AtrillionGamesLtd_PlayerMove Documentation

Overview

This script implements a first-person character controller in Unity, featuring dynamic movement based on input from both the user and in-game physics. It provides functionality for player movement (walking, sprinting, crouching, sliding), camera control (looking around), gravity direction handling, wall running, jumping, crouching, and interaction with slopes and stairs.

The system uses Unity's Rigidbody physics and Collider components to simulate realistic movement, and it integrates with the new Unity Input System for user input management. The player can interact with moving platforms, walls, and various slopes, making the movement system highly adaptable.

Fields

Player Movement and State

- `private GameObject wherePlayerLastStood;`
  Used for tracking moving parts and moving platforms.
- `private PlayerControls inputActions;`
  Input actions for the player (e.g., movement, camera look).

Debug Fields

- `private Vector3 prevPlayerPos;`
  The previous position of the player.
- `private Vector3 playerVelocity;`
  Current velocity of the player.
- `private Vector3 absolutePlayerVelocity;`
  The velocity of the player without considering external forces like gravity.
- `private Vector3 playerHeadVelocity;`
  The velocity of the player’s head (camera) for smoothed head movement calculation.
- `private Vector2 movementInput;`
  Raw movement input values (horizontal and vertical).
- `private Vector2 lookMovementInput;`
  Raw camera look input values (mouse X and Y movement).

Player Conditions

- `private bool isGrounded;`
  Whether the player is grounded.

- `private float isOnStairs;`
  A countdown timer indicating if the player is on stairs (or was on stairs not long ago).
- `private bool isDirectlyGrounded;`
  Whether the player is directly grounded, used to detect falling off edges at any orientation. Returns false if the player is walking off an edge but returns true if they are stepping down stairs.
- `private bool isSprinting;`
  Whether the player is sprinting.
- `private bool isCrouching;`
  Whether the player is crouching.
- `private bool isJumping;`
  Whether the player is jumping.
- `private bool isSliding;`
  Whether the player is sliding.
- `private bool isWallRunning;`
  Whether the player is wall running.
- `private bool isJumpDisabled;`
  Whether jumping is currently disabled (due to having just jumped to prevent bouncing off of surfaces by holding the jump key).

Player Details

- `private float playerRadius;`
  Radius of the player for collision detection.
- `private float playerStandingHeight;`
  The standing height of the player.
- `private float playerCrouchHeight;`
  The crouching height of the player.

Player Properties

- `private float sensitivity;`
  Camera sensitivity for looking around input.
- `private float maxWalkableSlope;`
  The maximum slope angle the player can walk on.
- `private float airtimeSpeedCap;`
  Speed cap for the player when in the air.
- `private float playerSpeed;`
  The base movement speed of the player.
- `private float sprintSpeedMultiplier;`
  Multiplier applied to speed when sprinting.
- `private float crouchSpeedMultiplier;`
  Multiplier applied to speed when crouching.
- `private float jumpHeight;`
  The jump height of the player.

- `private float stepHeight;`
  The maximum height of a step the player can climb.
- `private float rotateTowardsGravityRate;`
  Rate at which the player rotates to match the gravity direction.
- `private float gravityValue;`
  The gravitational force applied to the player.
- `private Vector3 gravityDirection;`
  The direction of gravity in the game world (Normalised).

Player Movement Properties

- `private Vector3 headOffset;`
  Offset used to position the player's head correctly.
- `private float playerAirControl;`
  The amount of control the player has when in the air.
- `private float playerAirResistance;`
  Resistance applied to the player's movement while in the air.
- `private float playerSlideControl;`
  The amount of control the player has while sliding.
- `private float playerWallRunControl;`
  The amount of control the player has while wall running.
- `private float playerWallRunFallOff;`
  How quickly the player loses grip on the wall during wall running.
- `private float playerWallRunJumpPower;`
  Jump power when wall running (how hard to push off the wall).

Private Fields for Collision and Physics

- `private float pointInWallRun;`
  The progression of the wall running state.
- `private float groundFriction;`
  The friction of the ground the player is standing on (updates to the dynamic friction of the surface that is being stood on).
- `private Vector3 slopeNormal;`
  The normal vector of the slope the player is standing on.
- `private Vector3 slopeDirection;`
  The direction down the slope the player is walking on.
- `private Vector3 slideDirection;`
  The direction the player slides when sliding down a slope.
- `private Transform lastSurface;`
  The last surface the player was standing on.
- `private float localAirSpeedCap;`
  The local cap for air movement speed. (Player can't accelerate after they left the ground without external forces pushing them)
- `private SphereCollider playerHeadCollider;`
  The collider for the player's head.

- `private CapsuleCollider playerBodyCollider;`
  The collider for the player’s body.
- `private Rigidbody playerBodyRigidBody;`
  Rigidbody for the player’s body.

Methods

Core Player Movement Methods

- `void Awake()`
  Initializes input actions for the new unity input system.
- `void OnEnable()`
  Enables the input actions when the object is enabled.
- `void OnDisable()`
  Disables the input actions when the object is disabled.
- `void setHeadPosition(float headHeight)`
  Adjusts the position of the player's head based on the current height (standing or crouching) and moves the head collider in accordance with the height.
- `void setPlayerGravityDirection(Vector3 _gravityDirection, float rateOfChange = 1f)`
  Changes the gravity direction over time, creating smooth transitions between gravity directions (if no rate is passed it will default to snapping to the gravity direction that is passed in). rateOfChange of “0” will rotate gravity by 0% towards the passed in direction. While a rate of 1f will move it by 100%.
- `void setPlayerFOV(float _FOV)`
  Sets the field of view (FOV) of the player's camera.
- `void setCameraSensitivity(float _Sensitivity)`
  Sets the camera sensitivity for looking around input.
- `void AddScriptsIfMissing()`
  Ensures that the necessary components (colliders, rigidbodies, etc.) are added and initialized.
- `void RotateTransformToDirection(Transform _transform, Vector3 targetDirection, float rotationSpeed = 10f)`
  Rotates the player's transform towards the target direction (typically used for aligning with gravity), rotationSpeed adds smoothing to the movement to avoid the player camera snapping to a new orientation.
- `void playerLookAround()`
  Handles the camera's rotation based on mouse input for look movement as well as movement smoothing to update the camera position every frame rather than every physics iteration to solve camera jitter.
- `void standingOnMovingSurface(Transform Surface, Vector3 position)`
  Handles the case where the player is standing on a moving platform.

- `void whatIsPlayerStoodOn(ref Vector3 slopeDirection, ref Vector3 slideDirection, ref float groundFriction)`
  Determines what surface the player is standing on and updates movement accordingly.
- `Vector3 getPlayerMoveDirection(Vector3 slopeDirection)`
  Computes the player's movement direction, considering the surface slope.
- `void handleCrouching(ref float playerMovementSpeed)`
  Manages the crouching behavior, including initiating sliding if the player moves too fast while crouching.
- `void handleSprinting(ref float playerMovementSpeed)`
  Increases the player's movement speed if sprinting.
- `void handleMovement(float playerMovementSpeed, Vector3 move)`
  Applies movement to the player's velocity, considering different states (grounded, in air, sliding, wall running).
- `void handleJumping()`
  Handles jumping, either on the ground or while wall running.
- `void handleSliding(Vector3 slideDirection, float groundFriction)`
  Manages the player's sliding behavior down slopes.
- `void handleStairs(Vector3 movementDirection)`
  Handles interactions with stairs, allowing the player to step up.

Input Handling

- `void playerInputs()`
  Handles traditional input (for the legacy Unity input system).
- `void GetPlayerInputs()`
  Handles input through the new Unity Input System. (This is used by Default)

Physics Collisions

- `private void OnCollisionStay(Collision collision)`
  Handles ongoing collisions, including detecting when the player is grounded, wall running, or interacting with slopes.
- `void OnCollisionExit(Collision collisionInfo)`
  Resets the player's grounded and wall running state when exiting a collision.

Core Updates

- `void Update()`
  Updates player input and camera look around each frame.
- `void FixedUpdate()`
  Handles the actual player movement in fixed physics steps, including movement logic, jumping, and sliding.

Usage Instructions

If you want to use the new input system make sure you have it installed (via the unity package manager).

1. **Drag and drop the Player Prefab into your scene**: If you have other main cameras active either disable them or make sure the player camera is the main one.
2. **Modify Player Properties (optional)**: Adjust the serialized fields (e.g., player speed, gravity direction, jump height) to customize the player movement.

OR

1. **Attach to Player**: Attach this script to the player GameObject. Add the necessary Rigidbody and Collider components (sphere and capsule collider). (If not this script will add them anyway at runtime)
2. **Input Setup**:
    - Set up input actions via the Unity Input System for movement, camera look, jumping, sprinting, and crouching.
    - Add the player input component to your player to receive the inputs

      OR

    - Use the playerControls input action asset that comes with the package and add it to the player input component
3. **Camera Setup**: The script automatically assigns the camera if no camera is found, but you can set a custom camera in the Inspector. Make sure the camera has a parent holder. (This is important for gravity manipulation to ensure everything rotates correctly)
4. **Modify Player Properties**: Adjust the serialized fields (e.g., player speed, gravity direction, jump height) to customize the player movement.

Contact us at : atrilliongames@gmail.com