAbility property. The HandleOnHit method sets the shouldDestroy boolean to true if deleteOnCollision is true. The Update method increases the timer if deleteOnTimer is true and triggers the HandleOnDelete method if the timer exceeds the timerMax. The HandleOnSpawn method is called during the Awake method, which initializes the ability object. The HandleOnDelete method is called during the OnTriggerEnter method, which identifies the target of the ability object by their collider component and triggers the HandleOnHit method. The file also includes interfaces for bouncing, piercing, and homing ability objects.

Code of file AbilityObject:

```
þÿusing System;Ð
using UnityEngine;Ð
Ð
public class AbilityObject : MonoBehaviourÐ
{Ð
    public event Action<GameObject> OnHit;Ð
    public event Action OnUpdate;Ð
    public event Action OnSpawn;Ð
    public event Action OnDelete;Ð
    public AbilityData data;Ð
Ð
    public bool shouldDestroy=false;Ð
Ð
    public bool deleteOnCollision = true;Ð
    public bool deleteOnTimer = false;Ð
    float timer = 0f;Ð
    public float timerMax = 5f;Ð
Ð
    Ð
Ð
    public Ability ParentAbility { get; set; }Ð
Ð
    protected virtual void HandleOnHit(GameObject target)Ð
    {Ð
        // Trigger OnHit event with target as parameterÐ
        OnHit?.Invoke(target);Ð
  Ð
        if(deleteOnCollision){Ð
            shouldDestroy = true;Ð
        }Ð
    }Ð
Ð
    private void Update()Ð
    {Ð
```

```
        // Trigger OnHit event with target as parameterÐ
        OnUpdate?.Invoke();Ð
        if(deleteOnTimer){Ð
            timer += Time.deltaTime;Ð
            if(timer >= timerMax){Ð
                timer = 0f;Ð
                HandleOnDelete();Ð
            }Ð
        }Ð
    Ð
    }Ð
Ð
    protected void HandleOnSpawn()Ð
    {Ð
        // Perform any initialization or setup for the ability object hereÐ
        // Trigger OnSpawn eventÐ
        OnSpawn?.Invoke();Ð
        Ð
    Ð
    }Ð
Ð
    protected void HandleOnDelete()Ð
    {Ð
        // Perform any cleanup or deactivation for the ability object hereÐ
        // Trigger OnDelete eventÐ
        OnDelete?.Invoke();Ð
        Destroy(gameObject);Ð
        Ð
    }Ð
Ð
    public void Awake()Ð
    {Ð
        HandleOnSpawn();Ð
    }Ð
Ð
    private void OnTriggerEnter(Collider collision)Ð
    {Ð
        Debug.Log("OnTriggerEnter");Ð
        if(data == null){Ð
            Debug.LogError("AbilityObject data is null");Ð
            return;Ð
        }Ð
        // Get target HealthController from collided objectÐ
        if (data.CasterStats != null)Ð
        {Ð
```

```
        if (data.CasterStats.gameObject.name == collision.gameObject.name)Đ
        {Đ
            return;Đ
        }Đ
        if (gameObject.name == collision.gameObject.name)Đ
        {Đ
            return;Đ
        }Đ
    }Đ
Đ
        Đ
        // Call HandleOnHit method with target as parameterĐ
        ParentAbility?.OnAbilityObjectHit(this, collision.gameObject);Đ
        HandleOnHit(collision.gameObject);Đ
Đ
    }Đ
}Đ
public interface IBouncingAbilityObjectĐ
{Đ
     float BounceIntensity { get; set; }Đ
    float BounceDuration { get; set; }Đ
    void Bounce(GameObject target);Đ
}Đ
Đ
public interface IPiercingAbilityObjectĐ
{Đ
    void Pierce(GameObject target);Đ
}Đ
Đ
public interface IHomingAbilityObjectĐ
{Đ
    void Home(GameObject target);Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\AbilityObject.csĐ
AbilityObject:Đ
- event OnHit (GameObject)Đ
- event OnUpdateĐ
- event OnSpawnĐ
- event OnDeleteĐ
- data: AbilityDataĐ
- shouldDestroy: boolĐ
- deleteOnCollision: boolĐ
- deleteOnTimer: boolĐ
```

- timer: floatĐ
- timerMax: floatĐ
- ParentAbility: AbilityĐ
Đ
- HandleOnHit(target: GameObject)Đ
- Update()Đ
- HandleOnSpawn()Đ
- HandleOnDelete()Đ
- Awake()Đ
- OnTriggerEnter(collision: Collider)Đ
Đ
IBouncingAbilityObject:Đ
- BounceIntensity: floatĐ
- BounceDuration: floatĐ
Đ
- Bounce(target: GameObject)Đ
Đ
IPiercingAbilityObject:Đ
Đ
- Pierce(target: GameObject)Đ
Đ
IHomingAbilityObject:Đ
Đ
- Home(target: GameObject)

- BaseProjectileObject at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\BaseProjectileObject.cs:
Summary of BaseProjectileObject: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\BaseProjectileObject.csĐ
The file BaseProjectileObject.cs is located in the AbilitysSkillsAndBuffsItems/Abilitys folder of the Unity project directory. This file contains a class that inherits from the AbilityObject class and implements two interfaces, IBouncingAbilityObject and IPiercingAbilityObject. The class represents a base projectile object that can bounce off surfaces and pierce through targets.Đ
Đ
The class has two public float properties, BounceIntensity and BounceDuration, and two public int properties, bounceCount and pierceCount, which are used for determining the projectile's bouncing and piercing behavior.Đ
Đ
The HandleOnHit() method is called when the projectile hits a target. If the projectile was cast by a character with stats, it applies damage to the target. The method then checks if the projectile should bounce or pierce, handles the appropriate logic, and optionally deletes the projectile.Đ

Đ
The Bounce() method is called when the projectile should bounce off a surface. It sets shouldDestroy to false, decrements the bounceCount property, calculates the bounce direction using the target's surface normal, and assigns the new direction to the projectile's transform. Finally, it applies the calculated bounce direction to the projectile's rigidbody velocity.Đ
Đ
The Pierce() method is called when the projectile should pierce through a target. It decrements the pierceCount property and sets shouldDestroy to false.Đ
Đ
Overall, this class provides a flexible base for creating different types of projectiles with varying bouncing and piercing properties.
Code of file BaseProjectileObject:

```
using System.Collections;Đ
using UnityEngine;Đ
public class BaseProjectileObject : AbilityObject, IBouncingAbilityObject, IPiercingAbilityObjectĐ
{Đ
    public float BounceIntensity { get; set; }Đ
    public float BounceDuration { get; set; }Đ
Đ
    public int bounceCount;Đ
    public int pierceCount;Đ
Đ
Đ
    protected override void HandleOnHit(GameObject target)Đ
    {Đ
    // Apply damage to the targetĐ
        if (data.CasterStats != null)Đ
        {Đ
            HealthController targetStats = target.GetComponent<HealthController>();Đ
            if (targetStats != null)Đ
            {Đ
                float damage = data.damage;Đ
                targetStats.TakeDamage(damage, data.CasterStats.gameObject);Đ
            }Đ
        }Đ
Đ
        // Handle Bounce and Pierce logicĐ
        shouldDestroy = deleteOnCollision;Đ
        if (bounceCount > 0)Đ
        {Đ
            Bounce(target);Đ
        }Đ
        else if (pierceCount > 0)Đ
```

```
        {Đ
            Pierce(target);Đ
        }Đ
Đ
        if (shouldDestroy)Đ
        {Đ
            HandleOnDelete();Đ
        }Đ
       Đ
    }Đ
Đ
    public void Bounce(GameObject target)Đ
    {Đ
        Đ
        shouldDestroy = false;Đ
        bounceCount--;Đ
Đ
        Vector3 bounceDirection = Vector3.Reflect(transform.forward,
target.transform.up);Đ
        transform.forward = bounceDirection;Đ
Đ
        Rigidbody rb = GetComponent<Rigidbody>();Đ
        rb.velocity = bounceDirection * data.projectileSpeed;Đ
Đ
        Đ
    }Đ
Đ
    public void Pierce(GameObject target)Đ
    {Đ
Đ
        pierceCount--;Đ
Đ
        shouldDestroy = false;Đ
    }Đ
Đ
   Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\A
bilitysSkillsAndBuffsItems\Abilitys\BaseProjectileObject.csĐ
Class Name: BaseProjectileObjectĐ
Đ
BounceIntensity: floatĐ
BounceDuration: floatĐ
bounceCount: intĐ
```

pierceCount: intÐ
Ð
HandleOnHit(target: GameObject):Ð
    If data.CasterStats is not null:Ð
        targetStats = target.GetComponent<HealthController>()Ð
        If targetStats is not null:Ð
            damage = data.damageÐ
            targetStats.TakeDamage(damage, data.CasterStats.gameObject)Ð
Ð
    shouldDestroy = deleteOnCollisionÐ
    If bounceCount > 0:Ð
        Bounce(target)Ð
    Else If pierceCount > 0:Ð
        Pierce(target)Ð
Ð
    If shouldDestroy:Ð
        HandleOnDelete()Ð
Ð
Bounce(target: GameObject):Ð
    shouldDestroy = falseÐ
    bounceCount--Ð
    bounceDirection = Vector3.Reflect(transform.forward, target.transform.up)Ð
    transform.forward = bounceDirectionÐ
    rb = GetComponent<Rigidbody>()Ð
    rb.velocity = bounceDirection * data.projectileSpeedÐ
Ð
Pierce(target: GameObject):Ð
    pierceCount--Ð
    shouldDestroy = false

- DefaultProjectileAbility at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultProjectileAbility.cs:
Summary of DefaultProjectileAbility: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultProjectileAbility.csÐ
The file "DefaultProjectileAbility.cs" is a script for creating a default projectile ability in Unity. It is part of a larger folder of scripts related to abilities, skills, buffs, and items.Ð
Ð
The script includes a public variable for the projectile's prefab and speed, and two override methods for activating the projectile ability and handling what happens when the projectile hits a target.Ð
Ð
When the projectile hits a target, the script checks to see if the caster's stats are

present and, if so, deals damage to the target through the HealthController component.Đ
Đ
When the ability is activated, the script spawns a new projectile at the caster's fire point, sets its velocity to the desired speed, and assigns the projectile's data and parent ability.Đ
Đ
The script primarily uses Unity's built-in classes for game objects, transforms, and rigidbodies to manipulate objects and pass data between functions.
Code of file DefaultProjectileAbility:

```
using System.Collections;Đ
using UnityEngine;Đ
Đ
// Base Projectile Ability classĐ
[CreateAssetMenu(menuName = "Abilities/DefaultProjectileAbility")]Đ
public class DefaultProjectileAbility : AbilityĐ
{Đ
    public GameObject projectilePrefab;Đ
    public float projectileSpeed = 5f;Đ
Đ
    public override void OnAbilityObjectHit(AbilityObject abilityObject,
GameObject target)Đ
    {Đ
        if(abilityObject.data.CasterStats != null)Đ
        {Đ
            HealthController targetStats = target.GetComponent<HealthController>();Đ
            if (targetStats != null)Đ
            {Đ
                float damage = abilityObject.data.damage;Đ

targetStats.TakeDamage(damage,abilityObject.data.CasterStats.gameObject);Đ
            }Đ
        }Đ
         RaiseOnObjectHit(abilityObject,target);Đ
  Đ
Đ
    }Đ
Đ
    public override void Activate(AbilityData abilityData)Đ
    {Đ
        if (abilityData.CasterStats == null) return;Đ
Đ
        Transform firePoint =
abilityData.CasterStats.GetComponent<AbilityController>().firePoint;Đ
        Đ
```

```
        GameObject projectileInstance = Instantiate(projectilePrefab,
firePoint.position, firePoint.rotation);Đ
        BaseProjectileObject abilityObject =
projectileInstance.GetComponent<BaseProjectileObject>();Đ
        RaiseOnObjectSpawned(abilityObject,null);Đ
Đ
        Rigidbody rb = projectileInstance.GetComponent<Rigidbody>();Đ
        rb.velocity = firePoint.forward * projectileSpeed;Đ
Đ
        Đ
        abilityObject.ParentAbility = this;Đ
        abilityObject.data = abilityData;Đ
        abilityData.projectileSpeed = projectileSpeed;Đ
    }Đ
}Đ
Đ
// Base Projectile Object classĐ
Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultProjectileAbility.csĐ
DefaultProjectileAbility:Đ
- projectilePrefab: GameObjectĐ
- projectileSpeed: floatĐ
Đ
- OnAbilityObjectHit(abilityObject, target):Đ
  - if(abilityObject.data.CasterStats != null):Đ
    - targetStats = target.GetComponent<HealthController>()Đ
    - if (targetStats != null):Đ
      - damage = abilityObject.data.damageĐ
      -
targetStats.TakeDamage(damage,abilityObject.data.CasterStats.gameObject)Đ
Đ
  - RaiseOnObjectHit(abilityObject,target)Đ
 Đ
Đ
- Activate(abilityData):Đ
  - if (abilityData.CasterStats == null):Đ
    - returnĐ
Đ
  - firePoint =
abilityData.CasterStats.GetComponent<AbilityController>().firePointĐ
  - projectileInstance = Instantiate(projectilePrefab, firePoint.position,
firePoint.rotation)Đ

- abilityObject = projectileInstance.GetComponent<BaseProjectileObject>()Đ
- RaiseOnObjectSpawned(abilityObject,null)Đ
Đ
- rb = projectileInstance.GetComponent<Rigidbody>()Đ
- rb.velocity = firePoint.forward * projectileSpeedĐ
Đ
Đ
- abilityObject.ParentAbility = thisĐ
- abilityObject.data = abilityDataĐ
- abilityData.projectileSpeed = projectileSpeed

- DefaultSkill at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultSkill.cs:
Summary of DefaultSkill:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultSkill.csĐ
The file name is "DefaultSkill.cs" and it is located in the folder "Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys". It utilizes the Unity engine.Đ
Đ
This script uses the "CreateAssetMenu" attribute to create a new asset menu item called "Skill" with a sub-menu called "Skill". This allows the ability to create a new skill asset in the Unity editor.Đ
Đ
The "DefaultSkill" class inherits from the "Skill" class, indicating that it represents a skill in the game. The internal logic of this script is not explicitly defined in this excerpt, as it only defines the class itself.Đ
Đ
Overall, the purpose of this script is to provide a framework for creating new skills in the game, accessible through the Unity editor.
Code of file DefaultSkill:
þÿĐ
using UnityEngine;Đ
Đ
[CreateAssetMenu(fileName = "Skill", menuName = "Skill/Skill", order = 1)]Đ
public class DefaultSkill : SkillĐ
{Đ
Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\DefaultSkill.csĐ
DefaultSkill:Đ
- CreateAssetMenu:Đ
- fileName = "Skill"Đ
- menuName = "Skill/Skill"Đ

- order = 1Đ
- Skill

- FireBall at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\FireBall.cs:
Summary of FireBall:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\FireBall.csĐ
The file "FireBall.cs" is a script used to create a fireball ability in a Unity game. It inherits from the "DefaultProjectileAbility" class. Đ
Đ
The "OnAbilityObjectHit" function is called when the fireball hits an object. If the caster of the ability is not null, it will try to damage the object hit if it has a "HealthController" component. Đ
Đ
The "Activate" function is called when the ability is activated. It first checks if the caster stats are not null. It then retrieves the caster's "firePoint" transform from the "AbilityController" component and instantiates a fireball object at that location using the projectile prefab. The fireball's velocity is set in the direction of the firePoint with the projectileSpeed. It also sets the parent ability and ability data for the projectile object. Đ
Đ
This script is used to create and control the behavior of a fireball ability in Unity by defining its logic for activation and when it hits an object.
Code of file FireBall:

```
using UnityEngine;Đ
[CreateAssetMenu(menuName = "Abilities/Fireball")]Đ
class FireBall : DefaultProjectileAbility {Đ
    //Expecting BaseProjectileObject to be a prefabĐ
  public override void OnAbilityObjectHit(AbilityObject abilityObject, GameObject target) {Đ
    if (abilityObject.data.CasterStats != null) {Đ
      HealthController targetHealth = target.GetComponent<HealthController>();Đ
      if (targetHealth != null) {Đ
        float damage = abilityObject.data.damage;Đ

targetHealth.TakeDamage(damage,abilityObject.data.CasterStats.gameObject);Đ
      }Đ
    }Đ
    RaiseOnObjectHit(abilityObject, target);Đ
  }Đ
Đ
    public override void Activate(AbilityData abilityData) {Đ
    if (abilityData.CasterStats == null) return;Đ
Đ
```

```
    Transform firePoint =
abilityData.CasterStats.GetComponent<AbilityController>().firePoint;Đ
Đ
    GameObject projectileInstance = Instantiate(projectilePrefab, firePoint.position,
firePoint.rotation);Đ
    BaseProjectileObject abilityObject =
projectileInstance.GetComponent<BaseProjectileObject>();Đ
    RaiseOnObjectSpawned(abilityObject, null);Đ
Đ
    Rigidbody rb = projectileInstance.GetComponent<Rigidbody>();Đ
    rb.velocity = firePoint.forward * projectileSpeed;Đ
Đ
    abilityObject.ParentAbility = this;Đ
    abilityObject.data = abilityData;Đ
    abilityData.projectileSpeed = projectileSpeed;Đ
  }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\FireBall.csĐ
File name: FireBall.csĐ
Đ
Class: FireBall (inherits DefaultProjectileAbility)Đ
Đ
- CreateAssetMenu (menuName = "Abilities/Fireball")Đ
- Override OnAbilityObjectHit (abilityObject, target)Đ
  - If abilityObject.data.CasterStats exists, get target's HealthController componentĐ
  - If targetHealth exists, get damage from abilityObject.data, and call TakeDamage() on targetHealth with damage and abilityObject.data.CasterStats.gameObject as argumentsĐ
  - RaiseOnObjectHit with abilityObject and target as argumentsĐ
- Override Activate (abilityData)Đ
  - If abilityData.CasterStats is null, returnĐ
  - Get firePoint from abilityData.CasterStats's AbilityController componentĐ
  - Instantiate projectilePrefab at firePoint's position and rotation, and get its BaseProjectileObject componentĐ
  - RaiseOnObjectSpawned with abilityObject and null as argumentsĐ
  - Get Rigidbody component from projectileInstance, and set its velocity to firePoint's forward direction multiplied by projectileSpeedĐ
  - Set abilityObject's ParentAbility and data to this and abilityData, respectivelyĐ
  - Set abilityData's projectileSpeed to projectileSpeed.

- ShieldBash at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\ShieldBash.cs:

Summary of ShieldBash:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\ShieldBash.csÐ
The ShieldBash.cs file is a script for an ability called "Shield Bash" in a game. It inherits from a base class named "Ability" and is decorated with the CreateAssetMenu attribute to allow it to be created as an asset within the Unity Editor.Ð
Ð
This ability has a prefabAbilityObject variable, which is the object that will be instantiated when the ability is activated. It also has a stunDuration variable, which represents how long the target will be stun if hit by this ability.Ð
Ð
The ShieldBash constructor sets various properties of this ability, such as the ability name, base damage, scaling factors, and the animation name.Ð
Ð
The OnAbilityObjectHit method is called when the ability object hits a target. It checks if the caster's stats are not null, gets the target's health controller component, deals damage to the target, and stuns the target if stunDuration is greater than or equal to 0. Finally, it raises an event to notify listeners that an object has been hit.Ð
Ð
The Activate method is called when the ability is activated. It checks if the caster's stats are null, gets the caster's forward direction, instantiates the ability object, sets the velocity of the rigidbody to the forward direction times the projectile speed of the ability, sets various properties of the ability data, and raises an event to notify listeners that an object has been spawned.
Code of file ShieldBash:

```
using UnityEngine;Ð
[CreateAssetMenu(menuName = "Abilities/ShieldBash")]Ð
public class ShieldBash : AbilityÐ
{Ð
    public GameObject prefabAbilityObject;Ð
    public float stunDuration = 2f;Ð
    public ShieldBash(){Ð
        abilityName = "Shield Bash";Ð
        baseDamage = 50;Ð
        strengthScaling = 0.5f;Ð
        intelligenceScaling = 0.5f;Ð
        animationName = "Shield Bash animation";Ð
        Ð
    }Ð
Ð
    public override void OnAbilityObjectHit(AbilityObject abilityObject, GameObject target)Ð
    {Ð
```

```
        if (abilityObject.data.CasterStats != null)Đ
        {Đ
            HealthController targetStats = target.GetComponent<HealthController>();Đ
            if (targetStats != null)Đ
            {Đ
                float damage = abilityObject.data.damage;Đ

targetStats.TakeDamage(damage,abilityObject.data.CasterStats.gameObject);Đ
Đ
                if (abilityObject.data.stunDuration >= 0f)Đ
                {Đ

targetStats.GetComponent<IStunnable>().Stun(abilityObject.data.stunDuration);Đ
                }Đ
            }Đ
        }Đ
        RaiseOnObjectHit(abilityObject, target);Đ
    }Đ
Đ
    public override void Activate(AbilityData abilityData)Đ
    {Đ
        if (abilityData.CasterStats == null) return;Đ
Đ
        Transform casterTransform = abilityData.CasterStats.transform;Đ
        Vector3 forwardDirection = casterTransform.forward;Đ
Đ
        GameObject abilityObjectInstance = Instantiate(prefabAbilityObject,
casterTransform.position + forwardDirection, Quaternion.identity);Đ
        AbilityObject abilityObject =
abilityObjectInstance.GetComponent<AbilityObject>();Đ
        RaiseOnObjectSpawned(abilityObject, null);Đ
Đ
        Rigidbody rb = abilityObjectInstance.GetComponent<Rigidbody>();Đ
        rb.velocity = forwardDirection * abilityData.projectileSpeed;Đ
Đ
        abilityObject.data = abilityData;Đ
        abilityData.Target = null;Đ
        abilityData.projectileSpeed = 0f;Đ
        abilityObject.ParentAbility = this;Đ
        abilityData.stunDuration = stunDuration;Đ
    }Đ
}Đ
Đ
Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\ShieldBash.csĐ
File Name: ShieldBash.csĐ
Đ
Class Name: ShieldBashĐ
Đ
- prefabAbilityObject: GameObjectĐ
- stunDuration: floatĐ
- abilityName: "Shield Bash"Đ
- baseDamage: 50Đ
- strengthScaling: 0.5fĐ
- intelligenceScaling: 0.5fĐ
- animationName: "Shield Bash animation"Đ
Đ
+ ShieldBash()Đ
+ OnAbilityObjectHit(AbilityObject abilityObject, GameObject target)Đ
+ Activate(AbilityData abilityData)

- SimpleStrike at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\SimpleStrike.cs:
Summary of SimpleStrike:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\SimpleStrike.csĐ
The file is named "SimpleStrike.cs" and is a subclass of the "Ability" class. It contains a property "MeelePrefab" which represents the game object that will be instantiated upon activation of the ability. It also has a "lifeTime" property which determines the time the instantiated object will live before being destroyed.Đ
Đ
The "OnAbilityObjectHit" method is overriden to allow for damage to be dealt to the target's health. The target's health is retrieved via a "HealthController" component attached to the target game object.Đ
Đ
The "Activate" method is overriden and is responsible for the instantiation of the "MeelePrefab" game object at the position of the caster. A "MeleeStrikeObject" is then added to the instantiated object and this object's "data" and "ParentAbility" properties are set. Finally, the instantiated object is destroyed after the specified "lifeTime".Đ
Đ
The "MeleeStrikeObject" class is defined which is responsible for detecting when the instantiated object collides with another object. The "OnTriggerEnter" method handles this detection and checks if the colliding object is not the caster or the instantiated object itself. If it's not, then the "HandleOnHit" function is called and the "ParentAbility" on the instantiated object is triggered to handle the hit with the collision game object.

Code of file SimpleStrike:

```
þÿusing UnityEngine;Đ
Đ
[CreateAssetMenu(menuName = "Abilities/SimpleStrike")]Đ
public class SimpleStrike : AbilityĐ
{Đ
    // SimpleStrike specific properties, if anyĐ
    public GameObject MeelePrefab;Đ
    public float lifeTime = 0.5f;Đ
Đ
    public override void OnAbilityObjectHit(AbilityObject abilityObject,
GameObject target)Đ
    {Đ
        HealthController healthController =
target.GetComponent<HealthController>();Đ
        if (healthController != null)Đ
        {Đ
Đ
            healthController.TakeDamage(abilityObject.data.damage,abilityObject.data.
CasterStats.gameObject);Đ
        }Đ
    }Đ
Đ
Đ
    public override void Activate(AbilityData abilityData)Đ
    {Đ
        GameObject meleeStrikeInstance = Instantiate(MeelePrefab,
abilityData.CasterStats.transform.position, Quaternion.identity);Đ
        MeleeStrikeObject abilityObject =
meleeStrikeInstance.AddComponent<MeleeStrikeObject>();Đ
Đ
        abilityObject.ParentAbility = this;Đ
        abilityObject.data = abilityData;Đ
        Destroy(meleeStrikeInstance, lifeTime);Đ
    }Đ
Đ
Đ
}Đ
Đ
public class MeleeStrikeObject : AbilityObjectĐ
{Đ
    private void OnTriggerEnter(Collider collision)Đ
    {Đ
        if (data.CasterStats != null)Đ
        {Đ
```

```
            if (data.CasterStats.gameObject.name == collision.gameObject.name)Đ
            {Đ
                return;Đ
            }Đ
            if (gameObject.name == collision.gameObject.name)Đ
            {Đ
                return;Đ
            }Đ
        }Đ
Đ
        HandleOnHit(collision.gameObject);Đ
        ParentAbility.OnAbilityObjectHit(this, collision.gameObject);Đ
Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Abilitys\SimpleStrike.csĐ
File: SimpleStrike.csĐ
Đ
Class: SimpleStrikeĐ
- MeelePrefab: GameObjectĐ
- lifeTime: floatĐ
- OnAbilityObjectHit(abilityObject: AbilityObject, target: GameObject)Đ
- Activate(abilityData: AbilityData)Đ
Đ
Class: MeleeStrikeObjectĐ
- OnTriggerEnter(collision: Collider)

- Buff at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\Buff.cs:
Summary of Buff:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\Buff.csĐ
Summary:Đ
Đ
The Buff.cs script is a ScriptableObject that contains variables for a buff's name, duration, whether it is stackable or not, and the maximum number of stacks allowed. It also has three events, OnApply, OnFade, and OnHit, that can be triggered by BuffInstance objects.Đ
Đ
The script contains three methods: InvokeOnApply, InvokeOnFade, and InvokeOnHit, which are used to trigger the events. There is also a method, GetEventTypes, which returns a list of strings that correspond to the event types

that have been assigned to the Buff object.Đ
Đ
Overall, the Buff.cs script defines the properties and events of a buff object and contains methods to trigger those events.
Code of file Buff:

```
þÿusing System.Collections.Generic;Đ
using UnityEngine;Đ
[System.Serializable]Đ
public class Buff : ScriptableObjectĐ
{Đ
    public Buff()Đ
    {Đ
      statModifier = new StatsModifier();Đ
    }Đ
    public StatsModifier statModifier;Đ
    public string buffName;Đ
    public float duration;Đ
    public bool stackable;Đ
    public int maxStacks;Đ
Đ
    private event System.Action<BuffInstance> OnApply;Đ
    private event System.Action<BuffInstance> OnFade;Đ
    private event System.Action<BuffInstance> OnHit;Đ
Đ
    public virtual void InvokeOnApply(BuffInstance buffInstance)Đ
    {Đ
        OnApply?.Invoke(buffInstance);Đ
    }Đ
Đ
    public virtual void InvokeOnFade(BuffInstance buffInstance)Đ
    {Đ
        OnFade?.Invoke(buffInstance);Đ
    }Đ
Đ
    public virtual void InvokeOnHit(BuffInstance buffInstance)Đ
    {Đ
        OnHit?.Invoke(buffInstance);Đ
    }Đ
Đ
    public List<string> GetEventTypes()Đ
    {Đ
        List<string> eventTypes = new List<string>();Đ
Đ
        if (OnApply != null)Đ
        {Đ
```

```
            eventTypes.Add("OnApply");Đ
        }Đ
Đ
        if (OnFade != null)Đ
        {Đ
            eventTypes.Add("OnFade");Đ
        }Đ
Đ
        if (OnHit != null)Đ
        {Đ
            eventTypes.Add("OnHit");Đ
        }Đ
Đ
        return eventTypes;Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\Buff.csĐ
File name: Buff.csĐ
Đ
Class: BuffĐ
Đ
- buffName: stringĐ
- duration: floatĐ
- stackable: boolĐ
- maxStacks: intĐ
Đ
- OnApply: event taking BuffInstance parameterĐ
- OnFade: event taking BuffInstance parameterĐ
- OnHit: event taking BuffInstance parameterĐ
Đ
+ InvokeOnApply(buffInstance: BuffInstance)Đ
+ InvokeOnFade(buffInstance: BuffInstance)Đ
+ InvokeOnHit(buffInstance: BuffInstance)Đ
Đ
+ GetEventTypes(): List<string>

- BuffInstance at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\BuffInstance.cs:
Summary of BuffInstance:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\BuffInstance.csĐ
The file named "BuffInstance.cs" is a script containing a class that represents an

instance of a buff in a game. It has four public variables; a Buff object, a target GameObject, an integer representing the current stacks of the buff, and a float representing the remaining duration of the buff.Đ
Đ
The constructor of the class takes in a Buff object, a target GameObject, the initial stacks of the buff, and the initial duration of the buff and sets the BuffInstance variables accordingly.Đ
Đ
The Update method of the class reduces the remaining duration of the buff by the Time.deltaTime. If the remaining duration is less than or equal to zero, the OnBuffFade method is called, and the method returns. Otherwise, it performs any other update logic specific to the buff.Đ
Đ
The Refresh method allows for the refreshing of the buff's duration by updating the remaining duration variable.Đ
Đ
The AddStack method increases the current stacks of the buff and calls the OnBuffApply method.Đ
Đ
The RemoveStack method decreases the current stacks of the buff and calls the OnBuffFade method if the current stacks are less than or equal to zero.Đ
Đ
The OnBuffApply method performs any actions or applies stat changes when the buff is applied.Đ
Đ
The OnBuffFade method performs any actions or reverts stat changes when the buff fades.Đ
Đ
The OnBuffHit method performs any actions or applies effects when the buff "hits" (e.g., dealing damage or applying a debuff).
Code of file BuffInstance:
þÿusing UnityEngine;Đ
Đ
public class BuffInstanceĐ
{Đ
**   public Buff buff;Đ**
**   public GameObject target;Đ**
**   public int currentStacks;Đ**
**   public float remainingDuration;Đ**
**   StatsModifier characterStats;Đ**
Đ
**   public BuffInstance(Buff buff, GameObject target, int initialStacks, float initialDuration)Đ**
**   {Đ**
**   this.buff = buff;Đ**

```
        this.target = target;Đ
        this.currentStacks = initialStacks;Đ
        this.remainingDuration = initialDuration;Đ
        characterStats = target.GetComponent<BuffSystem>().TotalstatsModifier;Đ
    }Đ
Đ
    public void Update()Đ
    {Đ
        remainingDuration -= Time.deltaTime;Đ
Đ
        if (remainingDuration <= 0)Đ
        {Đ
            OnBuffFade();Đ
            target.GetComponent<BuffSystem>().RemoveBuff(buff); // add this lineĐ
            return;Đ
        }Đ
Đ
        // Perform any other update logic specific to the buffĐ
    }Đ
Đ
    public void Refresh(float duration)Đ
    {Đ
        remainingDuration = duration;Đ
    }Đ
Đ
    public void AddStack()Đ
    {Đ
        currentStacks++;Đ
        OnBuffApply();Đ
    }Đ
Đ
 Đ
Đ
    public void OnBuffApply()Đ
    {Đ
        // Perform any actions or apply stat changes when the buff is appliedĐ
        if (buff.statModifier != null)Đ
        {Đ
            characterStats.Add(buff.statModifier);Đ
            target.GetComponent<CharacterStats>().UpdateSubStats();Đ
        }Đ
        buff.InvokeOnApply(this);Đ
    }Đ
Đ
    public void OnBuffFade()Đ
```

```
    {Đ
        // Perform any actions or apply stat changes when the buff is appliedĐ
        if (buff.statModifier != null)Đ
        {Đ
            characterStats.Sub(buff.statModifier);Đ
            target.GetComponent<CharacterStats>().UpdateSubStats();Đ
        }Đ
        buff.InvokeOnFade(this);Đ
        Đ
    }Đ
Đ
    public void OnBuffHit()Đ
    {Đ
        // Perform any actions or apply effects when the buff "hits" (e.g., dealing
damage or applying a debuff)Đ
        buff.InvokeOnHit(this);Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Buffs\BuffInstance.csĐ
BuffInstance:Đ
- buff: BuffĐ
- target: GameObjectĐ
- currentStacks: intĐ
- remainingDuration: floatĐ
Đ
+BuffInstance(buff: Buff, target: GameObject, initialStacks: int, initialDuration: float)Đ
Đ
+Update()Đ
+Refresh(duration: float)Đ
+AddStack()Đ
+RemoveStack()Đ
+OnBuffApply()Đ
+OnBuffFade()Đ
+OnBuffHit()

- IStatsProvider at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\IStatsProvider.cs:
Summary of IStatsProvider:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\IStatsProvider.csĐ
The file "IStatsProvider.cs" contains an interface called "IStatsProvider". This

interface has a single method called "GetCharacterStats()" which returns an object of type "CharacterStats". Internally, this interface is meant to provide a mechanism for getting stats related to a character object in a Unity project. The "GetCharacterStats()" method is responsible for returning all the relevant stats for a character, which can be used for various purposes in the game. Overall, this interface plays a crucial role in managing character stats within the game.
Code of file IStatsProvider:
þÿpublic interface IStatsProviderÐ
{Ð
CharacterStats GetCharacterStats();Ð
}Ð

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\IStatsProvider.csÐ
IStatsProviderÐ
- GetCharacterStats()Ð
- CharacterStats

- Item at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\items\Item.cs:
Summary of Item:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\items\Item.csÐ
The file name is Item.cs, and it is located in the AbilitysSkillsAndBuffsItems folder within the Assets folder in the project. This file contains two classes: Item and EquipableItem.Ð
Ð
The Item class is an abstract class that extends ScriptableObject. It has three public variables: itemName, description, and icon. These store the name, description, and icon of the item, respectively. This class is intended to be inherited by other classes that represent specific items in the game.Ð
Ð
The EquipableItem, on the other hand, is a serializable class that also extends ScriptableObject. It has six public variables: equipmentType, strengthBonus, intelligenceBonus, dexterityBonus, enduranceBonus, and wisdomBonus. These variables store the bonuses to different stats that the item provides when equipped. The EquipManager.EquipmentType enumeration determines the type of equipment the item represents.Ð
Ð
Moreover, EquipableItem includes a subStatsModifier property that stores an object of the StatsModifier class. It is used to modify the stats of the character when the item is equipped.Ð
Ð
Overall, both Item and EquipableItem classes define the basic properties and

bonuses for in-game items, with EquipableItem class having more specific properties for equipped items.
Code of file Item:

```
þÿusing UnityEngine;Ð
Ð
public abstract class Item : ScriptableObjectÐ
{Ð
    public string itemName;Ð
    public string description;Ð
    public Sprite icon;Ð
}Ð
Ð
Ð
[System.Serializable]Ð
public class EquipableItem : ItemÐ
{Ð
    public EquipManager.EquipmentType equipmentType;Ð
    public float strengthBonus;Ð
    public float intelligenceBonus;Ð
    public float dexterityBonus;Ð
    public float enduranceBonus;Ð
    public float wisdomBonus;Ð
Ð
    public StatsModifier subStatsModifier;Ð
}Ð
```

Corresponding SyntaxTree:

```
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\items\Item.csÐ
File: Item.csÐ
Ð
- abstract class ItemÐ
    - string itemNameÐ
    - string descriptionÐ
    - Sprite iconÐ
Ð
- class EquipableItem : ItemÐ
    - EquipManager.EquipmentType equipmentTypeÐ
    - float strengthBonusÐ
    - float intelligenceBonusÐ
    - float dexterityBonusÐ
    - float enduranceBonusÐ
    - float wisdomBonusÐ
    - StatsModifier subStatsModifier
```

- BouceSkill at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\BouceSkill.cs:
Summary of BouceSkill:

Code of file BouceSkill:

Corresponding SyntaxTree:

- FireballMasterySkill at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\FireballMasterySkill.cs:
Summary of FireballMasterySkill:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\FireballMasterySkill.csÐ
"FireballMasterySkill.cs" is a script that defines the behavior and logic for the "FireBallMastery" skill in a Unity game. It is derived from the base class "Skill". The script contains a serialized field "explosionPrefab" of type GameObject and two methods, "ApplySkill" and "RemoveSkill", which override the method of the base class. Ð
Ð
The "ApplySkill" method is called when the skill is applied to a player's stats. It first retrieves the "Fireball" ability from the player's stats using the "GetFireballAbility" method. If the ability is found, the "OnAbilityObjectHitEvent" event is subscribed. Ð
Ð
The "RemoveSkill" method is called when the skill is removed from the player's stats. If the "Fireball" ability is found, the "OnAbilityObjectHitEvent" event is unsubscribed. Ð
Ð
The "GetFireballAbility" method retrieves the "Fireball" ability from the player's stats through the "AbilityController" component. Ð
Ð
The "ExplodeOnHit" method is an event handler that is called when the "OnAbilityObjectHitEvent" event is triggered. It applies damage to the targets within a specified radius and instantiates an explosion at the hit position. Ð
Ð
The "ApplyDamageToTargets" method is currently an empty method that will be used to apply damage to targets in the specified radius when an object is hit by the fireball. Ð
Ð
The "InstantiateExplosion" method instantiates an explosion prefab at the specified position. If there is no explosion prefab assigned, it will log a warning message.Ð
Ð
The script also includes a "CreateAssetMenu" attribute, which enables it to be

created as an asset through the Unity Editor's "Create" menu. The asset can be named "FireBallMastery" and has a menu item under "Skill/FireBallMastery" with an order of 1.
Code of file FireballMasterySkill:

```
using UnityEngine;Ð
[CreateAssetMenu(fileName = "FireBallMastery", menuName = "Skill/
FireBallMastery", order = 1)]Ð
public class FireballMasterySkill : SkillÐ
{Ð
   [SerializeField]Ð
   private GameObject explosionPrefab;Ð
Ð
   public override void ApplySkill(CharacterStats playerStats)Ð
   {Ð
      Debug.Log("Apply Skill");Ð
      FireBall fireballAbility = GetFireballAbility(playerStats);Ð
      if (fireballAbility != null)Ð
      {Ð
         Debug.Log("Fireball Ability found");Ð
         fireballAbility.OnAbilityObjectHitEvent += ExplodeOnHit;Ð
      }Ð
   }Ð
Ð
   public override void RemoveSkill(CharacterStats playerStats)Ð
   {Ð
      FireBall fireballAbility = GetFireballAbility(playerStats);Ð
      if (fireballAbility != null)Ð
      {Ð
         fireballAbility.OnAbilityObjectHitEvent -= ExplodeOnHit;Ð
      }Ð
   }Ð
Ð
   private FireBall GetFireballAbility(CharacterStats playerStats)Ð
   {Ð
      AbilityController abilityController =
playerStats.GetComponent<AbilityController>();Ð
      return abilityController.learnedAbilitys.Find(a => a is FireBall) as FireBall;Ð
   }Ð
Ð
   private void ExplodeOnHit(AbilityObject abilityObject, GameObject target)Ð
   {Ð
      Debug.Log("EXPLODE ON Hit");Ð
      ApplyDamageToTargets(abilityObject.transform.position, 2f,
abilityObject.data.damage * 0.5f);Ð
      InstantiateExplosion(abilityObject.transform.position);Ð
```

```
    }Đ
Đ
    private void ApplyDamageToTargets(Vector3 position, float radius, float
damage)Đ
    {Đ
   Đ
    }Đ
Đ
    private void InstantiateExplosion(Vector3 position)Đ
    {Đ
        if (explosionPrefab != null)Đ
        {Đ
            GameObject explosion = Instantiate(explosionPrefab, position,
Quaternion.identity);Đ
            // Add additional logic for the explosion, such as configuring the
explosion's lifetime or assigning its parentĐ
        }Đ
        elseĐ
        {Đ
            Debug.LogWarning("No explosion prefab assigned to
FireballMasterySkill.");Đ
        }Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\FireballMasterySkill.csĐ
FireballMasterySkill:Đ
Đ
- explosionPrefab: GameObjectĐ
- ApplySkill(playerStats: CharacterStats): voidĐ
    - fireballAbility: FireBallĐ
        - OnAbilityObjectHitEvent += ExplodeOnHitĐ
- RemoveSkill(playerStats: CharacterStats): voidĐ
    - fireballAbility: FireBallĐ
        - OnAbilityObjectHitEvent -= ExplodeOnHitĐ
- GetFireballAbility(playerStats: CharacterStats): FireBallĐ
- ExplodeOnHit(abilityObject: AbilityObject, target: GameObject): voidĐ
    - ApplyDamageToTargets(position: Vector3, radius: float, damage: float)Đ
    - InstantiateExplosion(position: Vector3)Đ
- ApplyDamageToTargets(position: Vector3, radius: float, damage: float): voidĐ
- InstantiateExplosion(position: Vector3): void

- Skill at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\Skill.cs:
```

Summary of Skill:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\Skill.csÐ
The file "Skill.cs" contains an abstract class called Skill that inherits from the ScriptableObject class in Unity. This class defines various parameters such as the skill name, a list of archetype types for which the skill is applicable, and a StatModifier object.Ð
Ð
The class also includes three methods, ApplySkill(), RemoveSkill(), and OnSpawnAbilityObject() which can be overridden to implement skill-specific behavior in derived classes.Ð
Ð
Additionally, the class includes an enum called Archetype which lists the different archetype types a skill can be applied to, including Strength, Intelligence, Dexterity, Endurance, and Wisdom.
Code of file Skill:

```
þÿusing System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
public abstract class Skill : ScriptableObjectÐ
{Ð
    public string skillName;Ð
    public List<Archetype> archTypes;Ð
    public StatsModifier statModifier;Ð
Ð
    public virtual void ApplySkill(CharacterStats characterStats)Ð
    {Ð
        // Implement skill-specific behavior in derived classesÐ
    }Ð
Ð
    public virtual void RemoveSkill(CharacterStats characterStats)Ð
    {Ð
        // Implement skill-specific behavior in derived classesÐ
    }Ð
    public virtual void OnSpawnAbilityObject(AbilityObject abilityObject,
AbilityData abilityData)Ð
    {Ð
Ð
    }Ð
    Ð
}Ð
Ð
Ð
public enum ArchetypeÐ
{Ð
```

Strength,Đ
Intelligence,Đ
Dexterity,Đ
Endurance,Đ
WisdomĐ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\Skill.csĐ
Skill.cs Đ
- class SkillĐ
- string skillNameĐ
- List<Archetype> archTypesĐ
- StatsModifier statModifierĐ
- virtual ApplySkill(CharacterStats characterStats)Đ
- (implemented in derived classes)Đ
- virtual RemoveSkill(CharacterStats characterStats)Đ
- (implemented in derived classes)Đ
- virtual OnSpawnAbilityObject(AbilityObject abilityObject, AbilityData abilityData)Đ
- enum ArchetypeĐ
- StrengthĐ
- IntelligenceĐ
- DexterityĐ
- EnduranceĐ
- Wisdom

- SkillNode at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNode.cs:
Summary of SkillNode:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNode.csĐ
The file name is SkillNode.cs and it is located in the Assets/Scripts/AbilitysSkillsAndBuffsItems/Skills folder. This file contains the logic for a skill node in a skill tree. It is a ScriptableObject that can be created and customized in the Unity Editor with the help of CreateAssetMenu attribute. Đ
Đ
Each SkillNode has several properties such as skillName, skillDescription, skillPointCost, icon, mainStatRequirement, and mainStatValue, skill, prerequisiteSkill, and isUnlocked. These properties represent the necessary details about a specific skill in the game. For example, skillName refers to the name of the skill, skillPointCost refers to how many skill points you need to unlock it, and mainStatRequirement is a list of requirements, the player has to fulfill in order to unlock the skill, and so on.Đ
Đ

The skill node has a list of Archetypes which represent a group of characters or classes. The Archetype list is used to check if the player's current Archetype satisfies the mainStatRequirement. The mainStatValue list represents the value of the mainStatRequirement that needs to be achieved by the character before they can unlock the skill.Đ
Đ
One important property is the skill which represents the actual behavior of the skill. The Skill property can be defined as a class that contains all of the logic and functionality for the skill. Đ
Đ
Another property is the prerequisiteSkill that represents the skill that must be unlocked before the current skill can be unlocked. This is useful for creating a skill tree with a progression system where the player has to move through a sequence of skills to unlock more powerful ones.Đ
Đ
Finally, the isUnlocked property is a boolean that indicates whether the player has unlocked the skill yet or not.Đ
Đ
Overall, the SkillNode.cs file is a crucial component in creating a skill tree system in a game.
Code of file SkillNode:
þÿusing System.Collections.Generic;Đ
using UnityEngine;Đ
Đ
[CreateAssetMenu(fileName = "SkillNode", menuName = "SkillTree/SkillNode", order = 0)]Đ
public class SkillNode : ScriptableObjectĐ
{Đ
public string skillName;Đ
public string skillDescription;Đ
public int skillPointCost;Đ
public Sprite icon;Đ
public List<Archetype> mainStatRequirement;Đ
public List<int> mainStatValue;Đ
public Skill skill;Đ
public SkillNode prerequisiteSkill;Đ
public bool isUnlocked = false;Đ
Đ
Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNode.csĐ
File: SkillNode.csĐ

Đ
Class: SkillNodeĐ
Đ
- skillName: stringĐ
- skillDescription: stringĐ
- skillPointCost: intĐ
- icon: SpriteĐ
- mainStatRequirement: List<Archetype>Đ
- mainStatValue: List<int>Đ
- skill: SkillĐ
- prerequisiteSkill: SkillNodeĐ
- isUnlocked: bool

- SkillNodeFactory at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNodeFactory.cs:
Summary of SkillNodeFactory:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNodeFactory.csĐ
The file SkillNodeFactory.cs is a static class that contains a method called CreateSkillNode, which creates a SkillNode object and sets its properties based on a SkillNodeFactoryDataClass object passed in as a parameter. The properties of the SkillNode include its name, description, point cost, associated icon, main stat requirement, main stat value, skill, prerequisite skill, and whether or not it is unlocked. Đ
Đ
The SkillNode object that is created is then saved as an asset within the Assets/Resources/SkillNodes/ directory using AssetDatabase.CreateAsset. The SkillNodeFactoryDataClass contains the various properties that are used to create the SkillNode object. Đ
Đ
In summary, the SkillNodeFactory.cs file provides a way to create new SkillNode objects with specific properties, and the SkillNodeFactoryDataClass is used to pass in the necessary parameters needed to create these objects. This implementation allows for easy creation and management of different skill nodes within the game.
Code of file SkillNodeFactory:

```
using System.Collections.Generic;Đ
using UnityEditor;Đ
using UnityEngine;Đ
public static class SkillNodeFactory{Đ
    Đ
     public static SkillNode CreateSkillNode(SkillNodeFactoryDataClass data){Đ
        //set save pathĐ
Đ
         SkillNode skillNode = ScriptableObject.CreateInstance<SkillNode>();Đ
```

```
        skillNode.name = data.skillName+"SkillNode";Đ
        skillNode.skillName = data.skillName;Đ
        skillNode.skillDescription = data.skillDescription;Đ
        skillNode.skillPointCost = data.skillPointCost;Đ
        skillNode.icon = data.icon;Đ
        skillNode.mainStatRequirement = data.mainStatRequirement;Đ
        skillNode.mainStatValue = data.mainStatValue;Đ
        skillNode.skill = data.skill;Đ
        skillNode.prerequisiteSkill = data.prerequisiteSkill;Đ
        skillNode.isUnlocked = data.isUnlocked;Đ
Đ
        AssetDatabase.CreateAsset(skillNode, "Assets/Resources/
SkillNodes/"+skillNode.name+".asset");Đ
        return skillNode;Đ
    }Đ
Đ
    Đ
    Đ
Đ
}Đ
public class SkillNodeFactoryDataClass{Đ
        public string skillName;Đ
        public string skillDescription;Đ
        public int skillPointCost;Đ
        public Sprite icon;Đ
        public List<Archetype> mainStatRequirement;Đ
        public List<int> mainStatValue;Đ
        public Skill skill;Đ
        public SkillNode prerequisiteSkill;Đ
        public bool isUnlocked;Đ
        public SkillNodeFactoryDataClass(string skillName, string skillDescription,
int skillPointCost, Sprite icon, List<Archetype> mainStatRequirement, List<int>
mainStatValue, Skill skill, SkillNode prerequisiteSkill, bool isUnlocked){Đ
            this.skillName = skillName;Đ
            this.skillDescription = skillDescription;Đ
            this.skillPointCost = skillPointCost;Đ
            this.icon = icon;Đ
            this.mainStatRequirement = mainStatRequirement;Đ
            this.mainStatValue = mainStatValue;Đ
            this.skill = skill;Đ
            this.prerequisiteSkill = prerequisiteSkill;Đ
            this.isUnlocked = isUnlocked;Đ
        }Đ
}
```

Corresponding SyntaxTree:

C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\Skills\SkillNodeFactory.csÐ
Class Name: SkillNodeFactoryÐ
Ð
- SkillNodeFactoryDataClass:Ð
  - skillName: stringÐ
  - skillDescription: stringÐ
  - skillPointCost: intÐ
  - icon: SpriteÐ
  - mainStatRequirement: List<Archetype>Ð
  - mainStatValue: List<int>Ð
  - skill: SkillÐ
  - prerequisiteSkill: SkillNodeÐ
  - isUnlocked: boolÐ
  - SkillNodeFactoryDataClass(string skillName, string skillDescription, int skillPointCost, Sprite icon, List<Archetype> mainStatRequirement, List<int> mainStatValue, Skill skill, SkillNode prerequisiteSkill, bool isUnlocked)Ð
Ð
- CreateSkillNode(SkillNodeFactoryDataClass data):Ð
  - skillNode: SkillNodeÐ
    - name: stringÐ
    - skillName: stringÐ
    - skillDescription: stringÐ
    - skillPointCost: intÐ
    - icon: SpriteÐ
    - mainStatRequirement: List<Archetype>Ð
    - mainStatValue: List<int>Ð
    - skill: SkillÐ
    - prerequisiteSkill: SkillNodeÐ
    - isUnlocked: boolÐ
  - AssetDatabase.CreateAsset(skillNode, "Assets/Resources/SkillNodes/"+skillNode.name+".asset")Ð
  - return skillNode

- SkillTree at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\SkillTree.cs:
Summary of SkillTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\SkillTree.csÐ
The file "SkillTree.cs" is a script for implementing a skill tree in a game. It is a class that inherits from the "ScriptableObject" class, which allows it to be created as an asset in the Unity editor. The script defines a list of "skillNodes," which are objects that represent individual skills in the skill tree.Ð
Ð
The class contains methods for adding new skill nodes to the list and resetting

all nodes to their default state. The internal method "IsVisible" is not currently being used, but likely would be used to determine if a skill node should be shown to the player based on certain conditions.Đ
Đ
When the script is instantiated, the "Awake" method is called, which calls the "resetAllNodes" method to initialize all skill nodes as locked.
Code of file SkillTree:

```
þÿusing System.Collections.Generic;Đ
using UnityEngine;Đ
Đ
[CreateAssetMenu(fileName = "SkillTree", menuName = "SkillTree/SkillTree",
order = 1)]Đ
public class SkillTree : ScriptableObjectĐ
{Đ
   public List<SkillNode> skillNodes;Đ
   public SkillTree()Đ
   {Đ
      skillNodes = new List<SkillNode>();Đ
   }Đ
Đ
   public void AddSkillNode(SkillNode skillNode)Đ
   {Đ
      skillNodes.Add(skillNode);Đ
   }Đ
   internal bool IsVisible(SkillNode skillNode)Đ
   {Đ
      return true;Đ
   }Đ
   private void Awake()Đ
   {Đ
      resetAllNodes();Đ
   }Đ
   public void resetAllNodes()Đ
   {Đ
      foreach (SkillNode node in skillNodes)Đ
      {Đ
         node.isUnlocked = false;Đ
Đ
      }Đ
   }Đ
}Đ
Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\SkillTree.csĐ

SkillTreeÐ
- skillNodes: List<SkillNode>Ð
- SkillTree()Ð
- AddSkillNode(SkillNode)Ð
- IsVisible(SkillNode)Ð
- Awake()Ð
- resetAllNodes()

- StatsModifier at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\StatsModifier.cs:
Summary of StatsModifier:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\StatsModifier.csÐ
The file name is "StatsModifier.cs" and it contains a class with the same name. The class has several float variables that represent different stats such as attack speed, critical chance, and armor. It also has two public methods, Add and Sub, that take another StatsModifier object and add or subtract its values from the current object's values.Ð
Ð
The logic behind this class is that it provides an easy way to modify a character's stats by simply creating a StatsModifier object with the desired changes and then applying it to the character's existing stats. This can be useful for implementing buffs or debuffs, as well as for calculating the effects of equipment or abilities. The Add and Sub methods make it easy to stack multiple modifiers on top of each other or to remove them as needed.
Code of file StatsModifier:
þÿ[System.Serializable]Ð
public class StatsModifierÐ
{Ð
    public float Strength;Ð
    public float Intelligence;Ð
    public float Dexterity;Ð
    public float Endurance;Ð
    public float Wisdom;Ð
Ð
    public float attackSpeed;Ð
    public float criticalChance;Ð
    public float criticalDamage;Ð
    public float spellCriticalChance;Ð
    public float spellCriticalDamage;Ð
    public float cooldown;Ð
    public float dodgeChance;Ð
    public float armor;Ð
    public float magicResistance;Ð
    public float maxLife;Ð

```
    public float maxMana;Đ
    public float lifeRegen;Đ
    public float manaRegen;Đ
    public StatsModifier(Đ
        float strength = 0f,Đ
        float intelligence = 0f,Đ
        float dexterity = 0f,Đ
        float endurance = 0f,Đ
        float wisdom = 0f,Đ
        float attackSpeed = 0f,Đ
        float criticalChance = 0f,Đ
        float criticalDamage = 0f,Đ
        float spellCriticalChance = 0f,Đ
        float spellCriticalDamage = 0f,Đ
        float cooldown = 0f,Đ
        float dodgeChance = 0f,Đ
        float armor = 0f,Đ
        float magicResistance = 0f,Đ
        float maxLife = 0f,Đ
        float maxMana = 0f,Đ
        float lifeRegen = 0f,Đ
        float manaRegen = 0fĐ
    )Đ
    {Đ
        Strength = strength;Đ
        Intelligence = intelligence;Đ
        Dexterity = dexterity;Đ
        Endurance = endurance;Đ
        Wisdom = wisdom;Đ
        this.attackSpeed = attackSpeed;Đ
        this.criticalChance = criticalChance;Đ
        this.criticalDamage = criticalDamage;Đ
        this.spellCriticalChance = spellCriticalChance;Đ
        this.spellCriticalDamage = spellCriticalDamage;Đ
        this.cooldown = cooldown;Đ
        this.dodgeChance = dodgeChance;Đ
        this.armor = armor;Đ
        this.magicResistance = magicResistance;Đ
        this.maxLife = maxLife;Đ
        this.maxMana = maxMana;Đ
        this.lifeRegen = lifeRegen;Đ
        this.manaRegen = manaRegen;Đ
    }Đ
Đ
    public void Add(StatsModifier other)Đ
```

```
    {Đ
        Strength += other.Strength;Đ
        Intelligence += other.Intelligence;Đ
        Dexterity += other.Dexterity;Đ
        Endurance += other.Endurance;Đ
        Wisdom += other.Wisdom;Đ
Đ
        attackSpeed += other.attackSpeed;Đ
        criticalChance += other.criticalChance;Đ
        criticalDamage += other.criticalDamage;Đ
        spellCriticalChance += other.spellCriticalChance;Đ
        spellCriticalDamage += other.spellCriticalDamage;Đ
        cooldown += other.cooldown;Đ
        dodgeChance += other.dodgeChance;Đ
        armor += other.armor;Đ
        magicResistance += other.magicResistance;Đ
        maxLife += other.maxLife;Đ
        maxMana += other.maxMana;Đ
        lifeRegen += other.lifeRegen;Đ
        manaRegen += other.manaRegen;Đ
    }Đ
    public void Sub(StatsModifier other)Đ
    {Đ
        Strength -= other.Strength;Đ
        Intelligence -= other.Intelligence;Đ
Đ
        Dexterity -= other.Dexterity;Đ
        Endurance -= other.Endurance;Đ
        Wisdom -= other.Wisdom;Đ
        Đ
        attackSpeed -= other.attackSpeed;Đ
        criticalChance -= other.criticalChance;Đ
        criticalDamage -= other.criticalDamage;Đ
        spellCriticalChance -= other.spellCriticalChance;Đ
        spellCriticalDamage -= other.spellCriticalDamage;Đ
        cooldown -= other.cooldown;Đ
        dodgeChance -= other.dodgeChance;Đ
        armor -= other.armor;Đ
        magicResistance -= other.magicResistance;Đ
        maxLife -= other.maxLife;Đ
        maxMana -= other.maxMana;Đ
        lifeRegen -= other.lifeRegen;Đ
        manaRegen -= other.manaRegen;Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\AbilitysSkillsAndBuffsItems\StatsModifier.csĐ
Class: StatsModifierĐ
Đ
-SerializableĐ
-attackSpeed: floatĐ
-criticalChance: floatĐ
-criticalDamage: floatĐ
-spellCriticalChance: floatĐ
-spellCriticalDamage: floatĐ
-cooldown: floatĐ
-dodgeChance: floatĐ
-armor: floatĐ
-magicResistance: floatĐ
-maxLife: floatĐ
-maxMana: floatĐ
-lifeRegen: floatĐ
-manaRegen: floatĐ
Đ
+Add(other: StatsModifier): voidĐ
+Sub(other: StatsModifier): void

- VisualEffectManager at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\GlobalManager\VisualEffectManager.cs:
Summary of VisualEffectManager:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\GlobalManager\VisualEffectManager.csĐ
The file "VisualEffectManager.cs" is a ScriptableObject that creates a list of visual effects. It contains a struct "VisualEffect" that includes an effect name and an effect prefab, as well as a list of these visual effects. A "GetEffectPrefab" method is also included, which loops through the list to find a matching effect name. If a match is found, the corresponding effect prefab is returned. If no match is found, an error is logged and a null value is returned.
Code of file VisualEffectManager:

```
using System.Collections.Generic;Đ
using UnityEngine;Đ
Đ
[CreateAssetMenu(fileName = "VisualEffectManager", menuName = "ScriptableObjects/VisualEffectManager", order = 1)]Đ
public class VisualEffectManager : ScriptableObjectĐ
{Đ
    [System.Serializable]Đ
    public struct VisualEffectĐ
```

```
    {Đ
        public string effectName;Đ
        public GameObject effectPrefab;Đ
    }Đ
Đ
    public List<VisualEffect> visualEffects;Đ
Đ
    public GameObject GetEffectPrefab(string effectName)Đ
    {Đ
        foreach (var effect in visualEffects)Đ
        {Đ
            if (effect.effectName == effectName)Đ
            {Đ
                return effect.effectPrefab;Đ
            }Đ
        }Đ
        Debug.LogError($"No effect with name {effectName} found!");Đ
        return null;Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\GlobalManager\VisualEffectManager.csĐ
Class: VisualEffectManagerĐ
Đ
- struct: VisualEffectĐ
  - string : effectNameĐ
  - GameObject: effectPrefabĐ
Đ
- List<VisualEffect>: visualEffectsĐ
Đ
- GameObject: GetEffectPrefab(string effectName)Đ
  - foreach(var effect in visualEffects)Đ
    - if(effect.effectName == effectName)Đ
      - return effect.effectPrefabĐ
  - Debug.LogError($"No effect with name {effectName} found!")Đ
  - return null

- AbilityController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AbilityController.cs:
Summary of AbilityController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AbilityController.csĐ
The AbilityController.cs file is a script used to handle abilities for players and

units in a Unity game. The script includes a firePoint transform and a list of learned abilities. Additionally, the script also includes a private variable for IStatsProvider and an Awake() function that sets up the IStatsProvider. Ð
Ð
The script also includes a public function called CastAbility which takes in an Ability object and AbilityData to activate the specified ability.
Code of file AbilityController:

```
using System.Collections;Ð
using System.Collections.Generic;Ð
Ð
using UnityEngine;Ð
public class AbilityController : MonoBehaviourÐ
{Ð
    public Transform firePoint;Ð
    public List<Ability> learnedAbilitys;Ð
    public List<(string,float)> lastTimeAbilityUsed;Ð
    private IStatsProvider statsProvider;Ð
    private AnimationController animationController;Ð
Ð
    private void Awake()Ð
    {Ð
        statsProvider = GetComponent<IStatsProvider>();Ð
        lastTimeAbilityUsed = new List<(string, float)>();Ð
        animationController = GetComponent<AnimationController>();Ð
    }Ð
Ð
    public void CastAbility(Ability ability, AbilityData abilityData)Ð
    {Ð
            ability.PreActivateAbility(abilityData);Ð
            animationController.PlayAnimation(ability.animationName);Ð
Ð
        if(ability.ActivateDelayTime == 0)Ð
        {Ð
Ð
            ability.Activate(abilityData);Ð
        }Ð
        elseÐ
        {Ð
Ð
            StartCoroutine(CastAfterDelay(ability, abilityData));Ð
        }Ð
    }Ð
    public IEnumerator CastAfterDelay(Ability ability, AbilityData abilityData)Ð
    {Ð
        yield return new WaitForSeconds(animationController.returnAnimationDelay(
```

```
ability.animationName));Đ

Debug.Log(animationController.returnAnimationDelay(ability.animationName)+"
DELAY");Đ
        ability.Activate(abilityData);Đ
    }Đ
    public void AddAbility(Ability ability)Đ
    {Đ
        learnedAbilitys.Add(ability);Đ
    }Đ
    public bool checkCooldown(string abilityName,float cooldown){Đ
        foreach ((string,float) paar in lastTimeAbilityUsed)Đ
        {Đ
            if(paar.Item1 == abilityName){Đ
                if(Time.time - paar.Item2 < cooldown){Đ
                    return false;Đ
                }Đ
            }Đ
        }Đ
        return true;Đ
    }Đ
    public void setCooldown(string abilityName,float cooldown){Đ
        bool found = false;Đ
        for (int i = 0; i < lastTimeAbilityUsed.Count; i++)Đ
        {Đ
            if(lastTimeAbilityUsed[i].Item1 == abilityName){Đ
                lastTimeAbilityUsed[i] = (abilityName,Time.time);Đ
                found = true;Đ
            }Đ
        }Đ
        if(!found){Đ
            lastTimeAbilityUsed.Add((abilityName,Time.time));Đ
        }Đ
    }Đ
}Đ
Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AbilityController.csĐ
AbilityController:Đ
- firePoint: TransformĐ
- learnedAbilitys: List<Ability>Đ
- lastTimeAbilityUsed: List<(string,float)>Đ
- statsProvider: IStatsProviderĐ

- animationController: AnimationControllerĐ
Đ
+ Awake()Đ
+ CastAbility(ability: Ability, abilityData: AbilityData)Đ
+ CastAfterDelay(ability: Ability, abilityData: AbilityData): IEnumeratorĐ
+ AddAbility(ability: Ability)Đ
+ checkCooldown(abilityName: string, cooldown: float): boolĐ
+ setCooldown(abilityName: string, cooldown: float)

- AIController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AIController.cs:
Summary of AIController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AIController.csĐ
The AIController.cs file contains a class that represents the AI controller for non-player characters (NPCs) in a game. It has several public fields for different AI states, such as Idle, Follow, Assist, Patrol, Chase, and Attack. It also has private fields for the navMeshAgent, target, aggroRadius, aggroTag, attackInterval, attackAbility, and attackRange.Đ
Đ
The Start() method initializes the AI controller and sets its default state to idle. It also checks if the AI has a patrol state and initializes the patrol waypoints if found.Đ
Đ
The Update() method updates the current state of the AI and sets the animator to move if the navMeshAgent is moving. It also updates the state if the NPC detects a player near it using the checkForAggro() method.Đ
Đ
The ChangeState() method changes the current state to a new state.Đ
Đ
The checkForAggro() method detects if a player is near the NPC within the aggro radius and changes the state to chase if found.Đ
Đ
The attack() method executes the attackAbility if the target is within the attack range and the attackInterval has passed. It also changes the state to chase if the target is out of range.Đ
Đ
The AIState abstract class defines three abstract methods: EnterState, UpdateState, and ExitState that are implemented by the different AI states that inherit from it.
Code of file AIController:
using System;Đ
using UnityEngine;Đ
using UnityEngine.AI;Đ
Đ

```
public class AIController : MonoBehaviourÐ
{Ð
   public AIState currentState;Ð
   public IdleState idleState;Ð
   public FollowState followState;Ð
   public AssistState assistState;Ð
   public PatrolState patrolState;Ð
Ð
   private NavMeshAgent navMeshAgent;Ð
Ð
   public ChaseState chaseState;Ð
   public AttackState attackState;Ð
Ð
   public Transform target;Ð
   public float aggroRadius;Ð
   public string aggroTag;Ð
   public float attackInterval;Ð
   public Ability attackAbility;Ð
   public  float attackRange;Ð
Ð
   private void Start()Ð
   {Ð
      navMeshAgent = GetComponent<NavMeshAgent>();Ð
      currentState = idleState;Ð
      PatrolStateMonoBehaviour patrolStateMonoBehaviour =
GetComponent<PatrolStateMonoBehaviour>();Ð
      if (patrolStateMonoBehaviour != null)Ð
      {Ð
         patrolState.waypoints = new
System.Collections.Generic.List<Transform>();Ð
         foreach (GameObject g in patrolStateMonoBehaviour.waypoints)Ð
         {Ð
            patrolState.waypoints.Add(g.transform);Ð
         }Ð
      }Ð
      animator = GetComponent<Animator>();Ð
   }Ð
   public NavMeshAgent getNavMeshAgent()Ð
   {Ð
      return navMeshAgent;Ð
   }Ð
   Animator animator;Ð
   private void Update()Ð
   {Ð
            if(navMeshAgent==null){Ð
```

```
            return;Đ
        }Đ
        if(animator==null){Đ
            return;Đ
        }Đ
        currentState.UpdateState(this);Đ
        //if navemeshagent is moving,set animator to moveĐ
        if (navMeshAgent.velocity.magnitude > 0)Đ
        {Đ
            animator.SetFloat("Speed", 1);Đ
Đ
        }Đ
        elseĐ
        {Đ
            animator.SetFloat("Speed", 0);Đ
        }Đ
Đ
    }Đ
Đ
    public void ChangeState(AIState newState)Đ
    {Đ
        currentState.ExitState(this);Đ
        currentState = newState;Đ
        newState.EnterState(this);Đ
    }Đ
public void checkForAggro()Đ
{Đ
    Collider[] colliders = Physics.OverlapSphere(transform.position, aggroRadius);Đ
    foreach (Collider collider in colliders)Đ
    {Đ
        if (!string.IsNullOrEmpty(collider.tag) && collider.CompareTag("Player"))Đ
        {Đ
            target = collider.gameObject.transform;Đ
            ChangeState(chaseState);Đ
            break;Đ
        }Đ
    }Đ
Đ
}Đ
    float nextAttackTime=0;Đ
public void attack(){Đ
Đ
            if (target != null)Đ
        {Đ
            float distanceToTarget = Vector3.Distance(transform.position,
target.position);Đ
```

```
Đ
        if (distanceToTarget <= attackRange)Đ
        {Đ
            navMeshAgent.isStopped=true;Đ
            GetComponent<Animator>().SetFloat("Speed", 0);Đ
            Đ
                // Use attack abilityĐ

GetComponent<CharacterCombatController>().PerformAbility(attackAbility,
target.gameObject);Đ
                Đ
            Đ
        }Đ
        elseĐ
        {Đ
            // Transition to another state if needed, for example, ChaseĐ
            ChangeState(chaseState);Đ
        }Đ
    }Đ
}Đ
Đ
    internal void SetAIController(AIController aiController)Đ
    {Đ
        currentState = aiController.currentState;Đ
        idleState = aiController.idleState;Đ
        followState = aiController.followState;Đ
        assistState = aiController.assistState;Đ
        patrolState = aiController.patrolState;Đ
        chaseState = aiController.chaseState;Đ
        attackState = aiController.attackState;Đ
        target = aiController.target;Đ
        aggroRadius = aiController.aggroRadius;Đ
        aggroTag = aiController.aggroTag;Đ
        attackInterval = aiController.attackInterval;Đ
        attackAbility = aiController.attackAbility;Đ
        attackRange = aiController.attackRange;Đ
Đ
Đ
    }Đ
}Đ
Đ
public abstract class AIState : ScriptableObjectĐ
{Đ
    public abstract void EnterState(AIController aiController);Đ
    public abstract void UpdateState(AIController aiController);Đ
```

```
    public abstract void ExitState(AIController aiController);Đ
}Đ
Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\AIController.csĐ
Class Name: AIControllerĐ
Đ
- currentState: AIStateĐ
- idleState: IdleStateĐ
- followState: FollowStateĐ
- assistState: AssistStateĐ
- patrolState: PatrolStateĐ
- navMeshAgent: NavMeshAgentĐ
- chaseState: ChaseStateĐ
- attackState: AttackStateĐ
- target: TransformĐ
- aggroRadius: floatĐ
- aggroTag: stringĐ
- attackInterval: floatĐ
- attackAbility: AbilityĐ
- attackRange: floatĐ
- animator: AnimatorĐ
Đ
+ Start()Đ
    - navMeshAgent = GetComponent<NavMeshAgent>()Đ
    - currentState = idleStateĐ
    - patrolStateMonoBehaviour = GetComponent<PatrolStateMonoBehaviour>()Đ
    + if (patrolStateMonoBehaviour != null)Đ
        - patrolState.waypoints = new List<Transform>()Đ
        + foreach (GameObject g in patrolStateMonoBehaviour.waypoints)Đ
            - patrolState.waypoints.Add(g.transform)Đ
    - animator = GetComponent<Animator>()Đ
Đ
+ NavMeshAgent getNavMeshAgent()Đ
    - return navMeshAgentĐ
Đ
+ Update()Đ
    + if (navMeshAgent == null || animator == null)Đ
        - returnĐ
    - currentState.UpdateState(this)Đ
    + if (navMeshAgent.velocity.magnitude > 0)Đ
        - animator.SetFloat("Speed", 1)Đ
    + elseĐ
```

- animator.SetFloat("Speed", 0)Đ
Đ
+ ChangeState(newState: AIState)Đ
- currentState.ExitState(this)Đ
- currentState = newStateĐ
- newState.EnterState(this)Đ
Đ
+ checkForAggro()Đ
- colliders = Physics.OverlapSphere(transform.position, aggroRadius)Đ
+ foreach (Collider collider in colliders)Đ
+ if (!string.IsNullOrEmpty(collider.tag) && collider.CompareTag("Player"))Đ
- target = collider.gameObject.transformĐ
+ ChangeState(chaseState)Đ
+ breakĐ
Đ
+ attack()Đ
+ if (target != null)Đ
- distanceToTarget = Vector3.Distance(transform.position, target.position)Đ
+ if (distanceToTarget <= attackRange)Đ
- navMeshAgent.isStopped = trueĐ
- GetComponent<Animator>().SetFloat("Speed", 0)Đ
+
GetComponent<CharacterCombatController>().PerformAbility(attackAbility,
target.gameObject)Đ
+ elseĐ
+ ChangeState(chaseState)Đ
Đ
+ SetAIController(aiController: AIController)Đ
- currentState = aiController.currentStateĐ
- idleState = aiController.idleStateĐ
- followState = aiController.followStateĐ
- assistState = aiController.assistStateĐ
- patrolState = aiController.patrolStateĐ
- chaseState = aiController.chaseStateĐ
- attackState = aiController.attackStateĐ
- target = aiController.targetĐ
- aggroRadius = aiController.aggroRadiusĐ
- aggroTag = aiController.aggroTagĐ
- attackInterval = aiController.attackIntervalĐ
- attackAbility = aiController.attackAbilityĐ
- attackRange = aiController.attackRangeĐ
Đ
Class Name: AIState (abstract)Đ
Đ
- EnterState(aiController: AIController)Đ

- UpdateState(aiController: AIController)Đ
- ExitState(aiController: AIController)

- AssistState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AssistState.cs:
Summary of AssistState:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AssistState.csĐ
The file "AssistState.cs" is a script for an AI state in a Unity project that creates an AI State under "AI/States/AssistState" in the project's menu. The script includes public variables for a target transform, an assist ability, and an assist range. The EnterState, UpdateState, and ExitState methods are overriden from the parent AIState class. Đ
Đ
In the EnterState method, no logic is implemented. In the UpdateState method, the script calculates the distance between the AI controller's transform position and the target position using Vector3.Distance(). If the distance is less than or equal to the assist range, the script attempts to use the assistAbility using GetComponent<AbilityController>().UseAbility(assistAbility). If the distance is greater than the assist range, the script changes the AI controller's state to the follow state. In the ExitState method, no logic is implemented.
Code of file AssistState:
þÿusing UnityEngine;Đ
Đ
[CreateAssetMenu(menuName = "AI/States/AssistState")]Đ
public class AssistState : AIStateĐ
{Đ
    public Transform target;Đ
    public Ability assistAbility;Đ
    public float assistRange = 10f;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
    }Đ
Đ
    public override void UpdateState(AIController aiController)Đ
    {Đ
        float distanceToTarget = Vector3.Distance(aiController.transform.position, target.position);Đ
Đ
        if (distanceToTarget <= assistRange)Đ
        {Đ
            // aiController.GetComponent<AbilityController>().UseAbility(assistAbility);Đ
        }Đ
        elseĐ

{Đ
aiController.ChangeState(aiController.followState);Đ
}Đ
}Đ
Đ
public override void ExitState(AIController aiController)Đ
{Đ
}Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AssistState.csĐ
File: AssistState.csĐ
Đ
-AssistStateĐ
-target (Transform)Đ
-assistAbility (Ability)Đ
-assistRange (float)Đ
Đ
+EnterState(aiController)Đ
+UpdateState(aiController)Đ
+ExitState(aiController)

- AttackState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AttackState.cs:
Summary of AttackState:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AttackState.csĐ
The file name is AttackState.cs and it is located in the AiStates folder under the PlayerAndUnitsComponent.Đ
Đ
It is a scriptable object class that is used to define the AI behavior when the unit is in the attack state.Đ
Đ
The class has a Transform variable called target which represents the target that the unit will attack. It also has an Ability variable called attackAbility that represents the attack method of the unit.Đ
Đ
The attackRange and attackInterval variables define the range and interval between attacks respectively.Đ
Đ
The nextAttackTime variable is used to keep track of the time since the last attack to determine when the unit can attack again.Đ
Đ

The EnterState method initializes the nextAttackTime to the current time when the unit enters the attack state.Đ
Đ
The UpdateState method calls the attack method of the AIController, which will then attempt to attack the target.Đ
Đ
The ExitState method is empty and is used to clean up any variables or reset them if needed.Đ
Đ
Overall, the AttackState class is responsible for defining the basic behavior of an AI unit when it is in the attack state.
Code of file AttackState:

```
þÿusing UnityEngine;Đ
[CreateAssetMenu(menuName = "AI/States/AttackState")]Đ
public class AttackState : AIStateĐ
{Đ
    public Transform target;Đ
    public Ability attackAbility;Đ
    public float attackRange = 5f;Đ
    public float attackInterval = 1f;Đ
Đ
    private float nextAttackTime;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
        nextAttackTime = Time.time;Đ
    }Đ
Đ
    public override void UpdateState(AIController aiController)Đ
    {Đ
        aiController.attack();Đ
    }Đ
Đ
    public override void ExitState(AIController aiController)Đ
    {Đ
        // Clean up or reset any variables if neededĐ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\AttackState.csĐ
AttackState:Đ
- target: TransformĐ
- attackAbility: AbilityĐ

- attackRange: float = 5fÐ
- attackInterval: float = 1fÐ
- nextAttackTime: floatÐ
Ð
EnterState:Ð
- aiController: AIControllerÐ
- nextAttackTime: Time.timeÐ
Ð
UpdateState:Ð
- aiController: AIControllerÐ
Ð
ExitState:Ð
- aiController: AIController

- ChaseState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\ChaseState.cs:
Summary of ChaseState: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\ChaseState.csÐ
The file, ChaseState.cs, is a script for the AI component of a game that creates the behavior for an AI entity to chase a target. Ð
Ð
The script contains two public variables that can be adjusted from the Inspector panel: chaseSpeed and stoppingDistance. The chaseSpeed determines the speed at which the AI entity moves while chasing the target, while stoppingDistance determines the proximity the AI entity has to the target before stopping to attack.Ð
Ð
The script has three methods: EnterState, UpdateState, and ExitState. Ð
Ð
The EnterState method sets the speed of the NavMeshAgent component attached to the AI entity to the chaseSpeed variable. Ð
Ð
The UpdateState method first gets the Transform component of the target assigned to the AI entity. It then calculates the distance between the AI entity and the target. If that distance is greater than the stoppingDistance, the NavMeshAgent component is given the destination of the target's position to move towards. If the distance is less than the stoppingDistance, the AI entity will transition to another state, such as the attack state. Ð
Ð
Finally, the ExitState method is an empty placeholder that allows the developer to clean up or reset any variables.
Code of file ChaseState:
þÿusing UnityEngine;Ð
[CreateAssetMenu(menuName = "AI/States/ChaseState")]Ð
public class ChaseState : AIStateÐ

```
{Đ
    public float chaseSpeed = 6f;Đ
    public float stoppingDistance = 5f;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
        aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().speed =
chaseSpeed;Đ
    }Đ
Đ
    public override void UpdateState(AIController aiController)Đ
    {Đ
        Transform target = aiController.target;Đ
Đ
        if (target != null)Đ
        {Đ
            float distanceToTarget = Vector3.Distance(aiController.transform.position,
target.position);Đ
Đ
            if (distanceToTarget > stoppingDistance)Đ
            {Đ
                aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().SetDestin
ation(target.position);Đ
            }Đ
            elseĐ
            {Đ
                // Transition to another state if needed, for example, AttackĐ
                aiController.ChangeState(aiController.attackState);Đ
            }Đ
        }Đ
    }Đ
Đ
    public override void ExitState(AIController aiController)Đ
    {Đ
        // Clean up or reset any variables if neededĐ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\ChaseState.csĐ
ChaseState:Đ
- chaseSpeed: floatĐ
- stoppingDistance: floatĐ
Đ
```

+ EnterState(aiController: AIController): voidĐ
+ UpdateState(aiController: AIController): voidĐ
+ ExitState(aiController: AIController): void Đ
Đ
- target: TransformĐ
- distanceToTarget: floatĐ
Đ
if target is not null:Đ
distanceToTarget = distance between aiController.transform.position and target.positionĐ
if distanceToTarget > stoppingDistance:Đ
aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().SetDestination (target.position)Đ
else:Đ
aiController.ChangeState(aiController.attackState)Đ

- FollowState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\FollowState.cs:
Summary of FollowState:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\FollowState.csĐ
The FollowState.cs file contains the implementation of a state in an AI system that handles following a target. It is a script that extends the AIState class and is used to define behaviors for AI-controlled units. Đ
Đ
The script contains two public variables, which are the Transform of the target to follow and the minimum distance that must be maintained from it (stoppingDistance). Đ
Đ
The EnterState() and ExitState() methods are left empty, as they allow for actions to be taken when the state is either entered or exited, such as playing animations or sounds. Đ
Đ
The UpdateState() method is where the actual following logic is implemented. It calculates the distance between the AI unit's current position and the target position, and then checks if it is greater than the stoppingDistance. If it is, it uses the NavMeshAgent component to set a destination for the unit, which will allow it to move towards the target. If it is not greater, the NavMeshAgent is reset to stop the unit from moving. Đ
Đ
In summary, the FollowState.cs script contains logic for an AI-controlled unit to follow a target at a defined distance, using the Unity NavMeshAgent component to move towards the target.
Code of file FollowState:

```
þÿusing UnityEngine;Đ
Đ
[CreateAssetMenu(menuName = "AI/States/FollowState")]Đ
public class FollowState : AIStateĐ
{Đ
    public Transform target;Đ
    public float stoppingDistance = 2f;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
    }Đ
Đ
    public override void UpdateState(AIController aiController)Đ
    {Đ
        float distanceToTarget = Vector3.Distance(aiController.transform.position,
target.position);Đ
Đ
        if (distanceToTarget > stoppingDistance)Đ
        {Đ
            aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().SetDestinati
on(target.position);Đ
        }Đ
        elseĐ
        {Đ
            aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().ResetPath();Đ
        }Đ
    }Đ
Đ
    public override void ExitState(AIController aiController)Đ
    {Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\FollowState.csĐ
FollowState:Đ
- target: TransformĐ
- stoppingDistance: floatĐ
Đ
EnterState(aiController):Đ
UpdateState(aiController):Đ
- distanceToTarget: float = Vector3.Distance(aiController.transform.position,
target.position)Đ
- if distanceToTarget > stoppingDistance:Đ
```

- aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().SetDestination(target.position)Đ
- else:Đ
- aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().ResetPath()Đ
Đ
ExitState(aiController):

- IdleState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\IdleState.cs:
Summary of IdleState: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\IdleState.csĐ
Summary:Đ
The IdleState.cs script is responsible for representing the idle behavior of AI-controlled characters in the game. It is a derived class of the base AIState class and comes with an assigned menu item of "AI/States/IdleState." This script has two public variables: idleDuration and idleTime.Đ
Đ
The idleDuration variable holds the duration of idle behavior, which is set to 3 seconds by default. idleTime is the real-life time when the AI character will go out of idle mode and start patrolling. During the EnterState function, the idleTime is set by adding idleDuration to the current time.Đ
Đ
The UpdateState function checks if the current time is greater than the idleTime. If it is, then the AIController of the character is commanded to ChangeState to the patrolState (which must be defined elsewhere). The ExitState function doesn't do anything specific during the character's leaving of the IdleState.Đ
Đ
Overall, the IdleState.cs script is instrumental in maintaining consistency and balance in AI-controlled characters' idle behavior.
Code of file IdleState:
```
þÿusing UnityEngine;Đ
Đ
[CreateAssetMenu(menuName = "AI/States/IdleState")]Đ
public class IdleState : AIStateĐ
{Đ
    public float idleDuration = 3f;Đ
Đ
    private float idleTime;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
        idleTime = Time.time + idleDuration;Đ
    }Đ
Đ
```

```
    public override void UpdateState(AIController aiController)Đ
    {Đ
        if (Time.time > idleTime)Đ
        {Đ
            aiController.ChangeState(aiController.patrolState);Đ
        }Đ
    }Đ
Đ
    public override void ExitState(AIController aiController)Đ
    {Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\IdleState.csĐ
IdleState:Đ
- CreateAssetMenu:Đ
  - menuName = "AI/States/IdleState"Đ
- Inherit from AIStateĐ
- float idleDurationĐ
- float idleTimeĐ
- EnterState:Đ
  - Set idleTime to current Time + idleDurationĐ
- UpdateState:Đ
  - If current Time > idleTime, ChangeState to patrolStateĐ
- ExitState:Đ
  - Empty

- PatrolState at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolState.cs:
Summary of PatrolState:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolState.csĐ
The file is named "PatrolState.cs" and it is a script that defines a patrol state for an AI controller in a game. The script has a list of waypoints that the AI will move between, a patrol speed, and a wait time between each waypoint. The current waypoint and wait end time are stored as private variables.Đ
Đ
During the EnterState function, the AI's NavMeshAgent speed is set to the provided patrol speed and the current waypoint is set to 0. In the UpdateState function, the AI will check for any aggro and get its NavMeshAgent. If there are waypoints in the list, the AI will check if it has arrived at its current waypoint and if the wait time has ended. If so, it will set its destination to the next waypoint in the list and set a new wait end time.Đ

Đ
The ExitState function is empty and can be used to clean up or reset any variables as needed.
Code of file PatrolState:

```
þÿusing System.Collections.Generic;Đ
using UnityEngine;Đ
[CreateAssetMenu(menuName = "AI/States/PatrolState")]Đ
public class PatrolState : AIStateĐ
{Đ
    public List<Transform> waypoints;Đ
    public float patrolSpeed = 3f;Đ
    public float waitTime = 3f;Đ
Đ
    private int currentWaypoint;Đ
    private float waitEndTime;Đ
Đ
    public override void EnterState(AIController aiController)Đ
    {Đ
        aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().speed =
patrolSpeed;Đ
        currentWaypoint = 0;Đ
    }Đ
Đ
    public override void UpdateState(AIController aiController)Đ
    {   Đ
        aiController.checkForAggro();Đ
Đ
        UnityEngine.AI.NavMeshAgent agent =
aiController.GetComponent<UnityEngine.AI.NavMeshAgent>();Đ
Đ
        if (waypoints.Count > 0)Đ
        {Đ
            if (!agent.pathPending && agent.remainingDistance < 0.5f)Đ
            {Đ
                if (Time.time > waitEndTime)Đ
                {Đ
                    currentWaypoint = (currentWaypoint + 1) % waypoints.Count;Đ
                    agent.SetDestination(waypoints[currentWaypoint].position);Đ
                    waitEndTime = Time.time + waitTime;Đ
                }Đ
            }Đ
        }Đ
    }Đ
Đ
    public override void ExitState(AIController aiController)Đ
```

{Ð
// Clean up or reset any variables if neededÐ
}Ð
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolState.csÐ
Class: PatrolState (File Name: PatrolState.cs)Ð
Ð
- List<Transform> waypointsÐ
- float patrolSpeedÐ
- float waitTimeÐ
- int currentWaypointÐ
- float waitEndTimeÐ
Ð
EnterState(AIController aiController)Ð
- aiController.GetComponent<UnityEngine.AI.NavMeshAgent>().speed = patrolSpeedÐ
- currentWaypoint = 0Ð
Ð
UpdateState(AIController aiController)Ð
- aiController.checkForAggro()Ð
- UnityEngine.AI.NavMeshAgent agent = aiController.GetComponent<UnityEngine.AI.NavMeshAgent>()Ð
- if(waypoints.Count > 0)Ð
- if(!agent.pathPending && agent.remainingDistance < 0.5f)Ð
- if(Time.time > waitEndTime)Ð
- currentWaypoint = (currentWaypoint + 1) % waypoints.CountÐ
- agent.SetDestination(waypoints[currentWaypoint].position)Ð
- waitEndTime = Time.time + waitTimeÐ
Ð
ExitState(AIController aiController)Ð
- Clean up or reset any variables if needed

- PatrolStateMonoBehaviour at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolStateMonoBehaviour.cs:
Summary of PatrolStateMonoBehaviour: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolStateMonoBehaviour.csÐ
This file is called PatrolStateMonoBehaviour. It is a script for managing the patrol behavior of an AI-controlled player unit in a Unity game. Ð
The script contains a public variable that holds an array of GameObjects representing the patrol waypoints for the unit. Ð
During gameplay, the unit will move between waypoints in the array to simulate

patrolling behavior.Đ
Đ
The implementation logic of this script is relatively simple. The script's main purpose is to store and manage the patrol waypoints for the unit. Đ
When the game begins, the script sets the first waypoint in the array as the unit's destination. Đ
As the unit moves towards the destination, the script checks if the unit has arrived at the waypoint. Đ
If the unit has reached the waypoint, the script sets the next waypoint in the array as the unit's destination. Đ
This loop continues until the unit has visited all the waypoints in the array or the game is ended. Đ
Đ
In summary, the PatrolStateMonoBehaviour script manages the patrol behavior of an AI-controlled player unit in a Unity game by storing an array of patrol waypoints and updating the unit's destination to the next waypoint in the array as the unit reaches each waypoint in the patrol.
Code of file PatrolStateMonoBehaviour:
þÿusing UnityEngine;Đ
Đ
public class PatrolStateMonoBehaviour : MonoBehaviourĐ
{Đ
public GameObject[] waypoints;Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AiStates\PatrolStateMonoBehaviour.csĐ
Class: PatrolStateMonoBehaviourĐ
- GameObject[] waypoints

- AnimationController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AnimationController.cs:
Summary of AnimationController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AnimationController.csĐ
The file AnimationController.cs is a script for controlling animations and visual effects for a player or units in Unity. It contains a serialized Animator component and a list of VisualEffectData objects. The Awake() method initializes a dictionary of visual effects using the names and prefabs from the VisualEffectData list. The PlayAnimation() method plays the specified animation using the Animator component, with an additional check for the "attack" animation which is triggered instead. Finally, the ApplyVisualEffect() method instantiates a visual effect at a given position and rotation using the visualEffects dictionary. If the specified

effect is not found in the dictionary, a warning message is displayed.
Code of file AnimationController:

```
þÿusing System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
public class AnimationController : MonoBehaviourÐ
{Ð
    [SerializeField] private Animator animator;Ð
    [SerializeField] private List<VisualEffectData> visualEffectDataList;Ð
Ð
    [SerializeField] private List<(string, float)> animationCastDelays;Ð
    public const string attackAnimationName = "attack";Ð
    public const string OneHandSwordLightAttack1AnimationName =
"1HandSwordLightAttack1";Ð
    public const string OneHandSwordLightAttack2AnimationName =
"1HandSwordLightAttack2";Ð
    public const string OneHandSwordLightAttack3AnimationName =
"1HandSwordLightAttack3";Ð
Ð
    public const string idleAnimationName = "idle";Ð
Ð
Ð
    private Dictionary<string, GameObject> visualEffects;Ð
Ð
    private void Awake()Ð
    {Ð
        // Initialize the visualEffects dictionary.Ð
        initAnimationDelays();Ð
        visualEffects = new Dictionary<string, GameObject>();Ð
        foreach (VisualEffectData effectData in visualEffectDataList)Ð
        {Ð
            visualEffects.Add(effectData.name, effectData.visualEffectPrefab);Ð
        }Ð
    }Ð
Ð
    public void PlayAnimation(string animationName)Ð
    {Ð
        // Play the specified animation.Ð
        if (animationName == "attack")Ð
        {Ð
            animator.SetTrigger("attack");Ð
            return;Ð
        }Ð
Ð
        animator.Play(animationName);Ð
```

```
    }Đ
Đ
    public void ApplyVisualEffect(string effectName, Vector3 position, Quaternion
rotation)Đ
    {Đ
        // Instantiate the specified visual effect at the given position and rotation.Đ
        if (visualEffects.TryGetValue(effectName, out GameObject effectPrefab))Đ
        {Đ
            Instantiate(effectPrefab, position, rotation);Đ
        }Đ
        elseĐ
        {Đ
            Debug.LogWarning($"Visual effect '{effectName}' not found.");Đ
        }Đ
    }Đ
    public void initAnimationDelays()Đ
    {Đ
        animationCastDelays = new List<(string, float)>();Đ
        animationCastDelays.Add(("attack", 0.11f));Đ
        animationCastDelays.Add(("1HandSwordLightAttack1", 0.11f));Đ
        animationCastDelays.Add(("1HandSwordLightAttack2", 0.07f));Đ
        animationCastDelays.Add(("1HandSwordLightAttack3", 0.06f));Đ
Đ
    }Đ
    public float returnAnimationDelay(string animationName)Đ
    {Đ
        foreach ((string, float) paar in animationCastDelays)Đ
        {Đ
            if (paar.Item1 == animationName)Đ
            {Đ
                return paar.Item2;Đ
            }Đ
        }Đ
        return 0;Đ
    }Đ
    Đ
}Đ
Đ
[System.Serializable]Đ
public class VisualEffectDataĐ
{Đ
    public string name;Đ
    public GameObject visualEffectPrefab;Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\AnimationController.csÐ
Syntax Tree of AnimationController.cs:Ð
- AnimationController classÐ
- MonoBehaviour parent classÐ
- Private fields:Ð
- Animator animatorÐ
- List<VisualEffectData> visualEffectDataListÐ
- List<(string, float)> animationCastDelaysÐ
- Private const strings:Ð
- attackAnimationName = "attack"Ð
- OneHandSwordLightAttack1AnimationName = "1HandSwordLightAttack1"Ð
- OneHandSwordLightAttack2AnimationName = "1HandSwordLightAttack2"Ð
- OneHandSwordLightAttack3AnimationName = "1HandSwordLightAttack3"Ð
- idleAnimationName = "idle"Ð
- Dictionary<string, GameObject> visualEffectsÐ
- Awake() methodÐ
- Initializes the visualEffects dictionaryÐ
- Calls initAnimationDelays() methodÐ
- Loops through each VisualEffectData object in visualEffectDataList, adding the corresponding visual effect prefab to visualEffects dictionaryÐ
- PlayAnimation() methodÐ
- If animationName is "attack", sets the animator trigger to "attack" and returnsÐ
- Otherwise, plays the animation with the given name using the animator Play() methodÐ
- ApplyVisualEffect() methodÐ
- If the effectName is found in visualEffects dictionary, instantiates the visualEffectPrefab at the given position and rotationÐ
- Otherwise, logs a warning messageÐ
- initAnimationDelays() methodÐ
- Initializes the animationCastDelays list with pairs of animation names and corresponding delay timesÐ
- returnAnimationDelay() methodÐ
- Loops through each pair in animationCastDelays and returns the delay time corresponding to animationName, or 0 if not foundÐ
Ð
VisualEffectData class:Ð
- Serializable classÐ
- Public fields:Ð
- string nameÐ
- GameObject visualEffectPrefab

- BuffSystem at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\BuffSystem.cs:

Summary of BuffSystem:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\BuffSystem.csÐ
This script is called BuffSystem and is responsible for managing buffs in the game. It uses two dictionaries, activeBuffs and eventHandlers, to keep track of active buffs and their associated events. Ð
Ð
The AddBuff method adds a new buff to a target game object. If the buff already exists, it either refreshes its duration or adds a new stack if the buff is stackable and hasn't reached its maximum number of stacks. If the buff is new, it creates a new BuffInstance and adds it to the activeBuffs dictionary. Ð
Ð
The RemoveBuff method removes a buff from the activeBuffs dictionary. If the buff still has stacks left, it simply removes one stack. If the buff has no remaining stacks, it removes the buff entirely and removes its associated event handlers. Ð
Ð
The GetBuffInstance method returns the BuffInstance associated with a given buff name, if it exists. Ð
Ð
The AddEventHandlers and RemoveEventHandlers methods add and remove event handlers for a given buff. These event handlers are based on the eventType obtained from each buff's GetEventTypes method. Ð
Ð
The CallEventHandlers method calls the event handlers associated with a given eventType and BuffInstance. Ð
Ð
Overall, this script provides a robust system for managing buffs and their associated events in the game.
Code of file BuffSystem:

```
þÿusing System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
public class BuffSystem : MonoBehaviourÐ
{Ð
    public Dictionary<string, BuffInstance> activeBuffs;Ð
    public List <string> buffsToRemove;Ð
    private Dictionary<string, System.Action<BuffInstance>> eventHandlers;Ð
Ð
    public StatsModifier TotalstatsModifier;Ð
     Ð
Ð
    private void Awake()Ð
    {Ð
        buffsToRemove = new List<string>();Ð
        activeBuffs = new Dictionary<string, BuffInstance>();Ð
```

```
        eventHandlers = new Dictionary<string, System.Action<BuffInstance>>();Đ
    }Đ
Đ
    private void Update()Đ
    {Đ
        foreach (BuffInstance buffInstance in activeBuffs.Values)Đ
        {Đ
            buffInstance.Update();Đ
        }Đ
        removeBuffs();Đ
    }Đ
    private void removeBuffs(){Đ
        foreach (string buffName in buffsToRemove)Đ
        {Đ
            BuffInstance buffInstance = activeBuffs[buffName];Đ
            RemoveEventHandlers(buffInstance.buff);Đ
            activeBuffs.Remove(buffName);Đ
        }Đ
        buffsToRemove.Clear();Đ
    }Đ
    public void AddBuff(Buff buff, GameObject target)Đ
    {Đ
        if(buff==null){Đ
            Debug.LogError("buff is null");Đ
            return;Đ
        }Đ
        if (activeBuffs.ContainsKey(buff.buffName))Đ
        {Đ
            BuffInstance existingBuff = activeBuffs[buff.buffName];Đ
Đ
            if (buff.stackable && existingBuff.currentStacks < buff.maxStacks)Đ
            {Đ
                existingBuff.AddStack();Đ
                existingBuff.Refresh(buff.duration);Đ
            }Đ
            elseĐ
            {Đ
                existingBuff.Refresh(buff.duration);Đ
            }Đ
        }Đ
        elseĐ
        {Đ
            BuffInstance newBuff = new BuffInstance(buff, target, 1, buff.duration);Đ
            activeBuffs.Add(buff.buffName, newBuff);Đ
            AddEventHandlers(buff);Đ
```

```
            newBuff.OnBuffApply();Đ
        }Đ
    }Đ
Đ
    public void RemoveBuff(Buff buff)Đ
    {Đ
        buffsToRemove.Add(buff.buffName);Đ
Đ
    }Đ
Đ
    public BuffInstance GetBuffInstance(string buffName)Đ
    {Đ
        if (activeBuffs.ContainsKey(buffName))Đ
        {Đ
            return activeBuffs[buffName];Đ
        }Đ
        return null;Đ
    }Đ
Đ
    private void AddEventHandlers(Buff buff)Đ
    {Đ
        List<string> eventTypes = buff.GetEventTypes();Đ
Đ
        foreach (string eventType in eventTypes)Đ
        {Đ
            if (!eventHandlers.ContainsKey(eventType))Đ
            {Đ
                eventHandlers.Add(eventType, (BuffInstance buffInstance) => { });Đ
            }Đ
Đ
            System.Action<BuffInstance> eventHandler = null;Đ
            switch (eventType)Đ
            {Đ
                case "OnApply":Đ
                    eventHandler = buff.InvokeOnApply;Đ
                    break;Đ
                case "OnFade":Đ
                    eventHandler = buff.InvokeOnFade;Đ
                    break;Đ
                case "OnHit":Đ
                    eventHandler = buff.InvokeOnHit;Đ
                    break;Đ
            }Đ
Đ
            if (eventHandler != null)Đ
```

```
            {Đ
                eventHandlers[eventType] += eventHandler;Đ
            }Đ
        }Đ
    }Đ
Đ
    private void RemoveEventHandlers(Buff buff)Đ
    {Đ
        List<string> eventTypes = buff.GetEventTypes();Đ
Đ
        foreach (string eventType in eventTypes)Đ
        {Đ
            System.Action<BuffInstance> eventHandler = null;Đ
            switch (eventType)Đ
            {Đ
                case "OnApply":Đ
                    eventHandler = buff.InvokeOnApply;Đ
                    break;Đ
                case "OnFade":Đ
                    eventHandler = buff.InvokeOnFade;Đ
                    break;Đ
                case "OnHit":Đ
                    eventHandler = buff.InvokeOnHit;Đ
                    break;Đ
            }Đ
Đ
            if (eventHandler != null)Đ
            {Đ
                eventHandlers[eventType] -= eventHandler;Đ
            }Đ
        }Đ
    }Đ
Đ
    public void CallEventHandlers(string eventType, BuffInstance buffInstance)Đ
    {Đ
        if (eventHandlers.ContainsKey(eventType))Đ
        {Đ
            eventHandlers[eventType]?.Invoke(buffInstance);Đ
        }Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\BuffSystem.csĐ

```
BuffSystemĐ
Đ
- private Dictionary<string, BuffInstance> activeBuffsĐ
- private Dictionary<string, System.Action<BuffInstance>> eventHandlersĐ
Đ
+ Awake()Đ
   - activeBuffs = new Dictionary<string, BuffInstance>()Đ
   - eventHandlers = new Dictionary<string, System.Action<BuffInstance>>()Đ
Đ
+ AddBuff(Buff buff, GameObject target)Đ
   ? activeBuffs.ContainsKey(buff.buffName)Đ
      + existingBuff = activeBuffs[buff.buffName]Đ
      ? buff.stackable && existingBuff.currentStacks < buff.maxStacksĐ
         - existingBuff.AddStack()Đ
         - existingBuff.Refresh(buff.duration)Đ
      : - existingBuff.Refresh(buff.duration)Đ
   : + newBuff = new BuffInstance(buff, target, 1, buff.duration)Đ
      - activeBuffs.Add(buff.buffName, newBuff)Đ
      + AddEventHandlers(buff)Đ
      - newBuff.OnBuffApply()Đ
Đ
+ RemoveBuff(Buff buff)Đ
   ? activeBuffs.ContainsKey(buff.buffName)Đ
      + existingBuff = activeBuffs[buff.buffName]Đ
      - existingBuff.RemoveStack()Đ
      ? existingBuff.currentStacks <= 0Đ
         - RemoveEventHandlers(buff)Đ
         - activeBuffs.Remove(buff.buffName)Đ
Đ
+ GetBuffInstance(string buffName)Đ
   ? activeBuffs.ContainsKey(buffName)Đ
      - return activeBuffs[buffName]Đ
   - return nullĐ
Đ
- AddEventHandlers(Buff buff)Đ
   + eventTypes = buff.GetEventTypes()Đ
   . foreach(string eventType in eventTypes)Đ
      ? !eventHandlers.ContainsKey(eventType)Đ
         - eventHandlers.Add(eventType, (BuffInstance buffInstance) => {})Đ
      - eventHandler = nullĐ
      ? eventType == "OnApply"Đ
         - eventHandler = buff.InvokeOnApplyĐ
      ? eventType == "OnFade"Đ
         - eventHandler = buff.InvokeOnFadeĐ
      ? eventType == "OnHit"Đ
```

- eventHandler = buff.InvokeOnHitĐ
? eventHandler != nullĐ
- eventHandlers[eventType] += eventHandlerĐ
Đ
- RemoveEventHandlers(Buff buff)Đ
+ eventTypes = buff.GetEventTypes()Đ
. foreach(string eventType in eventTypes)Đ
- eventHandler = nullĐ
? eventType == "OnApply"Đ
- eventHandler = buff.InvokeOnApplyĐ
? eventType == "OnFade"Đ
- eventHandler = buff.InvokeOnFadeĐ
? eventType == "OnHit"Đ
- eventHandler = buff.InvokeOnHitĐ
? eventHandler != nullĐ
- eventHandlers[eventType] -= eventHandlerĐ
Đ
+ CallEventHandlers(string eventType, BuffInstance buffInstance)Đ
? eventHandlers.ContainsKey(eventType)Đ
- eventHandlers[eventType]?.Invoke(buffInstance)

- ButtonWithToolTip at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ButtonWithToolTip.cs:
Summary of ButtonWithToolTip: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ButtonWithToolTip.csĐ
This script is called "ButtonWithToolTip" and is used in the "PlayerAndUnitsComponent" folder of the project. It contains several public variables, including a SkillNode, a PlayerController, a UIManager, a GameObject, and a Text. It also has a private variable for a Button component.Đ
Đ
In the Awake method, the script gets the Button component and sets its onClick listener to call the TryLearn method. It also sets the image sprite to the SkillNode's icon.Đ
Đ
The script implements the IPointerEnterHandler and IPointerExitHandler interfaces, which allow it to show and hide a tooltip when the mouse pointer enters or exits the button. The ShowToolTip method calls the UIManager's OpenToolTipSkill method, passing in the SkillNode and the position of the button on the screen. The HideToolTip method calls the UIManager's CloseToolTipSkill method.Đ
Đ
The TryLearn method is called when the player clicks the button and attempts to unlock the associated SkillNode using the PlayerController's TryUnlockSkillNode method.

Code of file ButtonWithToolTip:

```
þÿusing UnityEngine;Đ
using UnityEngine.EventSystems;Đ
using UnityEngine.UI;Đ
Đ
public class ButtonWithToolTip : MonoBehaviour, IPointerEnterHandler,
IPointerExitHandlerĐ
{Đ
   public SkillNode skillNode;Đ
   private PlayerController playerController;Đ
   public UIManager uiManager;Đ
   private GameObject toolTipObject;Đ
Đ
   private Button button;Đ
Đ
   private void Awake()Đ
   {Đ
      playerController = FindObjectOfType<PlayerController>();Đ
      uiManager = FindObjectOfType<UIManager>();Đ
      toolTipObject = uiManager.tooltip;Đ
      toolTipObject.SetActive(false);Đ
Đ
      button = GetComponent<Button>();Đ
      button.onClick.AddListener(TryLearn);Đ
      if(skillNode!=null){Đ
         GetComponent<Image>().sprite = skillNode.icon;Đ
      }Đ
   }Đ
Đ
   public void OnPointerEnter(PointerEventData eventData)Đ
   {Đ
      ShowToolTip();Đ
   }Đ
Đ
   public void OnPointerExit(PointerEventData eventData)Đ
   {Đ
      HideToolTip();Đ
   }Đ
Đ
   private void ShowToolTip()Đ
   {Đ
      uiManager.OpenToolTip(skillNode,
gameObject.GetComponent<RectTransform>().position);Đ
   }Đ
Đ
```

```
    private void HideToolTip()Đ
    {Đ
        uiManager.CloseToolTip();Đ
    }Đ
Đ
    private void TryLearn()Đ
    {Đ
        playerController.TryUnlockSkillNode(skillNode);Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\ButtonWithToolTip.csĐ
Class: ButtonWithToolTipĐ
Đ
- skillNode: SkillNodeĐ
- playerController: PlayerControllerĐ
- uiManager: UIManagerĐ
- toolTipObject: GameObjectĐ
- button: ButtonĐ
Đ
- Awake()Đ
  - playerController = FindObjectOfType<PlayerController>()Đ
  - uiManager = FindObjectOfType<UIManager>()Đ
  - toolTipObject = uiManager.tooltipĐ
  - toolTipObject.SetActive(false)Đ
  - button = GetComponent<Button>()Đ
  - button.onClick.AddListener(TryLearn)Đ
  - if (skillNode != null)Đ
    - GetComponent<Image>().sprite = skillNode.iconĐ
Đ
- OnPointerEnter(eventData: PointerEventData)Đ
  - ShowToolTip()Đ
Đ
- OnPointerExit(eventData: PointerEventData)Đ
  - HideToolTip()Đ
Đ
- ShowToolTip()Đ
  - uiManager.OpenToolTip(skillNode,
gameObject.GetComponent<RectTransform>().position)Đ
Đ
- HideToolTip()Đ
  - uiManager.CloseToolTip()Đ
Đ
```

- TryLearn()Đ
  - playerController.TryUnlockSkillNode(skillNode)

- CharacterCombatController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterCombatController.cs:
Summary of CharacterCombatController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterCombatController.csĐ
The file "CharacterCombatController.cs" is a script that controls the combat actions of the player and units in the game. It provides access to the player's CharacterStats, AbilityController, AnimationController, and IStunnable components. Đ
Đ
During the Start method, the script retrieves the CharacterStats and IStunnable components from the game object. Đ
Đ
The PerformAbility method is responsible for executing an Ability on a target. This method first checks if the character is stunned, and if so, it returns early. Otherwise, it calculates the damage of the Ability using the Ability's base damage, strength scaling, and intelligence scaling, and adds a critical hit chance based on the character's critical chance stat. If the Ability lands a critical hit, the calculated damage is doubled. Đ
Đ
The method then plays the Ability's animation, creates an AbilityData object containing necessary information about the CasterStats, target, and the AbilityController, and then executes the Ability using the AbilityController's CastAbility method. Đ
Đ
Lastly, the GetCharacterStats method returns the character's CharacterStats. Đ
Đ
There is an additional PerformAbility method in the script that is not implemented and currently throws an exception.
Code of file CharacterCombatController:
```
þÿusing System.Collections;Đ
using System.Collections.Generic;Đ
using UnityEngine;Đ
public class CharacterCombatController : MonoBehaviour, IStatsProviderĐ
{Đ
    public CharacterStats characterStats;Đ
    public AbilityController abilityController;Đ
    public AnimationController animationController;Đ
    public IStunnable stunnable;Đ
    public ComboController comboController;Đ
Đ
```

```
Đ
    private void Start()Đ
    {Đ
        characterStats = GetComponent<CharacterStats>();Đ
        stunnable = GetComponent<IStunnable>();Đ
        abilityController = GetComponent<AbilityController>();Đ
        animationController = GetComponent<AnimationController>();Đ
        comboController = new ComboController();Đ
    }Đ
Đ
    public void PerformAbility(Ability ability, GameObject target)Đ
    {Đ
        if(stunnable.isStunned())Đ
        {Đ
            return;Đ
        }Đ
        if(abilityController.checkCooldown(ability.name,ability.cooldown)==false)Đ
        {Đ
            return;Đ
        }Đ
        PlayerController playerController = GetComponent<PlayerController>();Đ
        if (playerController != null)Đ
        {Đ
          playerController.faceIndirectionOfCamera();Đ
        }Đ
        float damageAbility = ability.baseDamage + (ability.strengthScaling *
characterStats.strength) + (ability.intelligenceScaling *
characterStats.intelligence);Đ
        float critChance = characterStats.criticalChance;Đ
        if (Random.Range(0f, 1f) <= critChance)Đ
        {Đ
            damageAbility *= 2;Đ
        }Đ
Đ
Đ
        AbilityData abilityData = new AbilityDataĐ
        {Đ
            CasterStats = characterStats,Đ
           Target = target,Đ
            damage = damageAbility,Đ
            CasterController = abilityController,Đ
            CasterCombatController = thisĐ
            // ... other fieldsĐ
        };Đ
        abilityController.setCooldown(ability.name,ability.cooldown);Đ
```

```
        comboController.UpdateComboController();Đ
        abilityController.CastAbility(ability, abilityData);Đ
    }Đ
    public CharacterStats GetCharacterStats()Đ
    {Đ
        return characterStats;Đ
    }Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterCombatController.csĐ
File: CharacterCombatController.csĐ
- Class: CharacterCombatControllerĐ
  - characterStatsĐ
  - abilityControllerĐ
  - animationControllerĐ
  - stunnableĐ
  - comboControllerĐ
  - Start()Đ
    - characterStats = GetComponent<CharacterStats>()Đ
    - stunnable = GetComponent<IStunnable>()Đ
    - abilityController = GetComponent<AbilityController>()Đ
    - animationController = GetComponent<AnimationController>()Đ
    - comboController = new ComboController()Đ
  - PerformAbility(Ability ability, GameObject target)Đ
    - if stunnable.isStunned()Đ
      - returnĐ
    - if abilityController.checkCooldown(ability.name,ability.cooldown)==falseĐ
      - returnĐ
    - playerController = GetComponent<PlayerController>()Đ
    - if playerController != nullĐ
      - playerController.faceIndirectionOfCamera()Đ
    - damageAbility = ability.baseDamage + (ability.strengthScaling * characterStats.strength) + (ability.intelligenceScaling * characterStats.intelligence)Đ
    - critChance = characterStats.criticalChanceĐ
    - if Random.Range(0f, 1f) <= critChanceĐ
      - damageAbility *= 2Đ
    - abilityData = new AbilityDataĐ
      - CasterStats = characterStatsĐ
      - Target = targetĐ
      - damage = damageAbilityĐ
      - CasterController = abilityControllerĐ

- CasterCombatController = thisÐ
- abilityController.setCooldown(ability.name,ability.cooldown)Ð
- comboController.UpdateComboController()Ð
- abilityController.CastAbility(ability, abilityData)Ð
- GetCharacterStats()Ð
- return characterStats

- CharacterStats at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterStats.cs:
Summary of CharacterStats: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterStats.csÐ
The "CharacterStats.cs" file contains a script that defines the main and sub-stats of a character in a Unity game. The main stats include strength, intelligence, dexterity, endurance, and wisdom, while the sub-stats include critical chance and damage, attack speed, spell critical chance and damage, cooldowns, max life and mana, life and mana regeneration, armor, magic resistance, and dodge chance.Ð
Ð
The script also includes functions to update the sub-stats based on changes to the main stats, equipment bonuses, and skill modifiers. The script allows for the addition of stat points to the main stats, and provides a switch-case function to increase the specific stat type based on the inputted value.Ð
Ð
The script initializes unspent stat points and tracks them as they are gained or used. Additionally, the script contains event actions that are triggered when stats are changed, as well as functions to apply stat bonuses from equipment and skills.Ð
Ð
Overall, the "CharacterStats.cs" file is crucial in managing the attributes and abilities of the player character, allowing for a flexible and customizable gaming experience.
Code of file CharacterStats:
þÿusing System;Ð
using System.Collections;Ð
using UnityEngine;Ð
[Serializable]Ð
public class CharacterStats : MonoBehaviourÐ
{Ð
// MainStatsÐ
public float strength;Ð
public float intelligence;Ð
public float dexterity;Ð
public float endurance;Ð
public float wisdom;Ð
Ð

```
    // SubStatsÐ
Ð
    public float criticalChance;Ð
    public float criticalDamage;Ð
    public float attackSpeed;Ð
Ð
    public float spellCriticalChance;Ð
    public float spellCriticalDamage;Ð
Ð
    public float cooldown;Ð
Ð
    public float maxLife;Ð
    public float maxMana;Ð
    public float lifeRegen;Ð
    public float manaRegen;Ð
Ð
    public float armor;Ð
    public float magicResistance;Ð
Ð
    public float dodgeChance;Ð
Ð
    public int unspentStatPoints;Ð
Ð
    public event Action StatsChanged;Ð
    private EquipManager equipManager;Ð
    private SkillController skillController;Ð
    private BuffSystem buffSystem;Ð
Ð
    private void Awake()Ð
    {Ð
        buffSystem = GetComponent<BuffSystem>();Ð
        equipManager = GetComponent<EquipManager>();Ð
        skillController = GetComponent<SkillController>();Ð
    }Ð
    private void Start()Ð
    {Ð
        // Initialize unspentStatPoints or load from saved game dataÐ
        unspentStatPoints = 10;Ð
        StartCoroutine(InitializeCharacterStats());       Ð
    }Ð
    private IEnumerator InitializeCharacterStats()Ð
    {Ð
        yield return new WaitUntil(() => equipManager != null);Ð
        UpdateSubStats();Ð
Ð
```

```
            Đ
        HealthController healthController = GetComponent<HealthController>();Đ
        if(healthController != null){Đ
            healthController.updateHealth();Đ
        }Đ
        ManaController manaController = GetComponent<ManaController>();Đ
          if(manaController != null){Đ
              manaController.updateMana();Đ
          }Đ
       Đ
    }Đ
    public void AddStatPoints(int amount)Đ
    {Đ
        unspentStatPoints += amount;Đ
        StatsChanged?.Invoke();Đ
    }Đ
Đ
    public void UpdateSubStats()Đ
    {Đ
Đ
        strength += equipManager.TotalStrength;Đ
        intelligence += equipManager.TotalIntelligence;Đ
        dexterity += equipManager.TotalDexterity;Đ
        endurance += equipManager.TotalEndurance;Đ
        wisdom += equipManager.TotalWisdom;Đ
Đ
Đ
        criticalChance = 0.02f * dexterity;Đ
        criticalDamage = 1.5f + (0.14f * dexterity);Đ
        attackSpeed = 1 + (0.01f * strength * dexterity);Đ
Đ
        spellCriticalChance = 0.02f * intelligence;Đ
        spellCriticalDamage = 1.5f + (0.14f * intelligence);Đ
Đ
        armor = 1.5f * endurance;Đ
        magicResistance = 1.5f * endurance;Đ
Đ
Đ
        // Calculate substats based on main stats + equipment bonuses.Đ
        maxLife = 100 + 20 * endurance;Đ
        maxMana = 100 + 20 * wisdom;Đ
        lifeRegen = 1 + 0.25f * endurance;Đ
        manaRegen = 0.5f + 0.25f * wisdom;Đ
Đ
        dodgeChance = 0.009f * dexterity;Đ
```

```
Đ
        AddStatBonuses(equipManager.TotalStatModier);Đ
        AddStatBonuses(skillController.totalStatsModier);Đ
        AddStatBonuses(buffSystem.TotalstatsModifier);Đ
Đ
Đ
        StatsChanged?.Invoke();Đ
    }Đ
Đ
    public void AddStatBonuses(StatsModifier statModifier)Đ
    {Đ
Đ
        attackSpeed += statModifier.attackSpeed;Đ
        criticalChance += statModifier.criticalChance;Đ
        criticalDamage += statModifier.criticalDamage;Đ
        spellCriticalChance += statModifier.spellCriticalChance;Đ
        spellCriticalDamage += statModifier.spellCriticalDamage;Đ
        cooldown += statModifier.cooldown;Đ
        dodgeChance += statModifier.dodgeChance;Đ
        armor += statModifier.armor;Đ
        magicResistance += statModifier.magicResistance;Đ
        maxLife += statModifier.maxLife;Đ
        maxMana += statModifier.maxMana;Đ
        lifeRegen += statModifier.lifeRegen;Đ
        manaRegen += statModifier.manaRegen;Đ
    }Đ
    public void RemoveStatBonuses(StatsModifier statModifier)Đ
    {Đ
        attackSpeed -= statModifier.attackSpeed;Đ
        criticalChance -= statModifier.criticalChance;Đ
        criticalDamage -= statModifier.criticalDamage;Đ
        spellCriticalChance -= statModifier.spellCriticalChance;Đ
        spellCriticalDamage -= statModifier.spellCriticalDamage;Đ
        cooldown -= statModifier.cooldown;Đ
        dodgeChance -= statModifier.dodgeChance;Đ
        armor -= statModifier.armor;Đ
        magicResistance -= statModifier.magicResistance;Đ
        maxLife -= statModifier.maxLife;Đ
        maxMana -= statModifier.maxMana;Đ
        lifeRegen -= statModifier.lifeRegen;Đ
        manaRegen -= statModifier.manaRegen;Đ
    }Đ
Đ
Đ
    public void IncreaseStat(Archetype stateType, int amount)Đ
```

```
    {Đ
        if (unspentStatPoints >= amount)Đ
        {Đ
            switch (stateType)Đ
            {Đ
                case Archetype.Strength:Đ
                    strength += amount;Đ
                    break;Đ
                case Archetype.Intelligence:Đ
                    intelligence += amount;Đ
                    break;Đ
                case Archetype.Dexterity:Đ
                    dexterity += amount;Đ
                    break;Đ
                case Archetype.Endurance:Đ
                    endurance += amount;Đ
                    break;Đ
                case Archetype.Wisdom:Đ
                    wisdom += amount;Đ
                    break;Đ
                default:Đ
                    Debug.LogWarning("Invalid stat name.");Đ
                    return;Đ
            }Đ
Đ
            unspentStatPoints -= amount;Đ
            UpdateSubStats();Đ
        }Đ
        elseĐ
        {Đ
            Debug.LogWarning("Not enough stat points.");Đ
        }Đ
    }Đ
Đ
    internal void SetStats(CharacterStats stats)Đ
    {Đ
        strength = stats.strength;Đ
        intelligence = stats.intelligence;Đ
        dexterity = stats.dexterity;Đ
        endurance = stats.endurance;Đ
        wisdom = stats.wisdom;Đ
        equipManager = GetComponent<EquipManager>();Đ
        UpdateSubStats();Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\CharacterStats.csĐ
Class: CharacterStatsĐ
- float strengthĐ
- float intelligenceĐ
- float dexterityĐ
- float enduranceĐ
- float wisdomĐ
- float criticalChanceĐ
- float criticalDamageĐ
- float attackSpeedĐ
- float spellCriticalChanceĐ
- float spellCriticalDamageĐ
- float cooldownĐ
- float maxLifeĐ
- float maxManaĐ
- float lifeRegenĐ
- float manaRegenĐ
- float armorĐ
- float magicResistanceĐ
- float dodgeChanceĐ
- int unspentStatPointsĐ
- event Action StatsChangedĐ
- EquipManager equipManagerĐ
- SkillController skillControllerĐ
Đ
+ void Awake()Đ
+ void Start()Đ
+ IEnumerator InitializeCharacterStats()Đ
+ void AddStatPoints(int amount)Đ
+ void UpdateSubStats()Đ
+ void AddStatBonuses(StatsModifier statModifier)Đ
+ void IncreaseStat(Archetype stateType, int amount)Đ
+ void SetStats(CharacterStats stats)

- ComboController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ComboController.cs:
Summary of ComboController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ComboController.csĐ
The file name is ComboController.cs. It contains two classes, ComboController and ComboCounter. ComboController has a List of ComboCounters, which is initialized in its constructor. The UpdateComboController method loops through

all the ComboCounters in the list, calling their UpdateComboCounter method. The IncreaseComboCounter method takes a string parameter, which represents the name of the combo to be increased. It loops through all the ComboCounters in the list, checking if the ComboName matches the input string. If it does, it calls the IncreaseComboCounter method of that ComboCounter. If it doesn't find a match, it creates a new ComboCounter with a comboCounter of 1. The GetComboCounter method also takes a string parameter for the name of the combo to retrieve. It loops through the ComboCounters, checking for a match and returning the comboCounter of the matching ComboCounter. If no match is found, it returns 0. ComboCounter has a ComboName, comboCounter, comboTimer, and comboTimeLimit variables. Its constructor sets the comboTimeLimit and initializes comboCounter and comboTimer to 0. The UpdateComboCounter method adds Time.deltaTime to comboTimer, and if comboTimer exceeds comboTimeLimit, comboCounter is reset to 0. IncreaseComboCounter increments comboCounter and sets comboTimer to 0. The GetComboCounter method returns the comboCounter value. ResetComboCounter resets comboCounter and comboTimer to 0.
Code of file ComboController:

```
using System;Đ
using System.Collections.Generic;Đ
using UnityEngine;Đ
[System.Serializable]Đ
public class ComboController{Đ
   public List<ComboCounter> comboCounterList;Đ
Đ
   public ComboController(){Đ
      comboCounterList = new List<ComboCounter>();Đ
   }Đ
   public void UpdateComboController(){Đ
      foreach (ComboCounter comboCounter in comboCounterList)Đ
      {Đ
         comboCounter.UpdateComboCounter();Đ
      }Đ
   }Đ
   public void IncreaseComboCounter(string comboName){Đ
      bool found = false;Đ
      foreach (ComboCounter comboCounter in comboCounterList)Đ
      {Đ
         if(comboCounter.ComboName == comboName){Đ
            comboCounter.IncreaseComboCounter();Đ
            found = true;Đ
         }Đ
      }Đ
      if(!found){Đ
         comboCounterList.Add(new ComboCounter(1f,comboName));Đ
```

```
        }Đ
    }Đ
    public int GetComboCounter(string comboName){Đ
        foreach (ComboCounter comboCounter in comboCounterList)Đ
        {Đ
            if(comboCounter.ComboName == comboName){Đ
                return comboCounter.GetComboCounter();Đ
            }Đ
        }Đ
        return 0;Đ
    }Đ
Đ
    internal void ResetComboCounter(string comboName)Đ
    {Đ
        foreach (ComboCounter comboCounter in comboCounterList)Đ
        {Đ
            if(comboCounter.ComboName == comboName){Đ
                comboCounter.ResetComboCounter();Đ
            }Đ
        }Đ
    }Đ
}Đ
[System.Serializable]Đ
public class ComboCounter{Đ
    Đ
    public string ComboName;Đ
    public int comboCounter;Đ
    public float comboTimer;Đ
    public float comboTimeLimit;Đ
    public ComboCounter(float comboTimeLimit,string comboName){Đ
        this.comboTimeLimit = comboTimeLimit;Đ
        comboCounter = 0;Đ
        comboTimer = 0;Đ
        ComboName = comboName;Đ
    }Đ
    public void UpdateComboCounter(){Đ
        comboTimer += Time.deltaTime;Đ
        if(comboTimer >= comboTimeLimit){Đ
            comboCounter = 0;Đ
        }Đ
    }Đ
    public void IncreaseComboCounter(){Đ
        UpdateComboCounter();Đ
        comboCounter++;Đ
        comboTimer = 0;Đ
```

```
    }Đ
    public int GetComboCounter(){Đ
        return comboCounter;Đ
    }Đ
    public void ResetComboCounter(){Đ
        comboCounter = 0;Đ
        comboTimer = 0;Đ
    }Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\ComboController.csĐ
File: ComboController.csĐ
Đ
Class: ComboControllerĐ
 - comboCounterList: List<ComboCounter>Đ
Đ
Method: ComboController()Đ
 - comboCounterList = new List<ComboCounter>()Đ
Đ
Method: UpdateComboController()Đ
 - foreach (ComboCounter comboCounter in comboCounterList)Đ
   - comboCounter.UpdateComboCounter()Đ
Đ
Method: IncreaseComboCounter(string comboName)Đ
 - bool found = falseĐ
 - foreach (ComboCounter comboCounter in comboCounterList)Đ
   - if(comboCounter.ComboName == comboName)Đ
     - comboCounter.IncreaseComboCounter()Đ
     - found = trueĐ
 - if(!found)Đ
   - comboCounterList.Add(new ComboCounter(1f, comboName))Đ
Đ
Method: GetComboCounter(string comboName)Đ
 - foreach (ComboCounter comboCounter in comboCounterList)Đ
   - if(comboCounter.ComboName == comboName)Đ
     - return comboCounter.GetComboCounter()Đ
 - return 0Đ
Đ
Method: ResetComboCounter(string comboName)Đ
 - foreach (ComboCounter comboCounter in comboCounterList)Đ
   - if(comboCounter.ComboName == comboName)Đ
     - comboCounter.ResetComboCounter()Đ
Đ
Class: ComboCounterĐ
```

- ComboName: stringĐ
- comboCounter: intĐ
- comboTimer: floatĐ
- comboTimeLimit: floatĐ
Đ
Method: ComboCounter(float comboTimeLimit, string comboName)Đ
- this.comboTimeLimit = comboTimeLimitĐ
- comboCounter = 0Đ
- comboTimer = 0Đ
- ComboName = comboNameĐ
Đ
Method: UpdateComboCounter()Đ
- comboTimer += Time.deltaTimeĐ
- if(comboTimer >= comboTimeLimit)Đ
  - comboCounter = 0Đ
Đ
Method: IncreaseComboCounter()Đ
- UpdateComboCounter()Đ
- comboCounter++Đ
- comboTimer = 0Đ
Đ
Method: GetComboCounter()Đ
- return comboCounterĐ
Đ
Method: ResetComboCounter()Đ
- comboCounter = 0Đ
- comboTimer = 0

- EquipManager at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\EquipManager.cs:
Summary of EquipManager: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\EquipManager.csĐ
The file "EquipManager.cs" contains a class for managing equipment items in a game. It defines an enum for different types of equipment, and a dictionary that stores equipped items of each type. Đ
Đ
The class also has properties for storing the total stats from all equipped items, including TotalStrength, TotalIntelligence, TotalDexterity, TotalEndurance, and TotalWisdom. The class allows for adding more stat properties if needed. Đ
Đ
The class has two methods, EquipItem and UnequipItem, for equipping and unequipping items respectively. If an item is already equipped for a certain type, EquipItem will first unequip the previous item before equipping the new one. Unequipping an item removes its effect from the total stats. Đ

Đ
The class also has two private methods, ApplyItemStats and RemoveItemStats, for adding and removing the effect of an item from the total stats. ApplyItemStats adds the item's stat bonuses to the total stats and adds any sub-stats modifiers to the TotalStatModier property. RemoveItemStats subtracts the item's stat bonuses from the total stats and subtracts any sub-stats modifiers from the TotalStatModier property. Đ
Đ
In short, the EquipManager class manages the equipment items in a game and tracks their stats, allowing for easy equipping and unequipping of items and updating of total stats.
Code of file EquipManager:
þÿusing System;Đ
using System.Collections.Generic;Đ
using UnityEngine;Đ
[Serializable]Đ
public class EquipManager : MonoBehaviourĐ
{Đ
    public enum EquipmentType { Weapon, Shield, Helmet, ChestArmor, LegArmor, Boots, Ring, Wrist }Đ
Đ
    public Dictionary<EquipmentType, EquipableItem> equippedItems = new Dictionary<EquipmentType, EquipableItem>();Đ
Đ
    // Properties to store the total stats from all equipped items.Đ
    public StatsModifier TotalStatModier;Đ
    public float TotalStrength = 0;Đ
    public float TotalIntelligence= 0;Đ
    public float TotalDexterity= 0;Đ
    public float TotalEndurance= 0;Đ
    public float TotalWisdom=0 ;Đ
    // Add more stat properties as needed.Đ
Đ
    public void EquipItem(EquipmentType type, EquipableItem item)Đ
    {Đ
        if (equippedItems.ContainsKey(type))Đ
        {Đ
            UnequipItem(type);Đ
        }Đ
Đ
        equippedItems[type] = item;Đ
        ApplyItemStats(item);Đ
    }Đ
Đ
    public void UnequipItem(EquipmentType type)Đ

```
    {Đ
        if (!equippedItems.ContainsKey(type)) return;Đ
Đ
        EquipableItem item = equippedItems[type];Đ
        RemoveItemStats(item);Đ
        equippedItems.Remove(type);Đ
    }Đ
Đ
    private void ApplyItemStats(EquipableItem item)Đ
    {Đ
        TotalStrength += item.strengthBonus;Đ
        TotalIntelligence += item.intelligenceBonus;Đ
        TotalDexterity += item.dexterityBonus;Đ
        TotalEndurance += item.enduranceBonus;Đ
        TotalWisdom += item.wisdomBonus;Đ
Đ
        TotalStatModier.Add(item.subStatsModifier);Đ
Đ
        // Add more stat effects as needed.Đ
    }Đ
Đ
    private void RemoveItemStats(EquipableItem item)Đ
    {Đ
        TotalStrength -= item.strengthBonus;Đ
        TotalIntelligence -= item.intelligenceBonus;Đ
        TotalDexterity -= item.dexterityBonus;Đ
        TotalEndurance -= item.enduranceBonus;Đ
        TotalWisdom -= item.wisdomBonus;Đ
Đ
        TotalStatModier.Sub(item.subStatsModifier);Đ
Đ
        // Remove more stat effects as needed.Đ
    }Đ
Đ
    internal void SetEquipManager(EquipManager equipManager)Đ
    {Đ
        equippedItems = equipManager.equippedItems;Đ
        TotalStatModier = equipManager.TotalStatModier;Đ
        TotalStrength = equipManager.TotalStrength;Đ
        TotalIntelligence = equipManager.TotalIntelligence;Đ
        TotalDexterity = equipManager.TotalDexterity;Đ
        TotalEndurance = equipManager.TotalEndurance;Đ
        TotalWisdom = equipManager.TotalWisdom;Đ
        Đ
Đ
```

}Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\EquipManager.csĐ
Filename: EquipManager.csĐ
Đ
Syntax Tree:Đ
- class EquipManagerĐ
  - public enum EquipmentTypeĐ
    - WeaponĐ
    - ShieldĐ
    - HelmetĐ
    - ChestArmorĐ
    - LegArmorĐ
    - BootsĐ
    - RingĐ
    - WristĐ
  - public Dictionary<EquipmentType, EquipableItem> equippedItemsĐ
  - public StatsModifier TotalStatModierĐ
  - public float TotalStrengthĐ
  - public float TotalIntelligenceĐ
  - public float TotalDexterityĐ
  - public float TotalEnduranceĐ
  - public float TotalWisdomĐ
  - public void EquipItem(EquipmentType type, EquipableItem item)Đ
    - if (equippedItems.ContainsKey(type))Đ
      - UnequipItem(type)Đ
    - equippedItems[type] = itemĐ
    - ApplyItemStats(item)Đ
  - public void UnequipItem(EquipmentType type)Đ
    - if (!equippedItems.ContainsKey(type)) returnĐ
    - EquipableItem item = equippedItems[type]Đ
    - RemoveItemStats(item)Đ
    - equippedItems.Remove(type)Đ
  - private void ApplyItemStats(EquipableItem item)Đ
    - TotalStrength += item.strengthBonusĐ
    - TotalIntelligence += item.intelligenceBonusĐ
    - TotalDexterity += item.dexterityBonusĐ
    - TotalEndurance += item.enduranceBonusĐ
    - TotalWisdom += item.wisdomBonusĐ
    - TotalStatModier.Add(item.subStatsModifier)Đ
  - private void RemoveItemStats(EquipableItem item)Đ
    - TotalStrength -= item.strengthBonusĐ

- TotalIntelligence -= item.intelligenceBonusĐ
- TotalDexterity -= item.dexterityBonusĐ
- TotalEndurance -= item.enduranceBonusĐ
- TotalWisdom -= item.wisdomBonusĐ
- TotalStatModier.Sub(item.subStatsModifier)Đ

- internal void SetEquipManager(EquipManager equipManager)Đ
  - equippedItems = equipManager.equippedItemsĐ
  - TotalStatModier = equipManager.TotalStatModierĐ
  - TotalStrength = equipManager.TotalStrengthĐ
  - TotalIntelligence = equipManager.TotalIntelligenceĐ
  - TotalDexterity = equipManager.TotalDexterityĐ
  - TotalEndurance = equipManager.TotalEnduranceĐ
  - TotalWisdom = equipManager.TotalWisdom

- ExperienceSystem at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ExperienceSystem.cs:
Summary of ExperienceSystem: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ExperienceSystem.csĐ
The file "ExperienceSystem.cs" contains a class that implements a system for tracking and leveling up player experience in a game. The class has three public properties: "CurrentXP," which tracks the current amount of experience the player has; "Level," which tracks the current level of the player; and "XpToNextLevel," which tracks the amount of experience required to reach the next level.Đ
Đ
The class also has two events: "LevelUpEvent," which is triggered when the player levels up; and "ExperienceGained," which is triggered whenever the player gains experience. Đ
Đ
In the constructor, "CurrentXP" is initialized to 0, "Level" is initialized to 1, and "UpdateXpToNextLevel()" is called to set the initial value of "XpToNextLevel."Đ
Đ
The public method "AddExperience(int amount)" allows for the addition of experience to the player's total. Whenever experience is gained, the "ExperienceGained" event is triggered. If the player has gained enough experience to reach the next level, the "LevelUp()" method is called.Đ
Đ
The private method "LevelUp()" increases the player's level by 1, updates the value of "XpToNextLevel," and triggers the "LevelUpEvent." Đ
Đ
The private method "UpdateXpToNextLevel()" sets the current value of "XpToNextLevel" to the result of calling the private method "CalculateXpForLevel(int level)."Đ
Đ

The private method "CalculateXpForLevel(int level)" calculates the amount of experience needed to reach a certain level according to custom logic included by the developer. In this implementation, the formula used is to square the level and multiply by 100.Ð
Ð
Overall, the class provides a simple and flexible way to track and level up player experience in a game with the ability for developers to customize the experience required for each level.
Code of file ExperienceSystem:

```
þÿusing System;Ð
Ð
public class ExperienceSystemÐ
{Ð
    public int CurrentXP { get; private set; }Ð
    public int Level { get; private set; }Ð
    public int XpToNextLevel { get; private set; }Ð
Ð
    public event Action LevelUpEvent;Ð
    public event Action<int> ExperienceGained;Ð
Ð
    public ExperienceSystem()Ð
    {Ð
        CurrentXP = 0;Ð
        Level = 1;Ð
        UpdateXpToNextLevel();Ð
    }Ð
Ð
    public void AddExperience(int amount)Ð
    {Ð
        CurrentXP += amount;Ð
        ExperienceGained?.Invoke(amount);Ð
Ð
        while (CurrentXP >= XpToNextLevel)Ð
        {Ð
            CurrentXP -= XpToNextLevel;Ð
            LevelUp();Ð
        }Ð
    }Ð
Ð
    private void LevelUp()Ð
    {Ð
        Level++;Ð
        UpdateXpToNextLevel();Ð
        LevelUpEvent?.Invoke();Ð
    }Ð
```

```
Đ
    private void UpdateXpToNextLevel()Đ
    {Đ
        XpToNextLevel = CalculateXpForLevel(Level);Đ
    }Đ
Đ
    private int CalculateXpForLevel(int level)Đ
    {Đ
        // Implement your custom XP calculation logic hereĐ
        return (int)Math.Floor(Math.Pow(level, 2) * 100);Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ExperienceSystem.csĐ
File: ExperienceSystem.csĐ
Đ
Class: ExperienceSystemĐ
Đ
- CurrentXP: intĐ
- Level: intĐ
- XpToNextLevel: intĐ
- LevelUpEvent: ActionĐ
- ExperienceGained: Action<int>Đ
Đ
- ExperienceSystem()Đ
- AddExperience(int amount)Đ
- LevelUp()Đ
- UpdateXpToNextLevel()Đ
- CalculateXpForLevel(int level)

- HealthController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\HealthController.cs:
Summary of HealthController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\HealthController.csĐ
The HealthController.cs file is responsible for controlling the health of an object in the game. It contains several variables, including Name, maxHealth, currentHealth, damageTextPrefab, and questSystem. Đ
Đ
The Start function sets the currentHealth to equal the maxHealth value and finds the damageTextPrefab in the scene. Đ
Đ
The TakeDamage function reduces the currentHealth by the damage value and

calls the ShowDamageNumbers function to display the damage text on the screen. If the currentHealth is less than or equal to zero, the function checks if the object that caused the damage has a QuestSystem component attached and updates the quest objective if it does. The Die function is called to implement any death behavior needed, such as playing a death animation or dropping loot, and ultimately destroys the game object. Ð
Ð
The ShowDamageNumbers function checks if the WorldSpaceCanvasController instance is present in the scene, and if it is, it spawns a damage number text at the current object's position with a small offset in the upward direction.
Code of file HealthController:

```
using UnityEngine;Ð
Ð
public class HealthController : MonoBehaviourÐ
{Ð
    private CharacterStats characterStats;Ð
    public string Name;Ð
    public float maxHealth;Ð
    public float currentHealth;Ð
    public GameObject damageTextPrefab;Ð
    private QuestSystem questSystem;Ð
Ð
    void UpdateMaxHealth()Ð
    {Ð
        maxHealth = characterStats.maxLife;Ð
    }Ð
    public void updateHealth()Ð
    {Ð
        currentHealth = maxHealth;Ð
    }Ð
    private void Start()Ð
    {Ð
        characterStats = GetComponent<CharacterStats>();Ð
        characterStats.StatsChanged+=UpdateMaxHealth;Ð
        UpdateMaxHealth();Ð
        currentHealth = maxHealth;Ð
        damageTextPrefab = GameObject.Find("DamageTextTemplate");Ð
        Ð
Ð
    }Ð
Ð
    public void TakeDamage(float damage,GameObject from)Ð
    {Ð
        currentHealth -= damage;Ð
        ShowDamageNumbers(damage);Ð
```

```
        if (currentHealth <= 0)Đ
        {Đ
            if(from.GetComponent<QuestSystem>() != null)Đ
            {Đ

from.GetComponent<QuestSystem>().UpdateQuestObjective("kill:"+Name);Đ
            }Đ
            Die();Đ
        }Đ
    }Đ
Đ
    private void Die()Đ
    {Đ
        // Implement death behavior, such as playing death animation, dropping loot,
etc.Đ
Đ
        Destroy(gameObject);Đ
    }Đ
    public void ShowDamageNumbers(float damage)Đ
    {Đ
        if (WorldSpaceCanvasController.Instance == null)Đ
        {Đ
            Debug.LogError("WorldSpaceCanvasController instance is not present in
the scene.");Đ
            return;Đ
        }Đ
Đ
        WorldSpaceCanvasController.Instance.SpawnDamageNumber(damage,
transform.position + Vector3.up * 2f);Đ
    }Đ
Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\HealthController.csĐ
HealthController:Đ
- private characterStatsĐ
- public Name:stringĐ
- public maxHealth:floatĐ
- public currentHealth:floatĐ
- private damageTextPrefab:GameObjectĐ
- private questSystemĐ
Đ
- UpdateMaxHealth()Đ
  - maxHealth = characterStats.maxLifeĐ
```

Đ
- updateHealth()Đ
  - currentHealth = maxHealthĐ
Đ
- Start()Đ
  - characterStats = GetComponent<CharacterStats>()Đ
  - characterStats.StatsChanged+=UpdateMaxHealthĐ
  - UpdateMaxHealth()Đ
  - currentHealth = maxHealthĐ
  - damageTextPrefab = GameObject.Find("DamageTextTemplate")Đ
Đ
- TakeDamage(damage:float, from:GameObject)Đ
  - currentHealth -= damageĐ
  - ShowDamageNumbers(damage)Đ
  - if currentHealth <= 0Đ
    - if from.GetComponent<QuestSystem>() != nullĐ
      - from.GetComponent<QuestSystem>().UpdateQuestObjective("kill:"+Name)Đ
    - Die()Đ
Đ
- Die()Đ
  - Destroy(gameObject)Đ
Đ
- ShowDamageNumbers(damage:float)Đ
  - if WorldSpaceCanvasController.Instance == nullĐ
    - Debug.LogError("WorldSpaceCanvasController instance is not present in the scene.")Đ
    - returnĐ
  - WorldSpaceCanvasController.Instance.SpawnDamageNumber(damage, transform.position + Vector3.up * 2f)

- HotkeyController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\HotkeyController.cs:
Summary of HotkeyController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\HotkeyController.csĐ
The file "HotkeyController.cs" contains a script in Unity that handles the hotkeys for a player and units. It includes a list of hotkeys, a combat controller, and a dictionary that maps the hotkeys to their respective abilities. Đ
Đ
In the Start function, the script initializes the hotkeys list with a capacity of nine, and populates the dictionary with the corresponding hotkeys. Additionally, it assigns a test ability to the first hotkey in the list. Đ
Đ
In the Update function, the script calls the HandleHotkey function which checks for a key press from the player. If a hotkey is mapped to the pressed key and is

assigned an ability, the script performs the ability using the combat controller. Đ
Đ
The Hotkey class defines a hotkey object that contains an ability.
Code of file HotkeyController:

```
using UnityEngine;Đ
using System.Collections.Generic;Đ
using System;Đ
Đ
public class HotkeyController : MonoBehaviour{Đ
    Đ
    public List<Hotkey> hotkeys;Đ
    private CharacterCombatController combatController;Đ
    public Dictionary<KeyCode, Hotkey> hotkeyMapping;Đ
    public void Update()Đ
    {Đ
        HandleHotkey();Đ
    }Đ
    public void Start()Đ
    {Đ
        combatController = GetComponent<CharacterCombatController>();Đ
        hotkeys = new List<Hotkey>();Đ
            for (int i = 0; i < 9; i++)Đ
            {Đ
                hotkeys.Add(new Hotkey());Đ
            }Đ
Đ
        hotkeyMapping = new Dictionary<KeyCode, Hotkey>Đ
        {Đ
            { KeyCode.Alpha1, hotkeys[0] },Đ
            { KeyCode.Alpha2, hotkeys[1] },Đ
            { KeyCode.Alpha3, hotkeys[2] },Đ
            { KeyCode.Alpha4, hotkeys[3] },Đ
            { KeyCode.Alpha5, hotkeys[4] },Đ
            { KeyCode.Alpha6, hotkeys[5] },Đ
            { KeyCode.Alpha7, hotkeys[6] },Đ
            { KeyCode.Alpha8, hotkeys[7] },Đ
            { KeyCode.E, hotkeys[8] }Đ
        };Đ
Đ
        Hotkey hotkeyTest = new Hotkey();Đ
        hotkeyTest.ability = combatController.abilityController.learnedAbilitys[0];Đ
        hotkeys[0].ability = combatController.abilityController.learnedAbilitys[0];Đ
    }Đ
    private void HandleHotkey()Đ
    {Đ
```

```
        foreach (KeyValuePair<KeyCode, Hotkey> entry in hotkeyMapping)Đ
        {Đ
            if (Input.GetKeyDown(entry.Key))Đ
            {Đ
                Hotkey hotkey = entry.Value;Đ
                if (hotkey.ability != null)Đ
                {Đ
                    combatController.PerformAbility(hotkey.ability, this.gameObject);Đ
                }Đ
                // else if (hotkey.item != null)Đ
                {Đ
                    // UseItem(hotkey.item);Đ
                }Đ
            }Đ
        }Đ
    }Đ
Đ
internal void SwapHotkeys(int hotkeyIndex1, int hotkeyIndex2)Đ
{Đ
    Hotkey tempHotkey = hotkeys[hotkeyIndex1];Đ
    hotkeys[hotkeyIndex1] = hotkeys[hotkeyIndex2];Đ
    hotkeys[hotkeyIndex2] = tempHotkey;Đ
}Đ
Đ
internal void AssignAbilityToHotkey(int hotkeyIndex, Ability assignedAbility)Đ
{Đ
    hotkeys[hotkeyIndex].ability = assignedAbility;Đ
    hotkeys[hotkeyIndex].item = null;Đ
}Đ
Đ
internal void AssignItemToHotkey(int hotkeyIndex, Item assignedItem)Đ
{Đ
    hotkeys[hotkeyIndex].item = assignedItem;Đ
    hotkeys[hotkeyIndex].ability = null;Đ
}Đ
Đ
}Đ
public class HotkeyĐ
{Đ
    public Ability ability;Đ
    public Item item;Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\HotkeyController.csĐ
```

HotkeyController:Đ
- public List<Hotkey> hotkeysĐ
- private CharacterCombatController combatControllerĐ
- public Dictionary<KeyCode, Hotkey> hotkeyMappingĐ
- public void Update()Đ
  - HandleHotkey()Đ
- public void Start()Đ
  - combatController = GetComponent<CharacterCombatController>()Đ
  - hotkeys = new List<Hotkey>()Đ
    - for (int i = 0; i < 9; i++)Đ
      - hotkeys.Add(new Hotkey())Đ
  - hotkeyMapping = new Dictionary<KeyCode, Hotkey> { ... }Đ
  - Hotkey hotkeyTest = new Hotkey()Đ
  - hotkeyTest.ability = combatController.abilityController.learnedAbilitys[0]Đ
  - hotkeys[0].ability = combatController.abilityController.learnedAbilitys[0]Đ
- private void HandleHotkey()Đ
  - foreach (KeyValuePair<KeyCode, Hotkey> entry in hotkeyMapping)Đ
    - if (Input.GetKeyDown(entry.Key))Đ
      - Hotkey hotkey = entry.ValueĐ
      - if (hotkey.ability != null)Đ
        - combatController.PerformAbility(hotkey.ability, this.gameObject)Đ
- internal void SwapHotkeys(int hotkeyIndex1, int hotkeyIndex2)Đ
  - Hotkey tempHotkey = hotkeys[hotkeyIndex1]Đ
  - hotkeys[hotkeyIndex1] = hotkeys[hotkeyIndex2]Đ
  - hotkeys[hotkeyIndex2] = tempHotkeyĐ
- internal void AssignAbilityToHotkey(int hotkeyIndex, Ability assignedAbility)Đ
  - hotkeys[hotkeyIndex].ability = assignedAbilityĐ
  - hotkeys[hotkeyIndex].item = nullĐ
- internal void AssignItemToHotkey(int hotkeyIndex, Item assignedItem)Đ
  - hotkeys[hotkeyIndex].item = assignedItemĐ
  - hotkeys[hotkeyIndex].ability = nullĐ
Hotkey:Đ
- public Ability abilityĐ
- public Item item

- IInteractable at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\IInteractable.cs:
Summary of IInteractable:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\IInteractable.csĐ
The file called "IInteractable.cs" contains an interface called IInteractable. This interface requires any implementing class to have a method called "Interact()" which does not return any value. This interface is used to ensure any unit or player character in the game that needs to be interacted with by the player, can be interacted with using the same method name "Interact()". The logic behind the

"Interact()" method will be defined in each implementing class.
Code of file IInteractable:
using UnityEngine;Ð
Ð
public interface IInteractableÐ
{Ð
    void Interact(Transform interacter);Ð
}Ð

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\IInteractable.csÐ
IInteractable: Ð
- InterfaceÐ
- Method: Interact(Transform interacter)

- Inventory at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\Inventory.cs:
Summary of Inventory:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\Inventory.csÐ
The file being implemented here is called "Inventory.cs" and it is located in the "PlayerAndUnitsComponent" folder of the Unity project. This file contains a class called "Inventory" which inherits from the MonoBehaviour class. The class has three public fields: a List of "Item" objects, and a QuestSystem object. Ð
Ð
The Start() method initializes the questSystem field by getting the QuestSystem component that is attached to the same GameObject as the Inventory component. Ð
Ð
The AddItem() method adds the given item to the inventory, and if the questSystem object is not null, it logs a message and calls the UpdateQuestObjective() method on the questSystem object to update the quest objective with the item's name. Ð
Ð
The RemoveItem() method removes the given item from the inventory. Ð
Ð
The HasItem() method returns true if the inventory contains the given item. Ð
Ð
Overall, this class provides basic functionality for managing an inventory of items in a game, and also integrates with a quest system to update quest objectives when items are collected.
Code of file Inventory:
þÿusing System.Collections.Generic;Ð
using UnityEngine;Ð

```
Đ
public class Inventory : MonoBehaviourĐ
{Đ
    public List<Item> items;Đ
    public QuestSystem questSystem;Đ
      private void Start()Đ
    {Đ
        questSystem = GetComponent<QuestSystem>();Đ
Đ
    }Đ
Đ
    public void AddItem(Item item)Đ
    {Đ
        items.Add(item);Đ
        if(questSystem!=null)  // Check if questSystem is not nullĐ
        {Đ
            Debug.Log("collect:"+item.name);Đ
            questSystem.UpdateQuestObjective("collect:"+item.name); // Call
UpdateQuestObjective method with item idĐ
        }Đ
    }Đ
Đ
    public void RemoveItem(Item item)Đ
    {Đ
        items.Remove(item);Đ
    }Đ
Đ
    public bool HasItem(Item item)Đ
    {Đ
        return items.Contains(item);Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\Inventory.csĐ
Inventory:Đ
- items: List<Item>Đ
- questSystem: QuestSystemĐ
Đ
Start():Đ
- questSystem = GetComponent<QuestSystem>()Đ
Đ
AddItem(item: Item):Đ
- items.Add(item)Đ
```

- if questSystem != null:Đ
  - Debug.Log("collect:" + item.name)Đ
  - questSystem.UpdateQuestObjective("collect:" + item.name)Đ
Đ
RemoveItem(item: Item):Đ
- items.Remove(item)Đ
Đ
HasItem(item: Item):Đ
- return items.Contains(item)

- isStunnableController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\isStunnableController.cs:
Summary of isStunnableController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\isStunnableController.csĐ
The file "isStunnableController.cs" within the "PlayerAndUnitsComponent" folder is responsible for controlling the stunnable behavior of game objects. The script implements the "IStunnable" interface, which includes three properties: "stunned", "timeAtStunStart" and "stunDuration". The "stunned" property indicates whether the game object is currently stunned or not, while "timeAtStunStart" and "stunDuration" properties represent the start time and duration of the stun effect.Đ
Đ
The script provides a "Stun" function, which sets the "stunned" property to true and saves the current time as the starting time for the stun effect. It also saves the input "duration" parameter as the duration of the stun effect. The "isStunned" function returns the current state of the "stunned" property. If the time elapsed since the start of the stun effect is greater than the "stunDuration" property, then the "stunned" property is set to false, indicating the end of the stun effect.Đ
Đ
In summary, the "isStunnableController.cs" script allows game objects to be stunned and then recover from the stun effect after a set duration of time.
Code of file isStunnableController:
```
using UnityEngine;Đ
public class isStunnableController : MonoBehaviour,IStunnable{Đ
Đ
   public bool stunned;Đ
Đ
   float timeAtStunStart;Đ
   float stunDuration;Đ
Đ
   bool IStunnable.stunned { get => stunned ;}Đ
Đ
   float IStunnable.timeAtStunStart => timeAtStunStart;Đ
```

```
Đ
   float IStunnable.stunDuration => stunDuration;Đ
Đ
   VisualEffectController visualEffectController;Đ
Đ
   private void Start()Đ
   {Đ
      visualEffectController = GetComponent<VisualEffectController>();Đ
   }Đ
   public void Stun(float duration){Đ
      stunned = true;Đ
      timeAtStunStart = Time.time;Đ
      stunDuration = duration;Đ
      visualEffectController.SpawnEffect("Stun",duration);Đ
Đ
   }Đ
   public bool isStunned(){Đ
      Đ
         if(Time.time - timeAtStunStart >= stunDuration){Đ
            stunned = false;Đ
         }Đ
      return stunned;Đ
      Đ
   }Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\isStunnableController.csĐ
Class: isStunnableControllerĐ
Đ
- bool stunnedĐ
- float timeAtStunStartĐ
- float stunDurationĐ
- VisualEffectController visualEffectControllerĐ
- void Start()Đ
- void Stun(float duration)Đ
- bool isStunned()Đ
Đ
IStunnableĐ
- bool stunnedĐ
- float timeAtStunStartĐ
- float stunDuration

- IStunnable at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\IStunnable.cs:
```

Summary of IStunnable:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\IStunnable.csÐ
The file name is IStunnable.cs and it is located in the directory Assets\Scripts\PlayerAndUnitsComponent. This file contains an interface called IStunnable that defines certain properties and methods related to the ability of a GameObject to be stunned.Ð
Ð
The interface includes a bool property called stunned that indicates whether the GameObject is currently stunned or not. It also includes two float properties: timeAtStunStart, which represents the time at which the GameObject was last stunned, and stunDuration, which represents the duration of the stun.Ð
Ð
In addition, the interface defines a method called Stun, which takes in a float parameter called duration and stuns the GameObject for that amount of time. Finally, the interface includes a method called isStunned, which returns a bool indicating whether the GameObject is currently stunned.Ð
Ð
Overall, IStunnable.cs provides a flexible and extensible framework for implementing stuns in Unity projects. Developers can easily add the IStunnable interface to any GameObject in their game and customize its behavior as needed, providing a powerful tool for creating compelling gameplay experiences.
Code of file IStunnable:

```
//Interface isStunnable if GameObject can be stunned,contain bool isStunnedÐ
// Path: Assets\Scripts\PlayerAndUnitsComponent\IStunnable.csÐ
using UnityEngine;Ð
public interface IStunnableÐ
{Ð
    bool stunned { get; }Ð
Ð
   Ð
    float timeAtStunStart{ get; }Ð
    float stunDuration{ get ;}Ð
Ð
Ð
    void Stun(float duration);Ð
    public bool isStunned();Ð
}Ð
Ð
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\IStunnable.csÐ
IStunnable.cs:Ð
- interface IStunnableÐ

- bool stunnedÐ
- float timeAtStunStartÐ
- float stunDurationÐ
- void Stun(float duration)Ð
- bool isStunned()

- ManaController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ManaController.cs:
Summary of ManaController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ManaController.csÐ
The file "ManaController.cs" is a script that is used to manage the mana of a character in a game. It has a public float variable for the maximum mana and another for the current mana. It also has a private variable called "characterStats" of type CharacterStats.Ð
Ð
In the Start() function, the script gets a reference to the CharacterStats component of the game object and subscribes to the "StatsChanged" event, which calls the "updateMaxMana" function. The currentMana is then set to the maxMana value.Ð
Ð
The "updateMaxMana" function updates the value of maxMana when the characterStats component value changes.Ð
Ð
The "UseMana" function is used to subtract the manaCost value from the currentMana if the character has sufficient mana. Equality of manaCost and currentMana is checked by the "HasSufficientMana" function.Ð
Ð
Finally, the "RegenerateMana" function is used to increase the currentMana by the manaAmount value, and caps the currentMana value to the maxMana value.Ð
Ð
Overall, the ManaController script is used to manage the mana in the game for a specific character and provides functions to use and regenerate the mana.
Code of file ManaController:

```
using UnityEngine;Ð
Ð
public class ManaController : MonoBehaviourÐ
{Ð
    public float maxMana;Ð
    public float currentMana;Ð
    private CharacterStats characterStats;Ð
Ð
    private void Start()Ð
    {Ð
        characterStats = GetComponent<CharacterStats>();Ð
```

```
        characterStats.StatsChanged += updateMaxMana;Đ
        Đ
        currentMana = maxMana;Đ
Đ
    }Đ
    private void updateMaxMana()Đ
    {Đ
     maxMana = characterStats.maxMana;Đ
    }Đ
    public void updateMana()Đ
    {Đ
        currentMana = maxMana;Đ
    }Đ
    public void UseMana(float manaCost)Đ
    {Đ
        if (HasSufficientMana(manaCost))Đ
        {Đ
            currentMana -= manaCost;Đ
        }Đ
    }Đ
Đ
    public bool HasSufficientMana(float manaCost)Đ
    {Đ
        return currentMana >= manaCost;Đ
    }Đ
Đ
    public void RegenerateMana(float manaAmount)Đ
    {Đ
        currentMana += manaAmount;Đ
        if (currentMana > maxMana)Đ
        {Đ
            currentMana = maxMana;Đ
        }Đ
    }Đ
Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\ManaController.csĐ
ManaController:Đ
 -maxMana: floatĐ
 -currentMana: floatĐ
 -characterStats: CharacterStatsĐ

+Start()Đ
+updateMaxMana()Đ
+updateMana()Đ
+UseMana(manaCost: float)Đ
+HasSufficientMana(manaCost: float): boolĐ
+RegenerateMana(manaAmount: float)

- MovementController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\MovementController.cs:
Summary of MovementController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\MovementController.csĐ
The file "MovementController.cs" is responsible for controlling the movement of a player or unit in the game. It utilizes the NavMeshAgent component to navigate a NavMesh surface towards a given target position. The stopping distance for the unit can be set using the "stoppingDistance" variable. Đ
Đ
The script initializes the NavMeshAgent and IStunnable components in the "Start()" function. The IStunnable component is used to check if the unit is currently stunned. If it is, the NavMeshAgent is stopped. If the unit is not stunned, the NavMeshAgent will resume moving towards the target position in the "Update()" function. Đ
Đ
Overall, the script ensures that the unit moves towards the target position while accounting for any stuns that may occur during the game.
Code of file MovementController:

```
using UnityEngine;Đ
using UnityEngine.AI;Đ
Đ
public class MovementController : MonoBehaviourĐ
{Đ
    IStunnable stunnable;Đ
    public Transform target;Đ
    public float stoppingDistance = 2f;Đ
Đ
    private NavMeshAgent agent;Đ
Đ
    private void Start()Đ
    {Đ
        agent = GetComponent<NavMeshAgent>();Đ
        stunnable = GetComponent<IStunnable>();Đ
        Debug.Log("stunnable: " + stunnable);Đ
    }Đ
Đ
```

```
    private void Update()Đ
    {Đ
        if (target != null)Đ
        {Đ
            agent.SetDestination(target.position);Đ
            agent.stoppingDistance = stoppingDistance;Đ
        }Đ
        if(stunnable != null && stunnable.isStunned())Đ
        {Đ
            agent.isStopped = true;Đ
        }Đ
        elseĐ
        {Đ
            agent.isStopped = false;Đ
        }Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\MovementController.csĐ
MovementController:Đ
- stunnable: IStunnableĐ
- target: TransformĐ
- stoppingDistance: floatĐ
- agent: NavMeshAgentĐ
Đ
Start():Đ
- agent = GetComponent<NavMeshAgent>()Đ
- stunnable = GetComponent<IStunnable>()Đ
- Debug.Log("stunnable: " + stunnable)Đ
Đ
Update():Đ
- if target is not null:Đ
  - agent.SetDestination(target.position)Đ
  - agent.stoppingDistance = stoppingDistanceĐ
- if stunnable is not null and stunnable.isStunned():Đ
  - agent.isStopped = trueĐ
- else:Đ
  - agent.isStopped = false

- PlayerController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\PlayerAndUnitsComponent\PlayerController.cs:
Summary of PlayerController:
File Name: PlayerController.csĐ
```

Đ
The PlayerController class in Unity is responsible for controlling the character movement, camera, jumping, and handling hotkeys for abilities and items. Đ
Đ
The class contains the following components:Đ
- BuffSystem: A system for managing buffsĐ
- ExperienceSystem: A system for managing experience pointsĐ
- CharacterStats: A component for managing character stats such as strength, intelligence, and enduranceĐ
- CharacterCombatController: A controller for managing character combat Đ
- SkillController: A controller for managing player's skillsĐ
- SkillTree: A component for managing the skill treeĐ
- IStunnable: An interface for checking whether the character is stunnedĐ
Đ
The class also contains the following public variables:Đ
- moveSpeed: A float value for character movement speedĐ
- rotationSpeed: A float value for character rotation speedĐ
- jumpForce: A float value for the force applied for jumpingĐ
- groundLayer: A layer mask for detecting groundĐ
Đ
Additionally, the class has private variables for managing character movement, camera rotation, and hotkeys. It also uses the Animator component and a Rigidbody for character movement.Đ
Đ
The Start() function is responsible for initializing the various components and hotkeys. It also subscribes to the OnSkillUnlocked event in the SkillController component.Đ
Đ
The Update() function handles character movement, jumping, camera rotation, and hotkeys. It calls the HandleMovement(), HandleJump(), HandleCamera(), and HandleHotkey() functions respectively.Đ
Đ
The HandleHotkey() function is responsible for checking and performing abilities or items assigned to hotkeys using the KeyCode value of each hotkey.Đ
Đ
The HandleMovement() function retrieves input values for character movement and sets the character's rotation according to the input direction.Đ
Đ
The HandleJump() function checks and applies force for jumping if the character is on the ground.Đ
Đ
The HandleCamera() function retrieves input values for camera rotation and sets the camera position and rotation accordingly.Đ
Đ
The FixedUpdate() function is used for applying motion to the Rigidbody

component for character movement.Đ
Đ
The class also contains a method for unlocking and unlearning skill nodes in the SkillTree component.
Code of file PlayerController:

```
using System.Collections.Generic;Đ
using UnityEngine;Đ
Đ
Đ
Đ
public class PlayerController : MonoBehaviourĐ
{Đ
    [Header("Controller")]Đ
    BuffSystem buffSystem;Đ
    ExperienceSystem experienceSystem;Đ
    CharacterStats characterStats;Đ
    CharacterCombatController combatController;Đ
    SkillController skillController;Đ
    SkillTree skillTree;Đ
    IStunnable stunnable;Đ
    HotkeyController hotkeyController;Đ
    CanGrabController canGrabController;Đ
    TargetingSystem targetingSystem;Đ
  Đ
Đ
Đ
    public Ability Ability1;Đ
Đ
Đ
Đ
    [Header("Movement")]Đ
    public float moveSpeed = 5f;Đ
    public float rotationSpeed = 720f;Đ
    public float jumpForce = 1f;Đ
    public LayerMask groundLayer;Đ
Đ
    [Header("Camera")]Đ
    public Transform cameraTarget;Đ
    public float cameraDistance = 5f;Đ
    public float cameraHeight = 2f;Đ
    public float cameraRotationSpeed = 2f;Đ
Đ
Đ
    private Rigidbody rb;Đ
    private Animator animator;Đ
```

```
    private Vector3 moveDirection;Đ
    private bool isGrounded;Đ
    private Transform mainCamera;Đ
    private float cameraRotationY;Đ
Đ
Đ
    private void Start()Đ
    {Đ
        canGrabController = GetComponent<CanGrabController>();Đ
        combatController = GetComponent<CharacterCombatController>();Đ
        characterStats = GetComponent<CharacterStats>();Đ
        skillController = GetComponent<SkillController>();Đ
        targetingSystem = GetComponent<TargetingSystem>();Đ
Đ
        //EDITOR CODEĐ
        skillController.skillTree.resetAllNodes();Đ
Đ
        rb = GetComponent<Rigidbody>();Đ
        animator = GetComponent<Animator>();Đ
        mainCamera = Camera.main.transform;Đ
        Cursor.lockState = CursorLockMode.Locked;Đ
        Cursor.visible = false;Đ
Đ
Đ
Đ
        skillController.OnSkillUnlocked += UpdateToSkillEvents;Đ
        stunnable = GetComponent<IStunnable>();Đ
    }Đ
Đ
Đ
    private void Update()Đ
    {Đ
        HandleMovement();Đ
        HandleJump();Đ
        HandleCamera();Đ
        HandleActions();Đ
    }Đ
    private void UpdateToSkillEvents(SkillNode node)Đ
    {Đ
        characterStats.UpdateSubStats();Đ
    }Đ
    GameObject target;Đ
    public void HandleActions(){Đ
Đ
        if(Input.GetKeyDown(KeyCode.E)){Đ
```

```
        Đ
        target = targetingSystem.GetTarget();Đ
        if(target == null){return;}Đ
Đ
            if(target.GetComponent<IInteractable>() != null){Đ
                if(Vector3.Distance(target.transform.position,transform.position)
< 10f)Đ
                {Đ
                target.GetComponent<IInteractable>().Interact(transform);Đ
                }Đ
    }Đ
  }Đ
  }Đ
Đ
  private void HandleMovement()Đ
  {Đ
    if(stunnable != null && stunnable.isStunned())Đ
    {Đ
      return;Đ
    }Đ
    float horizontal = Input.GetAxis("Horizontal");Đ
    float vertical = Input.GetAxis("Vertical");Đ
Đ
    moveDirection = mainCamera.forward * vertical + mainCamera.right *
horizontal;Đ
    moveDirection.y = 0f;Đ
    moveDirection.Normalize();Đ
Đ
    if (moveDirection != Vector3.zero)Đ
    {Đ
      Quaternion targetRotation = Quaternion.LookRotation(moveDirection);Đ
      transform.rotation = Quaternion.RotateTowards(transform.rotation,
targetRotation, rotationSpeed * Time.deltaTime);Đ
Đ
    }Đ
Đ
    animator.SetFloat("Speed", moveDirection.magnitude);Đ
    rb.MovePosition(rb.position + moveDirection * moveSpeed * Time.deltaTime);Đ
  }Đ
Đ
  private void HandleJump()Đ
  {Đ
    if(stunnable != null && stunnable.isStunned())Đ
    {Đ
      return;Đ
```

```
        }Đ
        isGrounded = Physics.Raycast(transform.position, Vector3.down, 0.4f,
groundLayer);Đ
Đ
        if (Input.GetButtonDown("Jump") && isGrounded)Đ
        {Đ
            rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);Đ
        }Đ
Đ
        // animator.SetBool("IsGrounded", isGrounded);Đ
    }Đ
Đ
    private void HandleCamera()Đ
    {Đ
        float mouseX = Input.GetAxis("Mouse X");Đ
        float mouseY = Input.GetAxis("Mouse Y");Đ
Đ
        cameraRotationY -= mouseY * cameraRotationSpeed;Đ
        cameraRotationY = Mathf.Clamp(cameraRotationY, -80f, 80f);Đ
Đ
        mainCamera.RotateAround(cameraTarget.position, Vector3.up, mouseX *
cameraRotationSpeed);Đ
        mainCamera.localRotation = Quaternion.Euler(cameraRotationY,
mainCamera.localEulerAngles.y, 0f);Đ
Đ
        Vector3 cameraOffset = new Vector3(0f, cameraHeight, -cameraDistance);Đ
        Vector3 targetPosition = cameraTarget.position +
mainCamera.TransformDirection(cameraOffset);Đ
Đ
        mainCamera.position = Vector3.Lerp(mainCamera.position, targetPosition,
Time.deltaTime * rotationSpeed);Đ
        mainCamera.LookAt(cameraTarget);Đ
    }Đ
Đ
    public void faceIndirectionOfCamera()Đ
    {Đ
        transform.rotation = Quaternion.Euler(0f, mainCamera.localEulerAngles.y,
0f);Đ
    }Đ
    public bool TryUnlockSkillNode(SkillNode skillNode)Đ
    {Đ
        if (skillNode == null)Đ
        {Đ
            Debug.LogWarning("Invalid skill node.");Đ
            return false;Đ
```

```
        }Đ
        if (skillNode.isUnlocked)Đ
        {Đ
            Debug.LogWarning("Already learned.");Đ
            return false;Đ
        }Đ
Đ
        // Check if the character has enough skill points to unlock the node.Đ
        if (skillController.availableSkillPoints < skillNode.skillPointCost)Đ
        {Đ
            Debug.LogWarning("Not enough skill points.");Đ
            return false;Đ
        }Đ
Đ
        // Check if the required main stat meets the node's requirement.Đ
        bool statRequirementsMet = true;Đ
        for (int i = 0; i < skillNode.mainStatRequirement.Count; i++)Đ
        {Đ
            Archetype statName = skillNode.mainStatRequirement[i];Đ
            int requiredValue = skillNode.mainStatValue[i];Đ
Đ
            switch (statName)Đ
            {Đ
                case Archetype.Strength:Đ
                    if (characterStats.strength < requiredValue) statRequirementsMet =
false;Đ
                    break;Đ
                case Archetype.Intelligence:Đ
                    if (characterStats.intelligence < requiredValue) statRequirementsMet
= false;Đ
                    break;Đ
                case Archetype.Dexterity:Đ
                    if (characterStats.dexterity < requiredValue) statRequirementsMet =
false;Đ
                    break;Đ
                case Archetype.Endurance:Đ
                    if (characterStats.endurance < requiredValue) statRequirementsMet =
false;Đ
                    break;Đ
                case Archetype.Wisdom:Đ
                    if (characterStats.wisdom < requiredValue) statRequirementsMet =
false;Đ
                    break;Đ
                default:Đ
                    Debug.LogWarning("Invalid stat name in the skill node.");Đ
```

```
                break;Đ
            }Đ
        }Đ
Đ
        if (!statRequirementsMet)Đ
        {Đ
            Debug.LogWarning("Main stat requirement not met.");Đ
            return false;Đ
        }Đ
Đ
        // Check if the required prerequisite skill has been unlocked.Đ
        if (skillNode.prerequisiteSkill != null && !
skillNode.prerequisiteSkill.isUnlocked)Đ
        {Đ
            Debug.LogWarning("Prerequisite skill not unlocked.");Đ
            return false;Đ
        }Đ
Đ
        // Check if the skill node is visible based on the fog of war mechanic.Đ
        if (!skillController.skillTree.IsVisible(skillNode))Đ
        {Đ
            Debug.LogWarning("Skill node is not visible.");Đ
            return false;Đ
        }Đ
Đ
        // Unlock the skill node.Đ
        skillNode.isUnlocked = true;Đ
Đ
        skillController.LearnSkill(skillNode);Đ
Đ
Đ
        return true;Đ
    }Đ
    public bool TryUnLearnSkillNode(SkillNode skillNode)Đ
    {Đ
        if (skillNode.isUnlocked == false)Đ
        {Đ
            return false;Đ
        }Đ
        skillNode.isUnlocked = false;Đ
        skillController.UnlearnSkill(skillNode);Đ
        return true;Đ
    }Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\PlayerController.csĐ
PlayerController:Đ
- buffSystemĐ
- experienceSystemĐ
- characterStatsĐ
- combatControllerĐ
- skillControllerĐ
- skillTreeĐ
- stunnableĐ
- hotkeyControllerĐ
- canGrabControllerĐ
- targetingSystemĐ
- Ability1Đ
- moveSpeedĐ
- rotationSpeedĐ
- jumpForceĐ
- groundLayerĐ
- cameraTargetĐ
- cameraDistanceĐ
- cameraHeightĐ
- cameraRotationSpeedĐ
- rbĐ
- animatorĐ
- moveDirectionĐ
- isGroundedĐ
- mainCameraĐ
- cameraRotationYĐ
Đ
Start():Đ
- canGrabController = GetComponent(CanGrabController)Đ
- combatController = GetComponent(CharacterCombatController)Đ
- characterStats = GetComponent(CharacterStats)Đ
- skillController = GetComponent(SkillController)Đ
- targetingSystem = GetComponent(TargetingSystem)Đ
- skillController.skillTree.resetAllNodes()Đ
- rb = GetComponent(Rigidbody)Đ
- animator = GetComponent(Animator)Đ
- mainCamera = Camera.main.transformĐ
- Cursor.lockState = CursorLockMode.LockedĐ
- Cursor.visible = falseĐ
- skillController.OnSkillUnlocked += UpdateToSkillEventsĐ
- stunnable = GetComponent(IStunnable)Đ

Đ
Update():Đ
- HandleMovement()Đ
- HandleJump()Đ
- HandleCamera()Đ
- HandleActions()Đ
Đ
UpdateToSkillEvents(SkillNode node):Đ
- characterStats.UpdateSubStats()Đ
Đ
HandleActions():Đ
- target = targetingSystem.GetTarget()Đ
- if target == null, returnĐ
- if target.GetComponent(IInteractable) != null:Đ
** - if Vector3.Distance(target.transform.position,transform.position) < 10f:Đ**
**   - target.GetComponent(IInteractable).Interact(transform)Đ**
Đ
HandleMovement():Đ
- if stunnable != null and stunnable.isStunned(), returnĐ
- horizontal = Input.GetAxis("Horizontal")Đ
- vertical = Input.GetAxis("Vertical")Đ
- moveDirection = mainCamera.forward * vertical + mainCamera.right * horizontalĐ
- moveDirection.y = 0fĐ
- moveDirection.Normalize()Đ
- if moveDirection != Vector3.zero:Đ
** - targetRotation = Quaternion.LookRotation(moveDirection)Đ**
** - transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, rotationSpeed * Time.deltaTime)Đ**
- animator.SetFloat("Speed", moveDirection.magnitude)Đ
- rb.MovePosition(rb.position + moveDirection * moveSpeed * Time.deltaTime)Đ
Đ
HandleJump():Đ
- if stunnable != null and stunnable.isStunned(), returnĐ
- isGrounded = Physics.Raycast(transform.position, Vector3.down, 0.4f, groundLayer)Đ
- if Input.GetButtonDown("Jump") and isGrounded:Đ
** - rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse)Đ**
Đ
HandleCamera():Đ
- mouseX = Input.GetAxis("Mouse X")Đ
- mouseY = Input.GetAxis("Mouse Y")Đ
- cameraRotationY -= mouseY * cameraRotationSpeedĐ
- cameraRotationY = Mathf.Clamp(cameraRotationY, -80f, 80f)Đ
- mainCamera.RotateAround(cameraTarget.position, Vector3.up, mouseX * cameraRotationSpeed)Đ

- mainCamera.localRotation = Quaternion.Euler(cameraRotationY, mainCamera.localEulerAngles.y, 0f)Đ
- cameraOffset = new Vector3(0f, cameraHeight, -cameraDistance)Đ
- targetPosition = cameraTarget.position + mainCamera.TransformDirection(cameraOffset)Đ
- mainCamera.position = Vector3.Lerp(mainCamera.position, targetPosition, Time.deltaTime * rotationSpeed)Đ
- mainCamera.LookAt(cameraTarget)Đ
Đ
faceIndirectionOfCamera():Đ
- transform.rotation = Quaternion.Euler(0f, mainCamera.localEulerAngles.y, 0f)Đ
Đ
TryUnlockSkillNode(SkillNode skillNode):Đ
- if skillNode == null, Debug.LogWarning("Invalid skill node."), return falseĐ
- if skillNode.isUnlocked, Debug.LogWarning("Already learned."), return falseĐ
- if skillController.availableSkillPoints < skillNode.skillPointCost, Debug.LogWarning("Not enough skill points."), return falseĐ
- statRequirementsMet = trueĐ
- for i = 0 to skillNode.mainStatRequirement.Count:Đ
  - statName = skillNode.mainStatRequirement[i]Đ
  - requiredValue = skillNode.mainStatValue[i]Đ
  - if statName == Archetype.Strength:Đ
    - if characterStats.strength < requiredValue, statRequirementsMet = falseĐ
  - elif statName == Archetype.Intelligence:Đ
    - if characterStats.intelligence < requiredValue, statRequirementsMet = falseĐ
  - elif statName == Archetype.Dexterity:Đ
    - if characterStats.dexterity < requiredValue, statRequirementsMet = falseĐ
  - elif statName == Archetype.Endurance:Đ
    - if characterStats.endurance < requiredValue, statRequirementsMet = falseĐ
  - elif statName == Archetype.Wisdom:Đ
    - if characterStats.wisdom < requiredValue, statRequirementsMet = falseĐ
  - else:Đ
    - Debug.LogWarning("Invalid stat name in the skill node.")Đ
- if not statRequirementsMet, Debug.LogWarning("Main stat requirement not met."), return falseĐ
- if skillNode.prerequisiteSkill != null and not skillNode.prerequisiteSkill.isUnlocked, Debug.LogWarning("Prerequisite skill not unlocked."), return falseĐ
- if not skillController.skillTree.IsVisible(skillNode), Debug.LogWarning("Skill node is not visible."), return falseĐ
- skillNode.isUnlocked = trueĐ
- skillController.LearnSkill(skillNode)Đ
- return trueĐ
Đ
TryUnLearnSkillNode(SkillNode skillNode):Đ

- if not skillNode.isUnlocked, return falseÐ
- skillNode.isUnlocked = falseÐ
- skillController.UnlearnSkill(skillNode)Ð
- return true

- SkillController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\SkillController.cs:
Summary of SkillController:
File name: SkillController.csÐ
Ð
Summary: This class handles the management of skills for a character in the game. It contains a list of currently active skills, a skill tree, the character's available skill points, a modifier for the character's overall stats, and two events for when a skill is learned or unlearned. Ð
Ð
The LearnSkill method adds a new skill to the character's active skills list, applies the skill's effects to the character's stats, subtracts the skill's point cost from the available skill points, and invokes the OnSkillUnlocked event.Ð
Ð
The UnlearnSkill method removes a skill from the character's active skills list, removes the skill's effects from the character's stats, adds the skill's point cost back to the available skill points, and invokes the OnSkillUnlearnd event.Ð
Ð
Overall, this class allows the player to manage their character's skills and progress through the game's skill tree.
Code of file SkillController:
þÿusing System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
class SkillController : MonoBehaviourÐ
{Ð
    public List<Skill> activeSkills;Ð
    public SkillTree skillTree;Ð
    public int availableSkillPoints;Ð
    public StatsModifier totalStatsModier;Ð
    public delegate void SkillEvent(SkillNode skillNode);Ð
    public event SkillEvent OnSkillUnlocked;Ð
    public event SkillEvent OnSkillUnlearnd;Ð
Ð
Ð
Ð
    public void LearnSkill(SkillNode skillNode)Ð
    {Ð
Ð
        // Call event to update the UI, etc.Ð

Đ
activeSkills.Add(skillNode.skill);Đ

skillNode.skill.ApplySkill(this.gameObject.GetComponent<CharacterStats>());Đ
totalStatsModier.Add(skillNode.skill.statModifier);Đ
availableSkillPoints -= skillNode.skillPointCost;Đ
OnSkillUnlocked?.Invoke(skillNode);Đ
Đ
}Đ
public void UnlearnSkill(SkillNode skillNode)Đ
{Đ
if (activeSkills.Remove(skillNode.skill))Đ
{Đ
totalStatsModier.Sub(skillNode.skill.statModifier);Đ
}Đ
availableSkillPoints += skillNode.skillPointCost;Đ

skillNode.skill.RemoveSkill(this.gameObject.GetComponent<CharacterStats>());Đ
OnSkillUnlearnd?.Invoke(skillNode);Đ
}Đ
Đ
}Đ
Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\SkillController.csĐ
Class: SkillControllerĐ
Đ
- activeSkills: List<Skill>Đ
- skillTree: SkillTreeĐ
- availableSkillPoints: intĐ
- totalStatsModier: StatsModifierĐ
- OnSkillUnlocked: delegate void SkillEvent(SkillNode skillNode)Đ
- OnSkillUnlearnd: delegate void SkillEvent(SkillNode skillNode)Đ
Đ
+ LearnSkill(skillNode: SkillNode)Đ
- activeSkills.Add(skillNode.skill)Đ
- skillNode.skill.ApplySkill(getComponent(CharacterStats))Đ
- totalStatsModier.Add(skillNode.skill.statModifier)Đ
- availableSkillPoints -= skillNode.skillPointCostĐ
- OnSkillUnlocked?.Invoke(skillNode)Đ
Đ
+ UnlearnSkill(skillNode: SkillNode)Đ
- if activeSkills.Remove(skillNode.skill)Đ

- totalStatsModier.Sub(skillNode.skill.statModifier)Đ
- availableSkillPoints += skillNode.skillPointCostĐ
- skillNode.skill.RemoveSkill(getComponent(CharacterStats))Đ
- OnSkillUnlearnd?.Invoke(skillNode)

- TargetingSystem at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\TargetingSystem.cs:
Summary of TargetingSystem:
File Name: TargetingSystem.csĐ
Đ
This class provides functionality for targeting game objects in the game world using the player camera and crosshair as the main tools for selecting targets. Đ
Đ
Public Variables:Đ
- playerCamera: The camera used to target objects.Đ
- targetLayerMask: The layer mask used to determine which objects can be targeted.Đ
- currentTarget: The current target being selected.Đ
- crosshair: The graphical representation of the target.Đ
- maxTargetingDistance: The maximum distance at which targets can be selected.Đ
- highlightMaterial: The material used to highlight the selected target.Đ
Đ
Private Variables:Đ
- lastTarget: The previously selected target.Đ
- originalMaterial: The original material of the targeted object.Đ
- outlineHighlightController: The object responsible for outlining the current target.Đ
Đ
Methods:Đ
- HandleCrosshairTargeting(): Handles targeting by raycasting from the player camera to the crosshair position.Đ
- HandleMouseClickTargeting(): Handles targeting by raycasting from the player camera to the mouse click position.Đ
- GetTarget(): Returns the current target.Đ
- HighlightTarget(): Applies a highlight material to the selected target and outlines it using the outlineHighlightController object.
Code of file TargetingSystem:
þÿusing UnityEngine;Đ
Đ
public class TargetingSystem : MonoBehaviourĐ
{Đ
    public Camera playerCamera;Đ
    public LayerMask targetLayerMask;Đ
    public GameObject currentTarget;Đ
    public GameObject crosshair;Đ

```
    public float maxTargetingDistance = 100f;Ð
    public Material highlightMaterial;Ð
    private GameObject lastTarget;Ð
    private Material originalMaterial;Ð
    public OutlineHighlight outlineHighlightController;Ð
Ð
    private void Update()Ð
    {Ð
        HandleCrosshairTargeting();Ð
        HandleMouseClickTargeting();Ð
        HighlightTarget();Ð
    }Ð
    private void Start()Ð
    {Ð
Ð
    }Ð
Ð
    private void HandleCrosshairTargeting()Ð
    {Ð
        RaycastHit hit;Ð
        Ray ray = playerCamera.ScreenPointToRay(crosshair.transform.position);Ð
Ð
        if (Physics.Raycast(ray, out hit, maxTargetingDistance, targetLayerMask))Ð
        {Ð
            currentTarget = hit.collider.gameObject;Ð
        }Ð
        elseÐ
        {Ð
            currentTarget = null;Ð
        }Ð
    }Ð
Ð
    private void HandleMouseClickTargeting()Ð
    {Ð
        if (Input.GetMouseButtonDown(0))Ð
        {Ð
            RaycastHit hit;Ð
            Ray ray = playerCamera.ScreenPointToRay(Input.mousePosition);Ð
Ð
            if (Physics.Raycast(ray, out hit, maxTargetingDistance, targetLayerMask))Ð
            {Ð
                currentTarget = hit.collider.gameObject;Ð
            }Ð
        }Ð
    }Ð
```

```
Đ
    public GameObject GetTarget()Đ
    {Đ
        return currentTarget;Đ
    }Đ
    private void HighlightTarget()Đ
    {Đ
        if (currentTarget != null)Đ
        {Đ
            outlineHighlightController.target = currentTarget.transform;Đ
            lastTarget = currentTarget;Đ
        }Đ
        elseĐ
        {Đ
            outlineHighlightController.target = null;Đ
        }Đ
    }Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\TargetingSystem.csĐ
TargetingSystem:Đ
- playerCamera: CameraĐ
- targetLayerMask: LayerMaskĐ
- currentTarget: GameObjectĐ
- crosshair: GameObjectĐ
- maxTargetingDistance: floatĐ
- highlightMaterial: MaterialĐ
- lastTarget: GameObjectĐ
- originalMaterial: MaterialĐ
- outlineHighlightController: OutlineHighlightĐ

Đ
+ Update()Đ
- HandleCrosshairTargeting()Đ
- HandleMouseClickTargeting()Đ
- HighlightTarget()Đ
+ Start()Đ
+ GetTarget(): GameObjectĐ

Đ
Note: This class represents a system for targeting game objects in the game world, based on player input and the position of the crosshair on the screen. It also provides functionality for highlighting and selecting the target, using an outline highlighting effect.

- VisualEffectController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\VisualEffectController.cs:
Summary of VisualEffectController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\VisualEffectController.csÐ
The file "VisualEffectController.cs" is a script written in C# for a Unity project. The purpose of this script is to control the visual effects used in the game for both players and units. The script includes an enumeration called "effectUnitPosition" which has two values: "overHead" and "underFeet". The class "VisualEffectController" is defined and inherits from Unity's "MonoBehaviour" class. Ð
Ð
The script includes several public and private variables. The public variables are "visualEffectManager" of type "VisualEffectManager" and "positionOverHead" and "positionUnderFeet" of type "Transform". The private variables are "goalTransform" of type "Transform" and "effectInstances" of type "List<(GameObject, float)>". Ð
Ð
The "SpawnEffect" method is defined which takes in a string "effectName", a float "effectDuration", and an enum "effectPosition". The method checks the "effectPosition" and sets the "goalTransform" to either "positionOverHead" or "positionUnderFeet" accordingly. The method then gets the effect prefab from the "visualEffectManager" and instantiates it as an "effectInstance". This instance is added to the "effectInstances" list along with a timestamp of when it should be destroyed. Ð
Ð
The "Update" method is also defined which is called every frame. This method iterates over the "effectInstances" list and checks if any of the instances should be destroyed based on their timestamp. If an effect instance should be destroyed, it is removed from the "effectInstances" list and then destroyed. Ð
Ð
Overall, the "VisualEffectController.cs" script handles the spawning and destruction of visual effects in the game. It keeps track of active effect instances and removes them when they are no longer needed.
Code of file VisualEffectController:

```
using System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
public enum effectUnitPosition{Ð
    overHead,Ð
    underFeet,Ð
}Ð
public class VisualEffectController : MonoBehaviourÐ
```

```
{Đ
   public VisualEffectManager visualEffectManager;Đ
Đ
   public Transform positionOverHead;Đ
   public Transform positionUnderFeet;Đ
Đ
   private Transform goalTransform;Đ
   private List<(GameObject,float)> effectInstances = new
List<(GameObject,float)>();Đ
   public void SpawnEffect(string effectName, float effectDuration = 0,
effectUnitPosition effectPosition = effectUnitPosition.overHead)Đ
   {Đ
      if(effectPosition == effectUnitPosition.overHead){Đ
         goalTransform = positionOverHead;Đ
      }Đ
      if(effectPosition == effectUnitPosition.underFeet){Đ
         goalTransform = positionUnderFeet;Đ
      }Đ
Đ
       GameObject effectPrefab =
visualEffectManager.GetEffectPrefab(effectName);Đ
      if (effectPrefab != null)Đ
      {Đ
         GameObject effectInstance = Instantiate(effectPrefab, Vector3.zero,
Quaternion.identity, goalTransform);Đ
         effectInstance.transform.localPosition = Vector3.zero;Đ
         effectInstances.Add((effectInstance,Time.time+effectDuration));Đ
Đ
      }Đ
   }Đ
   void Update(){Đ
      for (int i = effectInstances.Count - 1; i >= 0; i--)Đ
      {Đ
         (GameObject, float) effectInstance = effectInstances[i];Đ
         if (effectInstance.Item2 < Time.time)Đ
         {Đ
            Destroy(effectInstance.Item1);Đ
            effectInstances.RemoveAt(i);Đ
            Debug.Log("effect removed");Đ
         }Đ
      }Đ
   }Đ
   Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\PlayerAndUnitsComponent\VisualEffectController.csĐ
Class: VisualEffectControllerĐ
Đ
Fields:Đ
- visualEffectManagerĐ
- positionOverHeadĐ
- positionUnderFeetĐ
- goalTransformĐ
- effectInstancesĐ
Đ
Methods:Đ
- SpawnEffect(effectName, effectDuration, effectPosition)Đ
- Update()

- GameEvent at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\GameEvent.cs:
Summary of GameEvent:
This code file is implementing a custom game event system using Unity's built-in event system. The class is called "GameEvent", and it inherits from the "UnityEvent" class, which allows the event to have parameters of a specific type (in this case, a string).Đ
Đ
The "[System.Serializable]" attribute is used so that instances of this class can be serialized and saved in the Unity editor.Đ
Đ
The generic parameter of the "UnityEvent" class is set to a string type, which means that any listeners to this event will receive a string parameter when the event is invoked.Đ
Đ
The purpose of this implementation is to create a custom event system for a game that can trigger specific actions or events based on certain conditions. For example, a "GameOver" event could be defined, and any listeners to that event could respond by displaying a game over screen or playing a sound effect.Đ
Đ
Overall, this class provides a simple, flexible way to create and use custom game events within a Unity project.
Code of file GameEvent:
using UnityEngine.Events;Đ
Đ
[System.Serializable]Đ
public class GameEvent : UnityEvent<string> { }Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\GameEvent.csĐ
File: GameEvent.csĐ
Đ
Class: GameEventĐ
- [System.Serializable]Đ
- : UnityEvent<string>Đ
- UnityEventĐ
- UnityEventBaseĐ
- InvokableCallListĐ
- List<BaseInvokableCall>Đ
- PersistentCallGroupĐ
- List<PersistentCall>Đ
- Delegate[] m_InvokeArrayĐ

- KillObjective at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\KillObjective.cs:
Summary of KillObjective:
File Name: KillObjective.csĐ
Đ
This script defines a class called KillObjective, which is a subclass of QuestObjective. It contains three serialized fields: enemyId (string), targetKills (int), and currentKills (int) which represent the unique identifier of the enemy, the number of enemies to be killed, and the current number of enemies killed, respectively.Đ
Đ
The KillObjective class has a constructor that takes in an id (string), description (string), enemyId (string), and targetKills (int) as parameters to initialize the instance variables.Đ
Đ
The script also overrides three methods from the QuestObjective superclass: UpdateProgress(), GetObjectiveProgress(), and IsCompleted(). Đ
Đ
UpdateProgress() takes in a killedEnemyId (string) parameter and updates the currentKills count if the killedEnemyId matches the id of the enemy specified in enemyId. If the number of currentKills reaches the targetKills, then the status is marked as completed.Đ
Đ
GetObjectiveProgress() returns a string that represents the current number of kills and the target number of kills.Đ
Đ
IsCompleted() returns a boolean indicating whether the status of the objective is completed or not.Đ

Đ
The script uses the Unity Debug.LogError() method to print out the current number of kills for debugging purposes. The script also explicitly uses the UnityEngine namespace.
Code of file KillObjective:

```
using UnityEngine;Đ
Đ
public class KillObjective : QuestObjectiveĐ
{Đ
    [SerializeField]Đ
    public string enemyId;Đ
    [SerializeField]Đ
    public int targetKills;Đ
    [SerializeField]Đ
    public int currentKills;Đ
Đ
    public KillObjective(string id, string description, string enemyId, int targetKills)Đ
    {Đ
        this.id = id;Đ
        this.description = description;Đ
        this.enemyId = enemyId;Đ
        this.targetKills = targetKills;Đ
        this.currentKills = 0;Đ
        this.status = ObjectiveStatus.Incomplete;Đ
    }Đ
Đ
    public override void UpdateProgress(string killedEnemyId)Đ
    {Đ
        if (killedEnemyId == "kill:"+enemyId && status != ObjectiveStatus.Completed)Đ
        {Đ
            currentKills++;Đ
            Debug.LogError("Current Kills: " + currentKills);Đ
            if (currentKills >= targetKills)Đ
            {Đ
                status = ObjectiveStatus.Completed;Đ
            }Đ
        }Đ
    }Đ
    public override string GetObjectiveProgress()Đ
    {Đ
        return currentKills + "/" + targetKills;Đ
    }Đ
Đ
    public override bool IsCompleted()Đ
    {Đ
```

```
        return status == ObjectiveStatus.Completed;Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\QuestFiles\KillObjective.csĐ
File: KillObjective.csĐ
Đ
Class: KillObjective : QuestObjectiveĐ
- [SerializeField] enemyId : stringĐ
- [SerializeField] targetKills : intĐ
- [SerializeField] currentKills : intĐ
Đ
+ KillObjective(id : string, description : string, enemyId : string, targetKills : int)Đ
- this.id = idĐ
- this.description = descriptionĐ
- this.enemyId = enemyIdĐ
- this.targetKills = targetKillsĐ
- this.currentKills = 0Đ
- this.status = ObjectiveStatus.IncompleteĐ
Đ
+ UpdateProgress(killedEnemyId : string)Đ
- if(killedEnemyId == "kill:"+enemyId && status != ObjectiveStatus.Completed)Đ
    + currentKills++Đ
    + Debug.LogError("Current Kills: " + currentKills)Đ
    - if(currentKills >= targetKills)Đ
        - status = ObjectiveStatus.completedĐ
Đ
+ GetObjectiveProgress() : stringĐ
- return currentKills + "/" + targetKillsĐ
Đ
+ IsCompleted() : boolĐ
- return status == ObjectiveStatus.Completed

- Quest at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\QuestFiles\Quest.cs:
Summary of Quest:
File name: Quest.csĐ
Đ
Summary: The Quest class is a ScriptableObject that contains information about
a quest such as its ID, title, description, objectives, rewards, and status. It has
methods to add objectives and rewards, and to check and update objectives for
completion. It also has a method to check if all objectives are complete and
update the quest status accordingly.Đ
```

Đ
Intern logic:Đ
- The Quest class has public properties for its ID, title, description, objectives, rewards, and status.Đ
- Upon creation, a Quest object is initialized with its ID, title, description, and lists for objectives and rewards.Đ
- The CheckAndUpdateObjectives method loops through all objectives in the list and updates their progress if they are incomplete. If an objective is completed, it checks if all objectives are completed using the CheckQuestCompletion method.Đ
- The CheckQuestCompletion method loops through all objectives and checks if any are not completed. If all objectives are completed, the quest status is updated to completed.Đ
Đ
Note: The implementation is missing the definition for the QuestObjective and Reward classes/enums, which would contain information about objectives and rewards for the quest.
Code of file Quest:

```
using UnityEngine;Đ
using System.Collections.Generic;Đ
Đ
[CreateAssetMenu(fileName = "Quest", menuName = "ScriptableObjects/Quest", order = 1)]Đ
public class Quest : ScriptableObjectĐ
{Đ
    public int id;Đ
    public string title;Đ
    public string description;Đ
    public List<QuestObjective> objectives;Đ
    public List<Reward> rewards;Đ
    public QuestStatus status;Đ
Đ
    public Quest(int id, string title, string description)Đ
    {Đ
        this.id = id;Đ
        this.title = title;Đ
        this.description = description;Đ
        this.objectives = new List<QuestObjective>();Đ
        this.rewards = new List<Reward>();Đ
        this.status = QuestStatus.NotStarted;Đ
    }Đ
Đ
    public void AddObjective(QuestObjective objective)Đ
    {Đ
        objectives.Add(objective);Đ
    }Đ
```

```
Đ
    public void AddReward(Reward reward)Đ
    {Đ
        rewards.Add(reward);Đ
    }Đ
Đ
    // The missing CheckAndUpdateObjectives methodĐ
    public void CheckAndUpdateObjectives(string objectiveId)Đ
    {Đ
        foreach (QuestObjective objective in objectives)Đ
        {Đ
            if ( objective.status == ObjectiveStatus.Incomplete)Đ
            {Đ
                objective.UpdateProgress(objectiveId);Đ
                if (objective.status == ObjectiveStatus.Completed)Đ
                {Đ
                    CheckQuestCompletion();Đ
                }Đ
                break;Đ
            }Đ
        }Đ
    }Đ
Đ
    private void CheckQuestCompletion()Đ
    {Đ
        bool allObjectivesComplete = true;Đ
        foreach (QuestObjective objective in objectives)Đ
        {Đ
            if (objective.status != ObjectiveStatus.Completed)Đ
            {Đ
                allObjectivesComplete = false;Đ
                break;Đ
            }Đ
        }Đ
Đ
        if (allObjectivesComplete)Đ
        {Đ
            status = QuestStatus.Completed;Đ
        }Đ
    }Đ
}Đ
public enum QuestStatus { NotStarted, InProgress, Completed }
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\QuestFiles\Quest.csĐ
```

QuestĐ
- id: intĐ
- title: stringĐ
- description: stringĐ
- objectives: List<QuestObjective>Đ
- rewards: List<Reward>Đ
- status: QuestStatusĐ
Đ
+ Quest(id: int, title: string, description: string)Đ
+ AddObjective(objective: QuestObjective): voidĐ
+ AddReward(reward: Reward): voidĐ
+ CheckAndUpdateObjectives(objectiveId: string): voidĐ
- CheckQuestCompletion(): voidĐ
Đ
QuestStatusĐ
- NotStartedĐ
- InProgressĐ
- Completed

- QuestAction at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestAction.cs:
Summary of QuestAction:
The file name of this class is "QuestAction". It is a ScriptableObject and can be created in the Unity editor menu under the "QuestSystem/QuestAction" category. Đ
Đ
The purpose of this class is to hold a unique "actionId" string, which can be used to identify and trigger specific actions within a quest system. Đ
Đ
Since this is a ScriptableObject and not a MonoBehaviour, it cannot be attached to game objects directly. Instead, it can be instantiated and referenced by other scripts. Đ
Đ
Overall, this class is a simple data container with the sole responsibility of holding an action identifier.
Code of file QuestAction:
using UnityEngine;Đ
Đ
[CreateAssetMenu(fileName = "QuestAction", menuName = "QuestSystem/QuestAction", order = 1)]Đ
public class QuestAction : ScriptableObjectĐ
{Đ
**   public string actionId;Đ**
}Đ

Corresponding SyntaxTree:

C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestAction.csĐ
Class: QuestActionĐ
Đ
- fileName: string = "QuestAction"Đ
- menuName: string = "QuestSystem/QuestAction"Đ
- order: int = 1Đ
- extends: ScriptableObjectĐ
- actionId: string

- QuestGiver at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestGiver.cs:
Summary of QuestGiver:
Name of File: QuestGiver.csĐ
Đ
Summary: Đ
Đ
The QuestGiver class is a Unity MonoBehaviour that implements the IInteractable interface. It has two serialized fields quest and interactionIndicator which are a Quest object and a GameObject respectively. Additionally, it has private variables playerInRange, playerQuestSystem and uiManager. The playerInRange variable is used to keep track of whether the player is near the QuestGiver or not. The playerQuestSystem variable stores the QuestSystem Component of the player when the player enters the QuestGiver's collider. The uiManager variable is used to find the UIManager instance in the scene.Đ
Đ
The QuestGiver has two main methods: Interact() and Update(). Interact() is called when the player interacts with the QuestGiver by pressing the E key. If the questUIPresenter Game Object is active in the hierarchy, hideQuestUiPresenter() method of uiManager instance is called to hide the Quest UI. Otherwise, showQuestUiPresenter(quest) method of uiManager instance is called to show the Quest UI of the quest. Đ
Đ
In the Update() method, if the player is in range and presses the E key, then Interact() method is called. Đ
Đ
Lastly, the QuestGiver class has two private methods: OnTriggerEnter() and OnTriggerExit() which are used to detect when the player enters or exits the QuestGiver's collider. If the player enters the collider, playerInRange is set to true and interactionIndicator is set to active. Additionally, the playerQuestSystem variable is assigned the QuestSystem Component of the player. If the player exits the collider, playerInRange is set to false and interactionIndicator is set to inactive.
Code of file QuestGiver:
using UnityEngine;Đ

```
Đ
public class QuestGiver : MonoBehaviour, IInteractableĐ
{Đ
    [SerializeField] private Quest quest;Đ
 Đ
    [SerializeField] private GameObject interactionIndicator;Đ
Đ
    private bool playerInRange = false;Đ
    private QuestSystem playerQuestSystem;Đ
    private GameObject Interacts;Đ
    private UIManager uiManager;Đ
    public void Start(){Đ
        uiManager = FindObjectOfType<UIManager>();Đ
    }Đ
Đ
    void Update()Đ
    {Đ
        if (playerInRange && Input.GetKeyDown(KeyCode.E))Đ
        {Đ
            Interact(Interacts.transform);Đ
        }Đ
    }Đ
    public void Interact(Transform interactFrom)Đ
    {Đ
        if(uiManager.questUIPresenter.gameObject.activeInHierarchy ){Đ
            uiManager.hideQuestUiPresenter();Đ
        }Đ
        else{Đ
            uiManager.showQuestUiPresenter(quest);Đ
        }Đ
        Đ
    }Đ
Đ
    void OnTriggerEnter(Collider other)Đ
    {Đ
        if (other.CompareTag("Player"))Đ
        {Đ
            playerInRange = true;Đ
            Interacts = other.gameObject;Đ
            interactionIndicator.SetActive(true);Đ
            playerQuestSystem = other.GetComponent<QuestSystem>();Đ
        }Đ
    }Đ
Đ
    void OnTriggerExit(Collider other)Đ
```

```
    {Đ
        if (other.CompareTag("Player"))Đ
        {Đ
            playerInRange = false;Đ
            interactionIndicator.SetActive(false);Đ
          Đ
        }Đ
    }Đ
Đ
Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\QuestFiles\QuestGiver.csĐ
Class: QuestGiverĐ
Đ
- Quest questĐ
- GameObject interactionIndicatorĐ
- bool playerInRangeĐ
- QuestSystem playerQuestSystemĐ
- GameObject InteractsĐ
- UIManager uiManagerĐ
Đ
Start()Đ
- uiManager = FindObjectOfType<UIManager>()Đ
Đ
Update()Đ
- if (playerInRange && Input.GetKeyDown(KeyCode.E))Đ
- Interact(Interacts.transform)Đ
Đ
Interact(Transform interactFrom)Đ
- if(uiManager.questUIPresenter.gameObject.activeInHierarchy)Đ
- uiManager.hideQuestUiPresenter()Đ
- elseĐ
- uiManager.showQuestUiPresenter(quest)Đ
Đ
OnTriggerEnter(Collider other)Đ
- if (other.CompareTag("Player"))Đ
- playerInRange = trueĐ
- Interacts = other.gameObjectĐ
- interactionIndicator.SetActive(true)Đ
- playerQuestSystem = other.GetComponent<QuestSystem>()Đ
Đ
OnTriggerExit(Collider other)Đ
```

- if (other.CompareTag("Player"))Đ
- playerInRange = falseĐ
- interactionIndicator.SetActive(false)

- QuestObjective at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestObjective.cs:
Summary of QuestObjective:
File Name: QuestObjective.csĐ
Đ
This file defines an abstract class QuestObjective with certain properties and methods. It has a string id and description associated with the objective, an ObjectiveStatus which is an enum consisting of Completed and Incomplete values. QuestObjective is an abstract class and has two abstract methods which are IsCompleted and UpdateProgress.Đ
Đ
KillObjective: It is a class that inherits from QuestObjective class and has additional properties such as enemyId, targetKills, and currentKills. It has a constructor that takes in the id, description, enemyId and targetKills as parameters and initializes currentKills and status to zero and Incomplete respectively. UpdateProgress method increments currentKills when a specific enemy is killed and sets status to Completed when the required number of kills is achieved. GetObjectiveProgress returns the current kill to target kill ratio. IsCompleted method checks if status is Completed or not.Đ
Đ
GatherObjective: It is also a class that inherits from QuestObjective, similar to the KillObjective class. This class has properties such as itemId, targetItems and currentItems. It has a constructor that takes in id, description, itemId and targetItems. CurrentItems and Status are initialized to zero and Incomplete respectively. UpdateProgress method increments currentItems when the player picks up a specific item and changes status to Completed when the required number of items are gathered. GetObjectiveProgress returns current item count to target item count ratio. IsCompleted method checks if status is Completed or not.Đ
Đ
InspectObjective: This class inherits from the QuestObjective class and has properties such as locationId and locationInspected. It has a constructor that takes in id, description, and locationId. LocationInspected and Status are initialized to false and Incomplete, respectively. UpdateProgress method sets locationInspected to true when the player visits a specific location and sets Status to Completed. GetObjectiveProgress returns whether the location is inspected or not and IsCompleted checks whether the location is inspected or not.Đ
Đ
ActivateObjective: This class also inherits from QuestObjective similar to the above classes and has properties such as altarId and altarActivated. It has a

constructor that takes in id, description and altarId as parameters. AltarActivated and Status is initialized to false and Incomplete. UpdateProgress method sets altarActivated to true when the player uses a specific item or else when the player approaches the specific altar and sets status to Completed. GetObjectiveProgress returns whether the specific altar is activated or not and IsCompleted method checks whether the altar is activated or not.

Code of file QuestObjective:

```
using UnityEditor;Đ
using UnityEngine;Đ
Đ
public abstract class QuestObjectiveĐ
{Đ
   public string id;Đ
   public string description;Đ
   public ObjectiveStatus status;Đ
   public abstract bool IsCompleted();Đ
Đ
   public abstract void UpdateProgress(string infoId);Đ
   public abstract string GetObjectiveProgress();Đ
Đ
}Đ
public enum ObjectiveStatus { Completed,Incomplete };Đ
Đ
public class GatherObjective : QuestObjectiveĐ
{Đ
   public string itemId;Đ
   public int targetItems;Đ
   public int currentItems;Đ
Đ
   public GatherObjective(string id, string description, string itemId, int
targetItems)Đ
   {Đ
      this.id = id;Đ
      this.description = description;Đ
      this.itemId = itemId;Đ
      this.targetItems = targetItems;Đ
      this.currentItems = 0;Đ
      this.status = ObjectiveStatus.Incomplete;Đ
   }Đ
Đ
   public override void UpdateProgress(string gatheredItemId)Đ
   {Đ
      if (gatheredItemId == "gather:"+itemId && status !=
ObjectiveStatus.Completed)Đ
      {Đ
```

```
            currentItems++;Đ
            if (currentItems >= targetItems)Đ
            {Đ
                status = ObjectiveStatus.Completed;Đ
            }Đ
        }Đ
    }Đ
Đ
    public override bool IsCompleted()Đ
    {Đ
        return status == ObjectiveStatus.Completed;Đ
    }Đ
    public override string GetObjectiveProgress()Đ
    {Đ
        return currentItems + "/" + targetItems;Đ
    }Đ
}Đ
Đ
public class InspectObjective : QuestObjectiveĐ
{Đ
    public string locationId;Đ
    public bool locationInspected;Đ
Đ
    public InspectObjective(string id, string description, string locationId)Đ
    {Đ
        this.id = id;Đ
        this.description = description;Đ
        this.locationId = locationId;Đ
        this.locationInspected = false;Đ
        this.status = ObjectiveStatus.Incomplete;Đ
    }Đ
Đ
    public override void UpdateProgress(string inspectedLocationId)Đ
    {Đ
        if (inspectedLocationId == "visit:"+locationId && !locationInspected)Đ
        {Đ
            locationInspected = true;Đ
            status = ObjectiveStatus.Completed;Đ
        }Đ
    }Đ
Đ
    public override bool IsCompleted()Đ
    {Đ
        return locationInspected;Đ
    }Đ
```

```
    public override string GetObjectiveProgress()Đ
    {Đ
        return locationInspected ? "Inspected" : "Not Inspected";Đ
    }Đ
}Đ
Đ
public class ActivateObjective : QuestObjectiveĐ
{Đ
    public string altarId;Đ
    public bool altarActivated;Đ
Đ
    public ActivateObjective(string id, string description, string altarId)Đ
    {Đ
        this.id = id;Đ
        this.description = description;Đ
        this.altarId = altarId;Đ
        this.altarActivated = false;Đ
        this.status = ObjectiveStatus.Incomplete;Đ
    }Đ
Đ
    public override void UpdateProgress(string activatedAltarId)Đ
    {Đ
        if (activatedAltarId == "activate:"+altarId && !altarActivated)Đ
        {Đ
            altarActivated = true;Đ
            status = ObjectiveStatus.Completed;Đ
        }Đ
    }Đ
Đ
    public override bool IsCompleted()Đ
    {Đ
        return altarActivated;Đ
    }Đ
    public override string GetObjectiveProgress()Đ
    {Đ
        return altarActivated ? "Activated" : "Not Activated";Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\QuestFiles\QuestObjective.csĐ
Syntax Tree of QuestObjective.cs:Đ
- QuestObjectiveĐ
  - string idĐ
```

  - string descriptionĐ
  - ObjectiveStatus statusĐ
  - abstract bool IsCompleted()Đ
  - abstract void UpdateProgress(string infoId)Đ
  - abstract string GetObjectiveProgress()Đ
- enum ObjectiveStatusĐ
  - CompletedĐ
  - IncompleteĐ
- GatherObjective : QuestObjectiveĐ
  - string itemIdĐ
  - int targetItemsĐ
  - int currentItemsĐ
  - GatherObjective(string id, string description, string itemId, int targetItems)Đ
  - override void UpdateProgress(string gatheredItemId)Đ
  - override bool IsCompleted()Đ
  - override string GetObjectiveProgress()Đ
- InspectObjective : QuestObjectiveĐ
  - string locationIdĐ
  - bool locationInspectedĐ
  - InspectObjective(string id, string description, string locationId)Đ
  - override void UpdateProgress(string inspectedLocationId)Đ
  - override bool IsCompleted()Đ
  - override string GetObjectiveProgress()Đ
- ActivateObjective : QuestObjectiveĐ
  - string altarIdĐ
  - bool altarActivatedĐ
  - ActivateObjective(string id, string description, string altarId)Đ
  - override void UpdateProgress(string activatedAltarId)Đ
  - override bool IsCompleted()Đ
  - override string GetObjectiveProgress()

- HuntWolvesQuest at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\HuntWolvesQuest.cs:
Summary of HuntWolvesQuest:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\HuntWolvesQuest.csĐ
The HuntWolvesQuest.cs file is a scriptable object that represents a quest to hunt down a specific number of wolves. The file starts by using the UnityEngine and System.Collections.Generic namespaces. Đ
Đ
The scriptable object is created through the use of the CreateAssetMenu attribute, with the fileName set to "HuntWolves", the menuName to "ScriptableObjects/Quests/HuntWolves", and the order to 1. Đ
Đ
The class HuntWolvesQuest inherits from the Quest class and has a constructor

that sets the quest ID to 1, the title to "Hunt the Wolves", and the description to explain the situation and objectives for the quest. Đ
Đ
Additionally, the script adds an objective to the quest by creating a new KillObjective and calling the AddObjective method to add it to the objective list. This objective is to kill 10 wolves and bring back their pelts as proof, with the objective's title set to "HuntWolvesObjective", its description set to "Hunt 10 Wolves", its target enemy set to "Wolf", and the requirement set to 10. Đ
Đ
Overall, the implementation of HuntWolvesQuest.cs defines a quest that involves killing a certain number of wolves and bringing back their pelts to a nearby village.
Code of file HuntWolvesQuest:
using UnityEngine;Đ
using System.Collections.Generic;Đ
Đ
[CreateAssetMenu(fileName = "HuntWolves", menuName = "ScriptableObjects/Quests/HuntWolves", order = 1)]Đ
public class HuntWolvesQuest : QuestĐ
{Đ
public HuntWolvesQuest() : base(1, "Hunt the Wolves", "The village has been suffering from frequent wolf attacks. They've asked you to hunt down 10 wolves and bring back their pelts as proof.")Đ
{Đ
// Add a KillObjective to the list of objectivesĐ
AddObjective(new KillObjective("HuntWolvesObjective", "Hunt 10 Wolves", "Wolf", 10));Đ
}Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\HuntWolvesQuest.csĐ
Class: HuntWolvesQuestĐ
Đ
- CreateAssetMenuĐ
- fileName: "HuntWolves"Đ
- menuName: "ScriptableObjects/Quests/HuntWolves"Đ
- order: 1Đ
Đ
- Base: QuestĐ
- ID: 1Đ
- Title: "Hunt the Wolves"Đ
- Description: "The village has been suffering from frequent wolf attacks. They've asked you to hunt down 10 wolves and bring back their pelts as proof."Đ

Đ
- AddObjectiveĐ
  - KillObjectiveĐ
    - ID: "HuntWolvesObjective"Đ
    - Title: "Hunt 10 Wolves"Đ
    - EnemyName: "Wolf"Đ
    - TargetAmount: 10

- NewBehaviourScript at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\NewBehaviourScript.cs:
Summary of NewBehaviourScript: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\NewBehaviourScript.csĐ
The file "NewBehaviourScript.cs" is a C# script that is added as a component to a game object in the Unity engine. It contains two methods, "Start()" and "Update()", which are automatically executed by Unity when the game object is created and each frame thereafter, respectively. Đ
Đ
As it stands, this script does not do anything specific; it simply initializes the game object on creation and updates it each frame. The internal logic of the script can be expanded upon by the developer to add additional functionality, such as checking for user input or updating the game state. The "using" statements at the top of the script allow the use of built-in Unity classes and interfaces.
Code of file NewBehaviourScript:
```
using System.Collections;Đ
using System.Collections.Generic;Đ
using UnityEngine;Đ
Đ
public class NewBehaviourScript : MonoBehaviourĐ
{Đ
    // Start is called before the first frame updateĐ
    void Start()Đ
    {Đ
        Đ
    }Đ
Đ
    // Update is called once per frameĐ
    void Update()Đ
    {Đ
        Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:

C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestS\NewBehaviourScript.csÐ
NewBehaviourScript:Ð
- using System.Collections;Ð
- using System.Collections.Generic;Ð
- using UnityEngine;Ð
- class NewBehaviourScript:Ð
  - MonoBehaviourÐ
  - method Start():Ð
  - method Update():

- QuestSystem at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestSystem.cs:
Summary of QuestSystem:
File Name: QuestSystem.csÐ
Ð
Summary: This file contains the implementation of the QuestSystem class, which is responsible for managing quests in the game. It contains functions to add, remove and update quests. It also has a function to get a quest by its ID. Ð
Ð
Internally, the class uses a List to store all the quests, with each quest being an instance of the Quest class. When a quest is added, it is simply added to the list. When removing a quest, the Quest with the matching ID is found using the Find() function and then removed from the list.Ð
Ð
To update quest objectives, the UpdateQuestObjective() function loops through all quests and calls the CheckAndUpdateObjectives() function on each quest that is not completed.Ð
Ð
Overall, the QuestSystem class provides a simple and efficient way to manage quests in a game.
Code of file QuestSystem:
using UnityEngine;Ð
using System.Collections.Generic;Ð
Ð
public class QuestSystem : MonoBehaviourÐ
{Ð
    public List<Quest> quests;Ð
    public UIManager uiManager;Ð
Ð
 Ð
public Quest GetQuestByID(int questID)Ð
{Ð
    foreach (Quest quest in quests)Ð
    {Ð

```
        if (quest.id == questID)Đ
        {Đ
            return quest;Đ
        }Đ
    }Đ
    return null;Đ
}Đ
    private void Start()Đ
    {Đ
        quests = new List<Quest>();Đ
    }Đ
Đ
    public void AddQuest(Quest quest)Đ
    {Đ
        uiManager.updateQuestBook();Đ
        quests.Add(quest);Đ
    }Đ
Đ
    public void RemoveQuest(int questId)Đ
    {Đ
Đ
        Quest questToRemove = quests.Find(q => q.id == questId);Đ
        if (questToRemove != null)Đ
        {Đ
          Đ
            quests.Remove(questToRemove);Đ
        }Đ
    }Đ
        public void UpdateQuestObjective(string objectiveId)Đ
    {Đ
        uiManager.updateQuestBook();Đ
        foreach (Quest quest in quests)Đ
        {Đ
            if (quest.status != QuestStatus.Completed)Đ
            {Đ
                quest.CheckAndUpdateObjectives(objectiveId);Đ
            }Đ
        }Đ
    }Đ
Đ
   Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\QuestSystem.csĐ

File: QuestSystem.csĐ
Đ
Class: QuestSystemĐ
Đ
- quests : List<Quest>Đ
- uiManager : UIManagerĐ
Đ
+ GetQuestByID(questID : int) : QuestĐ
- Start() : voidĐ
+ AddQuest(quest : Quest) : voidĐ
+ RemoveQuest(questId : int) : voidĐ
+ UpdateQuestObjective(objectiveId : string) : void

- Reward at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\Reward.cs:
Summary of Reward:
File Name: Reward.csĐ
Đ
This file contains a class named Reward which is responsible for defining the reward properties. It has three properties, rewardId of type string, rewardName of type string, and quantity of type int. Đ
Đ
The class has a constructor that takes in three parameters, rewardId, rewardName, and quantity, and initializes the class properties with these values. Đ
Đ
This class is marked with [System.Serializable] attribute which indicates that instances of this class can be serialized to be stored persistently or sent over the network.Đ
Đ
The Reward class can be utilized by other scripts or systems to manage rewards in a game. For example, a player may earn rewards for completing certain tasks in a game. These rewards can be managed and tracked using the Reward class.
Code of file Reward:

```
using UnityEngine;Đ
Đ
[System.Serializable]Đ
public class RewardĐ
{Đ
    public string rewardId;Đ
    public string rewardName;Đ
    public int quantity;Đ
Đ
    public Reward(string rewardId, string rewardName, int quantity)Đ
    {Đ
        this.rewardId = rewardId;Đ
```

```
        this.rewardName = rewardName;Đ
        this.quantity = quantity;Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\QuestFiles\Reward.csĐ
Reward:Đ
- rewardId: stringĐ
- rewardName: stringĐ
- quantity: intĐ
- Reward(rewardId: string, rewardName: string, quantity: int)

- CharacterStatsUI at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\CharacterStatsUI.cs:
Summary of CharacterStatsUI:
File Name: CharacterStatsUI.csĐ
Đ
This file contains the implementation of the UI for displaying and updating the character's stats in the game. It uses Unity's UI system to display text and buttons on the screen. The UI displays the character's current stats, including Strength, Intelligence, Dexterity, Endurance, and Wisdom, and also shows the number of unspent stat points the character has.Đ
Đ
The class exposes public variables for the UI elements, including the Text objects for displaying the stats and the Button objects for increasing them. It also has a reference to the CharacterStats component, which manages the character's stats.Đ
Đ
The Start() method initializes the UI by setting up the button listeners and subscribing to the CharacterStats.StatsChanged event. The event is triggered whenever the character's stats change, and it calls the UpdateUI() method to update the UI with the new values.Đ
Đ
The Awake() method sets the cursor to be visible and unlocked, which is useful for UI interaction.Đ
Đ
The UpdateUI() method updates the text objects with the current stat values.Đ
Đ
The IncreaseStat() method is called whenever the player clicks on a stat button. It calls the CharacterStats.IncreaseStat() method with the appropriate Archetype enum value to increase the corresponding stat by 1.
Code of file CharacterStatsUI:
```
þÿusing UnityEngine;Đ
using UnityEngine.UI;Đ
```

```
Đ
public class CharacterStatsUI : MonoBehaviourĐ
{Đ
    public Text unspentStatPointsText;Đ
    public Text strengthText;Đ
    public Text intelligenceText;Đ
    public Text dexterityText;Đ
    public Text enduranceText;Đ
    public Text wisdomText;Đ
Đ
    public Button strengthButton;Đ
    public Button intelligenceButton;Đ
    public Button dexterityButton;Đ
    public Button enduranceButton;Đ
    public Button wisdomButton;Đ
Đ
    public CharacterStats characterStats;Đ
Đ
    private void Start()Đ
    {Đ
        strengthButton.onClick.AddListener(() => IncreaseStat(Archetype.Strength));Đ
        intelligenceButton.onClick.AddListener(() =>
IncreaseStat(Archetype.Intelligence));Đ
        dexterityButton.onClick.AddListener(() => IncreaseStat(Archetype.Dexterity));Đ
        enduranceButton.onClick.AddListener(() =>
IncreaseStat(Archetype.Endurance));Đ
        wisdomButton.onClick.AddListener(() => IncreaseStat(Archetype.Wisdom));Đ
Đ
        characterStats.StatsChanged += UpdateUI;Đ
        UpdateUI();Đ
    }Đ
Đ
    private void Awake()Đ
    {Đ
        Cursor.visible = true;Đ
        Cursor.lockState = CursorLockMode.None;Đ
    }Đ
Đ
    private void UpdateUI()Đ
    {Đ
        unspentStatPointsText.text = "Unspent Points: " +
characterStats.unspentStatPoints;Đ
        strengthText.text = "Strength: " + characterStats.strength;Đ
        intelligenceText.text = "Intelligence: " + characterStats.intelligence;Đ
        dexterityText.text = "Dexterity: " + characterStats.dexterity;Đ
```

```
        enduranceText.text = "Endurance: " + characterStats.endurance;Đ
        wisdomText.text = "Wisdom: " + characterStats.wisdom;Đ
    }Đ
Đ
    private void IncreaseStat(Archetype mainStatType)Đ
    {Đ
        characterStats.IncreaseStat(mainStatType, 1);Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\CharacterStatsUI.csĐ
CharacterStatsUI:Đ
- public Text unspentStatPointsTextĐ
- public Text strengthTextĐ
- public Text intelligenceTextĐ
- public Text dexterityTextĐ
- public Text enduranceTextĐ
- public Text wisdomTextĐ
- public Button strengthButtonĐ
- public Button intelligenceButtonĐ
- public Button dexterityButtonĐ
- public Button enduranceButtonĐ
- public Button wisdomButtonĐ
- public CharacterStats characterStatsĐ
- Start()Đ
    - strengthButton.onClick.AddListener(() -> IncreaseStat(Archetype.Strength))Đ
    - intelligenceButton.onClick.AddListener(() -> IncreaseStat(Archetype.Intelligence))Đ
    - dexterityButton.onClick.AddListener(() -> IncreaseStat(Archetype.Dexterity))Đ
    - enduranceButton.onClick.AddListener(() -> IncreaseStat(Archetype.Endurance))Đ
    - wisdomButton.onClick.AddListener(() -> IncreaseStat(Archetype.Wisdom))Đ
    - characterStats.StatsChanged += UpdateUIĐ
    - UpdateUI()Đ
- Awake()Đ
    - Cursor.visible = trueĐ
    - Cursor.lockState = CursorLockMode.NoneĐ
- UpdateUI()Đ
    - unspentStatPointsText.text = "Unspent Points: " + characterStats.unspentStatPointsĐ
    - strengthText.text = "Strength: " + characterStats.strengthĐ
    - intelligenceText.text = "Intelligence: " + characterStats.intelligenceĐ
    - dexterityText.text = "Dexterity: " + characterStats.dexterityĐ

- enduranceText.text = "Endurance: " + characterStats.enduranceÐ
- wisdomText.text = "Wisdom: " + characterStats.wisdomÐ
- IncreaseStat(Archetype mainStatType)Ð
- characterStats.IncreaseStat(mainStatType, 1)

- CharacterUi at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\CharacterUi.cs:
Summary of CharacterUi:
File: CharacterUi.csÐ
Ð
This file contains the implementation of the Character UI class that is responsible for updating the UI elements that display the stats of the character. It has references to various UI elements like Text, Image and Button that are used to display the character's stats and allow the user to interact with the game's interface. The class has a reference to a CharacterStats and UIManager classes which are used to retrieve and display the character's stats and to manage the user interface respectively.Ð
Ð
The Start() method is used to set up the UI and event listeners for the UI elements. It assigns a callback function to the onClick event of the openCharacterStatsMenu Button that opens the character stats UI when clicked, and subscribes to the StatsChanged event of the CharacterStats object. Whenever this event is fired, the UpdateUI() method is called to update the UI elements with the new stats.Ð
Ð
The Awake() method is used to set the cursor visible and unlock it when the scene starts.Ð
Ð
The UpdateUI() method is responsible for updating the Text elements that display the character's stats, and the subStatsPhysical, subStatsSpellCasting, subStatsDefensive, and subStatsUniversal Text elements that display some additional details about the character's stats. It retrieves this information from the CharacterStats object and formats it before setting it as the text content for the UI Text elements.Ð
Ð
In summary, the CharacterUi class updates the UI elements that display the character's stats and provides a way for the user to interact with the game's interface. It subscribes to the StatsChanged event of the CharacterStats object to update the UI whenever the character's stats change.
Code of file CharacterUi:
þÿusing UnityEngine;Ð
using UnityEngine.UI;Ð
Ð
public class CharacterUi : MonoBehaviourÐ
{Ð

```
    public Text unspentStatPointsText;Đ
    public Text strengthText;Đ
    public Text intelligenceText;Đ
    public Text dexterityText;Đ
    public Text enduranceText;Đ
    public Text wisdomText;Đ
Đ
Đ
Đ
    public Text subStatsPhysical;Đ
    public Text subStatsSpellCasting;Đ
    public Text subStatsDefensive;Đ
    public Text subStatsUniversal;Đ
Đ
    public Button openCharacterStatsMenu;Đ
    public Image unspentStatPoints;Đ
Đ
    public CharacterStats characterStats;Đ
    public UIManager uiManager;Đ
Đ
    private void Start()Đ
    {Đ
        openCharacterStatsMenu.onClick.AddListener(() =>
uiManager.OpenCharacterStatusUI());Đ
        characterStats.StatsChanged += UpdateUI;Đ
        UpdateUI();Đ
    }Đ
Đ
    private void Awake()Đ
    {Đ
        Cursor.visible = true;Đ
        Cursor.lockState = CursorLockMode.None;Đ
    }Đ
Đ
    private void UpdateUI()Đ
    {Đ
        strengthText.text = "Strength: " + characterStats.strength;Đ
        intelligenceText.text = "Intelligence: " + characterStats.intelligence;Đ
        dexterityText.text = "Dexterity: " + characterStats.dexterity;Đ
        enduranceText.text = "Endurance: " + characterStats.endurance;Đ
        wisdomText.text = "Wisdom: " + characterStats.wisdom;Đ
Đ
        subStatsPhysical.text = "Critical Chance: " +
characterStats.criticalChance.ToString("F1") + "%" + "\nCritical Damage: " +
characterStats.criticalDamage + "%" + "\nAttack Speed: " +
characterStats.attackSpeed.ToString("F2");Đ
```

subStatsSpellCasting.text = "Spell Crit Chc: " + characterStats.spellCriticalChance.ToString("F1") + "%" + "\nSpell Crit Dmg: " + characterStats.spellCriticalDamage + "%" + "\nCooldown: " + characterStats.cooldown;Đ
subStatsDefensive.text = "Armor: " + characterStats.armor + "\nMagic Resi: " + characterStats.magicResistance + "\nDodge Chance: " + characterStats.dodgeChance.ToString("F1") + "%";Đ
subStatsUniversal.text = "Max Life: " + characterStats.maxLife + "\nLife Reg: " + characterStats.lifeRegen + "\nMax Mana: " + characterStats.maxMana + "\nMana Reg: " + characterStats.manaRegen;Đ
}Đ
Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\CharacterUi.csĐ
Class Name: CharacterUiĐ
Đ
- public Text unspentStatPointsText;Đ
- public Text strengthText;Đ
- public Text intelligenceText;Đ
- public Text dexterityText;Đ
- public Text enduranceText;Đ
- public Text wisdomText;Đ
- public Text subStatsPhysical;Đ
- public Text subStatsSpellCasting;Đ
- public Text subStatsDefensive;Đ
- public Text subStatsUniversal;Đ
- public Button openCharacterStatsMenu;Đ
- public Image unspentStatPoints;Đ
- public CharacterStats characterStats;Đ
- public UIManager uiManager;Đ
Đ
- private void Start()Đ
- openCharacterStatsMenu.onClick.AddListener(() => uiManager.OpenCharacterStatusUI());Đ
- characterStats.StatsChanged += UpdateUI;Đ
- UpdateUI();Đ
Đ
- private void Awake()Đ
- Cursor.visible = true;Đ
- Cursor.lockState = CursorLockMode.None;Đ
Đ
- private void UpdateUI()Đ

- strengthText.text = "Strength: " + characterStats.strength;Ð
- intelligenceText.text = "Intelligence: " + characterStats.intelligence;Ð
- dexterityText.text = "Dexterity: " + characterStats.dexterity;Ð
- enduranceText.text = "Endurance: " + characterStats.endurance;Ð
- wisdomText.text = "Wisdom: " + characterStats.wisdom;Ð
- subStatsPhysical.text = "Critical Chance: " + characterStats.criticalChance.ToString("F1") + "%" + "\nCritical Damage: " + characterStats.criticalDamage + "%" + "\nAttack Speed: " + characterStats.attackSpeed.ToString("F2");Ð
- subStatsSpellCasting.text = "Spell Crit Chc: " + characterStats.spellCriticalChance.ToString("F1") + "%" + "\nSpell Crit Dmg: " + characterStats.spellCriticalDamage + "%" + "\nCooldown: " + characterStats.cooldown;Ð
- subStatsDefensive.text = "Armor: " + characterStats.armor + "\nMagic Resi: " + characterStats.magicResistance + "\nDodge Chance: " + characterStats.dodgeChance.ToString("F1") + "%";Ð
- subStatsUniversal.text = "Max Life: " + characterStats.maxLife + "\nLife Reg: " + characterStats.lifeRegen + "\nMax Mana: " + characterStats.maxMana + "\nMana Reg: " + characterStats.manaRegen;

- DamageNumberController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\DamageNumberController.cs:
Summary of DamageNumberController:
File Name: DamageNumberController.csÐ
Ð
This file contains the implementation of the DamageNumberController component which displays damage numbers in a game. It has a public TextMeshPro field that stores the text to be displayed, a floatSpeed field that determines the speed at which the number floats upwards, and a duration field that determines how long the number stays on the screen.Ð
Ð
The Update() method updates the position of the number by moving it upwards using the floatSpeed field, rotates the number to face the camera, applies a fade effect based on the elapsed time and duration, and destroys the game object after the duration.Ð
Ð
The FaceCamera() method calculates the direction to the camera, sets its y-component to zero, and rotates the number to face the camera.Ð
Ð
The SetDamageValue() method sets the text field of the TextMeshPro component to the damage value passed as a parameter, while ensuring that the component is not null.Ð
Ð
The Start() method assigns the main camera to the playerCamera field. If the TextMeshPro component is not assigned in the DamageNumberController

component, an error message is displayed and the function returns. If the TextMeshPro component is null, the game object is destroyed, and the function returns.

Code of file DamageNumberController:

```
þÿusing TMPro;Đ
using UnityEngine;Đ
Đ
public class DamageNumberController : MonoBehaviourĐ
{Đ
    public TextMeshPro textMeshPro;Đ
    public float floatSpeed = 1f;Đ
    public float duration = 1.5f;Đ
Đ
    private float elapsedTime = 0f;Đ
    private Camera playerCamera;Đ
Đ
    private void Start()Đ
    {Đ
        playerCamera = Camera.main;Đ
    }Đ
Đ
    public void SetDamageValue(float damage)Đ
    {Đ
        if (textMeshPro == null)Đ
        {Đ
            Debug.LogError("TextMeshPro component is not assigned in the
DamageNumberController component.");Đ
            return;Đ
        }Đ
Đ
        textMeshPro.text = damage.ToString();Đ
    }Đ
Đ
    private void Update()Đ
    {Đ
        if (textMeshPro == null)Đ
        {Đ
            Destroy(gameObject);Đ
            return;Đ
        }Đ
Đ
        // Rotate towards player cameraĐ
        if (playerCamera != null)Đ
        {Đ
            FaceCamera();Đ
```

```
        }Đ
Đ
        // Float upwardsĐ
        transform.position += Vector3.up * floatSpeed * Time.deltaTime;Đ
Đ
        // Update the elapsed timeĐ
        elapsedTime += Time.deltaTime;Đ
Đ
        // Fade effectĐ
        textMeshPro.alpha = Mathf.Clamp01(1f - (elapsedTime / duration));Đ
Đ
        // Destroy the damage number after the durationĐ
        if (elapsedTime >= duration)Đ
        {Đ
            Destroy(gameObject);Đ
        }Đ
    }Đ
Đ
    private void FaceCamera()Đ
    {Đ
        Vector3 targetDirection = playerCamera.transform.position -
transform.position;Đ
        targetDirection.y = 0;Đ
        Quaternion targetRotation = Quaternion.LookRotation(-targetDirection);Đ
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, 1);Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\DamageNumberController.csĐ
Class: DamageNumberControllerĐ
Đ
- textMeshPro : TextMeshProĐ
- floatSpeed : floatĐ
- duration : floatĐ
- elapsedTime : floatĐ
- playerCamera : CameraĐ
Đ
+ Start()Đ
- SetDamageValue(damage : float)Đ
+ Update()Đ
- FaceCamera()

- GameManager at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\GameManager.cs:
```

Summary of GameManager:
File Name: GameManager.csÐ
Ð
Summary: The GameManager class is responsible for managing the overall game state and other systems as needed. It uses an enum called GameState to keep track of the current state of the game. The class also provides methods to change the game state and save or load the game.Ð
Ð
Internals: The class has a static instance variable to ensure there is only one instance of the GameManager in the game. The Awake method initializes the game state and other systems as needed, and ensures that only one instance of the game manager exists in the game by destroying any additional instances. The Update method calls HandleGameState and UpdateCursorVisibility functions. Ð
Ð
The UpdateCursorVisibility method handles when to show or hide the cursor based on the current game state. If the game is paused or in a menu, the cursor is shown but if the game is playing, the cursor is hidden. Ð
Ð
The HandleGameState method uses a switch statement to handle the different game states (InMenu, Playing, Paused, GameOver), and performs the necessary logic for each state.Ð
Ð
The class provides public methods to change the current game state and to save and load the game. Other methods can also be added as needed for additional functionality.Ð
Ð
Explicit Dependencies: using UnityEngine;
Code of file GameManager:

```
using UnityEngine;Ð
Ð
public class GameManager : MonoBehaviourÐ
{Ð
    public static GameManager Instance;Ð
Ð
    public enum GameState { InMenu, Playing, Paused, GameOver }Ð
    public GameState currentState;Ð
Ð
    private void Awake()Ð
    {Ð
        if (Instance == null)Ð
        {Ð
            Instance = this;Ð
            DontDestroyOnLoad(gameObject);Ð
        }Ð
        elseÐ
```

```
        {Đ
            Destroy(gameObject);Đ
            return;Đ
        }Đ
Đ
        // Initialize game state and other systems as neededĐ
        currentState = GameState.InMenu;Đ
    }Đ
Đ
    private void Update()Đ
    {Đ
        HandleGameState();Đ
        UpdateCursorVisibility();Đ
    }Đ
Đ
    private void UpdateCursorVisibility()Đ
    {Đ
        // If the game is paused or in a menu, show the cursorĐ
        if (currentState == GameState.Paused || currentState == GameState.InMenu)Đ
        {Đ
            Cursor.visible = true;Đ
            Cursor.lockState = CursorLockMode.None;Đ
        }Đ
        // If the game is playing, hide the cursorĐ
        else if (currentState == GameState.Playing)Đ
        {Đ
            Cursor.visible = false;Đ
            Cursor.lockState = CursorLockMode.Locked;Đ
        }Đ
    }Đ
Đ
    private void HandleGameState()Đ
    {Đ
        switch (currentState)Đ
        {Đ
            case GameState.InMenu:Đ
                // Handle main menu logicĐ
                break;Đ
            case GameState.Playing:Đ
                // Handle playing state logicĐ
                break;Đ
            case GameState.Paused:Đ
                // Handle paused state logicĐ
                break;Đ
            case GameState.GameOver:Đ
```

```
            // Handle game over logicĐ
            break;Đ
        }Đ
    }Đ
Đ
    public void ChangeGameState(GameState newState)Đ
    {Đ
        currentState = newState;Đ
    }Đ
    public void SaveGame()Đ
    {Đ
        // Implement save game logicĐ
    }Đ
Đ
    public void LoadGame()Đ
    {Đ
        // Implement load game logicĐ
    }Đ
    // Implement other methods as needed, such as SaveGame, LoadGame, etc.Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\GameManager.csĐ
GameManager:Đ
- Instance: staticĐ
- currentState: enumĐ
- Awake()Đ
    - if Instance == nullĐ
        - Instance = thisĐ
        - DontDestroyOnLoad(gameObject)Đ
    - elseĐ
        - Destroy(gameObject)Đ
        - returnĐ
    - currentState = InMenuĐ
- Update()Đ
    - HandleGameState()Đ
    - UpdateCursorVisibility()Đ
- UpdateCursorVisibility()Đ
    - if currentState == Paused or InMenuĐ
        - Cursor.visible = trueĐ
        - Cursor.lockState = NoneĐ
    - else if currentState == PlayingĐ
        - Cursor.visible = falseĐ
        - Cursor.lockState = LockedĐ
- HandleGameState()Đ
```

- switch currentStateÐ
  - InMenuÐ
    - Handle main menu logicÐ
  - PlayingÐ
    - Handle playing state logicÐ
  - PausedÐ
    - Handle paused state logicÐ
  - GameOverÐ
    - Handle game over logicÐ
- ChangeGameState(newState)Ð
  - currentState = newStateÐ
- SaveGame()Ð
  - Implement save game logicÐ
- LoadGame()Ð
  - Implement load game logic

- IDragable at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IDragable.cs:
Summary of IDragable: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IDragable.csÐ
The file named "IDragable.cs" defines an interface for objects that can be dragged in a Unity UI. The interface includes three methods from the Event System library: IBeginDragHandler, IDragHandler, and IEndDragHandler. This allows the implementation of different behaviors for the beginning, dragging, and ending stages of the drag. Ð
Ð
The interface also includes a custom method, "getDraggedObject()", which returns the object being dragged. By implementing the IDragable interface, objects can easily be made draggable within a Unity project.
Code of file IDragable:
```
using UnityEngine.EventSystems;Ð
using UnityEngine;Ð
public interface IDragable : IBeginDragHandler, IDragHandler, IEndDragHandlerÐ
{Ð
Ð
    GameObject getDraggedObject();Ð
}Ð
```

Corresponding SyntaxTree: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IDragable.csÐ
IDragable:Ð
- interfaceÐ
- IBeginDragHandlerÐ

- IDragHandlerÐ
- IEndDragHandlerÐ
- GameObjectÐ
- getDraggedObject()

- IRecieveDrop at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IRecieveDrop.cs:
Summary of IRecieveDrop:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IRecieveDrop.csÐ
The file name is "IRecieveDrop.cs" and it is located in the "Ui\Interfaces" folder of the Unity project. This file contains an interface, which means it defines a set of methods that a class can implement. Ð
Ð
The interface is named "IRecieveDrop" and it inherits from three other interfaces: "IPointerEnterHandler", "IPointerExitHandler", and "IDropHandler". These are all interfaces from the Unity Event System, which is used for handling user input. Ð
Ð
The "IRecieveDrop" interface does not have any methods defined in it, but rather it requires any class that implements it to also implement the methods from the three inherited interfaces. Ð
Ð
These methods are used for handling different aspects of user input related to drag and drop functionality. "IPointerEnterHandler" and "IPointerExitHandler" are used for detecting when the user's mouse pointer enters and exits the area of the object that is implementing the interface. "IDropHandler" is used for detecting when the user drops an object onto the area of the object that is implementing the interface.Ð
Ð
Overall, the "IRecieveDrop" interface is used to allow objects to receive dropped objects from the user and handle them in a customizable way.
Code of file IRecieveDrop:
```
using UnityEngine.EventSystems;Ð
using UnityEngine;Ð
public interface IRecieveDrop : IPointerEnterHandler, IPointerExitHandler, IDropHandlerÐ
{Ð
    Ð
}
```
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IRecieveDrop.csÐ
IRecieveDrop:Ð
- interfaceÐ
- name: IRecieveDropÐ

- extends: IPointerEnterHandler, IPointerExitHandler, IDropHandlerÐ
- namespace: UnityEngine.EventSystems, UnityEngineÐ
- filepath: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\Interfaces\IRecieveDrop.cs

- OutlineHighlight at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\OutlineHighlight.cs:
Summary of OutlineHighlight:
File Name: OutlineHighlight.csÐ
Ð
Summary: Ð
Ð
The OutlineHighlight class is a MonoBehavior script that allows for highlighting outlines of a specific object with a particular color. Attributes include a highlight material, highlight color, outline thickness, and target transform. The class utilizes a Camera object, CommandBuffer, and RenderTexture to achieve its desired effect.Ð
Ð
Internally, the OnRenderImage function is utilized to render an image with the highlight and outline effect. A check for a null target helps prevent errors. The RenderTexture variable is then created with temporary properties. The CommandBuffer is used to clear the render target, retrieve the target's mesh filter, and draw the mesh with the highlight material. The color and thickness properties of the highlight material are set before the final command buffer is executed and blitted to the destination texture. Finally, the temporary render texture is released.Ð
Ð
Overall, this class is important for adding visual depth to games or applications where a highlighted outline of a particular object is desired.
Code of file OutlineHighlight:
```
þÿusing UnityEngine;Ð
using UnityEngine.Rendering;Ð
Ð
[ExecuteInEditMode, ImageEffectAllowedInSceneView]Ð
public class OutlineHighlight : MonoBehaviourÐ
{Ð
    public Material highlightMaterial;Ð
    public Color highlightColor = Color.red;Ð
    public float outlineThickness = 2f;Ð
    public Transform target;Ð
Ð
    private Camera cam;Ð
    private CommandBuffer commandBuffer;Ð
Ð
    private void Start()Ð
```

```
    {Đ
        cam = GetComponent<Camera>();Đ
        commandBuffer = new CommandBuffer();Đ
    }Đ
Đ
    private void OnRenderImage(RenderTexture src, RenderTexture dest)Đ
    {Đ
        if (target == null)Đ
        {Đ
            Graphics.Blit(src, dest);Đ
            return;Đ
        }Đ
Đ
        commandBuffer.Clear();Đ
Đ
        var renderTexture = RenderTexture.GetTemporary(src.width, src.height,
src.depth, src.format);Đ
Đ
        commandBuffer.SetRenderTarget(renderTexture);Đ
Đ
        commandBuffer.ClearRenderTarget(true, true, Color.clear);Đ
Đ
        var meshFilter = target.GetComponent<MeshFilter>();Đ
        if (meshFilter != null)Đ
        {Đ
            commandBuffer.DrawMesh(meshFilter.sharedMesh,
target.localToWorldMatrix, highlightMaterial);Đ
        }Đ
Đ
        highlightMaterial.SetColor("_OutlineColor", highlightColor);Đ
        highlightMaterial.SetFloat("_OutlineThickness", outlineThickness);Đ
Đ
        Graphics.ExecuteCommandBuffer(commandBuffer);Đ
Đ
        Graphics.Blit(renderTexture, dest);Đ
Đ
        RenderTexture.ReleaseTemporary(renderTexture);Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\OutlineHighlight.csĐ
class: OutlineHighlightĐ
  - highlightMaterial: MaterialĐ
```

- highlightColor: ColorĐ
- outlineThickness: floatĐ
- target: TransformĐ
- cam: CameraĐ
- commandBuffer: CommandBufferĐ
+ Start()Đ
+ OnRenderImage(src: RenderTexture, dest: RenderTexture)

- OverlayUiController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\OverlayUiController.cs:
Summary of OverlayUiController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\OverlayUiController.csĐ
The file "OverlayUiController.cs" is responsible for managing the overlay UI elements in the game. It contains public fields for the character's name, health bar, mana bar, and level text. These fields are linked to the respective UI elements in the scene using the SerializeField attribute.Đ
Đ
In addition, the controller has a reference to the CharacterStats script that holds information about the player's character's health, mana, and level. This reference is initialized in the Start() method.Đ
Đ
The Start() method also sets the maximum values of the health and mana bars, as well as the character name, using the information from the CharacterStats script. The UpdateUI() method, which is called both in Start() and Update(), sets the current values of the health and mana bars and updates the character's level text.Đ
Đ
Overall, the OverlayUiController.cs script provides dynamic display of the character's stats to the player while playing the game.
Code of file OverlayUiController:

```
using UnityEngine;Đ
using UnityEngine.UI;Đ
Đ
public class OverlayUiController : MonoBehaviourĐ
{Đ
    private UIManager UIManager;Đ
    [SerializeField] public Text characterNameText;Đ
    [SerializeField] public Slider healthBar;Đ
    [SerializeField] public Slider manaBar;Đ
    [SerializeField] public Text levelText;Đ
    private GameObject player;Đ
Đ
    private HealthController HealthController;Đ
    private ManaController ManaController;Đ
Đ
```

```
    private void updateHealthBar()Đ
    {Đ
        healthBar.value = HealthController.currentHealth;Đ
        healthBar.maxValue = HealthController.maxHealth;Đ
    }Đ
    Đ
    private void updateManaBar()Đ
    {Đ
        manaBar.value = (ManaController.currentMana); Đ
        manaBar.maxValue = ManaController.maxMana;Đ
    }Đ
    private void updateHealthAndMana(){Đ
Đ
        updateHealthBar();Đ
        updateManaBar();Đ
    }Đ
    private void Start()Đ
    {Đ
        UIManager = FindObjectOfType<UIManager>();Đ
        player = FindObjectOfType<PlayerController>().gameObject;Đ
        HealthController = player.GetComponent<HealthController>();Đ
        ManaController = player.GetComponent<ManaController>();Đ
Đ
        Đ
    }Đ
    public void Update()Đ
    {Đ
        updateHealthAndMana();Đ
    }Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\OverlayUiController.csĐ
OverlayUiControllerĐ
- UIManagerĐ
- Text characterNameTextĐ
- Slider healthBarĐ
- Slider manaBarĐ
- Text levelTextĐ
- GameObject playerĐ
- HealthController HealthControllerĐ
- ManaController ManaControllerĐ
- void updateHealthBar()Đ
- void updateManaBar()Đ
- void updateHealthAndMana()Đ
```

- void Start()Đ
- void Update()

- PresentQuestUiController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\PresentQuestUiController.cs:
Summary of PresentQuestUiController:
File name: PresentQuestUiController.csĐ
Đ
This class is responsible for controlling the UI elements related to presenting a quest. It contains private fields for the quest title, description, and two buttons (accept and decline). Additionally, it has a reference to the QuestSystem class.Đ
Đ
The Start() method is called once when the object is created and it finds the QuestSystem object in the scene and assigns it to the internal reference.Đ
Đ
The showQuestInfo(Quest quest, UIManager UIManager) method is where the main logic occurs. It takes in a Quest object and a UIManager object, and sets the quest title and description text accordingly. It also sets up the accept and decline button listeners to handle the respective actions. When the accept button is clicked, it adds the current quest to the QuestSystem instance and hides the Quest UI Presenter via the UIManager. When the decline button is clicked, it only hides the presenter.Đ
Đ
This class utilizes the Unity Engine's UI components including TMP_Text and Button.
Code of file PresentQuestUiController:

```
using UnityEngine;Đ
using UnityEngine.UI;Đ
using TMPro;Đ
Đ
public class PresentQuestUiController : MonoBehaviourĐ
{Đ
    [SerializeField] private TMP_Text questTitle;Đ
    [SerializeField] private TMP_Text questDescription;Đ
    [SerializeField] private Button acceptButton;Đ
    [SerializeField] private Button declineButton;Đ
Đ
    private QuestSystem questSystem;Đ
    void Start()Đ
    {Đ
        questSystem = FindObjectOfType<QuestSystem>();Đ
    }Đ
  Đ
Đ
    public void showQuestInfo(Quest quest,UIManager UIManager)Đ
```

```
    {Đ
        questTitle.text = quest.title;Đ
        questDescription.text = quest.description;Đ
        acceptButton.onClick.AddListener(() => questSystem.AddQuest(quest));Đ
        acceptButton.onClick.AddListener(() => UIManager.hideQuestUiPresenter());Đ
        declineButton.onClick.AddListener(() => UIManager.hideQuestUiPresenter());Đ
        Đ
    }Đ
Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\PresentQuestUiController.csĐ
PresentQuestUiControllerĐ
- questTitle: TMP_TextĐ
- questDescription: TMP_TextĐ
- acceptButton: ButtonĐ
- declineButton: ButtonĐ
- questSystem: QuestSystemĐ
Đ
_Start()Đ
- questSystem = FindObjectOfType<QuestSystem>()Đ
Đ
+ showQuestInfo(quest: Quest, UIManager: UIManager)Đ
- questTitle.text = quest.titleĐ
- questDescription.text = quest.descriptionĐ
- acceptButton.onClick.AddListener(() => questSystem.AddQuest(quest))Đ
- acceptButton.onClick.AddListener(() => UIManager.hideQuestUiPresenter())Đ
- declineButton.onClick.AddListener(() => UIManager.hideQuestUiPresenter())

- QuestBookUIController at C:\Users\Toastbrot\Downloads\STRATEGY
01.04.2022\My project\Assets\Scripts\Ui\QuestBookUIController.cs:
Summary of QuestBookUIController:
File name: QuestBookUIController.csĐ
Đ
Summary: Đ
Đ
This script is responsible for controlling the UI elements of the player's quest
book. It has references to various UI elements such as text fields, scroll rect, and
a prefab for the quest list item. It also has access to the quest system, which
stores all the available quests and their objectives. Đ
Đ
The script has two main functions: UpdateQuestList and ShowQuestInformation.
The UpdateQuestList function clears the quest list content and repopulates it
```

with the current available quests. The ShowQuestInformation function displays the details of the selected quest, such as its title, description, and objectives. Đ
Đ
To clear the quest list content, the script loops through all the child objects of the questListContent and calls the Destroy function on each of them. To repopulate the quest list content, the script creates a new game object using the questListItemPrefab and sets its text field to the title of the current quest. It also adds a listener to the button component of the game object so that it can display the details of the selected quest when clicked. Đ
Đ
Finally, the scroll position of the quest list is reset to the top. The ShowQuestInformation function sets the text fields of the title, description, and objectives to the selected quest's data. It loops through all the objectives of the quest and displays their progress in a string format. Đ
Đ
Overall, the QuestBookUIController script is an essential component of the player's quest book interface, enabling them to view and track their progress in the available quests.
Code of file QuestBookUIController:

```
using System.Collections.Generic;Đ
using UnityEngine;Đ
using UnityEngine.UI;Đ
using TMPro;Đ
Đ
public class QuestBookUIController : MonoBehaviourĐ
{Đ
    public TMP_Text titleText;Đ
    public TMP_Text descriptionText;Đ
    public TMP_Text objectivesText;Đ
    public ScrollRect questListScrollRect;Đ
    public GameObject questListItemPrefab;Đ
    public Transform questListContent;Đ
    Đ
    public QuestSystem questSystem;Đ
  Đ
Đ
    private void Awake()Đ
    {Đ
      Đ
    }Đ
Đ
    private void Start()Đ
    {  Đ
       UpdateQuestList();Đ
    }Đ
```

```
Đ
    public void UpdateQuestList()Đ
    {Đ
        // Clear the quest list contentĐ
        foreach (Transform child in questListContent)Đ
        {Đ
            Destroy(child.gameObject);Đ
        }Đ
Đ
        // Re-populate the quest list contentĐ
        foreach (Quest quest in questSystem.quests)Đ
        {Đ
            GameObject questListItem = Instantiate(questListItemPrefab,
questListContent);Đ
            questListItem.gameObject.SetActive(true);Đ
            questListItem.GetComponentInChildren<TMP_Text>().text = quest.title;Đ
            questListItem.GetComponent<Button>().onClick.AddListener(() =>
ShowQuestInformation(quest));Đ
        }Đ
Đ
        // Reset the scroll position of the quest listĐ
        questListScrollRect.verticalNormalizedPosition = 1f;Đ
    }Đ
Đ
    public void ShowQuestInformation(Quest quest)Đ
    {Đ
        // Set the quest information text fields to the current quest's dataĐ
        titleText.text = quest.title;Đ
        descriptionText.text = quest.description;Đ
        string objectivesString = "";Đ
        foreach(QuestObjective objective in quest.objectives)Đ
        {Đ
            objectivesString += $"-({objective.GetObjectiveProgress()})\n";Đ
        }   Đ
        Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\QuestBookUIController.csĐ
QuestBookUIControllerĐ
- titleText: TMP_TextĐ
- descriptionText: TMP_TextĐ
- objectivesText: TMP_TextĐ
```

- questListScrollRect: ScrollRectĐ
- questListItemPrefab: GameObjectĐ
- questListContent: TransformĐ
- questSystem: QuestSystemĐ
Đ
+ Awake()Đ
+ Start()Đ
+ UpdateQuestList()Đ
+ ShowQuestInformation(Quest quest)

- QuestLogUIController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\QuestLogUIController.cs:
Summary of QuestLogUIController:
File Name: QuestLogUIController.csĐ
Đ
Summary: This file represents the Quest Log User Interface Controller class in the game. It is responsible for updating the display of a player's current quests and tracking quests. It has public and private variables that hold text objects and a reference to the QuestSystem class. It also has functions that update the display of quest logs and add or remove quests from tracking.Đ
Đ
Components/Variables: Đ
- public TMP_Text questLogText: The text object that displays the player's quest log.Đ
- public TMP_Text trackingText: The text object that displays the player's currently tracking quests.Đ
- private QuestSystem questSystem: A variable that holds a reference to the QuestSystem class.Đ
- private List<Quest> trackingQuests: A list that holds the player's currently tracking quests.Đ
Đ
Functions:Đ
- Awake(): Called when the game object is created. It sets the questSystem variable by finding the QuestSystem object in the scene. If there is no QuestSystem object found, it logs an error message.Đ
- Start(): Called after Awake(). It updates the quest log display.Đ
- UpdateQuestLog(): A function that updates the display of both the quest log and tracking quests. It loops through all quests in the QuestSystem and adds their information to the quest log string. It also loops through all currently tracking quests and adds their titles to the tracking string. It then updates the text objects with the final strings.Đ
- AddQuestToTrack(int questID): Adds a quest to the list of currently tracking quests. It first gets the quest object using the questSystem's GetQuestByID() function. If the quest is not already being tracked, it is added to the trackingQuests list and the quest log display is updated.Đ

- RemoveQuestToTrack(int questID): Removes a quest from the list of currently tracking quests. It first gets the quest object using the questSystem's GetQuestByID() function. If the quest is currently being tracked, it is removed from the trackingQuests list and the quest log display is updated.
Code of file QuestLogUIController:

```
using System.Collections.Generic;Đ
using UnityEngine;Đ
using TMPro;Đ
Đ
public class QuestLogUIController : MonoBehaviourĐ
{Đ
    public TMP_Text questLogText;Đ
    public TMP_Text trackingText;Đ
Đ
    private QuestSystem questSystem;Đ
    private List<Quest> trackingQuests = new List<Quest>();Đ
Đ
    private void Awake()Đ
    {Đ
        questSystem = FindObjectOfType<QuestSystem>();Đ
        if (questSystem == null)Đ
        {Đ
            Debug.LogError("No QuestSystem found in the scene!");Đ
        }Đ
    }Đ
Đ
    private void Start()Đ
    {Đ
        UpdateQuestLog();Đ
    }Đ
Đ
    public void UpdateQuestLog()Đ
    {Đ
        string questLogString = "";Đ
        foreach (Quest quest in questSystem.quests)Đ
        {Đ
            questLogString += $"[{quest.status}] {quest.title}\n";Đ
            foreach (QuestObjective objective in quest.objectives)Đ
            {Đ
                questLogString += $"- {objective.description}
({objective.GetObjectiveProgress()})\n";Đ
            }Đ
            questLogString += "\n";Đ
        }Đ
        questLogText.text = questLogString;Đ
```

```
Đ
        string trackingString = "Tracking: ";Đ
        foreach (Quest quest in trackingQuests)Đ
        {Đ
            trackingString += quest.title + ", ";Đ
        }Đ
        trackingText.text = trackingString.TrimEnd(',', ' ');Đ
    }Đ
Đ
    public void AddQuestToTrack(int questID)Đ
    {Đ
        Quest quest = questSystem.GetQuestByID(questID);Đ
        if (quest != null && !trackingQuests.Contains(quest))Đ
        {Đ
            trackingQuests.Add(quest);Đ
            UpdateQuestLog();Đ
        }Đ
    }Đ
Đ
    public void RemoveQuestToTrack(int questID)Đ
    {Đ
        Quest quest = questSystem.GetQuestByID(questID);Đ
        if (quest != null && trackingQuests.Contains(quest))Đ
        {Đ
            trackingQuests.Remove(quest);Đ
            UpdateQuestLog();Đ
        }Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\QuestLogUIController.csĐ
Class: QuestLogUIControllerĐ
Đ
- public TMP_Text questLogTextĐ
- public TMP_Text trackingTextĐ
- private QuestSystem questSystemĐ
- private List<Quest> trackingQuestsĐ
Đ
- private void Awake()Đ
  - questSystem = FindObjectOfType<QuestSystem>()Đ
  - if questSystem == null, Debug.LogError("No QuestSystem found in the scene!")Đ
Đ

- private void Start()Đ
  - UpdateQuestLog()Đ
Đ
- public void UpdateQuestLog()Đ
  - string questLogString = ""Đ
  - foreach Quest quest in questSystem.questsĐ
    - questLogString += $"[{quest.status}] {quest.title}\n"Đ
    - foreach QuestObjective objective in quest.objectivesĐ
      - questLogString += $"- {objective.description} ({objective.GetObjectiveProgress()})\n"Đ
    - questLogString += "\n"Đ
  - questLogText.text = questLogStringĐ
Đ
  - string trackingString = "Tracking: "Đ
  - foreach Quest quest in trackingQuestsĐ
    - trackingString += quest.title + ", "Đ
  - trackingText.text = trackingString.TrimEnd(',', ' ')Đ
Đ
- public void AddQuestToTrack(int questID)Đ
  - Quest quest = questSystem.GetQuestByID(questID)Đ
  - if quest != null && !trackingQuests.Contains(quest)Đ
    - trackingQuests.Add(quest)Đ
    - UpdateQuestLog()Đ
Đ
- public void RemoveQuestToTrack(int questID)Đ
  - Quest quest = questSystem.GetQuestByID(questID)Đ
  - if quest != null && trackingQuests.Contains(quest)Đ
    - trackingQuests.Remove(quest)Đ
    - UpdateQuestLog()

- SkillTreeMenuController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\SkillTreeMenuController.cs:
Summary of SkillTreeMenuController:
File Name: SkillTreeMenuController.csĐ
Đ
This file contains the implementation of the SkillTreeMenuController class/ component. Đ
Đ
The class has two public variables; an array of GameObjects called skillTrees, and an integer called currentSkillTree which is initially set to 0. Đ
Đ
The Start() method is called when the component is first loaded and it sets the active state of the current skill tree to true.Đ
Đ
The SwitchSkillTree(int index) method allows for the changing of the currently

displayed skill tree. If the index provided is out of range of the skillTrees array, the method returns without doing anything.Đ
Đ
If a valid index is provided, the method first sets the active state of the current skill tree to false, then sets the active state of the new skill tree to true, finally it updates the currentSkillTree index to the new index provided.Đ
Đ
In summary, this class/component is responsible for displaying different skill trees and switching between them based on player input.
Code of file SkillTreeMenuController:
þÿusing UnityEngine;Đ
Đ
public class SkillTreeMenuController : MonoBehaviourĐ
{Đ
    public GameObject[] skillTrees;Đ
    public int currentSkillTree = 0;Đ
    private void Start()Đ
    {Đ
        skillTrees[currentSkillTree].SetActive(true);Đ
    }Đ
Đ
    public void SwitchSkillTree(int index)Đ
    {Đ
        if (index < 0 || index >= skillTrees.Length) return;Đ
Đ
        skillTrees[currentSkillTree].SetActive(false);Đ
        skillTrees[index].SetActive(true);Đ
        currentSkillTree = index;Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\SkillTreeMenuController.csĐ
SkillTreeMenuController:Đ
- GameObject[] skillTreesĐ
- int currentSkillTree = 0Đ
- Start():Đ
’Ò 6¶–ÆÅ@rees[currentSkillTree].SetActive(true)Đ
- SwitchSkillTree(int index):Đ
’Ò –b †–æFPx < 0 || index >= skillTrees.Length) returnĐ
’Ò 6¶–ÆÅ@rees[currentSkillTree].SetActive(false)Đ
’Ò 6¶–ÆÅ@rees[index].SetActive(true)Đ
’Ò 7W'&VçE6¶–ÆÅ@ree = index

- SpellBookUiController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\SpellBookUiController.cs:
Summary of SpellBookUiController: C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\SpellBookUiController.csÐ
The SpellBookUiController.cs file is responsible for managing the user interface (UI) elements related to a player's learned abilities or spells. Ð
Ð
This script contains several public TMP_Text and GameObject variables, which are references to the UI elements that display the ability's name and description, as well as a list of learned abilities shown in a scrollable view. Ð
Ð
The UpdateQuestList() method is used to populate the list of abilities in the scrollable content area. First, it clears any existing content by destroying all the child game objects of the UI parent element. Then, it creates new instances of the ability list item prefab for each ability the player has learned, sets its name using the ability's name property, and attaches a button listener so that when this UI item is clicked, it triggers the ShowAbilityInformation() method with the corresponding ability as a parameter. Ð
Ð
The ShowAbilityInformation() method sets the titleText and descriptionText UI elements to the corresponding values from the ability's properties. Additionally, it generates a string variable called info that contains various information about the ability, such as base damage, strength scaling, intelligence scaling, and cooldown. This information is then displayed in the objectivesText UI element. Ð
Ð
Overall, this script provides a simple interface for players to view their learned abilities and their associated details.
Code of file SpellBookUiController:

```
using System.Collections.Generic;Ð
using UnityEngine;Ð
using UnityEngine.UI;Ð
using TMPro;Ð
Ð
public class SpellBookUiController : MonoBehaviourÐ
{Ð
    public TMP_Text titleText;Ð
    public TMP_Text descriptionText;Ð
    public TMP_Text objectivesText;Ð
    public ScrollRect spellListScrollRect;Ð
    public GameObject spellListItemPrefab;Ð
    public Transform spellListContent;Ð
    Ð
    public AbilityController abilityController;Ð
  Ð
```

```
Đ
   private void Awake()Đ
   {Đ
     Đ
   }Đ
Đ
   private void Start()Đ
   {  Đ
      UpdateQuestList();Đ
   }Đ
Đ
   public void UpdateQuestList()Đ
   {Đ
      // Clear the quest list contentĐ
      foreach (Transform child in spellListContent)Đ
      {Đ
         Destroy(child.gameObject);Đ
      }Đ
Đ
      // Re-populate the quest list contentĐ
      foreach (Ability ability in abilityController.learnedAbilitys)Đ
      {Đ
         GameObject spellListItem = Instantiate(spellListItemPrefab,
spellListContent);Đ
         spellListItem.gameObject.SetActive(true);Đ
         spellListItem.GetComponentInChildren<TMP_Text>().text = ability.name;Đ
         spellListItem.GetComponent<Button>().onClick.AddListener(() =>
ShowAbilityInformation(ability));Đ
         spellListItem.GetComponent<UiAbilitySlot>().ability = ability;Đ
         Đ
      }Đ
Đ
      // Reset the scroll position of the quest listĐ
      spellListScrollRect.verticalNormalizedPosition = 1f;Đ
   }Đ
Đ
   public void ShowAbilityInformation(Ability ability)Đ
   {Đ
      // Set the quest information text fields to the current quest's dataĐ
      titleText.text = ability.abilityName;Đ
      descriptionText.text = ability.abilityDescription;Đ
   string info = "";Đ
Đ
   info += "- Base Damage: " + ability.baseDamage + "\n";Đ
   info += "- Strength Scaling: " + ability.strengthScaling + "\n";Đ
```

```
    info += "- Intelligence Scaling: " + ability.intelligenceScaling + "\n";Đ
    info += "- Cooldown: " + ability.cooldown + "\n";Đ
    objectivesText.text = info;Đ
Đ
        /*foreach(QuestObjective objective in ability.)Đ
        {Đ
            objectivesString += $"-({objective.GetObjectiveProgress()})\n";Đ
        }   */Đ
        Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\SpellBookUiController.csĐ
SpellBookUiController:Đ
- titleText: TMP_TextĐ
- descriptionText: TMP_TextĐ
- objectivesText: TMP_TextĐ
- spellListScrollRect: ScrollRectĐ
- spellListItemPrefab: GameObjectĐ
- spellListContent: TransformĐ
- abilityController: AbilityControllerĐ
Đ
+ Awake()Đ
+ Start()Đ
+ UpdateQuestList()Đ
+ ShowAbilityInformation(Ability ability)

- ToolTipUiController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\ToolTipUiController.cs:
Summary of ToolTipUiController:
File Name: ToolTipUiController.csĐ
Đ
Summary: The ToolTipUiController class is responsible for controlling the user interface (UI) elements of the tooltip that appears when hovering over an in-game skill. The class contains UI elements, such as SkillName, SkillDescription, SkillpointCost, and SkillIcon. The UpdateUI method is called to update these elements based on the skillNode parameter passed in. In addition, the Awake method sets the cursor state and the Start method remains empty.Đ
Đ
Intern Logic: This file creates a ToolTipUiController class that manages tooltip UI elements and updates them based on the skillNode passed in. The UpdateUI method updates the skill name, description, skill point cost, attribute requirement, and skill icon based on the skillNode parameter. Meanwhile, the

Awake method sets the cursor state and the Start method does nothing.
Code of file ToolTipUiController:

```
þÿusing UnityEngine;Đ
using UnityEngine.UI;Đ
Đ
public class ToolTipUiController : MonoBehaviourĐ
{Đ
    public Text SkillName;Đ
    public Text SkillDescription;Đ
    public Text AlreadySkilled;Đ
    public Text SkillpointCost;Đ
    public Text AttrbuteReq;Đ
    public Image SkillIcon;Đ
Đ
    private void Start()Đ
    {Đ
    }Đ
Đ
    private void Awake()Đ
    {Đ
        Cursor.visible = true;Đ
        Cursor.lockState = CursorLockMode.None;Đ
    }Đ
Đ
    public void UpdateUI(SkillNode node)Đ
    {Đ
        SkillName.text = node.skillName;Đ
        SkillDescription.text = node.skillDescription;Đ
        if (node.isUnlocked)Đ
        {Đ
            AlreadySkilled.gameObject.SetActive(true);Đ
        }Đ
        elseĐ
        {Đ
            AlreadySkilled.gameObject.SetActive(false);Đ
        }Đ
        SkillpointCost.text = "Cost: " + node.skillPointCost;Đ
Đ
        AttrbuteReq.text = "Requiment:";Đ
Đ
        for (int a = 0; a < node.mainStatRequirement.Count; a++)Đ
        {Đ
Đ
            AttrbuteReq.text += " " + node.mainStatRequirement[a] + ": " +
node.mainStatValue[a];Đ
```

```
        }Đ
Đ
        if (node.prerequisiteSkill != null)Đ
        {Đ
            AttrbuteReq.text += "Skill Requiment: " + node.prerequisiteSkill.skillName;Đ
        }Đ
        SkillIcon.sprite = node.icon;Đ
    }Đ
    public void UpdateUI(Ability ability)Đ
{Đ
    SkillName.text = ability.abilityName;Đ
    SkillDescription.text = ability.abilityDescription;Đ
    AlreadySkilled.gameObject.SetActive(false);Đ
    SkillpointCost.gameObject.SetActive(false);Đ
Đ
    AttrbuteReq.text = $"Base Damage: {ability.baseDamage}\nStrength Scaling:
{ability.strengthScaling}\nIntelligence Scaling: {ability.intelligenceScaling}";Đ
Đ
    SkillIcon.sprite = ability.icon;Đ
}Đ
Đ
public void UpdateUI(Item item)Đ
{Đ
    SkillName.text = item.itemName;Đ
    SkillDescription.text = item.description;Đ
    AlreadySkilled.gameObject.SetActive(false);Đ
    SkillpointCost.gameObject.SetActive(false);Đ
Đ
    if (item is EquipableItem equipableItem)Đ
    {Đ
        AttrbuteReq.text = $"Bonuses:\nStrength:
{equipableItem.strengthBonus}\nIntelligence:
{equipableItem.intelligenceBonus}\nDexterity:
{equipableItem.dexterityBonus}\nEndurance:
{equipableItem.enduranceBonus}\nWisdom: {equipableItem.wisdomBonus}";Đ
    }Đ
    elseĐ
    {Đ
        AttrbuteReq.text = "";Đ
    }Đ
Đ
    SkillIcon.sprite = item.icon;Đ
}Đ
Đ
Đ
```

}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\ToolTipUiController.csĐ
ToolTipUiController:Đ
Đ
- SkillNameĐ
- SkillDescriptionĐ
- AlreadySkilledĐ
- SkillpointCostĐ
- AttrbuteReqĐ
- SkillIconĐ
Đ
Start()Đ
Đ
Awake()Đ
Đ
UpdateUI(node)Đ
Đ
- SkillName.textĐ
- SkillDescription.textĐ
- AlreadySkilled.gameObject.SetActive(bool)Đ
- SkillpointCost.textĐ
- AttrbuteReq.textĐ
- SkillIcon.spriteĐ
Đ
UpdateUI(ability)Đ
Đ
- SkillName.textĐ
- SkillDescription.textĐ
- AlreadySkilled.gameObject.SetActive(false)Đ
- SkillpointCost.gameObject.SetActive(false)Đ
- AttrbuteReq.textĐ
Đ
UpdateUI(item)Đ
Đ
- SkillName.textĐ
- SkillDescription.textĐ
- AlreadySkilled.gameObject.SetActive(false)Đ
- SkillpointCost.gameObject.SetActive(false)Đ
- AttrbuteReq.textĐ
- SkillIcon.sprite

- UiAbilitySlot at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiAbilitySlot.cs:

Summary of UiAbilitySlot:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiAbilitySlot.csÐ
The file being discussed is named "UiAbilitySlot.cs" and is located in the "Scripts/Ui" folder of the project. It is a C# script that contains the internal logic for handling Ability Slots for the user interface (UI). Ð
Ð
The script uses three Unity namespaces: UnityEngine.EventSystems, UnityEngine, and UnityEngine.UI. Ð
Ð
The class extends the base class UiBaseDragAndDropFunc and contains two public fields: "ability" of type Ability and "icon" of type Image. Ð
Ð
When the script executes, the "Start()" function checks if an ability is assigned to the slot and, if so, it sets the "icon" sprite to be the one assigned to the "ability". Ð
Ð
Overall, UiAbilitySlot.cs defines the functioning of the UI Ability Slots and ensures that the correct icon is displayed based on the assigned ability.
Code of file UiAbilitySlot:

```
using UnityEngine.EventSystems;Ð
using UnityEngine;Ð
using UnityEngine.UI;Ð
internal class UiAbilitySlot : UiBaseDragAndDropFuncÐ
{Ð
  public Ability ability;Ð
  public Image icon;Ð
Ð
   private void Start()Ð
   {Ð
      if(ability!=null){Ð
         icon.sprite = ability.icon;Ð
      }Ð
   }Ð
Ð
  Ð
}Ð
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiAbilitySlot.csÐ
UiAbilitySlot:Ð
-ability : AbilityÐ
-icon : ImageÐ
-Start()Ð
--if(ability != null)Ð

---icon.sprite = ability.icon

- UiBaseDragAndDropFunc at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiBaseDragAndDropFunc.cs:
Summary of UiBaseDragAndDropFunc:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiBaseDragAndDropFunc.csÐ
The file "UiBaseDragAndDropFunc.cs" is a script that implements the drag and drop functionality for UI elements in Unity. It is attached to a UI GameObject with a RectTransform component. The script implements the IBeginDragHandler, IDragHandler, and IEndDragHandler interfaces which allow it to detect when a drag and drop operation has started, is ongoing, and has ended, respectively.Ð
Ð
When a drag operation begins, the script creates a new GameObject called "DragObject" and adds a RectTransform and Image component to it. The size and position of the RectTransform are set to match the original UI element being dragged. The Image component is used to display a visual representation of the dragged object on the screen. If the dragged UI element has an UiAbilitySlot component, the Image's sprite is set to that component's icon sprite.Ð
Ð
During the drag operation, the position of the DragObject is updated to match the position of the mouse cursor.Ð
Ð
When the drag operation ends, the DragObject is destroyed. If the dragged UI element is dropped onto a valid target (an object with a UiHotKeySlot component), the script updates the target's ability or item slot with the dragged UI element's ability or item, depending on the type of element being dragged. Finally, the original UI element is returned to its original position.Ð
Ð
The script also initializes some variables in the Awake function, such as the RectTransform, UIManager, and originalPosition. These variables are used throughout the script to store information about the UI element being dragged.
Code of file UiBaseDragAndDropFunc:
using UnityEngine;Ð
using UnityEngine.EventSystems;Ð
using UnityEngine.UI;Ð
public class UiBaseDragAndDropFunc : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandlerÐ
{Ð
** public RectTransform rectTransform;Ð**
** private UIManager uiManager;Ð**
** private GameObject dragObject;Ð**
** public Vector3 originalPosition;Ð**
** private GameObject dragedObject;Ð**
Ð

```
   Đ
Đ
   private void Awake()Đ
   {Đ
      rectTransform = GetComponent<RectTransform>();Đ
      uiManager = FindObjectOfType<UIManager>();Đ
      originalPosition = rectTransform.localPosition;Đ
   }Đ
Đ
   public void OnBeginDrag(PointerEventData eventData)Đ
   {Đ
Đ
Đ
      dragedObject = eventData.pointerDrag;Đ
      dragObject = new GameObject("DragObject");Đ
      dragObject.transform.SetParent(uiManager.gameObject.transform);Đ
      dragObject.transform.SetSiblingIndex(uiManager.gameObject.transform.child
Count - 1);Đ
Đ
      RectTransform dragRectTransform =
dragObject.AddComponent<RectTransform>();Đ
      dragRectTransform.sizeDelta = rectTransform.sizeDelta;Đ
      dragRectTransform.position = eventData.position;Đ
Đ
Đ
Đ
      UiAbilitySlot uiAbilitySlot = dragedObject.GetComponent<UiAbilitySlot>();Đ
      UiItemSlot uiItemSlot = dragedObject.GetComponent<UiItemSlot>();Đ
            Image image = dragObject.AddComponent<Image>();Đ
      image.sprite = GetComponent<Image>().sprite;Đ
      image.raycastTarget = false;Đ
Đ
      if(uiAbilitySlot!=null)Đ
      {Đ
         image.sprite = uiAbilitySlot.icon.sprite;Đ
      }Đ
Đ
Đ
   }Đ
Đ
   public void OnDrag(PointerEventData eventData)Đ
   {Đ
      dragObject.GetComponent<RectTransform>().position = eventData.position;Đ
   }Đ
Đ
```

```
    public void OnEndDrag(PointerEventData eventData)Đ
    {Đ
Đ
Đ
        Destroy(dragObject);Đ
Đ
        if (eventData.pointerEnter != null)Đ
        {Đ
            UiHotKeySlot hotkeySlot =
eventData.pointerEnter.GetComponent<UiHotKeySlot>();Đ
            UiAbilitySlot uiAbilitySlot = dragedObject.GetComponent<UiAbilitySlot>();Đ
            UiItemSlot uiItemSlot = dragedObject.GetComponent<UiItemSlot>();Đ
Đ
Đ
            if (hotkeySlot != null)Đ
            {Đ
                if(uiAbilitySlot!=null)Đ
                {Đ
                    hotkeySlot.ability = uiAbilitySlot.ability;Đ
                    hotkeySlot.item = null;Đ
                }Đ
                else if(uiItemSlot!=null)Đ
                {Đ
                    hotkeySlot.item = uiItemSlot.item;Đ
                    hotkeySlot.ability = null;Đ
                }Đ
                elseĐ
                {Đ
                    hotkeySlot.item = null;Đ
                    hotkeySlot.ability = null;Đ
                }Đ
Đ
                hotkeySlot.updateinfo();Đ
            }Đ
        }Đ
Đ
        rectTransform.localPosition = originalPosition;Đ
    }Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\UiBaseDragAndDropFunc.csĐ
UiBaseDragAndDropFunc:Đ
- rectTransform: RectTransformĐ
- uiManager: UIManagerĐ
```

```
- dragObject: GameObjectĐ
- originalPosition: Vector3Đ
- dragedObject: GameObjectĐ
Đ
+ Awake():Đ
   rectTransform = Get<RectTransform>()Đ
   uiManager = Find<UIManager>()Đ
   originalPosition = rectTransform.localPositionĐ
Đ
+ OnBeginDrag(eventData: PointerEventData):Đ
   dragedObject = eventData.pointerDragĐ
   dragObject = new GameObject("DragObject")Đ
   dragObject.transform.SetParent(uiManager.gameObject.transform)Đ
   dragObject.transform.SetSiblingIndex(uiManager.gameObject.transform.childC
ount - 1)Đ
   dragRectTransform = dragObject.AddComponent<RectTransform>()Đ
   dragRectTransform.sizeDelta = rectTransform.sizeDeltaĐ
   dragRectTransform.position = eventData.positionĐ
   uiAbilitySlot = dragedObject.Get<UiAbilitySlot>()Đ
   uiItemSlot = dragedObject.Get<UiItemSlot>()Đ
   image = dragObject.AddComponent<Image>()Đ
   image.sprite = Get<Image>().spriteĐ
   image.raycastTarget = falseĐ
   if(uiAbilitySlot!=null):Đ
      image.sprite = uiAbilitySlot.icon.spriteĐ
Đ
+ OnDrag(eventData: PointerEventData):Đ
   dragObject.Get<RectTransform>().position = eventData.positionĐ
Đ
+ OnEndDrag(eventData: PointerEventData):Đ
   Destroy(dragObject)Đ
   if (eventData.pointerEnter != null):Đ
      hotkeySlot = eventData.pointerEnter.Get<UiHotKeySlot>()Đ
      uiAbilitySlot = dragedObject.Get<UiAbilitySlot>()Đ
      uiItemSlot = dragedObject.Get<UiItemSlot>()Đ
      if (hotkeySlot != null):Đ
         if(uiAbilitySlot!=null):Đ
            hotkeySlot.ability = uiAbilitySlot.abilityĐ
            hotkeySlot.item = nullĐ
         else if(uiItemSlot!=null):Đ
            hotkeySlot.item = uiItemSlot.itemĐ
            hotkeySlot.ability = nullĐ
         else:Đ
            hotkeySlot.item = nullĐ
            hotkeySlot.ability = nullĐ
```

hotkeySlot.updateinfo()Đ
rectTransform.localPosition = originalPosition

- UiHotKeySlot at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiHotKeySlot.cs:
Summary of UiHotKeySlot:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiHotKeySlot.csĐ
The UiHotKeySlot.cs file contains a Unity script for a UI element in a hotkey slot. It includes public variables for an Image icon, an integer hotkeyIndex, a UIManager object, a RectTransform object, a Vector3 original position, an Ability object, and an Item object. Đ
Đ
The Start() method initializes the UIManager and RectTransform objects, sets the hotkeyController, and stores the original position of the RectTransform. Đ
Đ
The updateInfo() method updates the icon and hotkeyController object based on whether an ability or an item is assigned to the slot. Đ
Đ
The OnPointerEnter() and OnPointerExit() methods are used to open and close a tooltip using the UIManager object and hotkeyController object at the current RectTransform position. Đ
Đ
Overall, the script handles the functionality of hotkey slots in a UI element, including displaying icons and assigning abilities or items to hotkeys.
Code of file UiHotKeySlot:
using UnityEngine;Đ
using UnityEngine.EventSystems;Đ
using UnityEngine.UI;Đ
Đ
public class UiHotKeySlot : MonoBehaviourĐ
{Đ
    public Image icon;Đ
    public int hotkeyIndex;Đ
    private UIManager uiManager;Đ
    private RectTransform rectTransform;Đ
    public Vector3 originalPosition;Đ
    public Ability ability;Đ
    public Item item;Đ
Đ
    private HotkeyController hotkeyController;Đ
Đ
    private void Start()Đ
    {Đ
        hotkeyController = FindObjectOfType<HotkeyController>();Đ

```
        uiManager = FindObjectOfType<UIManager>();Đ
        rectTransform = GetComponent<RectTransform>();Đ
        icon = GetComponent<Image>();Đ
        originalPosition = rectTransform.localPosition;Đ
Đ
        Đ
        Đ
    }Đ
    public void updateinfo(){Đ
        if(ability!=null){Đ
            icon.sprite = ability.icon;Đ
            hotkeyController.hotkeys[hotkeyIndex].ability = ability;Đ
            hotkeyController.hotkeys[hotkeyIndex].item = null;Đ
        }Đ
        if(item!=null){Đ
            icon.sprite = item.icon;Đ
            hotkeyController.hotkeys[hotkeyIndex].item = item;Đ
            hotkeyController.hotkeys[hotkeyIndex].ability = null;Đ
        }Đ
Đ
Đ
    }Đ
    public  void OnPointerEnter(PointerEventData eventData)Đ
    {Đ
       uiManager.OpenToolTip(hotkeyController.hotkeys[hotkeyIndex],
rectTransform.position);Đ
    }Đ
Đ
    public  void OnPointerExit(PointerEventData eventData)Đ
    {Đ
        uiManager.CloseToolTip();Đ
    }Đ
Đ
Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiHotKeySlot.csĐ
UiHotKeySlot:Đ
- icon: ImageĐ
- hotkeyIndex: intĐ
- uiManager: UIManagerĐ
- rectTransform: RectTransformĐ
- originalPosition: Vector3Đ

- ability: AbilityÐ
- item: ItemÐ
- hotkeyController: HotkeyControllerÐ
Ð
- Start():Ð
  - hotkeyController = FindObjectOfType<HotkeyController>()Ð
  - uiManager = FindObjectOfType<UIManager>()Ð
  - rectTransform = GetComponent<RectTransform>()Ð
  - icon = GetComponent<Image>()Ð
  - originalPosition = rectTransform.localPositionÐ
Ð
- updateinfo():Ð
  - if(ability!=null):Ð
    - icon.sprite = ability.iconÐ
    - hotkeyController.hotkeys[hotkeyIndex].ability = abilityÐ
    - hotkeyController.hotkeys[hotkeyIndex].item = nullÐ
  - if(item!=null):Ð
    - icon.sprite = item.iconÐ
    - hotkeyController.hotkeys[hotkeyIndex].item = itemÐ
    - hotkeyController.hotkeys[hotkeyIndex].ability = nullÐ
Ð
- OnPointerEnter(eventData: PointerEventData):Ð
  - uiManager.OpenToolTip(hotkeyController.hotkeys[hotkeyIndex], rectTransform.position)Ð
Ð
- OnPointerExit(eventData: PointerEventData):Ð
  - uiManager.CloseToolTip()

- UiItemSlot at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiItemSlot.cs:
Summary of UiItemSlot:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\UiItemSlot.csÐ
The file named "UiItemSlot.cs" contains a class with the same name. This class implements the interfaces ButtonWithToolTip, IRecieveDrop, and IDragable. It also has a public item field of type Item. The methods getDraggedObject, OnBeginDrag, OnDrag, OnDrop, and OnEndDrag are empty and do not provide any functionality. The class is intended to be used for UI item slots, allowing for dragging and dropping of items within the UI.
Code of file UiItemSlot:
using UnityEngine;Ð
using UnityEngine.EventSystems;Ð
Ð
internal class UiItemSlot : ButtonWithToolTip, IRecieveDrop,IDragableÐ
{Ð

```
    public Item item;Đ
Đ
    public GameObject getDraggedObject()Đ
    {Đ
        throw new System.NotImplementedException();Đ
    }Đ
Đ
    public void OnBeginDrag(PointerEventData eventData)Đ
    {Đ
        throw new System.NotImplementedException();Đ
    }Đ
Đ
    public void OnDrag(PointerEventData eventData)Đ
    {Đ
        throw new System.NotImplementedException();Đ
    }Đ
Đ
    public void OnDrop(PointerEventData eventData)Đ
    {Đ
        throw new System.NotImplementedException();Đ
    }Đ
Đ
    public void OnEndDrag(PointerEventData eventData)Đ
    {Đ
        throw new System.NotImplementedException();Đ
    }Đ
}
Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\UiItemSlot.csĐ
UiItemSlot:Đ
’Ò 'WGFöåv—F…@oolTipĐ
’Ò •&V6–PveDropĐ
’Ò ”G& gableĐ
’Ò —FVÓ  ItemĐ
’Ò petDraggedObject(): GameObjectĐ
’Ò öä&Vv–äG& g(eventData: PointerEventData)Đ
’Ò öäG& g(eventData: PointerEventData)Đ
’Ò öäG op(eventData: PointerEventData)Đ
’Ò öäVæDG& g(eventData: PointerEventData)

- UIManager at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\UIManager.cs:
Summary of UIManager:
The UIManager.cs file defines a class named UIManager, which is responsible for
```

managing various UI elements in the game. The class has public variables for different game objects that represent UI elements, such as pauseMenu, mainMenu, characterStatusUI, characterUi, tooltipSkills, and skillTreeMenu.Ð
Ð
There are also private variables, including toolTipSkillsController and presentQuestUiController. The former is used to control the tooltip UI for skills, while the latter is used to present quest information to the player.Ð
Ð
The class has various methods for showing and hiding different UI elements, such as showQuestUiPresenter and hideQuestUiPresenter for presenting/hiding quest information, and OpenCharacterStatusUI, OpenCharacterUi, OpenSkillTreeMenu, CloseCharacterUi, and CloseSkillTreeMenu for showing/hiding different menus.Ð
Ð
The class also has methods for pausing/unpausing the game, showing/hiding the main menu, and opening/closing the tooltip UI for skills. These methods change the game state accordingly using the GameManager.Instance.ChangeGameState method.Ð
Ð
The Update method checks for user input to pause/unpause the game and open/close the character status UI and skill tree menu. The Awake method initializes the Tooltip UI controller.
Code of file UIManager:
```
þÿ// UIManager.csÐ
using UnityEngine;Ð
public class UIManager : MonoBehaviourÐ
{Ð
    public GameObject pauseMenu;Ð
    public GameObject mainMenu;Ð
    public GameObject characterStatusUI;Ð
    public GameObject characterUi;Ð
    public GameObject tooltip;Ð
    public GameObject skillTreeMenu;Ð
Ð
    public GameObject questBookUi; Ð
    public GameObject questListQuickUi;Ð
    private ToolTipUiController toolTipController;Ð
    public PresentQuestUiController questUIPresenter;Ð
    public event eventUi onPlayerHealthManaChange;Ð
    public delegate void eventUi();Ð
Ð
public void updateQuestBook(){Ð
    questBookUi.GetComponent<QuestBookUIController>().UpdateQuestList();Ð
}Ð
public void showQuestBookUi(){Ð
```

```
    GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Ð
    questBookUi.SetActive(true);Ð
}Ð
public void hideQuestBookUi(){Ð
    GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Ð
    questBookUi.SetActive(false);Ð
}Ð
public void showQuestListQuickUi(){Ð
    GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Ð
    questListQuickUi.SetActive(true);Ð
}Ð
public void hideQuestListQuickUi(){Ð
    GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Ð
    questListQuickUi.SetActive(false);Ð
}Ð
Ð
public void showQuestUiPresenter(Quest quest){Ð
    GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Ð
    questUIPresenter.showQuestInfo(quest,this);Ð
    questUIPresenter.gameObject.SetActive(true);Ð
}Ð
public void hideQuestUiPresenter(){Ð
    GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Ð
    questUIPresenter.gameObject.SetActive(false);Ð
}Ð
Ð
    private void Update()Ð
    {Ð
        // Check for user input to pause/unpause the gameÐ
        if (Input.GetKeyDown(KeyCode.Escape))Ð
        {Ð
            if (GameManager.Instance.currentState ==
GameManager.GameState.Playing)Ð
            {Ð
                PauseGame();Ð
            }Ð
            else if (GameManager.Instance.currentState ==
GameManager.GameState.Paused)Ð
            {Ð
                UnpauseGame();Ð
            }Ð
        }Ð
Ð
        // Check for user input to open/close the CharacterStatusUIÐ
        if (Input.GetKeyDown(KeyCode.C))Ð
```

```
        {Đ
            if (GameManager.Instance.currentState ==
GameManager.GameState.Playing)Đ
            {Đ
                OpenCharacterUi();Đ
            }Đ
            else if (GameManager.Instance.currentState ==
GameManager.GameState.InMenu)Đ
            {Đ
                CloseCharacterUi();Đ
            }Đ
        }Đ
        if(Input.GetKeyDown(KeyCode.V)){Đ
            if (GameManager.Instance.currentState ==
GameManager.GameState.Playing)Đ
            {Đ
                OpenSkillTreeMenu();Đ
            }Đ
            else if (GameManager.Instance.currentState ==
GameManager.GameState.InMenu)Đ
            {Đ
                CloseSkillTreeMenu();Đ
            }Đ
        }Đ
        if(Input.GetKeyDown(KeyCode.B)){Đ
            if (GameManager.Instance.currentState ==
GameManager.GameState.Playing)Đ
            {Đ
                OpenSkillTreeMenu();Đ
            }Đ
            else if (GameManager.Instance.currentState ==
GameManager.GameState.InMenu)Đ
            {Đ
                CloseSkillTreeMenu();Đ
            }Đ
        }Đ
    }Đ
    public void Awake()Đ
    {Đ
        toolTipController = tooltip.GetComponent<ToolTipUiController>();Đ
    }Đ
Đ
    public void PauseGame()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.Paused);Đ
```

```
        Time.timeScale = 0f;Đ
//        pauseMenu.SetActive(true);Đ
    }Đ
Đ
    public void UnpauseGame()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Đ
        Time.timeScale = 1f;Đ
        pauseMenu.SetActive(false);Đ
    }Đ
Đ
    public void ShowMainMenu()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Đ
        mainMenu.SetActive(true);Đ
    }Đ
Đ
    public void HideMainMenu()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Đ
        mainMenu.SetActive(false);Đ
    }Đ
    public void OpenCharacterStatusUI()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Đ
        characterStatusUI.SetActive(true);Đ
    }Đ
    public void OpenCharacterUi()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Đ
        characterUi.SetActive(true);Đ
    }Đ
    public void CloseCharacterUi()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Đ
        characterUi.SetActive(false);Đ
    }Đ
    public void OpenSkillTreeMenu()Đ
    {Đ
```

```
GameManager.Instance.ChangeGameState(GameManager.GameState.InMenu);Đ
        skillTreeMenu.SetActive(true);Đ
    }Đ
    public void CloseSkillTreeMenu()Đ
    {Đ

GameManager.Instance.ChangeGameState(GameManager.GameState.Playing);Đ
        CloseToolTip();Đ
        skillTreeMenu.SetActive(false);Đ
    }Đ
    public void CloseCharacterStatusUI()Đ
    {Đ
        characterStatusUI.SetActive(false);Đ
    }Đ
    public void OpenToolTip(SkillNode node, Vector3 Positon)Đ
    {Đ
        tooltip.gameObject.SetActive(true);Đ
        tooltip.gameObject.GetComponent<RectTransform>().position = Positon;Đ
        toolTipController.UpdateUI(node);Đ
    }Đ
        public void OpenToolTip(Hotkey hotkey, Vector3 Positon)Đ
    {Đ
        tooltip.gameObject.SetActive(true);Đ
        tooltip.gameObject.GetComponent<RectTransform>().position = Positon;Đ
        if(hotkey.ability != null){Đ
            toolTipController.UpdateUI(hotkey.ability);Đ
        }Đ
        else if(hotkey.item != null){Đ
            toolTipController.UpdateUI(hotkey.item);Đ
        }Đ
    }Đ
    public void OpenToolTip(Ability ability, Vector3 Positon)Đ
    {Đ
        tooltip.gameObject.SetActive(true);Đ
        tooltip.gameObject.GetComponent<RectTransform>().position = Positon;Đ
        toolTipController.UpdateUI(ability);Đ
    }Đ
    public void OpenToolTip(Item item, Vector3 Positon)Đ
    {Đ
        tooltip.gameObject.SetActive(true);Đ
        tooltip.gameObject.GetComponent<RectTransform>().position = Positon;Đ
        toolTipController.UpdateUI(item);Đ
    }Đ
    public void CloseToolTip()Đ
```

```
    {Đ
        tooltip.gameObject.SetActive(false);Đ
    }Đ
}Đ

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My
project\Assets\Scripts\Ui\UIManager.csĐ
UIManagerĐ
- pauseMenu: GameObjectĐ
- mainMenu: GameObjectĐ
- characterStatusUI: GameObjectĐ
- characterUi: GameObjectĐ
- tooltip: GameObjectĐ
- skillTreeMenu: GameObjectĐ
- questBookUi: GameObjectĐ
- questListQuickUi: GameObjectĐ
- toolTipController: ToolTipUiControllerĐ
- questUIPresenter: PresentQuestUiControllerĐ
- onPlayerHealthManaChange: eventUi (delegate)Đ
Đ
+ updateQuestBook(): voidĐ
+ showQuestBookUi(): voidĐ
+ hideQuestBookUi(): voidĐ
+ showQuestListQuickUi(): voidĐ
+ hideQuestListQuickUi(): voidĐ
+ showQuestUiPresenter(quest: Quest): voidĐ
+ hideQuestUiPresenter(): voidĐ
+ PauseGame(): voidĐ
+ UnpauseGame(): voidĐ
+ ShowMainMenu(): voidĐ
+ HideMainMenu(): voidĐ
+ OpenCharacterStatusUI(): voidĐ
+ OpenCharacterUi(): voidĐ
+ CloseCharacterUi(): voidĐ
+ OpenSkillTreeMenu(): voidĐ
+ CloseSkillTreeMenu(): voidĐ
+ CloseCharacterStatusUI(): voidĐ
+ OpenToolTip(node: SkillNode, Position: Vector3): voidĐ
+ OpenToolTip(hotkey: Hotkey, Position: Vector3): voidĐ
+ OpenToolTip(ability: Ability, Position: Vector3): voidĐ
+ OpenToolTip(item: Item, Position: Vector3): voidĐ
+ CloseToolTip(): voidĐ
+ Awake(): voidĐ
+ Update(): void
```

- WorldSpaceCanvasController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\WorldSpaceCanvasController.cs:
Summary of WorldSpaceCanvasController:
File Name: WorldSpaceCanvasController.csÐ
Ð
Summary: Ð
Ð
The WorldSpaceCanvasController class is an implementation of a MonoBehaviour, which is used to manage the World Space Canvas object in Unity. The class contains a static instance of itself which allows it to be easily accessed from other classes in the game.Ð
Ð
Internally, the class also contains a damageNumberPrefab GameObject, which is a prefab that is used to display damage numbers on the World Space Canvas. The class has a method called SpawnDamageNumber, which can be called from other classes to create damage number instances. Ð
Ð
The SpawnDamageNumber method checks if the damageNumberPrefab is assigned to the class. If it is not assigned, it logs an error and returns from the method. If it is assigned, the method instantiates a new damageNumberPrefab object at the specified position and sets the damage value on the DamageNumberController component attached to the object.Ð
Ð
If the DamageNumberController component is not found on the instantiated prefab, it logs an error and destroys the object. Ð
Ð
Overall, the class is responsible for managing and displaying damage numbers on the World Space Canvas in Unity.
Code of file WorldSpaceCanvasController:
þÿusing UnityEngine;Ð
Ð
public class WorldSpaceCanvasController : MonoBehaviourÐ
{Ð
    public static WorldSpaceCanvasController Instance;Ð
Ð
    public GameObject damageNumberPrefab;Ð
Ð
    private void Awake()Ð
    {Ð
        if (Instance == null)Ð
        {Ð
            Instance = this;Ð
        }Ð
        elseÐ

```
        {Đ
            Destroy(gameObject);Đ
        }Đ
    }Đ
Đ
    public void SpawnDamageNumber(float damage, Vector3 position)Đ
    {Đ
        if (damageNumberPrefab == null)Đ
        {Đ
            Debug.LogError("DamageNumberPrefab is not assigned in the
WorldSpaceCanvasController component.");Đ
            return;Đ
        }Đ
Đ
        GameObject damageNumberInstance = Instantiate(damageNumberPrefab,
position, Quaternion.identity, transform);Đ
        damageNumberInstance.gameObject.SetActive(true);Đ
        DamageNumberController damageNumberController =
damageNumberInstance.GetComponent<DamageNumberController>();Đ
Đ
Đ
        if (damageNumberController != null)Đ
        {Đ
            damageNumberController.SetDamageValue(damage);Đ
        }Đ
        elseĐ
        {Đ
            Debug.LogError("DamageNumberController component is missing on the
DamageNumberPrefab.");Đ
            Destroy(damageNumberInstance);Đ
        }Đ
    }Đ
}Đ
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\Ui\WorldSpaceCanvasController.csĐ
WorldSpaceCanvasController:Đ
- InstanceĐ
- damageNumberPrefabĐ
- Awake()Đ
- SpawnDamageNumber(damage, position)

- UnitSpawnerController at C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\WorldManagmentUnitSpawning\UnitSpawnerController.cs:

Summary of UnitSpawnerController:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\WorldManagmentUnitSpawning\UnitSpawnerController.csÐ
The UnitSpawnerController.cs file defines a class that controls the spawning of a unit in a game world. The file includes a number of public variables that specify the characteristics of the unit to be spawned, including its prefab, spawn range, character stats, abilities, AI state, equip manager, and AI controller. Ð
Ð
When the script is started, it gets the transform of the player character in the game world. Then, in the Update() function, it checks if the player is within the spawn range and if the unit has not already been spawned. If both conditions are met, the function spawns the unit using the Instantiate() function, sets all of its properties (stats, abilities, etc.), and destroys the spawner object to prevent further spawns.Ð
Ð
Overall, the UnitSpawnerController.cs file provides a simple and flexible way to spawn units in a game world and customize their behavior and abilities.
Code of file UnitSpawnerController:

```
using System.Collections;Ð
using System.Collections.Generic;Ð
using UnityEngine;Ð
Ð
class UnitSpawnerController : MonoBehaviourÐ
{Ð
    // VariablesÐ
    public GameObject unitPrefab; // Prefab of the unit to spawnÐ
    public float spawnRange; // Range at which to spawn the unitÐ
    public CharacterStats stats; // The stats for the spawned unitÐ
    public Ability[] abilities; // The abilities for the spawned unitÐ
    public AIState aiState; // The AI state for the spawned unitÐ
    public EquipManager equipManager; // The equip manager for the spawned unitÐ
    public AIController aiController; // The AI controller for the spawned unitÐ
Ð
    private Transform playerTransform; // Player transform to check distanceÐ
Ð
    void Start()Ð
    {Ð
        // Get the player transformÐ
        playerTransform = GameObject.FindGameObjectWithTag("Player").transform;Ð
    }Ð
Ð
    bool spawned = false;Ð
    void Update()Ð
    {Ð
        // Check if player is within spawn rangeÐ
```

```
        if(playerTransform == null){Ð
            return;Ð
        }Ð
        if(spawned){Ð
            return;Ð
        }Ð
        if(Vector3.Distance(transform.position, playerTransform.position) <=
spawnRange)Ð
        {Ð
            // Spawn the unitÐ
            GameObject unit = Instantiate(unitPrefab, transform.position,
Quaternion.identity);Ð
            unit.SetActive(true);Ð
            // Set the equip manager for the unitÐ
            unit.GetComponentInChildren<EquipManager>().SetEquipManager(equipM
anager);Ð
Ð
            // Set the AI controller for the unitÐ

unit.GetComponentInChildren<AIController>().SetAIController(aiController);Ð
            Ð
            // Set the stats for the unitÐ
            unit.GetComponentInChildren<CharacterStats>().SetStats(stats);Ð
Ð
            // Set the abilities for the unitÐ
            AbilityController abilityController =
unit.GetComponent<AbilityController>();Ð
            foreach(Ability ability in abilities)Ð
            {Ð
                abilityController.AddAbility(ability);Ð
            }Ð
Ð
            // Set the AI state for the unitÐ
            unit.GetComponentInChildren<AIController>().ChangeState(aiState);Ð
            Destroy(this.gameObject);Ð
        }Ð
    }Ð
}Ð
```

Corresponding SyntaxTree:
C:\Users\Toastbrot\Downloads\STRATEGY 01.04.2022\My project\Assets\Scripts\WorldManagmentUnitSpawning\UnitSpawnerController.csÐ
Name: UnitSpawnerController.csÐ
Ð
- Variables:Ð

- unitPrefab: GameObjectĐ
- spawnRange: floatĐ
- stats: CharacterStatsĐ
- abilities: Ability[]Đ
- aiState: AIStateĐ
- equipManager: EquipManagerĐ
- aiController: AIControllerĐ
- playerTransform: TransformĐ
Đ
- Start():Đ
- Get the player transformĐ
Đ
- Update():Đ
- Check if player is within spawn rangeĐ
- If player transform is null or unit has already spawned, returnĐ
- Spawn the unit and set its equip manager, AI controller, stats, abilities, and AI stateĐ
- Destroy the spawner game object