GMAP-231-001 Final Project Design Plan

Jacob Zolda

2D Platformer

Fantasy themed platformer following the player's journey through varied levels. Mix of fast paced action and strategic platforming areas.

Setting & Theme

Various levels containing natural landscapes, villages, and fortress/castle structures. Follows the player character's journey through these environments while encountering enemies and obstacles.

Game Feel

Throughout the levels there will be more action-oriented combat sections that feel faster than the slower and strategic platforming sections for dynamic gameplay. Each level will have more types of enemies.

Progression

The levels will contain temporary power-ups and new permanent gameplay abilities can also be found. The game and story will progress as levels are completed.

Combat & Movement Abilities

Initially, the player will move, jump, and flip by a standard amount. The player uses melee combat with only fists at the start.

Player can find a **sword** as a primary weapon and permanent ability in Level 1

Player can find a permanent ability of a **jump boost** in Level 2

Player can find a **greatsword** as a primary weapon and permanent ability in Level 3

Temporary power-ups: speed boost, jump boost, attack buff, defense buff, HP increase

Include Mechanics & Skills

1. Player controlled 2D physics based movement
2. Projectile shooting for enemies and possibly player as secondary weapon
3. Platforming and jumping for navigating environment and obstacles
4. Temporary power-ups and permanent abilities
5. Simple AI behavior for enemy combat
6. Singleton based Game Manager
7. Level loading

Movement Pseudocode

```
// player movement using physics
class Player {
    Rigidbody2D rb
    float moveSpeed
    float jumpForce
    bool isGrounded

    void Start() {
        rb = GetComponent<Rigidbody2D>()
    }

    void Update() {
        Input()
        CheckGroundedStatus()
    }

    void Input() {
        // check for horizontal movement and jump button
    }

    void CheckGroundedStatus() {
        // check if the player is on the ground
    }
}
```

Enemy AI & Projectile Pseudocode

```
// enemy AI & shooting
class Enemy {
    Vector2 position
    Vector2 velocity
    float moveSpeed
    GameObject projectile

    void Update() {
        Patrol()
        AttackPlayer()
        ShootProjectile()
    }

    void Patrol() {
        // patrol logic
    }

    void AttackPlayer() {
        if (PlayerInRange()) {
            // attack player
        }
    }

    void PlayerInRange() {
        // check if player is within attack range
    }

    void ShootProjectile() {
        GameObject projectile = Instantiate(projectile)
        projectile.velocity = CalculateProjectileVelocity()
    }

    void CalculateProjectileVelocity() {
        // calculate the velocity vector for the projectile to reach the player
    }
}
```

Power-Ups Pseudocode

```
// power-up
class PowerUp {
    string type

    void OnCollision(Player player) {
        if (type == "speed") {
            // increase player speed
        }
        else if (type == "jump") {
            // increase player jump
        }
        else if (type == "attack") {
            // increase player attack power
        }
        else if (type == "defense") {
            // increase player defense power
        }
        else if (type == "HP") {
            // increase player health points
        }

    }
}
```

Game Manager Pseudocode

```
// singleton game manager
class GameManager {
    static GameManager instance
    int playerLives
    string currentLevel

    void Awake() {
        if (instance == null) {
            instance = this
            DontDestroyOnLoad(gameObject)
        } else {
            Destroy(gameObject)
        }
    }

    void StartGame() {
        // init game state
        playerLives = 3
        LoadLevel("Level1")
    }

    void LoadLevel(string levelName) {
        currentLevel = levelName
        SceneManager.LoadScene(levelName)
    }

    void YouDied() {
        playerLives -= 1
        if (playerLives <= 0) {
            EndGame()
        } else {
            // respawn player
            LoadLevel(currentLevel)
        }
    }

    void EndGame() {
        // show game over screen
        SceneManager.LoadScene("GameOver")
    }
}
```