Basing from prior research I have found a way to include looking up and down from the player perspective however, this makes the player start to slide a little. On top of this I have also found a way to lock the players perspective to the game to stop mis-clicks and improve base testing and gameplay.

[Tooltip("Speed multiplier for horizontal & vertical rotation.")]

public Vector2 turnSpeed = new Vector2(1, 1);

[Tooltip("Maximum rotation from the initial orientation.")]

public Vector2 degreeClamp = new Vector2(90, 80);

[Tooltip("Check this box if you want forward input to look downward.")]

public bool invertY;

Quaternion \_initialOrientation;

Vector2 \_currentAngles;

CursorLockMode \_previousLockState;

bool \_wasCursorVisible;

void OnEnable()

{

\_initialOrientation = transform.localRotation;

\_previousLockState = Cursor.lockState;

\_wasCursorVisible = Cursor.visible;

Cursor.lockState = CursorLockMode.Locked;

Cursor.visible = false;

}

void OnDisable()

{

Cursor.visible = \_wasCursorVisible;

Cursor.lockState = \_previousLockState;

transform.localRotation = \_initialOrientation;

}

Simply these lines lock the player’s cursor into the game and free the cursor after.

Rigid.MovePosition(transform.position + (transform.forward \* Input.GetAxis("Vertical") \* MoveSpeed) + (transform.right \* Input.GetAxis("Horizontal") \* MoveSpeed));

if (Input.GetKeyDown("space"))

Rigid.AddForce(transform.up \* JumpForce);

Vector2 motion = new Vector2(

Input.GetAxis("Mouse X"),

Input.GetAxis("Mouse Y"));

// Scale it by the turn speed, add it to our current angle, and clamp.

motion = Vector2.Scale(motion, turnSpeed);

\_currentAngles += motion;

\_currentAngles = Vector2.Min(\_currentAngles, degreeClamp);

\_currentAngles = Vector2.Max(\_currentAngles, -degreeClamp);

// Rotate to look in this direction, relative to our initial orientation.

Quaternion look = Quaternion.Euler(

-\_currentAngles.y, // Yaw

(invertY ? -1f : 1f) \* \_currentAngles.x, // Pitch

0); // Roll

transform.localRotation = \_initialOrientation \* look;

}

As for here this allows for player movement and horizontal + vertical aiming though not perfect by any means. This also allows for subtle user adjustments to suit the sensitivity to their liking.