```
○ ○ ○

///<Summary>
/// Handle player interaction with the phone.
/// Toggles call state, updates handset visuals, and triggers appropriate sound.
///<Summary>
public void Interact()
{
    if (Cooldown) return;
    Cooldown = true;
    _callInStart = false;

    _inCall = !_inCall;
    UpdatePhoneVisuals();
    HandleCallAudio();

    if (_isRining) _voiceMessage.SetActive(true);
    _gameEngineScript.InCall = _inCall;
}

///<Summary>
/// Updates the handset and phone visuals based on call state.
///<Summary>
private void UpdatePhoneVisuals()
{
    _gameEngineScript.Handset.SetActive(_inCall);
    _handset.SetActive(!_inCall);
}

///<Summary>
/// Handles all audio logic when starting, answering, or ending a call.
///<Summary>
private void HandleCallAudio()
{
    if (_inCall)
    {
        StartCoroutine(CooldownForEnd(((_isRining) ? _inCallDelay : _inCallEmptyDelay),
((_isRining) ? true : false)));
        StartCoroutine(CooldownForAnswer());
    }
    else
    {
        PlaySoundEffect(_callEndClip, 1, 1, false);
    }
}

///<Summary>
/// Coroutine for playing answer sounds and in-call loop audio.
///<Summary>
private IEnumerator CooldownForAnswer()
{
    PlaySoundEffect(_answerClip, 1, 0.8f, false);
    yield return new WaitForSeconds(1);
    if (_isRining)
    {
        PlaySoundEffect(_inCallClip, 0.3f, 1, true);
        _isRining = false;
    }
    else
    {
        PlaySoundEffect(_inEmptyCallClip, 0.5f, 1, true);
    }
}

///<Summary>
/// Coroutine to handle cooldown for ending call and optional triggering of tasks/events.
///<Summary>
private IEnumerator CooldownForEnd(float timeEnd, bool setNewTaskAndtriggerEvent)
{
    yield return new WaitForSeconds(timeEnd);
    if (setNewTaskAndtriggerEvent)
    {
        SetNewTaskAndTriggerEvent();
        PlaySoundEffect(_inEmptyCallClip, 1, 1, true);
    }
    Cooldown = false;
}

///<Summary>
/// Sets a new task and triggers an event if the components exist.
///<Summary>
private void SetNewTaskAndTriggerEvent()
{
    if (_newTask) _newTask.SetTask();
    if (_triggerEvent) _triggerEvent.StartEvent();
}

///<Summary>
/// Starts the phone ringing with optional voice message and custom delay.
///<Summary>
public void StartRining(GameObject voiceMessage = null, float inCallDelay = 0)
{
    if (voiceMessage) _voiceMessage = voiceMessage;
    if (inCallDelay > 0) _inCallDelay = inCallDelay;

    _isRining = true;
    PlaySoundEffect(_riningClip, 0.5f, 1, true);
}

///<Summary>
/// Plays the specified audio clip with volume, pitch, and loop options.
///<Summary>
private void PlaySoundEffect(AudioClip clip, float volume, float pitch, bool inLoop)
{
    _audioSource.volume = volume;
    _audioSource.loop = inLoop;
    _audioSource.pitch = pitch;
    _audioSource.clip = clip;
    _audioSource.Play();
}
```