MoonBlade

This is a 2D-platformer game where the player also fights some enemies like scorpions and knight.

Player Controls:

- Move Left – A
- Move Right – D
- Jump – W
- Attack – SpaceBar

To win the game you need to avoid falling in water and kill the main enemy i.e. knight available on last platform.

Coins are not just collectibles - they're a part of the story. They represent your courage and journey. The more you explore, the more of your spirit you leave behind.

About Game

Our background is using one of sample parallax in tile mode. Platforms are mostly static. Also there are some moving platforms which can also carry players along with them. Below is water which kills everything falling in it whether it is the player or the enemies.

- Player:

  Health – 100
  Attack Damage – 15
  Has its unique idle, run, attack and die animation

Talking about enemies we have:

1. Brown Scorpion

Health – 15
Attack Damage – 5
Dies with only one sword attack of player.
Has its corresponding idle, run, attack and die animation.

2. Orange Scorpion

Health – 30
Attack Damage – 10
Dies with two sword attacks of player.
It also has its corresponding idle, run, attack and die animation.

3. Main Enemy - Knight

Health – 50
Attack Damage – 15
Has its own idle, run, attack and die animation.

Also, all the enemies have attack rate of 0.5 i.e. they can attack twice a second (I tried making a simple enemy bot).

Along with these, this game also has coins on different platforms and UI that keeps track of it. The coins are also animated and disappears when player touches it increasing the coins collected by one. Also, the coins reset to zero whenever you restart the game.

I have also added HealthBar as a new feature for which I used a sprite of Brackeys (https://github.com/Brackeys/Health-Bar/blob/master/Health%20Bar/Assets/Sprites/Bar.png)

The HealthBar is green for player and red for enemies.

Also, I have added a background music and some sound effects like for sword attack, scorpion attack, coins collected and many more.

Talking about my scripts , the most important part of the game and the part on which I focused the most:

- **ParallaxEffect.cs**
  This script creates a parallax scrolling effect on background layers making it move relative to the camera movement to simulate depth.
  The background as a result moves in relation to the camera movement at a different speed depending on its distance in the z-axis, creating a depth illusion where background layers move slower than foreground layers.
  - **cam**: The main camera.
  - **followTarget**: Usually the player — the camera's target.
  - **startingPosition**: Initial position of the background layer.
  - **camMoveSinceStart**: How much the camera has moved since the scene started.
  - **distanceFromPlayer**: Z-distance from the background to the player.
  - **clippingPlane**: Used to compute how far the camera can see.
  - **parallaxFactor**: Scales background movement based on its depth.
  - **Update()**: Moves the background slightly based on camera movement to simulate depth.

- **PlayerController.cs**
  This script is the main controller for the player character. It handles player movement, jumps, attacks as well as work with other components like health tracking, SFX and input management. Some of its major task are:
  - Handles input using the new Unity Input System
  - Moves the player using physics (Rigidbody2D) with different speeds on ground and air.
  - Manages facing direction by flipping the sprite using local scale.
  - Plays attack animation and sword sound when grounded.
  - Also responsible for other animation like of the idle s, moving and death state.
  - Detects jump only if the player is grounded.
  - Interacts with Coin objects to update score and play sound.
  - Checks player's health and loads the next scene if the player dies.

  Depends on:
  - **TouchDIrection.cs** for ground detection
  - **HealthBook.cs** for health tracking
  - **SFXManager.cs** for sound effects

- **TouchDIrection.cs**
  This is to detect whether the player is currently standing on the ground. It does this by casting a tiny invisible ray downward. We need this script because the player shouldn't be able to jump or attack while floating in mid-air. This script helps enforce that rule and also updates animations so the game can show walking or falling visuals properly.
    - **IsGrounded**: Boolean updated based on downward raycast hits.
    - Uses **CapsuleCollider2D.Cast()** to check ground beneath the player.
    - Sets an animation parameter (**isGrounded**) to trigger grounded animation state.

- **MovingScript.cs**
  This script controls a simple moving platform which moves left and right between two points. It automatically switches direction when it reaches either end.
    - Uses a **Rigidbody2D** to move the platform left and right between two points.
    - **moveDistance**: How far it moves from the initial position.
    - **speed**: Movement speed.

- **KnightScript.cs**
  This is the most interesting script as I made this to make a basic enemy bot. The enemy patrols the area, detects the player, walks toward them, and attacks when close enough. It plays sound during attacks and dies when its health reaches zero. This gives AI behaviour to enemy.
    - Detects player using raycasting.
    - Walks toward the player if within range.
    - Attacks when close enough with a cooldown.
    - Plays attack sound using **SFXManager**.
    - If health of main enemy i.e. knight becomes zero, triggers win scene (**WIn**).

- **HealthBook.cs**
  Handles the health system for characters that can take damage — such as enemies. Connects with **HealthBar** to update visuals and an animator to show death state.
    - Contains health and max health.
    - Triggers death animation when health reaches 0.
    - Provides Hit(damage) method to reduce health

- **HealthBar.cs**
  Updates the health bar UI. When a character takes damage, the bar reflects the change visually
    - SetMaxHealth(int health) – Sets full bar on start.
    - SetHealth(int health) – Adjusts fill as damage is taken.

- **Attack.cs**
  Attached to the character's attack point. Detects collision with an enemy and reduces their health.
  When sword collides with something (**OnTriggerEnter2D**), it checks if it's a damageable object (**HealthBook**). If yes, it deals fixed damage.

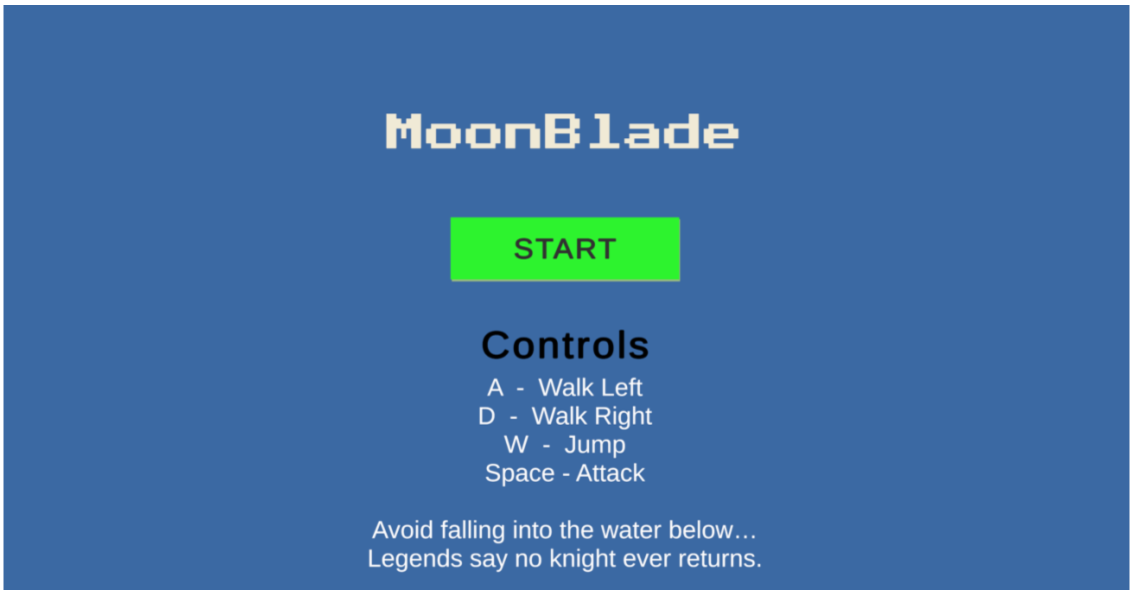
- **CoinManager.cs**
  Keeps track of total coins collected during the game and updates the on-screen UI.

- **SFXManager.cs**
  A central script to handle short sound effects like sword swings or coin pickups.
  Usage: Call **PlaySFX(AudioClip)** from other scripts to play sound effects without repeating audio setup logic.

- **KillTrigger.cs**
  Instantly kills anything that falls in water

Also we have used different scenes with order:

i) Start


MoonBlade

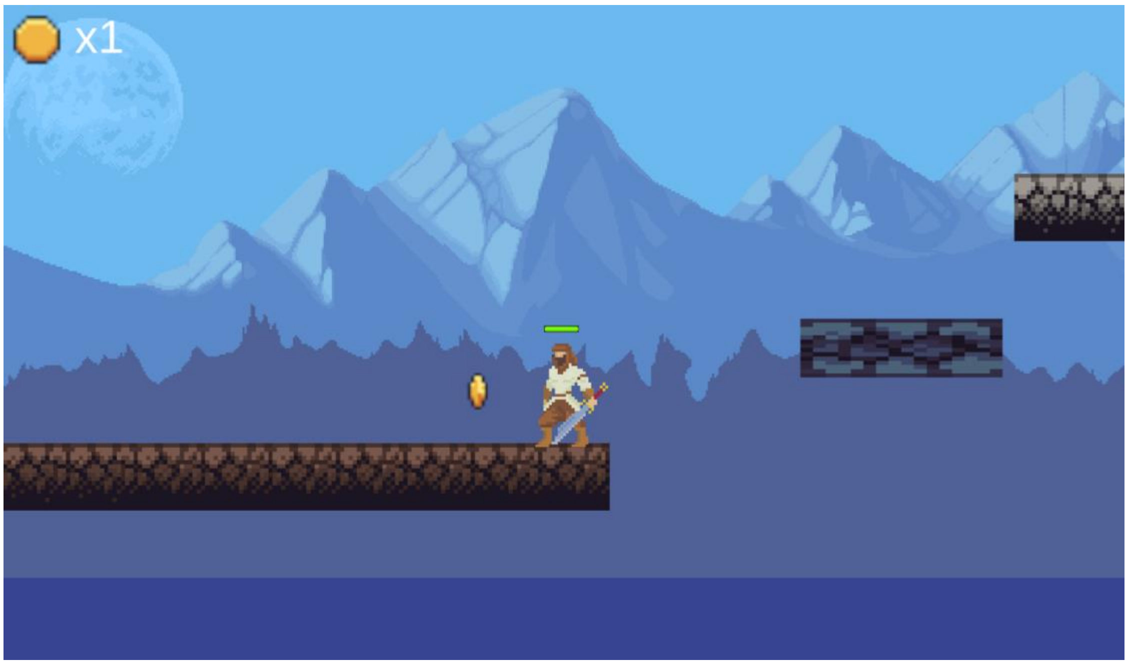
START

Controls

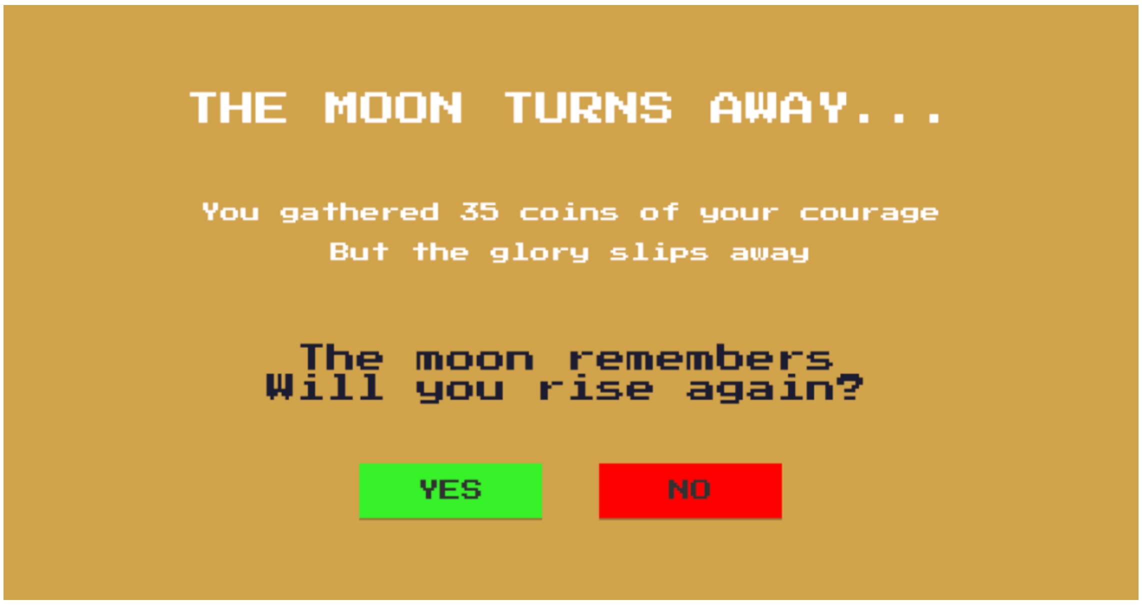
A - Walk Left
D - Walk Right
W - Jump
Space - Attack

Avoid falling into the water below…
Legends say no knight ever returns.

ii) Game



iii) Try


THE MOON TURNS AWAY...

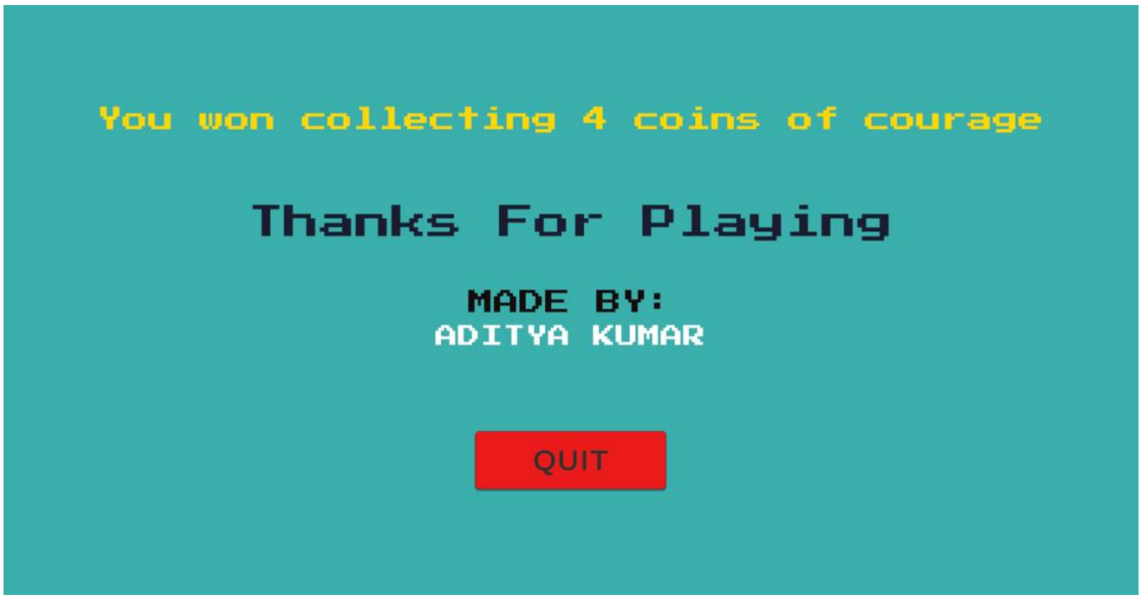
You gathered 35 coins of your courage
But the glory slips away

The moon remembers
Will you rise again?

YES

NO

iv) Win


You won collecting 4 coins of courage

Thanks For Playing

MADE BY:
ADITYA KUMAR

Some scripts like **Menu.cs**, **Exit.cs**, **Try.cs**, **WIn.cs**, **ScoreManager.cs**, **ScoreWin.cs** are for scene management handling buttons and text to be displayed on them.

This game has also been worked on animator, tilemaps, layers, tags, sprite sheets and have also tweaked with project settings.

Now, go enjoy the game.

This is Mac build of my game: https://drive.google.com/file/d/1gyGhpmca4CTW0TM0x8vL3-TiBZ69d5qt/view?usp=sharing