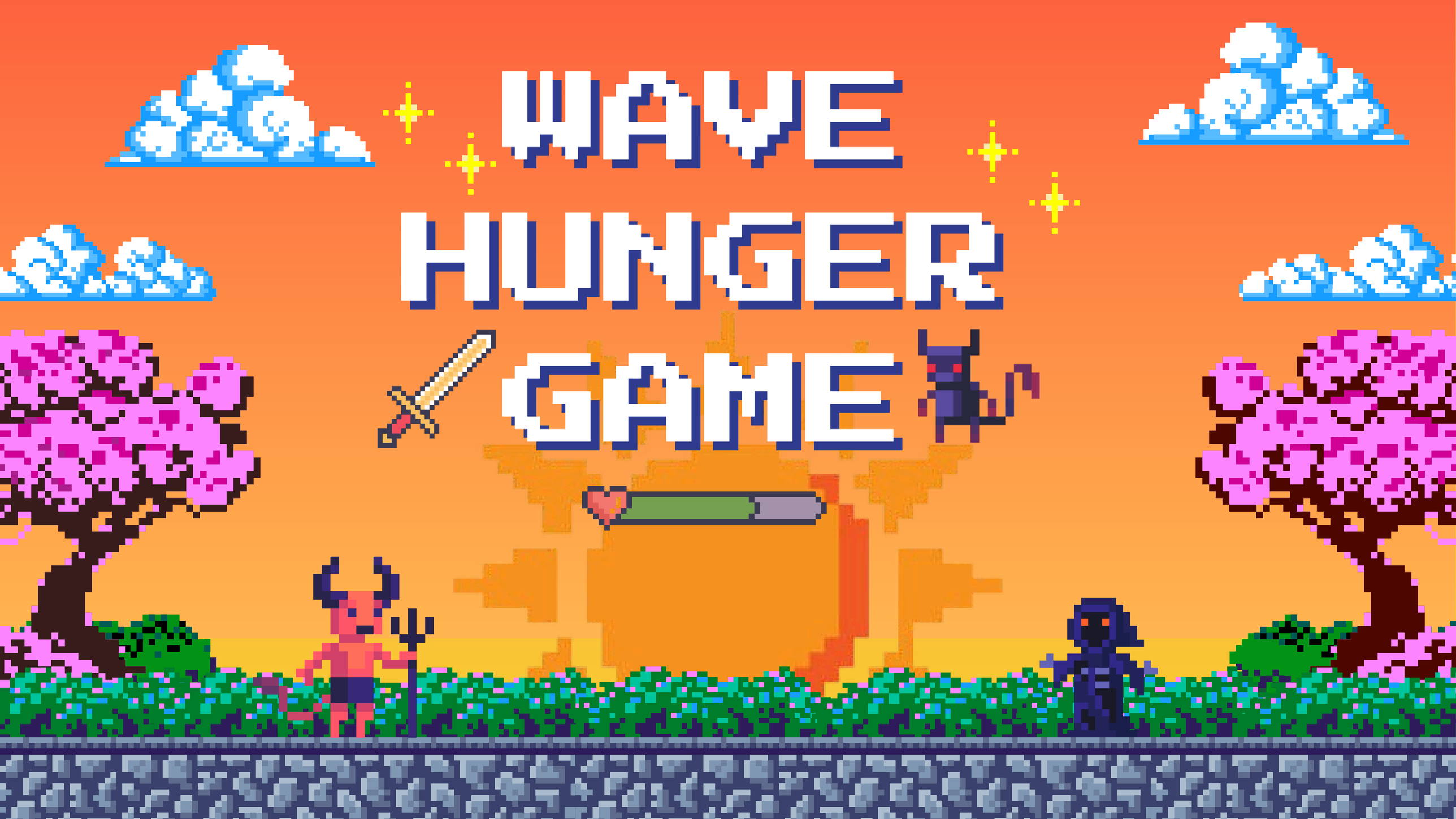

WAVE
HUNGER
GAME

OVERVIEW

แนวคิดเกม: เกม Turn Based RPG แบบข้อความ (Text-Based RPG)

เงื่อนไขการจบเกม:

- **Game Over:** หากพลังชีวิต (HP) ของฮีโร่ของคุณลดลงเหลือ 0 เกมจะจบลง
- **Win Game:** เอาชนะบอสได้ครบทั้ง 5 ตัว

รูปแบบการเล่น:Turn-based Combat: ผลัดกันโจมตี

- การเลือกคลาส: ⚔️ Warrior, 🏹 Archer, 🔮 Mage พร้อมสกิลเฉพาะตัว
- ระบบไอเท็ม: สุ่มดรอป, เลือก, สวมใส่เพื่อเพิ่มความแข็งแกร่ง
- ระบบโพชั่น (Potion):ผู้เล่นสามารถใช้โพชั่นในเทิร์นเพื่อฟื้นฟูพลังชีวิตหรือมานาได้
- Status Effects: พิษ, เลือดไหล, สตัน เพิ่มความซับซ้อนในการต่อสู้
- พัฒนาตัวละคร: อัปเกรดค่า HP,MP,DEF,ATK

```
You defeated Goblin King!

=== STAGE COMPLETE! CHOOSE UPGRADE ===
1. Heal (+30 HP)
2. Restore Mana (+20 MP)
3. Increase Attack (+5 ATK)
4. Increase Defense (+3 DEF)
Choose (1-4): _
```

Item Drop

```
=== ITEM SELECTION ===
Choose 1 item from the following 3 options:
1. Poison Dagger - Poisons enemies on hit [ATK +8]
2. Ice Shield - Freezes attackers occasionally [HP +20] [DEF +10]
3. Fire Sword - Burns enemies with fire damage [ATK +15]
Choose item (1-3):
```

UI Game

```
========================================
| STAGE 1 BATTLE                       |
========================================

=== YOUR STATUS ===
Wave - HP: 120/120, MP: 50/50, ATK: 20, DEF: 2

=== ENEMY STATUS ===
Goblin King - HP: 90/90, MP: 60/60, ATK: 15, DEF: 0


=== BATTLE LOG ===
No actions yet.

=== YOUR TURN ===
1. Attack
2. Use Skills
3. Inventory
4. Pass
Choose action:
```

REQUIREMENTS ดูแบบละเอียดได้ที่ Readme

Functional Requirements

- การสร้างตัวละคร: ตั้งชื่อ, เลือก 1 ใน 3 คลาส (Warrior, Archer, Mage)
- ระบบต่อสู้แบบเทิร์นเบส: การผลัดกันโจมตี, การเลือกแอ็กชัน (Attack, Skill, Potion, Pass)
- สกิลเฉพาะคลาส: 3 สกิลพร้อม Mana Cost สำหรับแต่ละคลาส
- ระบบคลังเก็บของ: ใช้ Health/Mana Potion
- ระบบไอเท็ม: ไอเท็มดรอป, เลือก 1 จาก 3 ไอเท็มพิเศษหลังบอส
- สถานะผิดปกติ: เช่น พิษ (Poison), สตัน (Stun), เพิ่ม/ลดพลังโจมตี
- ความคืบหน้าเกม: 5 ด่านบอส, อัปเกรดตัวละครหลังจบด่าน
- เงื่อนไขการจบเกม: Game Over (HP 0) / Victory (ชนะบอสครบ 5 ตัว)
- อินเทอร์เฟซผู้ใช้ (UI): แสดงผลผ่าน Console (Text-based)

Non-Functional Requirements

- ประสิทธิภาพ (Performance): การทำงานที่ราบรื่น,Animation ที่รวดเร็ว
- การใช้งาน (Usability): อินเทอร์เฟซที่ใช้งานง่ายและเข้าใจง่าย มีคำแนะนำที่ชัดเจน
- ความถูกต้อง (Reliability): จัดการ Input ที่ไม่ถูกต้องได้อย่างเหมาะสม
- การบำรุงรักษา (Maintainability): โครงสร้างโค้ดที่เป็นระเบียบหลัก OOP
- ความสะดวก (Portability): เกมควรสามารถคอมไพล์และทำงานได้บน ระบบปฏิบัติการทั่วไป ที่รองรับคอมไพเลอร์ C++

OOP CODE EXAMPLE


```cpp
// ========== BaseClassUnit ==========
class Unit
{
protected:
    string name;
    int maxHealth;
    int health;
    int maxMana;
    int mana;
    int baseAttack;
    int currentAttack;
    int defense;
    map<StatusEffect, int> statusEffects;
    vector<Item *> equipment;
    vector<Potion> potions;
    vector<string> battleLog;

public:
    Unit(string n, int hp, int mp, int atk, int def = 0)
        : name(n), maxHealth(hp), health(hp), maxMana(mp), mana(mp),
          baseAttack(atk), currentAttack(atk), defense(def) {}

    virtual ~Unit()
    {
        for (Item *item : equipment)
            delete item;
    }

    // Getters
    string getName() const { return name; }
    int getHealth() const { return health; }
    int getMaxHealth() const { return maxHealth; }
    int getMana() const { return mana; }
    int getMaxMana() const { return maxMana; }
    int getAttack() const { return currentAttack; }
    int getDefense() const { return defense; }
    bool isAlive() const { return health > 0; }
    const vector<Potion> &getPotions() const { return potions; }
    const vector<string> &getBattleLog() const { return battleLog; }
    void clearBattleLog() { battleLog.clear(); }
```

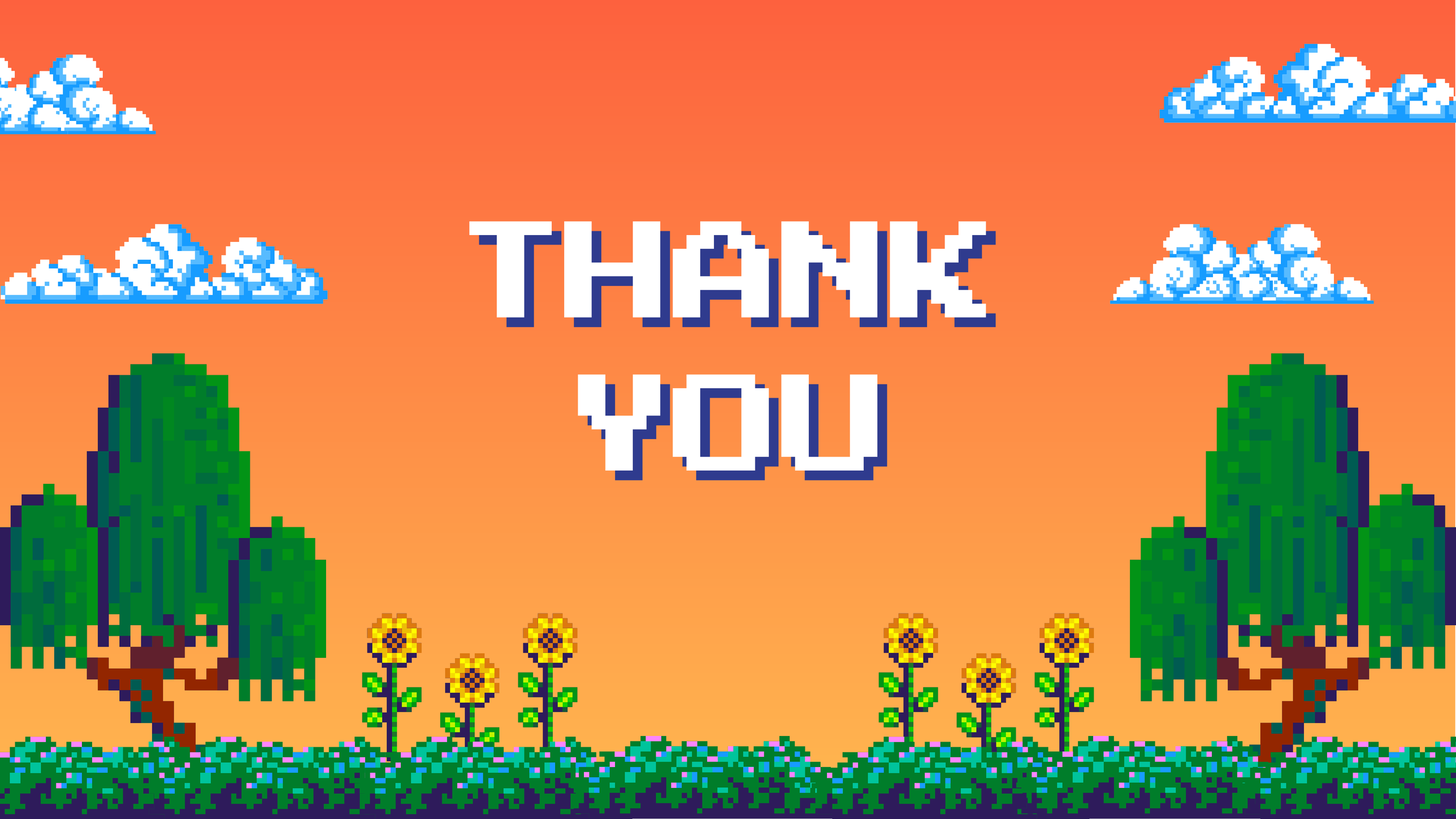
```cpp
// ========== DerivedClassUnit ===========
// Class player
class Warrior : public Unit
{
public:
    Warrior(string n) : Unit(n, 120, 50, 20, 2) {}

    string getSkill1Name() const override { return "Power Strike"; }
    string getSkill2Name() const override { return "Demoralizing Shout"; }
    string getSkill3Name() const override { return "Battle Rage"; }
    int getSkill1Cost() const override { return 15; }
    int getSkill2Cost() const override { return 10; }
    int getSkill3Cost() const override { return 20; }

    bool useSkill1(Unit &target) override
    {
        if (mana < getSkill1Cost())
        {
            battleLog.push_back("Not enough MP for Power Strike!");
            return false;
        }

        mana -= getSkill1Cost();
        int dmg = currentAttack + 25;
        int targetPrevHealth = target.getHealth();
        battleLog.push_back(name + " uses Power Strike on " + target.getName() + " for " + to_string(dmg) + " damage!");
        target.takeDamage(dmg);
        battleLog.push_back(target.getName() + "'s HP: " + to_string(targetPrevHealth) + " -> " +
                            to_string(target.getHealth()) + "/" + to_string(target.getMaxHealth()));
        return true;
    }
```

OOP CODE EXAMPLE


```cpp
// ========= BaseClassItem ========
class Item
{
protected:
    string name;
    string description;
    ItemType type;
    int attackBonus;
    int healthBonus;
    int defenseBonus;
    int manaBonus;

public:
    Item(string n, string desc, ItemType t, int atk = 0, int hp = 0, int def = 0, int mp = 0)
        : name(n), description(desc), type(t), attackBonus(atk), healthBonus(hp), defenseBonus(def), manaBonus(mp) {}

    virtual ~Item() {}

    string getName() const { return name; }
    string getDescription() const { return description; }
    ItemType getType() const { return type; }
    int getAttackBonus() const { return attackBonus; }
    int getHealthBonus() const { return healthBonus; }
    int getDefenseBonus() const { return defenseBonus; }
    int getManaBonus() const { return manaBonus; }

    virtual void applyEffect(class Unit &unit) {}
    virtual void displayInfo() const
    {
        cout << "\033[1;33m" << name << "\033[0m - " << description;
        if (attackBonus > 0)
            cout << " [ATK +" << attackBonus << "]";
        if (healthBonus > 0)
            cout << " [HP +" << healthBonus << "]";
        if (defenseBonus > 0)
            cout << " [DEF +" << defenseBonus << "]";
        if (manaBonus > 0)
            cout << " [MP +" << manaBonus << "]";
        cout << "\n";
    }
};
```

```cpp
class Potion
{
private:
    string name;
    int amount;
    bool isHealthPotion;

public:
    Potion(string n, int a, bool health) : name(n), amount(a), isHealthPotion(health) {}

    string getName() const { return name; }
    int getAmount() const { return amount; }
    bool isHealth() const { return isHealthPotion; }

    void displayInfo() const
    {
        cout << "\033[1;34m" << name << "\033[0m - Restores " << amount
             << (isHealthPotion ? " HP" : " MP") << "\n";
    }
};
```

```cpp
// ========= DerivedClassItem ==========
// SpecificItemClasses
class FireSword : public Item
{
public:
    FireSword() : Item("Fire Sword", "Burns enemies with fire damage", ItemType::WEAPON, 15) {}
    void applyEffect(class Unit &unit) override;
};

class IceShield : public Item
{
public:
    IceShield() : Item("Ice Shield", "Freezes attackers occasionally", ItemType::ARMOR, 0, 20, 10) {}
    void applyEffect(class Unit &unit) override;
};
```


THANK
YOU