รายงานการสร้างเกม

HUNT THE HUNTER

คำนำ

"Hunt the Hunter" เป็นเกมแนว Turn-Based RPG 3D ที่ออกแบบมาเพื่อทดสอบทักษะของผู้เล่นผ่านการวางแผนกลยุทธ์และการเอาตัวรอดจากเหล่าปีศาจในดันเจี้ยนผู้เล่นจะได้รับบทเป็น"ซอล"นักล่าปีศาจในตำนานที่ต้องออกเดินทางเพื่อล้างแค้นและทำลายล้างศัตรูที่พรากครอบครัวของเขาไป

เกมนี้นำเสนอประสบการณ์ที่ผสมผสานระหว่าง ความท้าทายในการต่อสู้, ระบบการพัฒนาตัวละคร และการสำรวจดันเจี้ยน ทุกการตัดสินใจของผู้เล่นจะส่งผลต่อแนวทางการเล่นและชะตากรรมของตัวละคร ระบบการต่อสู้แบบเทิร์นเบสช่วยให้ผู้เล่นต้องคิดและวางแผนล่วงหน้าเพื่อเอาชนะศัตรู ไม่ว่าจะเป็นมอนสเตอร์ธรรมดาหรือบอส

นอกจากเกมเพลย์ที่ลุ่มลึก "Hunt the Hunter" ยังมาพร้อมกับบรรยากาศ ยุคกลางแฟนตาซีที่มืดมน พร้อมเรื่องราวเข้มข้นที่เต็มไปด้วยการล้างแค้นและความขัดแย้งระหว่างมนุษย์กับปีศาจ โลกของเกมถูกออกแบบให้มีเอกลักษณ์เฉพาะตัว ทั้งดันเจี้ยนสุดอันตราย อาวุธและอุปกรณ์มากมายให้ค้นหา รวมถึงระบบศัตรูที่มี AI

GDD

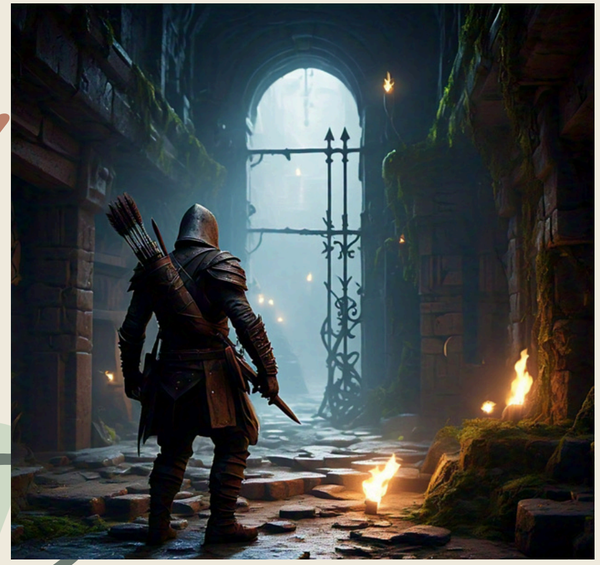
GAME DEVELOPMENT DOCUMENT

Hunt the hunter

Hunt the hunter เป็นเกม Turnbased Rpg 3D ที่ผู้เล่นจะได้รับบทเป็นนักล่าปีศาจทำการตะลุยดันเจี้ยนและอัพเกรดสกิลของตัวละคร เพื่อเตรียมตัวให้พร้อมสำหรับดันเจี้ยนต่อไป

ธีม

ยุคกลาง แฟนตาซี มืดมน



วัตถุประสงค์

1. ควบคุมตัวละครเพื่อโค่นล้มเหล่าปีศาจ
2. พัฒนาตัวละครและความสามารถ
3. โค่นล้มบอสภายในดันเจี้ยน
4. เคลียร์ดันเจี้ยนเพื่อไปดันเจี้ยนถัดไป

เป้าหมาย

สร้างเกมแนว Turnbased Rpg 3D ที่มีความสนุกสนานระหว่างความท้าทายในการเคลียร์ด่านและการพัฒนาตัวละคร รวมถึงสามารถเล่นซ้ำได้ เพื่อมอบประสมการณ์การเล่นเกมที่ตื่นเต้น

ผู้เล่นจะได้สวมบทบาทเป็นนักล่าปีศาจที่ต้องการจะล้างบางเหล่าปีศาจพร้อมกับพัฒนาความสามารถของตัวละครเพื่อเตรียมพร้อมสำหรับดันเจี้ยนต่อไป

กลุ่มเป้าหมาย

ผู้เล่นที่ชื่นชอบในเกม Turnbased Rpg 3D และเกมที่มีธีมยุคกลาง แฟนตาซี

ความสนใจ

ผู้เล่นที่สนใจยุคกลางแฟนตาซี,เกมแนว Turnbased Rpg 3D,การพัฒนาตัวละคร

ประเภทความท้าทาย

1. ความท้าทายด้านการเอาชีวิตรอด: ผู้เล่นต้องเอาชีวิตรอดและจัดการกับศัตรูเพื่อเคลียร์ดันเจี้ยน
2. ความท้าทายจากดันเจี้ยนที่สุ่มสร้าง: ผู้เล่นต้องรับมือกับดันเจี้ยนที่สุ่มสร้างใหม่
3. ความท้าทายด้านศัตรู: ผู้เล่นต้องรับมือกับศัตรูหลากหลายรูปแบบ
4. ความท้าทายด้านระบบความถาวร: ผู้เล่นต้องรับมือกับระบบการตายถาวร ถ้าหากตายสแต็กที่อัพเกรดมาจะหายไปทั้งหมด

ระดับความท้าทาย

1.ปรับระดับความยากอัตโนมัติ

รูปแบบความท้าทาย

1. โหมดการรุกราน: ผู้เล่นต้องเข้าไปในดันเจี้ยนและกำจัดศัตรู
2. โหมดการอยู่รอด: ผู้เล่นต้องอยู่รอดในสถานการณ์ที่ยากลำบาก
3. โหมดการเอาชนะ: ผู้เล่นต้องเอาชนะเหล่าศัตรู

เป้าหมายและผลตอบแทน

1.short term goal ทำลายศัตรู รางวัลคือค่าประสบการณ์ที่สามารถนำไปอัพเกรดตัวละครได้
2.medium term goal ปราบบอส รางวัลคือค่าประสบการณ์ที่มากขึ้นรวมถึงไอเท็มพิเศษ
3.long term goal ปราบบอสสุดท้าย รางวัลคือ ฉากจบ บทสรุปของเรื่องราว

PILLAR 1: การวางกลยุทธ์แบบปรับตัว

1.การวางแผนในการโจมตีศัตรู
2.การจัดการทรัพยากรและสกิล
3.การพัฒนาตัวละคร
4.การใช้กลยุทธ์เพื่อชนะศัตรู
5.การปรับตัวให้เข้ากับสถานการณ์ที่เปลี่ยนแปลง

PILLAR 2: การสำรวจและพัฒนา

1.การสำรวจแผนที่และด่านต่างๆ
2.การอัพเกรดตัวละคร
3.การค้นหาอุปกรณ์ใหม่ๆ
4.การปลดล็อคความสามารถและตัวละครใหม่

การเชื่อมโยงระหว่าง PILLAR 1 และ 2

1. การสำรวจช่วยสนับสนุนการวางกลยุทธ์
2. การพัฒนาตัวละครเพื่อเพิ่มความยืดหยุ่นในการปรับกลยุทธ์
3. การค้นหาอุปกรณ์ช่วยจัดการทรัพยากรและเพิ่มประสิทธิภาพการต่อสู้
4. การสำรวจเปิดโอกาสในการพัฒนากลยุทธ์ใหม่
5. การพัฒนาความสามารถเพิ่มความได้เปรียบในสถานการณ์ที่เปลี่ยนแปลง

วัตถุประสงค์ของ PILLAR 1 และ 2

1.สร้างประสบการณ์การเล่นที่มีความท้าทายและสนุก
2.ส่งเสริมการวางแผนและกลยุทธ์
3.สร้างความพึงพอใจในการเล่น

STORY
AND
WORLD SETTING

เรื่องราว

ในอาณาจักรยุคกลางแฟนตาซี ผู้เล่นจะได้รับบทเป็น"ซอล"นักล่าปีศาจในตำนานที่ผันตัวมาใช้ชีวิตสงบสุขกับลูกและเมีย แต่วันนึง"คาซิน"ปีศาจที่ทรงพลังต้องการจะจบเชื้อสายของซอล เขาจึงนำกองทัพออกล่าซอลและครอบครัวจนท้ายที่สุด ซอลได้สูญเสียเมียและลูก เขาจึงต้องกลับมาล่าพวกปีศาจอีกครั้ง โดยเล็งเป้าหมายหลักไปที่ศาสนา"prey nightmare"ของคาซิน

โลกของเกม

1.ดันเจี้ยน: สร้างดันเจี้ยนที่มีลักษณะเฉพาะและน่าสนใจ และเข้ากับบอสภายในด่าน
2.ประวัติศาสตร์: สร้างประวัติศาสตร์ของโลกภายในเกมให้น่าสนใจ
3.สิ่งมีชีวิต: สร้างปีศาจที่มีเอกลักษณ์เฉพาะตัว

โลกของเกมฉากและพื้นที่

1.ดันเจี้ยน(ของบอสแต่ละตัว)
2.ด้านนอกดันเจี้ยน

ตัวละคร

1.ตัวละครหลัก: สร้างตัวละครหลักที่มีบุคลิกและความสามารถที่น่าสนใจ
2.ตัวละคร npc: สร้างตัวละคร npc ที่สำคัญต่อเนื้อเรื่องและเกมเพลย์

ตัวละคร ระบบเติบโตตัวละคร

1.ระบบสแต็ท: ตัวละครมีค่าพลังพื้นฐานและสามารถอัพเกรดได้
2.ระบบทักษะ: ตัวละครสามารถเลือกสกิลเพื่อใช้งานได้
3.ระบบอุปกรณ์: ตัวละครสามารถใช้อุปกรณ์เสริมและอุปกรณ์พิเศษได้


DEVIL BANE

GAMEPLAY

ระบบการเล่น

1.ระบบตัวละคร: ตัวละครมีทักษะและความสามารถพิเศษ
2.ระบบอุปกรณ์: ผู้เล่นสามารถเลือกใช้อุปกรณ์ได้

ระบบการต่อสู้

1.การควบคุม: ผู้เล่นสามารถเดินได้อย่างอิสระ
2.สกิล: ผู้เล่นสามารถเลือกใช้และพัฒนาสกิลได้
3.สภาพแวดล้อม: สภาพแวดล้อมของแผนที่มีผลต่อความยากในการต่อสู้

การเคลื่อนที่

ผู้เล่นสามารถเคลื่อนที่ได้ด้วยการคลิ๊กซ้าย

ระบบการพัฒนาตัวละคร

1.ระบบเลเวล: ตัวละครจะเลเวลอัพเมื่อได้รับค่าประสบการณ์ การเลเวลอัพจะเพิ่มค่าสถานะและปลดล็อกทักษะใหม่
2.ตัวละครมีค่าสถานะพื้นฐาน: Hp,Mp,Atk,Spd
3.ระบบสกิล: ใช้แต้มในเกมสำหรับอัพเกรดสกิล
4.อุปกรณ์: ตัวละครสามารถสวมใส่อาวุธ และไอเท็มพิเศษได้

ระบบสกิล

1.ทักษะโจมตี: เช่น สายฟ้าฟาด
2.ทักษะสนับสนุน: เช่น การรักษาพลังชีวิต
3.ทักษะอรรถประโยชน์: เช่น การฟื้นฟูมานา

ระบบสำรวจ

1.ห้องมอนสเตอร์: ห้องที่เมื่อเข้าไปจะเจอกับมอนสเตอร์
2.ห้องไอเท็ม: ห้องที่เมื่อเข้าไปจะพบกับกล่องไอเท็ม
3.ห้องร้านค้า: ห้องที่เมื่อเข้าไปจะพบกับร้านค้าไอเท็ม

GRAPHIC AND SOUND

กราฟิกและเสียง

1.กราฟฟิก 3D
2.แอนิเมชัน
3.เสียงประกอบ
4.เพลงประกอบ

รูปแบบศิลป์

1.สไตล์: สไตล์กราฟิกที่เป็นแนว พิกเซล แฟนตาซี มืดมน
2.สีสัน: ใช้สีที่สื่อถึงบรรยากาศของยุคกลาง แฟนตาซี มืดมน

บรรยากาศ

1.ฉาก: ดันเจี้ยน หนองน้ำ ห้องอัศวิน สวนใต้ดิน ห้องแม่มด โบสถ์
2.แรงบันดาลใจ: วัฒนธรรมและสถาปัตยกรรมยุคกลางของยุโรป

ตัวละคร

1.ฝ่ายผู้เล่น: ซอล
2.ฝ่ายศัตรู: มอนสเตอร์ ปีศาจ

รูปแบบกราฟิก

1.ตัวละคร 3D สไตล์พิกเซล
2.เอฟเฟต์พิเศษ เช่น แสงไฟจากคบเพลิง เงา การโจมตีของผู้เล่น และการโจมตีของบอส

เสียง

1.เพลงประกอบ: สร้างเพลงประกอบที่น่ากลัว ลุ้นระทึก
2.เสียงเอฟเฟกต์: เสียงการโจมตี เสียงบรรยากาศ

UX/UI

เมนูหลัก

1.เริ่มเกม
2.เล่นต่อ
3.ตัวเลือก
4.ออก

หน้าต่างในเกม

1.แสดงข้อมูลตัวละคร: Hp,Mp

หน้าต่างอัปเกรด

1.แสดงรายการอาวุธ/สกิลที่ใช้ได้
2.แสดงสแต็ทตัวละคร

ANOTHER SYSTEM

ระบบควบคุม

1.การเคลื่อนไหว: ผู้เล่นสามารถควบคุมตัวละครได้
2.การโจมตี: ผู้เล่นสั่งโจมตีได้
3.การใช้สกิล: ผู้เล่นสามารถสั่งใช้สกิลได้
4.การสลับอาวุธ: ผู้เล่นสามารถสลับอาวุธได้

ระบบ AI

1.Ai สำหรับปรับความยากของด่าน
2.Ai สำหรับปรับระดับของร้านค้า
3.Ai สำหรับควบคุมศัตรู

TOOLS

เครื่องมือพัฒนา

Unity Engine: สำหรับพัฒนากราฟิกและระบบเกม

ภาษาสำหรับพัฒนา

C#: ใช้เขียนโค้ดสำหรับระบบเกม

เครื่องมือ3มิติ

Blender: สำหรับสร้างโมเดล 3 มิติ

เครื่องมือ2มิติ

Gimp

PROTOTYPE

โครงสร้างแผนที่

1.แผนที่ขนาด 10x10 ตาราง
2.มีสิ่งกีดขวาง เช่น กล่อง กำแพง

ตัวละครเบื้องต้น

1.ตัวละครผู้เล่น: ซอล
2.ตัวละครศัตรู: ออร์คเอ

ตัวละครเบื้องต้น

ชื่อ: ซอล

พื้นเพ:อดีตตำนานนักล่าปีศาจที่ผันตัวมาใช้ชีวิตสงบสุขกับลูกและเมีย แต่ว่าวันหนึ่งเกิดการรุกรานจากมอนส์เตอร์ ของศาสนา
“Prey nightmare”เหตุการณ์นั้นทำให้เขาเสียเมียและลูกไป ทำให้เขาต้องกลับไปทำงานเดิมอีกครั้ง โดยเล็งเป้าหมายหลักไปที่“Prey nightmare”

รูปลักษณ์:
สวมแจ็คเก็ตหนังดูเก่าๆที่แสดงถึงประสบการณ์ของเขา และบาดแผลตรงหน้าที่ถูกทิ้งไว้หลังจากเหตุการณ์เลวร้าย
อาวุธ:ดาบไร้ชื่อ(Nameless sword) และ หน้าไม้เดวิลเบน(Devil bane crossbow)

ตัวละครเบื้องต้น

ค่าสถานะเริ่มต้น:
Hp: 100
ทักษะ:
Slash :โจมตี 15 ดาเมจ
Shoot :โจมตี 20 ดาเมจ
Defend :ป้องกัน 3 วิ
Run:หนีจากการต่อสู้

ตัวละครเบื้องต้น

ตัวละครเบื้องต้น

ชื่อ: ออร์คเอ

พื้นเพ:เขาเป็นมอนส์เตอร์บรรพกาลที่ทรงพลังอย่างมาก แต่กลับถูกกำราบลงโดยกลุ่มนักล่าปีศาจ ทว่าก่อนที่เขาจะตาย เขาได้ย้อนเวลามาเพื่อแก้ไขก่อนที่ทุกอย่างจะเกิดขึ้น โดยการฆ่าซอล เพื่อไม่ให้เกิดลูกหลานของนักล่าที่แข็งแกร่งในอนาคต

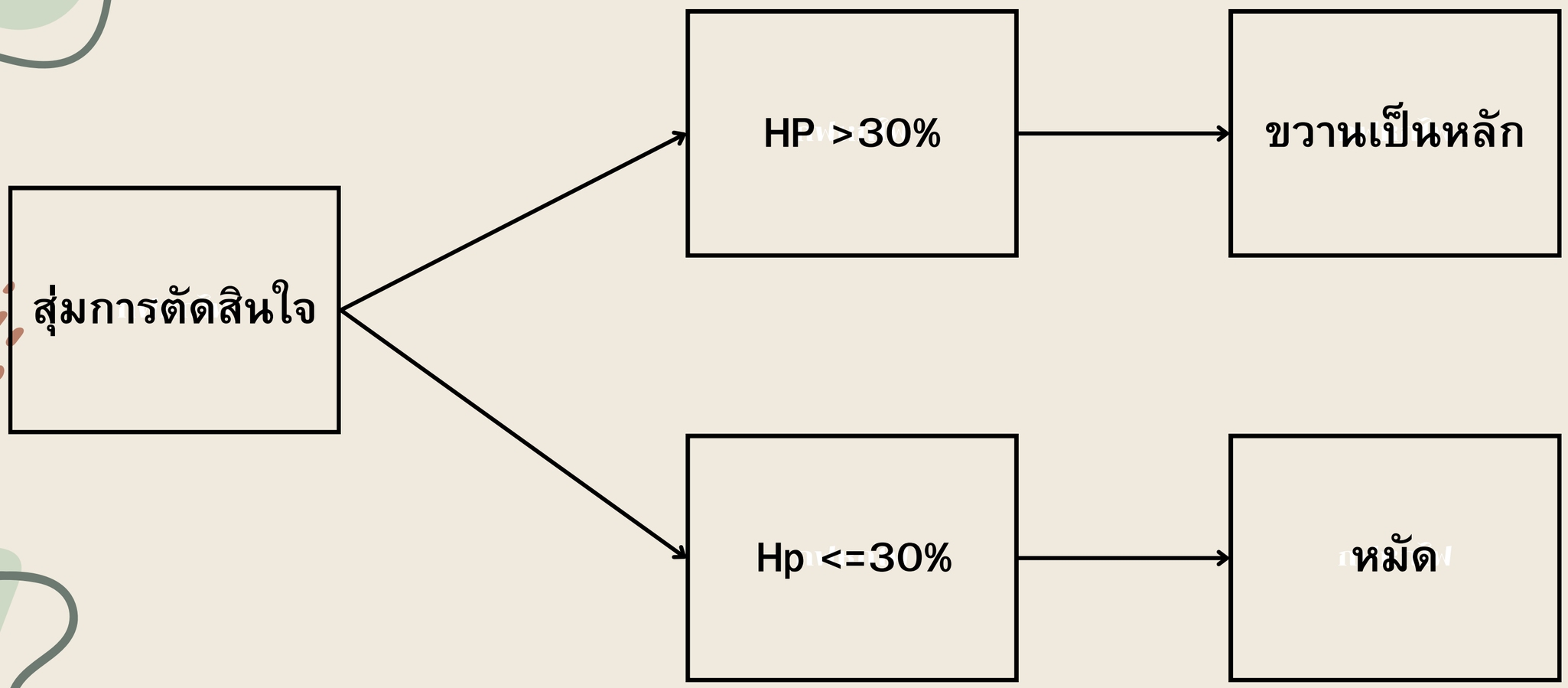
รูปลักษณ์:
ออร์คขนาดมหึมา ผิวสีเขียวเข้มจนเกือบดำ ผิวหนังเต็มไปด้วยรอยแผลเป็นจากสงคราม กล้ามเนื้อของเขาแน่นราวกับหิน
อาวุธ:ขวานปีศาจ(Demon Axe)

ตัวละครเบื้องต้น

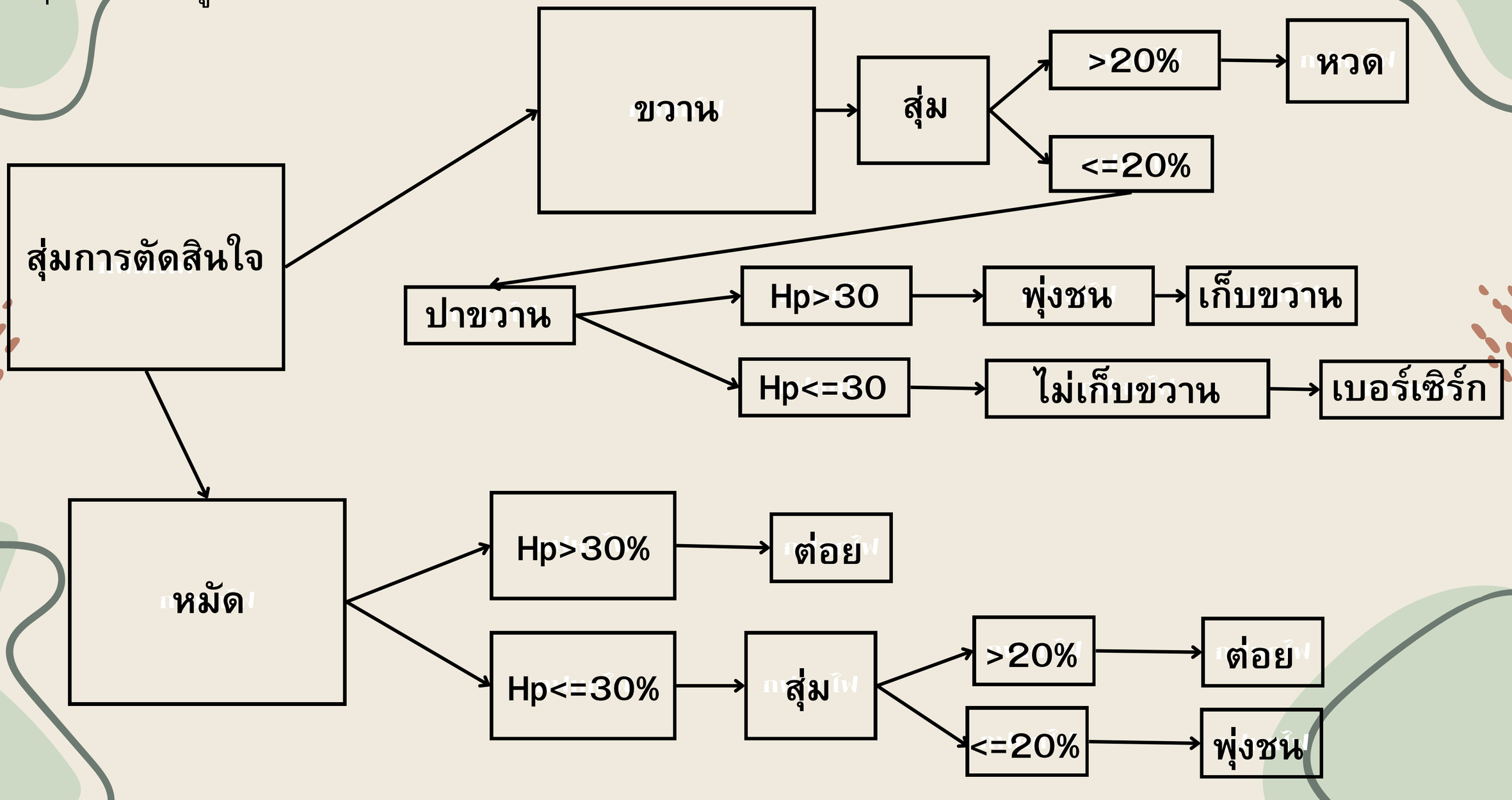
ค่าสถานะเริ่มต้น:
Hp: 150

ตัวละครเบื้องต้น

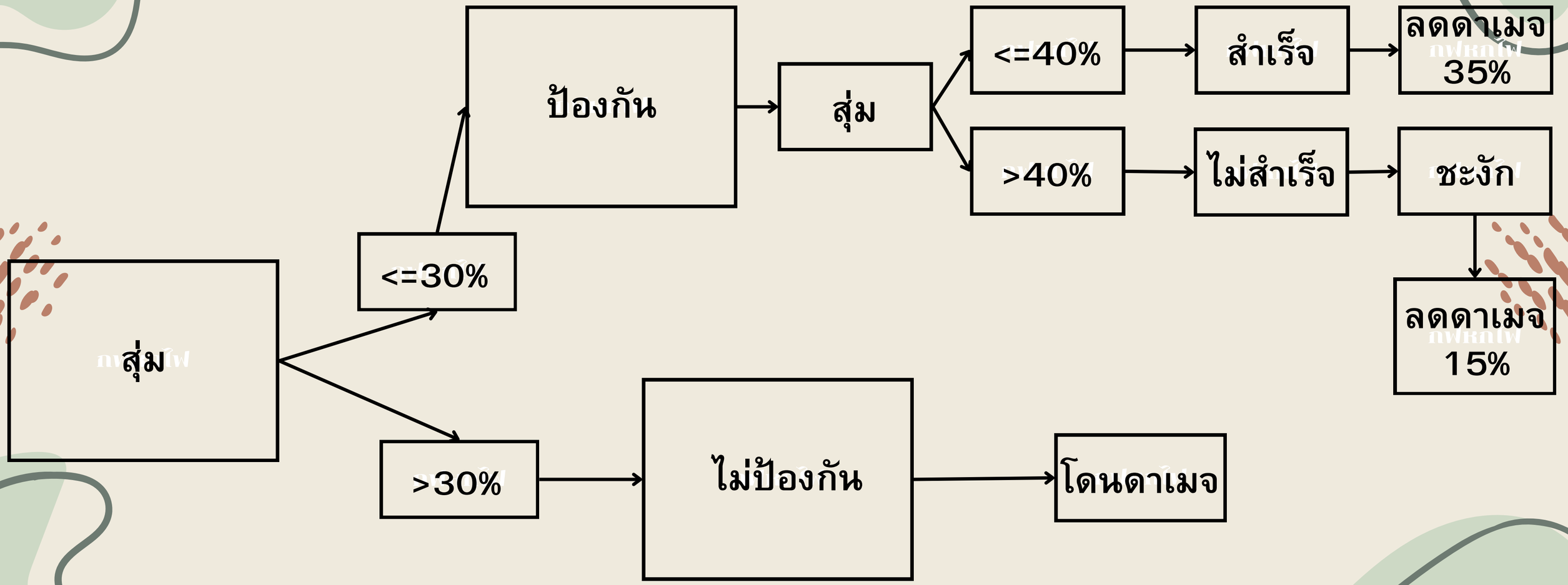
พฤติกรรมศัตรู เลือกอาวุธ



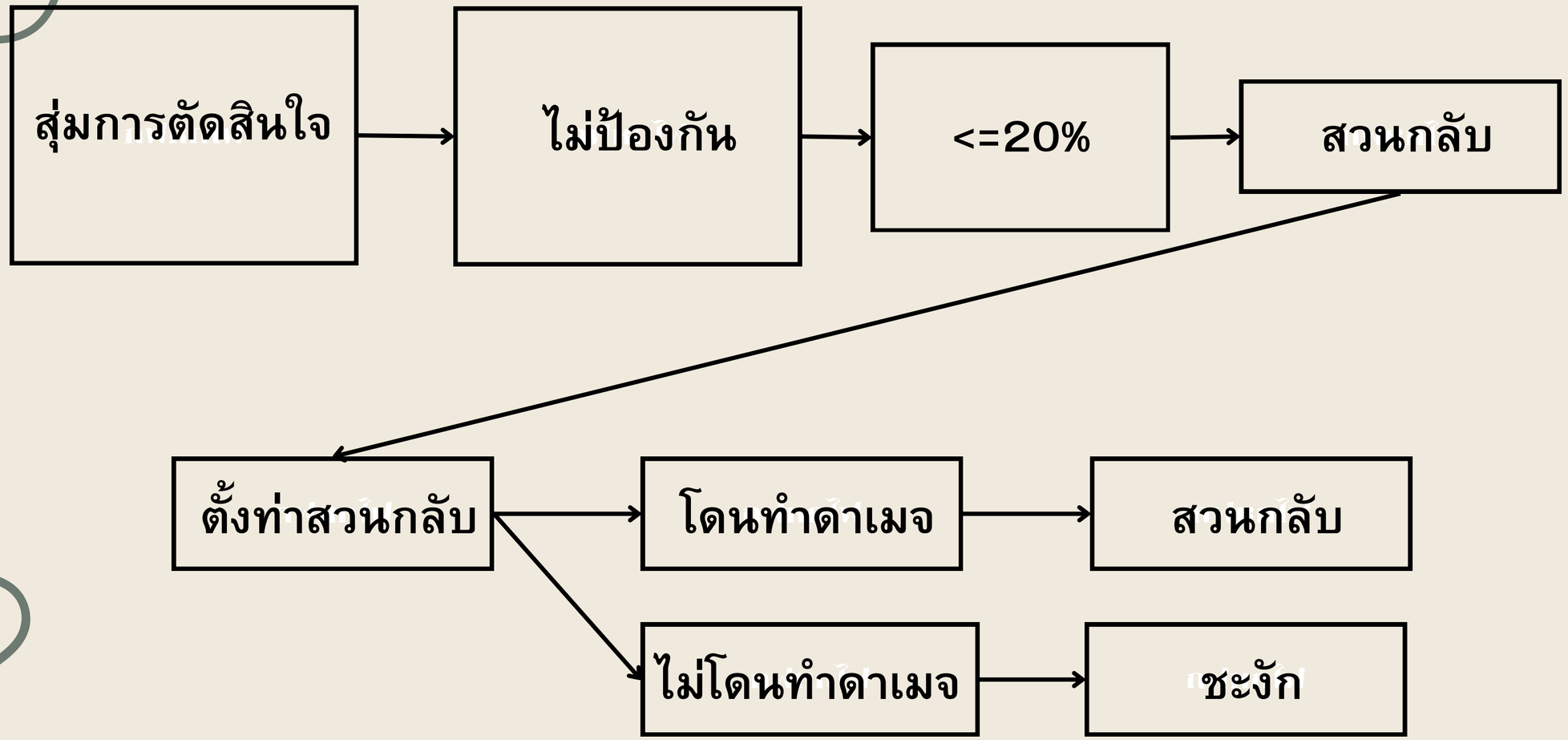
พฤติกรรมศัตรู อัตราการโจมตี



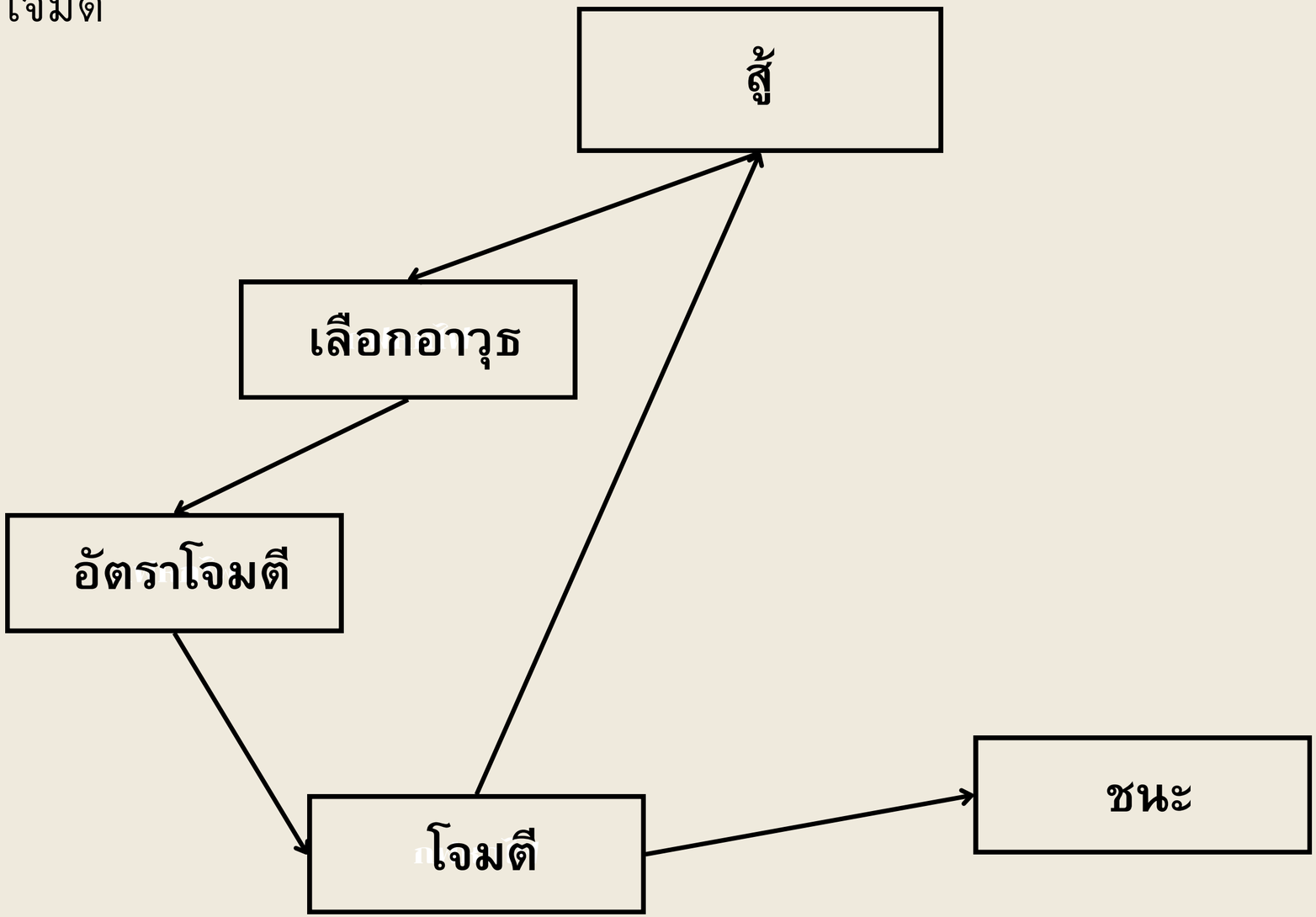
พฤติกรรมศัตรู ป้องกัน



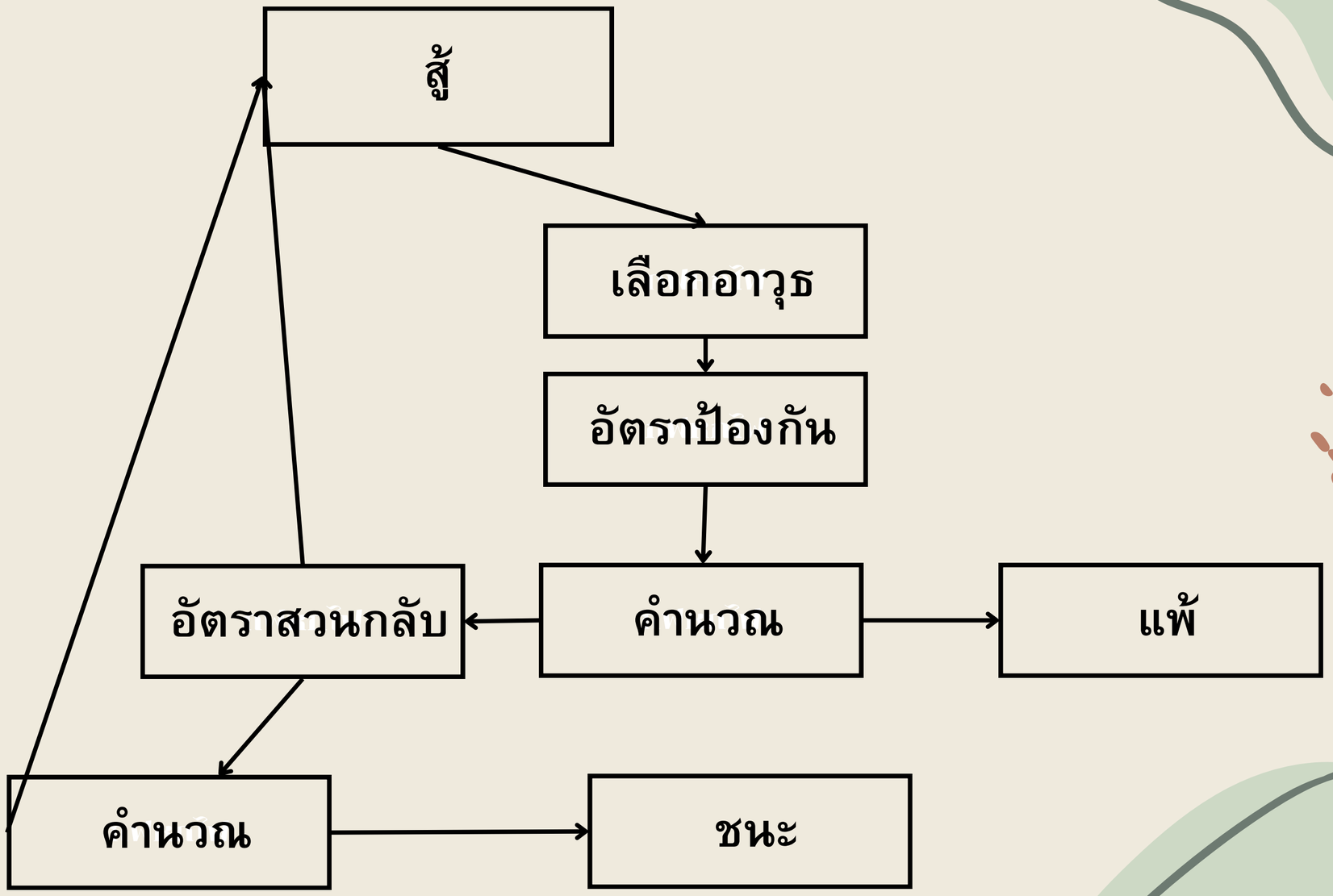
พฤติกรรมศัตรู สวนกลับ



ผังพฤติกรรม โจมตี



ผังพฤติกรรม ป้องกัน



CODE:MAINMENUSCENE

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenuScene : MonoBehaviour
{
    public void QuitGame()
    {

        #if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
        #else

        Application.Quit();
        #endif
    }

    public void LoadLobby()
    {
        SceneManager.LoadScene("Lobby");
    }
}
```

CODE:TEXTCONTROLLER

```
using UnityEngine;
using UnityEngine.UI;

public class TextController : MonoBehaviour
{
    public Text descriptionText;

    void Start()
    {
        UpdateText("Welcome Sol!!");
    }

    public void UpdateText(string newText)
    {
        descriptionText.text = newText;
    }
}
```

CODE:GATEDOOR

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Gatedoor : MonoBehaviour
{
    private void OnCollisionEnter(Collision collision)
    {
        Debug.Log("Collision with: " + collision.gameObject.name);

        if (collision.gameObject.CompareTag("Player"))
        {
            Debug.Log("Player hit the gate. Loading Dungeon Scene...");
            SceneManager.LoadScene("Dungeon");
        }
    }
}
```

CODE:RANDOMBOX

```
using UnityEngine;

public class RandomBox : MonoBehaviour
{

    private string[] items = { "Potion", "Sword", "Shield", "Gold", "Rare Gem" };

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            string randomItem = GetRandomItem();
            Debug.Log("คุณได้รับ: " + randomItem);
            Destroy(gameObject);
        }
    }

    private string GetRandomItem()
    {
        int randomIndex = Random.Range(0, items.Length);
        return items[randomIndex];
    }
}
```

CODE:PLAYERSCRIPTS

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class PlayerScripts : MonoBehaviour
{
    public float moveSpeed = 5.0f;
    private Vector3 targetPosition;

    void Update()
    {

        if (Input.GetMouseButtonDown(0))
        {

            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;


            if (Physics.Raycast(ray, out hit))
            {

                targetPosition = hit.point;
            }
        }


        if (targetPosition != Vector3.zero)
        {
            transform.position = Vector3.MoveTowards(transform.position, targetPosition, moveSpeed * Time.deltaTime);
        }
    }
}
```

CODE:SCENEMANAGERS

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneManagers : MonoBehaviour
{
    public void QuitGame()
    {
        Application.Quit();
    }

    public void LoadLevel1()
    {
        SceneManager.LoadScene("Lobby");
    }
}
```

CODE:ENEMYSPAWNER

```
using UnityEngine;

public class EnemySpawner : MonoBehaviour
{
    public GameObject enemyPrefab;
    public Transform spawnPoint;
    private bool isEnemySpawned = false;

    void Start()
    {

        if (!isEnemySpawned)
        {
            SpawnEnemy();
            isEnemySpawned = true;
        }
    }

    public void SpawnEnemy()
    {

        Instantiate(enemyPrefab, spawnPoint.position, spawnPoint.rotation);
    }

    public void ResetSpawnStatus()
    {
        isEnemySpawned = false;
    }
}
```

CODE:ENEMY

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Enemy : MonoBehaviour
{
    public string battleScene = "BattleScene";
    public float moveSpeed = 3.0f;
    public float chaseDistance = 10f;
    private Transform player;

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void Update()
    {

        float distanceToPlayer = Vector3.Distance(transform.position, player.position);


        if (player != null && distanceToPlayer <= chaseDistance)
        {

            transform.position = Vector3.MoveTowards(transform.position, player.position, moveSpeed * Time.deltaTime);
        }

        else
        {

            transform.position = transform.position;
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            SceneManager.LoadScene(battleScene);
        }
    }
}
```

CODE:HPMANAGER

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class HPManager : MonoBehaviour
{
    public Slider hpBar;
    public Text hpText;
    private float maxHP = 100f;
    private float currentHP;

    public float MaxHP => maxHP;
    public float CurrentHP => currentHP;

    void Start()
    {
        currentHP = maxHP;
        UpdateHPUI();
    }

    public void TakeDamage(float damage)
    {
        currentHP -= damage;
        currentHP = Mathf.Clamp(currentHP, 0, maxHP);
        UpdateHPUI();

        if (currentHP <= 0)
        {
            Debug.Log("Player died.");
            SceneManager.LoadScene("LoseScene");
        }
    }

    void UpdateHPUI()
    {
        hpBar.value = currentHP / maxHP;
        hpText.text = currentHP + "/" + maxHP;
    }
}
```

CODE:PLAYERACTIONS

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PlayerActions : MonoBehaviour
{
    public HPManager enemyHP;
    public HPManager playerHP;
    public Button attackSlashButton, attackShootButton, defendButton, runButton;

    private bool isDefending = false;

    void Start()
    {
        attackSlashButton.onClick.AddListener(AttackSlash);
        attackShootButton.onClick.AddListener(AttackShoot);
        defendButton.onClick.AddListener(Defend);
        runButton.onClick.AddListener(Run);
    }

    void AttackSlash()
    {
        if (enemyHP != null)
        {
            enemyHP.TakeDamage(15);
            Debug.Log("Player ใช้ฟัน!");
        }
    }

    void AttackShoot()
    {
        if (enemyHP != null)
        {
            enemyHP.TakeDamage(20);
            Debug.Log("Player ใช้ยิง!");
        }
    }

    void Defend()
    {
        isDefending = true;
        Debug.Log("Player ป้องกัน!");
        Invoke("ResetDefend", 3f);
    }

    void ResetDefend()
    {
        isDefending = false;
    }

    void Run()
    {
        Debug.Log("Player หนีจากการต่อสู้!");
        SceneManager.LoadScene("Dungeon");
    }

    public bool IsDefending()
    {
        return isDefending;
    }
```

CODE:ENEMYHPMANAGER

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class EnemyHPManager : MonoBehaviour
{
    public Slider hpBar;
    public Text hpText;
    private float maxHP = 150f;
    private float currentHP;

    public float MaxHP => maxHP;
    public float CurrentHP => currentHP;

    void Start()
    {
        currentHP = maxHP;
        UpdateHPUI();
    }

    public void TakeDamage(float damage)
    {
        currentHP -= damage;
        currentHP = Mathf.Clamp(currentHP, 0, maxHP);
        UpdateHPUI();

        if (currentHP <= 0)
        {
            Debug.Log("Enemy died.");
            // เรียกฟังก์ชัน OnEnemyDeath() ใน SceneManagerController เมื่อศัตรูตาย
            SceneManager.LoadScene("WinScene");
        }
    }

    void UpdateHPUI()
    {
        hpBar.value = currentHP / maxHP;
        hpText.text = currentHP + "/" + maxHP;
    }
}
```

CODE:ENEMYACTIONS

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class EnemyActions : MonoBehaviour
{
    public HPManager playerHP;
    public HPManager enemyHP;

    private bool isDefending = false;

    void Start()
    {
        InvokeRepeating("EnemyDecision", 2f, 3f);
    }

    void EnemyDecision()
    {
        if (enemyHP.CurrentHP <= enemyHP.MaxHP * 0.2f)
        {
            TryToEscape();
        }
        else
        {
            AttackOrDefend();
        }
    }

    void AttackOrDefend()
    {
```

```
    {
        if (playerHP.CurrentHP < playerHP.MaxHP * 0.2f)
        {
            Debug.Log("ศัตรูเห็นว่าผู้เล่นอ่อนแอและโจมตีต่อ!");
            Attack();
        }
        else if (enemyHP.CurrentHP > enemyHP.MaxHP * 0.3f)
        {
            Attack();
        }
        else
        {
            Defend();
        }
    }

    void Attack()
    {
        int damage = 20;
        playerHP.TakeDamage(damage);
        Debug.Log("ศัตรูโจมตีผู้เล่น! ความเสียหาย: " + damage);
    }

    void Defend()
    {
        isDefending = true;
        Debug.Log("ศัตรูป้องกันตัวเอง!");
        Invoke("ResetDefend", 3f);
    }
```

```
    void TryToEscape()
    {
        float escapeChance = Random.Range(0f, 1f);
        if (escapeChance > 0.5f)
        {
            Debug.Log("ศัตรูหลบหนีสำเร็จ!");
            Escape();
        }
        else
        {
            Debug.Log("ศัตรูพยายามหนี แต่ล้มเหลว! ศัตรูป้องกันตัวเอง");
            Defend();
        }
    }

    void Escape()
    {
        Debug.Log("ศัตรูหนีจากการต่อสู้!");
        SceneManager.LoadScene("Dungeon");
    }

    void ResetDefend()
    {
        isDefending = false;
    }

    public bool IsDefending()
    {
        return isDefending;
    }
```

เอกสารอ้างอิง

1.เกม Vampire survival
2.เกม Soul knight
3.คู่มือการพัฒนาเกม Unity 6

THANK YOU