non-Euclidean Record

Recording

PART ONE

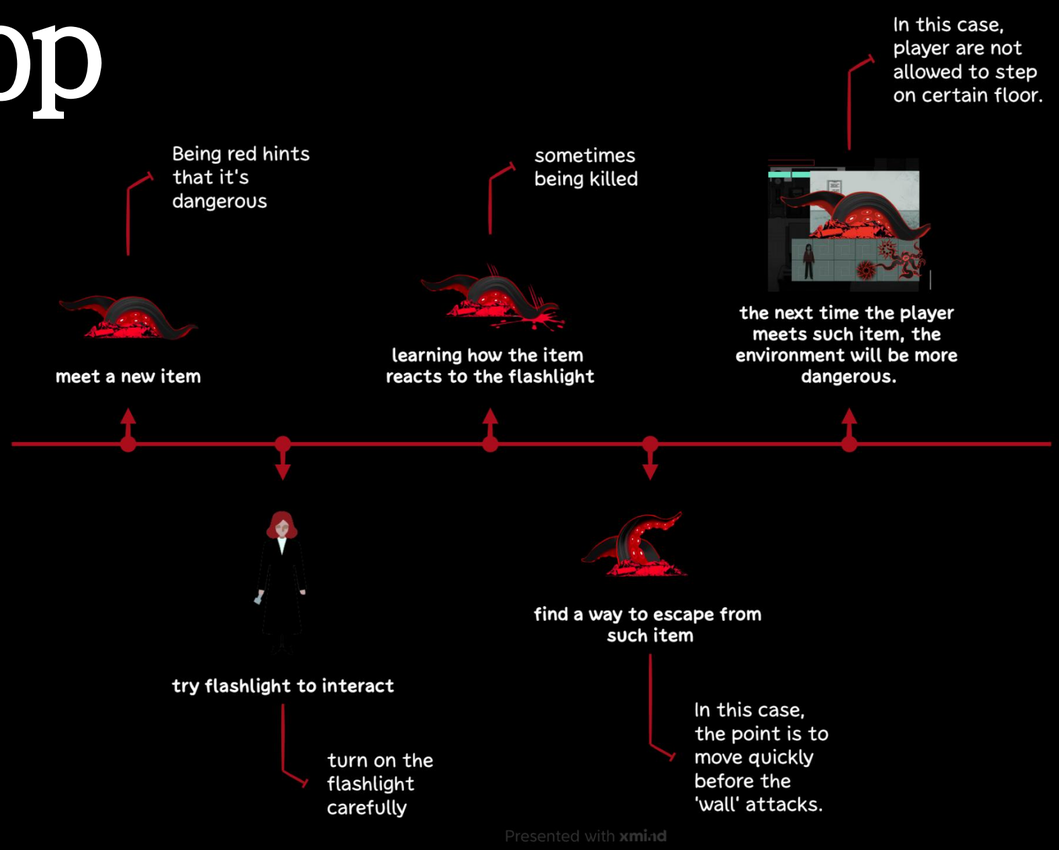
Design Concept

Exploration
Descriptive: Illuminating, understanding unfamiliar environments, fear, darkness and confinement, unknown corners, curiosity, danger, peering over cliff edges
Prescriptive: Navigating fear
Form Contradiction: Exploration brings fear vs. Exploration helps you overcome fear
Shift in Emphasis: Exploration helps you uncover the source and essence of fear
Intellectual Sensation: Heartbeat sensation - Excitement - (Persistence sensation - Directional sense - Planar spatial awareness - Foreign object sensation - Matching sensation - Swallowing sensation)

PART TWO

Game Loop

How does player discover a new item



Then the player will meet another item that has never seen before

PART THREE

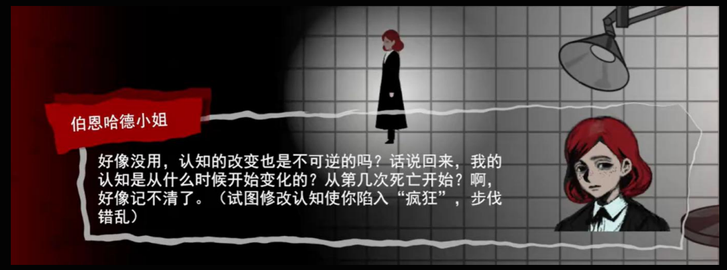
System Design

Three monster mechanics: Each monster interacts differently with the flashlight. Players are encouraged to learn all three mechanics through the process of dying.

Core gameplay: Interact with unknown creatures using a flashlight with limited battery life.

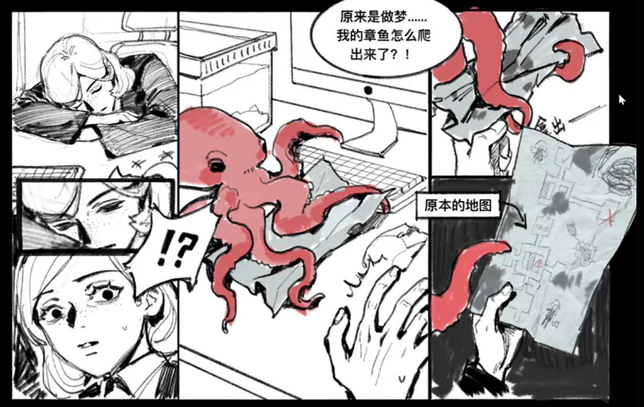
Teleport to the character's position and attack after being exposed to light for too long.

It disappears after being illuminated and consumes surrounding objects.



Death Loop: Each time the player dies, they receive a new debuff (reduced movement speed, inverted vertical movement keys), encouraging players to avoid death whenever possible. The game determines the appearance of monsters players encounter based on the number of deaths. Tentacles initially appear as walls; dying repeatedly transforms them into tentacles.

Explodes upon exposure to light, killing surrounding objects.



The protagonist descends into madness with each death, only to discover in the end that it was all a dream—an octopus crawling onto his map, which explains the appearance of tentacles and the disappearing wrinkles on the map.

PART FOUR

Implementation


```
private void ProcessFlashlightRotation(float baseAngle)
{
    flashlight.transform.rotation = Quaternion.Euler(0, 0, baseAngle - 90);
}

private void DetectObjectsInLight(float baseAngle)
{
    illuminatedObjects.Clear();
    float currentOuterAngle = flashlight.pointLightOuterAngle;
    float startAngle = baseAngle - (currentOuterAngle / 2);
    // 确保 rayCount > 1 避免除零
    float angleStep = (rayCount > 1) ? currentOuterAngle / (rayCount - 1) : 0;
    float currentAngle = startAngle;

    for (int i = 0; i < rayCount; i++)
    {
        Vector2 rayDirection = new Vector2(Mathf.Cos(currentAngle * Mathf.Deg2Rad), Mathf.Sin(currentAngle * Mathf.Deg2Rad));
        RaycastHit2D hit = Physics2D.Raycast(flashlight.transform.position, rayDirection, flashlight.pointLightOuterRadius, obstacleLayers);
        if (hit.collider != null)
        {
            GameObject hitObject = hit.collider.gameObject;
            if (!illuminatedObjects.Contains(hitObject))
            {
                illuminatedObjects.Add(hitObject);
            }
        }
        Debug.DrawRay(flashlight.transform.position, rayDirection * (hit.collider ? hit.distance : flashlight.pointLightOuterRadius), Color.green);
        currentAngle += angleStep;
    }
}
```

Flashlight Logic in Playerlogic Traverse the flashlight's range to detect collisions with objects.

```
public void ResetState()
{
    StopAllCoroutines();
    transform.position = initialPosition;
    transform.rotation = initialRotation;
    SetHierarchyEnabled(true, true);
    wasIlluminated = false;
    isAttackSequenceRunning = false;
    if (myAnimator != null && GameManager.Instance != null)
    {
        int currentDeaths = GameManager.Instance.GetDeathCount();
        if (currentDeaths >= enragedThreshold)
        {
            if (myAnimator.runtimeAnimatorController != enragedAnimatorController)
            {
                myAnimator.runtimeAnimatorController = enragedAnimatorController;
            }
        }
        else
        {
            if (myAnimator.runtimeAnimatorController != normalAnimatorController)
            {
                myAnimator.runtimeAnimatorController = normalAnimatorController;
            }
        }
    }
}
```

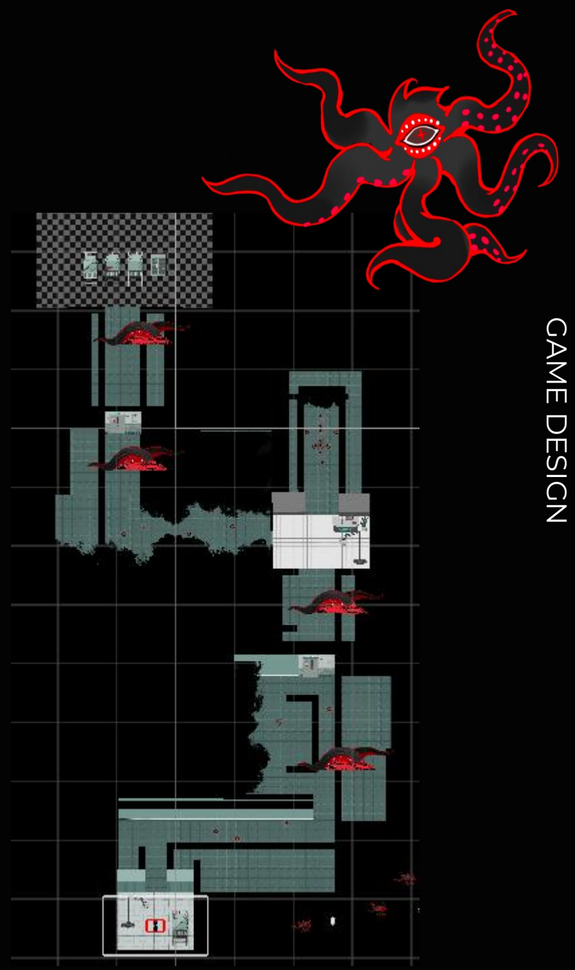
```
private IEnumerator PlayerDiedSequence()
{
    isPlayerDead = true;
    deathCount++;
    Debug.Log($"玩家已死亡！当前死亡次数：{deathCount}");
    if (deathCount >= maxDeath)
    {
        Debug.Log($"已达到最大死亡次数（{maxDeath}）！游戏结束。");
        if (UIManager.Instance != null)
        {
            UIManager.Instance.SetScreenBlack(true);
            yield return new WaitForSeconds(1.5f);
        }
        SceneManager.LoadScene(endingSceneName);
        yield break;
    }
    if (UIManager.Instance != null)
    {
        yield return StartCoroutine(UIManager.Instance.PlayRedFlash());
        UIManager.Instance.SetScreenBlack(true);
    }

    yield return new WaitForSeconds(0.5f);
    if (deathCount > 0 && DialogueManager.Instance != null) { DialogueManager.Instance.PlayDeathSequence(deathCount); }
    UpdateDeathEffects();
    ResetAllWorldObjects();
    usedSpawnsThisLife.Clear();
    ApplyNextDebuff();
    RespawnPlayer();
    if (player != null)
    {
        player.RefillBattery();
        player.EnhanceAuraLight(deathCount);
    }

    if (UIManager.Instance != null)
    {
        StartCoroutine(UIManager.Instance.PlayRespawnFadeIn());
    }
}
```

GameManager
When a player dies, if the player has died a certain number of times, all world items are restored, then the restore code within the items is invoked.
After the player dies, use the interface in PlayerLogic to apply a debuff to the player and spawn them back.

WallLogic's tentacle logic achieves different animator usage based on death count by accessing the death count from the game manager.



Implement the mechanism module first, then proceed with map stitching.

PART FIVE

Live Demo

https://youtu.be/miF5KJkAirQ


POSTMODERN

PART ONE

Game Worldview

In a fictional postmodern society, a person's head morphs according to the ideology they embody: a devout rationalist grows a computer head, a hedonist a champagne bottle, a militarist a star-marked goat. Yet transformation demands two conditions. The ideology must be wholly internalized, lived as one's very essence, and it must be contagious, spreading through contact with another believer. This society has long operated under such laws. These metamorphoses reflect the postmodern decay of thought: reflection is fragmented, meaning diluted into hollow forms. After the death of God, nihilism reigns; appearance replaces essence; the sublime collapses; metaphysics is consumed; life's meaning reduced to survival's empty ritual. Amid this dissolution, our protagonist struggles to reclaim what it means to be human before humanity itself is devoured by its age.

Postmodern

PART TWO

MAIN CHARACTERS

Woodman

The protagonist of the game, who begins anew ten years later to uncover the cause of his friend's death.

Shu

A dear friend who died ten years ago under unknown circumstances.

Elmslay/Gimlet

Elmsly, a former mobster turned straight, now runs a bar popular with the Woodlanders. One day, they approached him with a request.

Gimlet, a shiftless drifter and part-time chronicler for the Human Club, offers to help the Woodlanders investigate it. A card magic enthusiast, he seems an unlikely ally.

Elmsly and Gimlet are, in fact, the same person.

Gene

A specially appointed researcher at the protagonist's company. While investigating the head-shifting phenomenon, he is also meticulously planning an escape.

Wu Mu

The genius of the Human Club. Curious about Shu's death. Willing to help the protagonist investigate Shu's death. But as he delves deeper into Shu's death, he seems to grow increasingly bewildered.

PART THREE

Game Storyline and Gameplay

The game features three intersecting storylines, with players choosing daily which path to advance. Unselected storylines continue to develop independently, meaning different players may experience entirely distinct narratives.

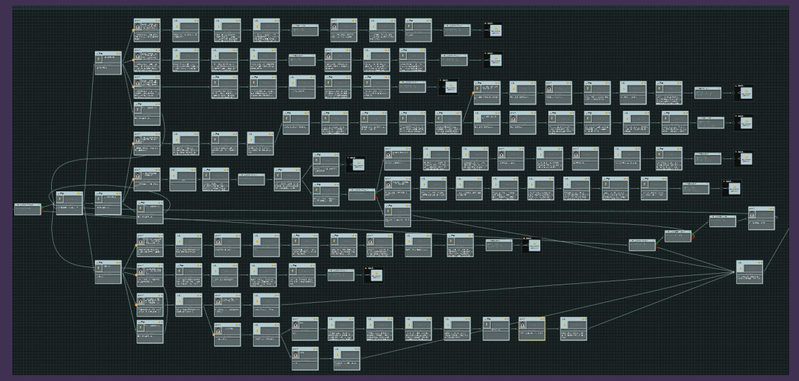
As players advance the story, everyone the protagonist meets during this exploration gradually begins to transform into computer heads. Under these circumstances, players must delve into each storyline, understand the life crises these individuals face, and provide them with the appropriate assistance to restore them to human form. Through this process, players must uncover the true cause of Shu's death.

PART FOUR

Implementation

The in-game text alone totals approximately 200,000 characters, and the game has produced a corresponding amount of content. The game is not yet complete, and it is estimated that the final product will contain around 900,000 characters of text.


```
public void PerformAction(NPCActionType actionType, Vector3[] targetPositions, bool shouldDisappear = false, MovementDirection? newDirection = null)
{
    Debug.Log("NPCActionController: 执行动作 " + actionType);
    switch (actionType)
    {
        case NPCActionType.SpecialAnimation:
            PlaySpecialAnimation();
            break;
        case NPCActionType.MoveAndWalkAnimation:
        case NPCActionType.MoveAndSpecialAnimation:
            if (targetPositions.Length > 1)
            {
                StartCoroutine(MoveThroughTargets(targetPositions, shouldDisappear));
            }
            else
            {
                StartCoroutine(MoveAndAnimateCoroutine(targetPositions[0], shouldDisappear));
            }
            break;
        case NPCActionType.Disappear:
            gameObject.SetActive(false);
            break;
        case NPCActionType.ChangeDirection:
            if (newDirection.HasValue)
            {
                npcAnimatorController.SetDirection(newDirection.Value);
            }
            break;
        case NPCActionType.ChangePosition:
            if (targetPositions != null && targetPositions.Length > 0)
                transform.position = targetPositions[0];
            break;
        default:
            Debug.LogWarning("未处理的 NPCActionType: " + actionType);
            break;
    }
}
```



The game utilizes Articy Draft to facilitate the creation of diverse dialogue options. It also employs its functionality to import these options into Unity.

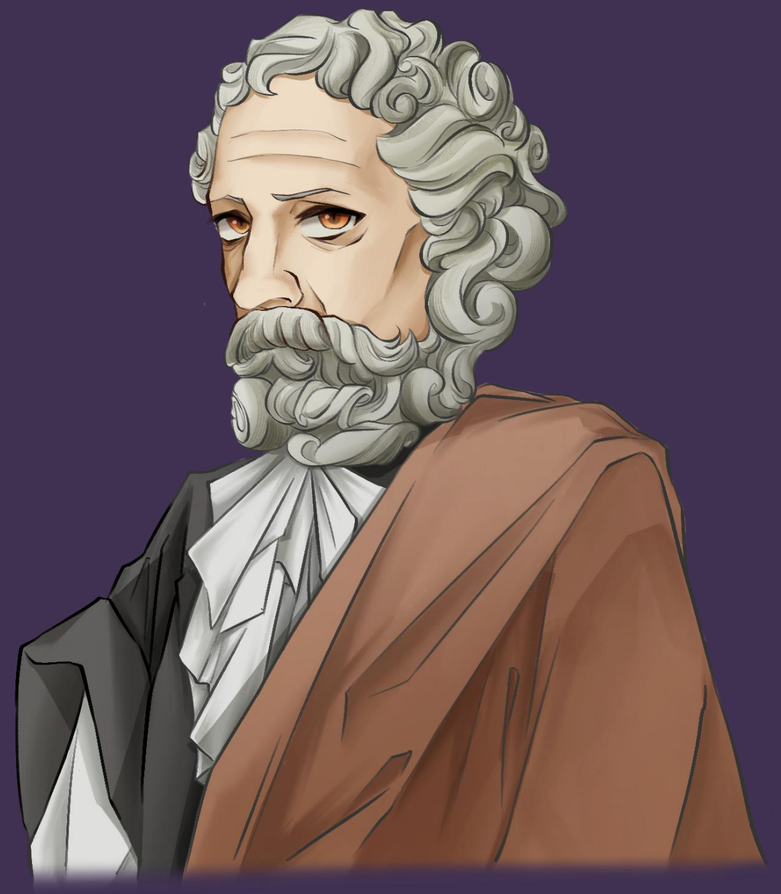
Further enables NPCs to play specific animations and move during conversations.

PART FIVE

Gameplay footage

A 20-minute gameplay-captured dialogue sequence from the Human Club storyline.

https://youtu.be/MLPrcYoDFTw




HUNTER

PART ONE

Conceptual Design of the MDA Framework

Aesthetics
Challenge：
Players will encounter progressively challenging wild monsters on the mission map, each differing in attack frequency, patterns, numbers, and damage output. Overcoming these foes requires persistent effort. During this process, players can craft recovery items using monster drops, fostering a sense of progression. Upon defeating these foes, players experience a profound sense of accomplishment. This stems from the monsters' high damage output and aggressive attacks, which force players into a cycle of observing enemy behavior and making split-second decisions. The relentless process of learning and decision-making culminates in a deeply satisfying sense of achievement. 。

Discovery：
Upon entering the mission map, players explore to locate monsters and engage in hunts. Discovering collectibles and confirming monster distributions during exploration enhances the player experience. Additionally, if players choose not to complete the mission in a single attempt, their familiarity with the map allows them to adopt more efficient strategies upon subsequent visits. The map undergoes terrain changes as the mission progresses, delivering fresh experiences to players.
Submission：
Players achieve immersion through interactions with environments and monsters, while certain progression elements further motivate them to engage with the game, thereby providing entertainment.

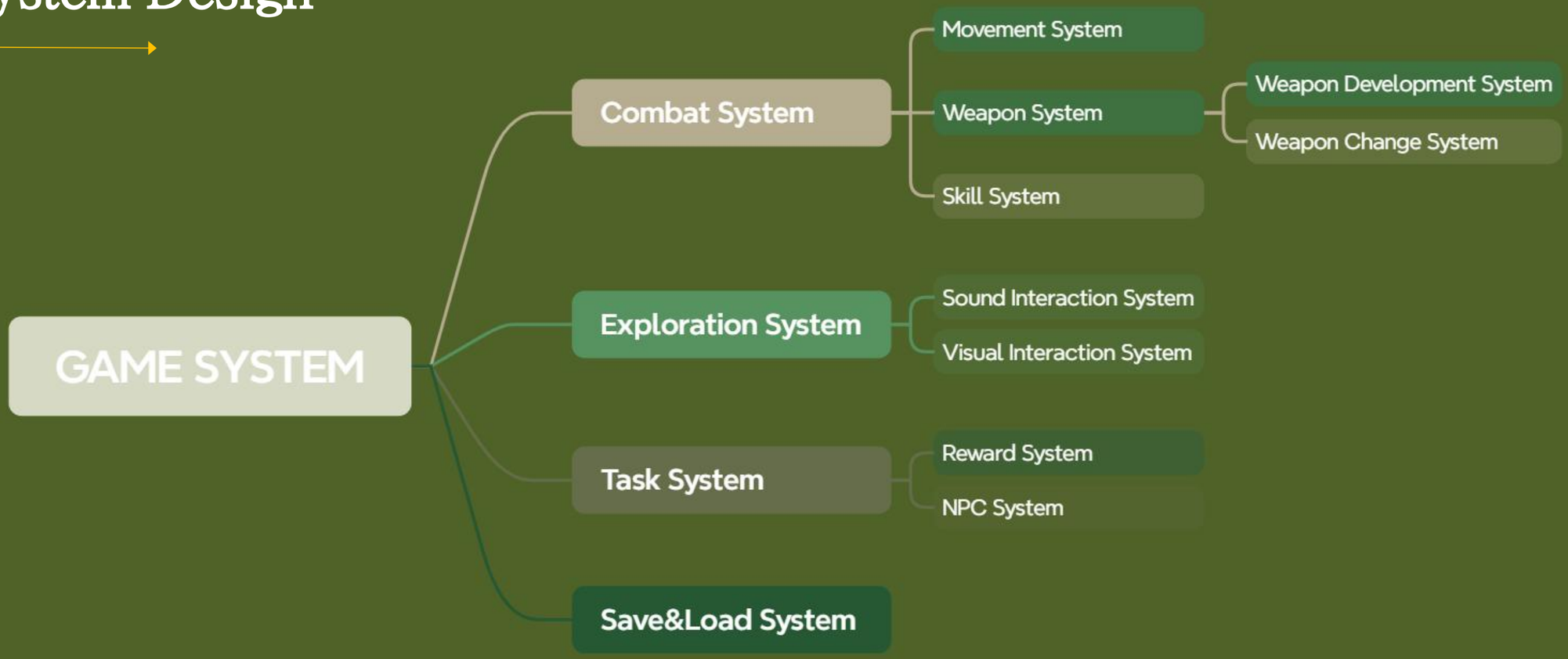
Dynamics
Backpack: Contains loot picked up while exploring the world, drops from slain monsters, materials consumed during crafting/alchemy, and enhanced equipment. Players can sell loot to earn gold.
Skill CD: Skills have a cooldown period after use.
Weapon Switching: Players can switch between different weapons, each with distinct animation sets.
Selling and Purchasing Materials
Crafting Items

Mechanics
Movement: Can move
Crouch: Players are less likely to be detected by monsters while crouched
Weapon Skills
Eagle Vision
Taking Damage
Rolling
Weapon Forging
Shop

HUNTING

PART TWO

System Design



PART THREE

GAMEPLAY footage

https://youtu.be/63yPC1wwQIQ