Project Title: 2D Platformer Game in Unity

Introduction

In today's gaming industry, classic 2D platformer games remain popular due to their simple controls, challenging levels, and fast-paced reward loops. This project proposes the development of a minimal yet engaging 2D platformer game in Unity. The game will include movement, jumping, wall jump & slide, shooting, enemies & traps, a basic State System, and multiple levels, all combined to produce a playable MVP (Minimum Viable Product) within 5 weeks.

Objectives

• Implement smooth movement and jumping mechanics for the player.

• Add wall jump and slide features for vertical level design.

• Implement shooting mechanics (projectile/raycast) and a basic combat system.

• Add enemies and traps to provide challenging gameplay.

• Build a basic State System for both player and enemies

(Idle/Run/Jump/Fall/Wall/Attack/Hit/Death).

• Design multiple levels with progression checkpoints and completion goals.

• Deliver a stable playable build (APK/PC) and documentation within 5 weeks.

Functional Requirements

1. Gameplay Mechanics

• Player movement: left/right, run, jump, variable jump height (based on button hold),

coyote time, and jump buffer.

• Wall slide & wall jump: sliding with descent speed clamp; wall jump with impulse vector.

• Shooting: input-based shooting with projectile prefabs, fire rate, ammo (optional).

• Health & Damage: health system for player and enemies with i-frames.

• Enemies & Traps: patrol enemies, chasing enemies, turrets, spikes, saws, falling platforms.

• Collectibles: optional coins/gems and health pickups.

2. Player Controller

• Physics: Unity 2D Rigidbody/Colliders or custom kinematic controller.

• State System: Idle, Run, Jump, Fall, WallSlide, WallJump, Shoot, Hit, Death with transitions.

• Animation: Animator and state-based animation blending.

3. Combat & Shooting

• Projectile prefab (speed, lifetime, damage).

• Enemy hitbox/hurtbox with collision-based damage.

• Knockback and death logic.

4. Enemies & Traps

• Patrol AI (waypoint/edge detect).

• Chase AI (distance threshold).

• Traps: spikes, moving/falling platforms.

• Optional: boss enemy.

5. Level Design & Progression

• At least 3–5 levels of increasing difficulty.

• Checkpoints and Level Complete flags/portals.

• Difficulty curve: tutorial → intermediate → advanced mechanics.

6. Audio

• SFX: jump, shoot, hit, collect, death.

• Background music per level.

7. Save/Progress

• Basic save system using PlayerPrefs/JSON: unlocked levels, audio preferences, high score.

8. Performance & Build

• Target 60 FPS.

• Support for Android (APK) and Windows/Mac builds.

9. Accessibility (Optional)

• Rebindable keys, color-blind-friendly sprites, vibration support on Android.

Sprint Plan (5 Weeks, Agile/Scrum)

Sprint 1 (Week 1)

Backlog Items:

• Backlog 1: Finalize requirements and wireframes (main menu, HUD, level flow), Initialize Unity project and folder structure, Setup base scene with placeholder player and ground.

• Backlog 2: Left-right movement, changing player speed ,Implement jumping , Upgrading Graphics, Editing our Player.

• Backlog 3: Flipping Player Left- Right, Basics of Animatio , Idle Animation ,Run Animation , Jumping Animation.

• Backlog 4: Cleaning up the code, Raycast & boxcast ,Implementing boxcasting , Layers & layerMasks , Detecting wall collision , Sticking to walls , Climbing walls, Jumping Away from the wall

Sprint 2 (Week 2)

Backlog Items:

• Backlog 1: Basic Attack, Adding Delays, Attack Animation, Creating a Fireball.

• Backlog 2: Object Pooling, Pooling Fireballs , Fixing Bugs, Cleaning Up , Importing Sprites.

• Backlog 3: Ground Tiles, Wall Tiles, Ceiling, Additional Details, Background.

• Backlog 4: Room Camera, Minor Fixes , Camera Follow Player, Camera Lookahead.

Sprint 3 (Week 3)

Backlog Items:

• Backlog 1: Prefab Fireballs, Sorting Layers, Health Component, Health UI, Making the healthbar , Hurt & Die Animations work, Creating a Saw Trap, Health Pickups

• Backlog 2: Fixes, Creating the Layers, iFrames, Tweaking, Final Result, Patreon Shoutout

• Backlog 3: Spikes, Firetrap, Arrowtrap , Inheritance , Enemy Arrows , Spikehead , Reseting The Rooms, Patreon Shoutout

Sprint 4 (Week 4)

Backlog Items:

• Backlog 1: Design 3–5 levels with tilemaps/prefabs.

• Backlog 2: Implement HUD, Pause Menu, and Settings.

• Backlog 3: Add checkpoints and level complete functionality.

• Backlog 4: Implement save and progress system.

• Backlog 5: Optimize performance and add mobile controls.

Deliverable: Multiple levels with UI and progression system.

Sprint 5 (Week 5)

Backlog Items:

• Backlog 1: Perform functional testing on PC and Android.

• Backlog 2: Fix bugs and tune gameplay balance.

• Backlog 3: Add polish (particles, camera follow, subtle screenshake).

• Backlog 4: Prepare documentation with diagrams, screenshots, and known issues.

• Backlog 5: Generate final builds for Windows/Mac and Android.

Deliverable: Stable builds and documentation ready for submission.

Features

1. Movement & Jumping – smooth controls with advanced jump mechanics.
2. Wall Jump & Slide – vertical traversal.
3. Shooting – projectile-based combat.
4. Enemies & Traps – patrol/chase enemies and environmental hazards.
5. Basic State System – player and enemy states.
6. Multiple Levels – at least 3 playable levels.
7. UI & HUD – health, pause, menus.

Tools and Technology

• Engine/IDE: Unity (2021 LTS or later).

• Language: C#.

• Version Control: Git/GitHub.

• Art & Audio: Placeholder assets from Kenney/Itch.io.

• Build Targets: Windows/Mac + Android.

Target Users

• Casual players who enjoy platformer games.

• Mobile and PC users looking for short-session gameplay.

• Students learning game development.

Software Development Model

Agile Scrum is suitable because:

• Supports 5-week sprint-based delivery.

• Allows testing and feedback at each sprint.

• Flexible scope adjustment to ensure MVP delivery.

Risks & Mitigation

• Scope creep: Stick to sprint plan; keep optional features separate.

• Performance issues on mobile: Use sprite atlas, pooling, and optimized physics.

• Delays in art/audio: Start with placeholders, replace later.

Testing Plan

• Unit-like testing for collision layers, damage, and respawn.

• Playtesting for controls, difficulty tuning.

• Device testing for FPS, input latency, and scaling.

Expected Deliverables

• Playable Build: Windows/Mac executable and Android APK.

• Project Files: Well-structured Unity project.

• Documentation: Features list, architecture, control, screenshots.

Conclusion

This project aims to deliver a focused MVP of a classic 2D platformer game—smooth movement, wall mechanics, shooting, enemies, traps, a basic state system, and multiple levels. By following Agile Scrum, the project will achieve a stable, playable build with documentation within 5 weeks, ready for submission.

…………………..　　　　　　　　　　　　　　　　…………………….

Signature of student　　　　　　　　　　　　　　　Signature of teachers