Project Documentation
Recreating Apex Legends Ping System

Table of Contents

Devlog – Recreating the ping system from apex legends

#1 How to make the basic move ping

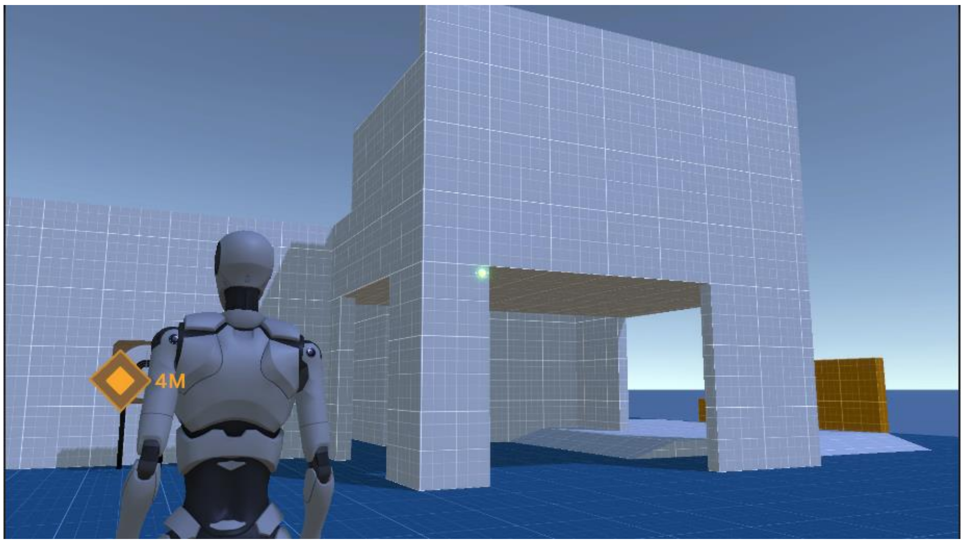
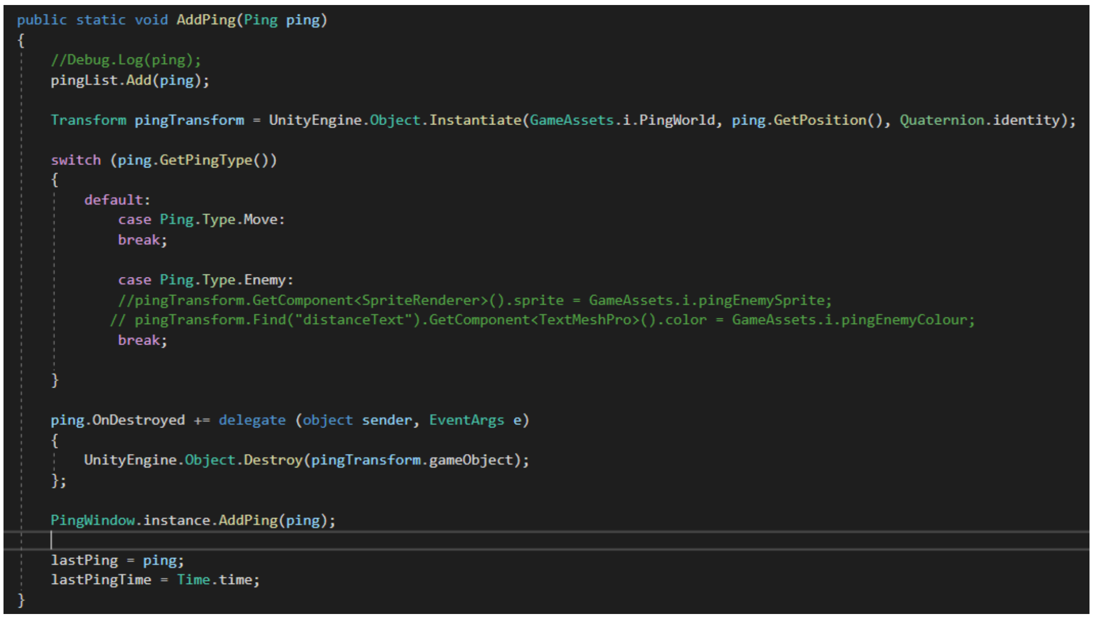


Figure 1 Worldspace ping (UI)

Apex legend's ping is context-sensitive. Until prototyping the concept it took me a while to understand that there should be two cameras one for showing the ping offscreen and the other to show in world space and have separate cameras for their respective canvas.



```
public static void AddPing(Ping ping)
{
    //Debug.Log(ping);
    pingList.Add(ping);

    Transform pingTransform = UnityEngine.Object.Instantiate(GameAssets.i.PingWorld, ping.GetPosition(), Quaternion.identity);

    switch (ping.GetPingType())
    {
        default:
            case Ping.Type.Move:
            break;

            case Ping.Type.Enemy:
            //pingTransform.GetComponent<SpriteRenderer>().sprite = GameAssets.i.pingEnemySprite;
           // pingTransform.Find("distanceText").GetComponent<TextMeshPro>().color = GameAssets.i.pingEnemyColour;
            break;

    }

    ping.OnDestroyed += delegate (object sender, EventArgs e)
    {
        UnityEngine.Object.Destroy(pingTransform.gameObject);
    };

    PingWindow.instance.AddPing(ping);

    lastPing = ping;
    lastPingTime = Time.time;
}
```

The method mentioned here in *Figure 2* explains how the ping system would be made for various types of pings. The pings would destroy depending on its priority and time. The problem here is the priority of the system to show the important pings when the player is offscreen and not the other pings.

Problems:

1. Managing the screen overlay and world space camera

2. Rotating the world space ping with respect to the Player

Solutions:

1. Having different scripts and cameras for each type of UI interface for differentiating and management

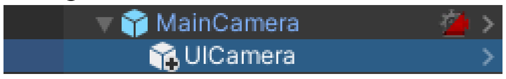
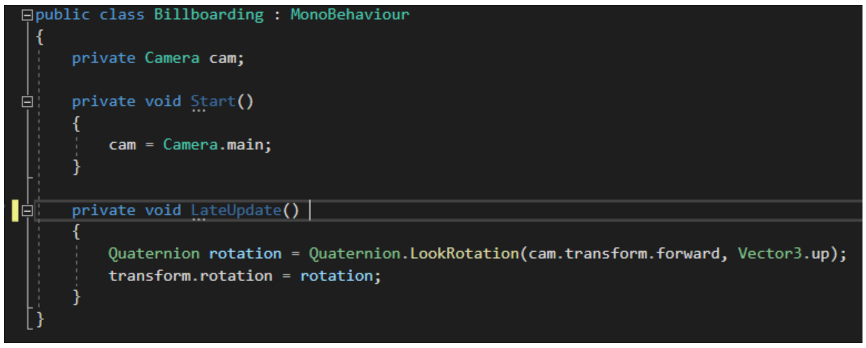


Figure 2 Setup for the cameras

2. Billboarding



```
public class Billboarding : MonoBehaviour
{
    private Camera cam;

    private void Start()
    {
        cam = Camera.main;
    }

    private void LateUpdate()
    {
        Quaternion rotation = Quaternion.LookRotation(cam.transform.forward, Vector3.up);
        transform.rotation = rotation;
    }
}
```

Figure 3 Billboard function

- Billboarding is used so sprites in world space turn/rotate to the player. This really helps in showing in the exact ping system in Apex legends

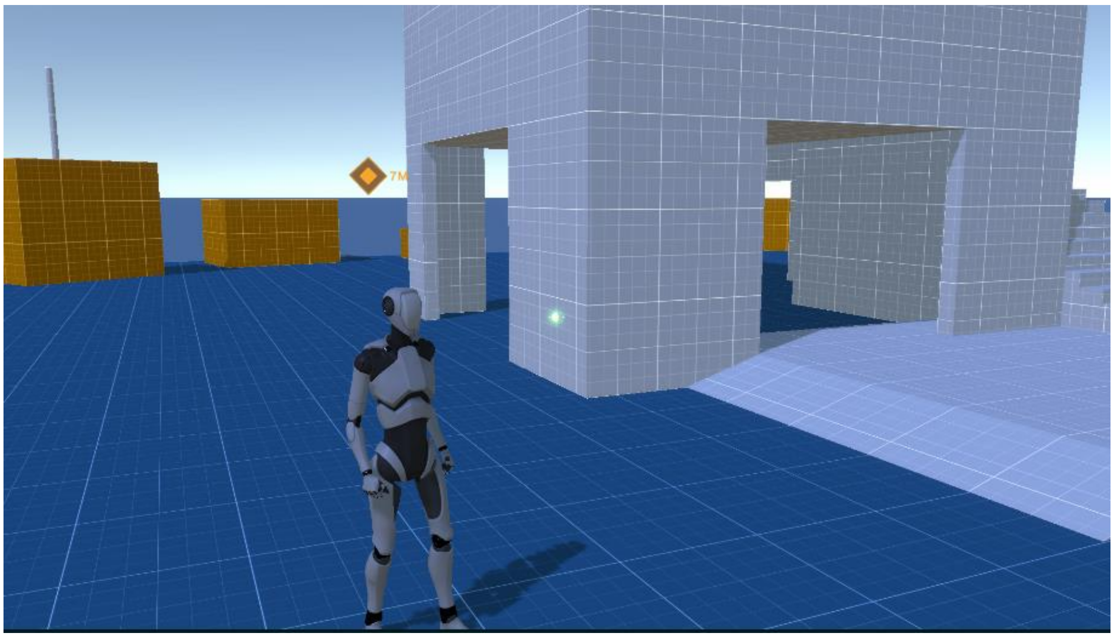


Figure 4 Result of Billboarding (https://media.giphy.com/media/sy5FMZ3XzqyqjlhgOd/giphy.gif)

This is the result where the worldspace ping is rotating towards the player bases on the camera position.

Limitations:

It is designed for one ping specifically (Move ping). The system is not universal for all pings and the code must be changed to achieve that. As this is a prototype, it was a great experience to just recreate a basic ping exactly from a AAA studio game.

Peculiarities:

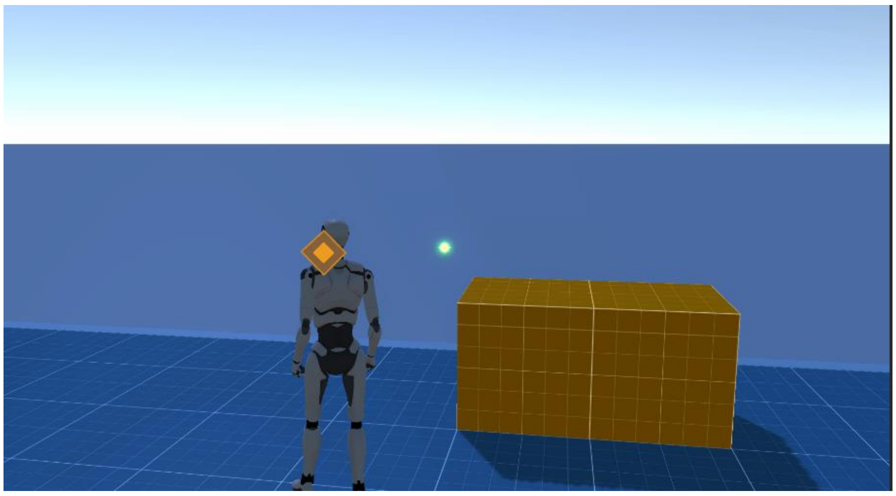
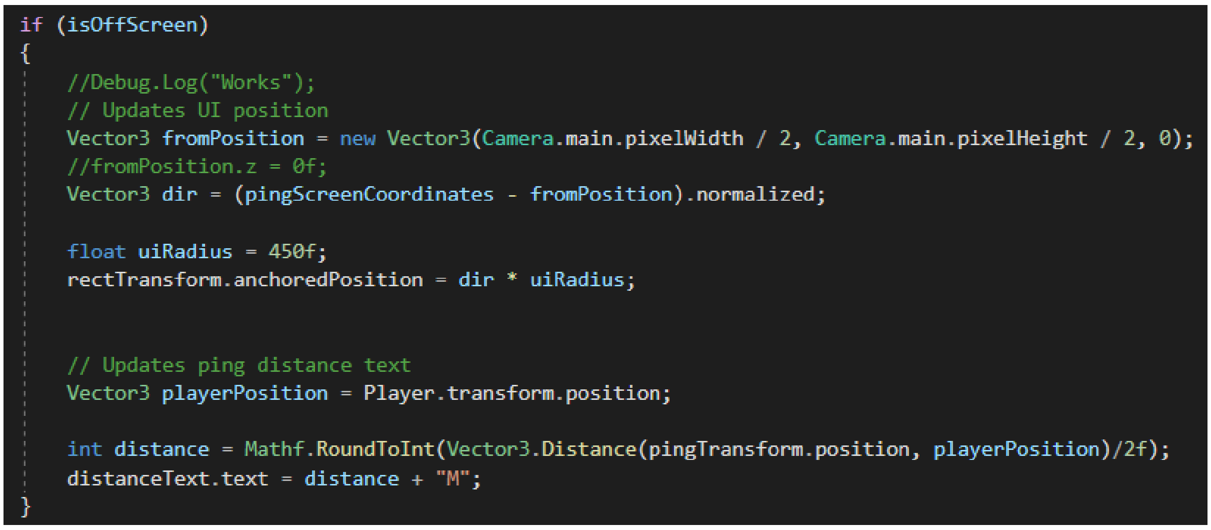


Figure 5 Offscreen UI

Here the ping is behind the player and must be visible as an off-screen UI. The odd thing here is that the offscreen ping is right behind the player's head and not exactly where the ping position is coming from. Also, the distance is not displayed yet. This is some odd behaviour that must be investigated while developing the prototype.

Solution to this odd behaviour:



```
if (isOffScreen)
{
    //Debug.Log("Works");
    // Updates UI position
    Vector3 fromPosition = new Vector3(Camera.main.pixelWidth / 2, Camera.main.pixelHeight / 2, 0);
    //fromPosition.z = 0f;
    Vector3 dir = (pingScreenCoordinates - fromPosition).normalized;

    float uiRadius = 450f;
    rectTransform.anchoredPosition = dir * uiRadius;


    // Updates ping distance text
    Vector3 playerPosition = Player.transform.position;

    int distance = Mathf.RoundToInt(Vector3.Distance(pingTransform.position, playerPosition)/2f);
    distanceText.text = distance + "M";
}
```

Figure 6 Math behind offscreen ping

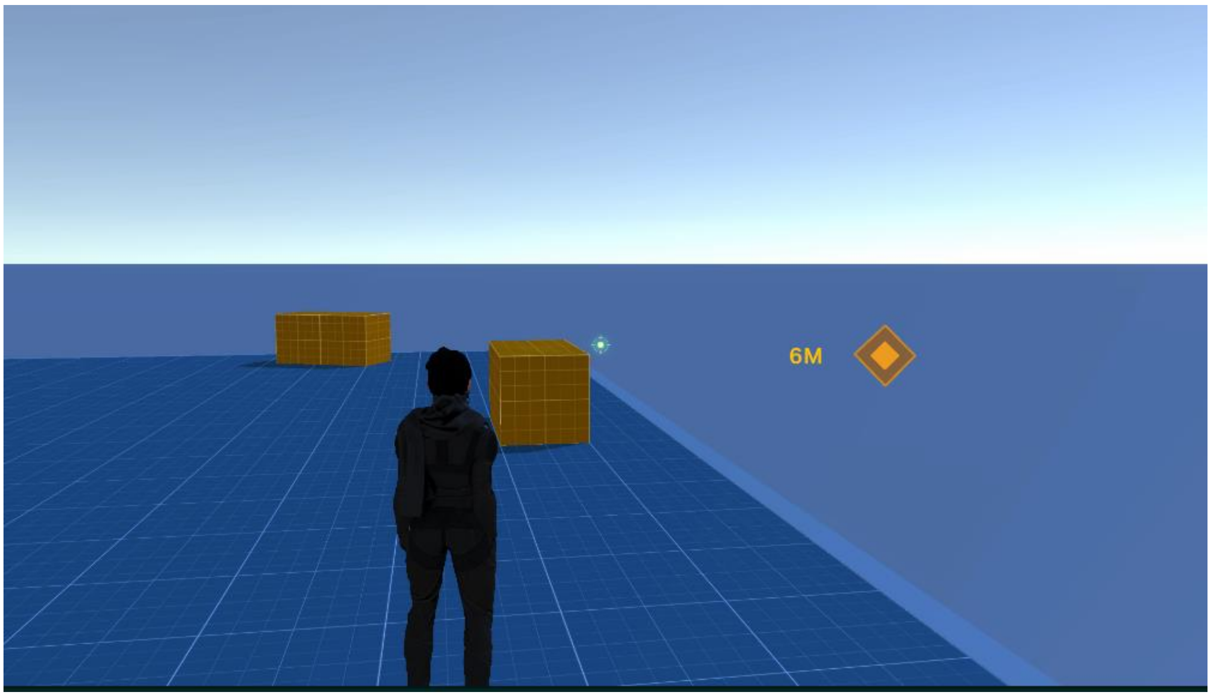


Figure 7 OffScreen (Solution)

The direction is the difference between the screenspace UI and worldspace UI. It finds the intersection point and then displays the off-screen UI with a hard-set radius.

#2 Creating the Ping Wheel

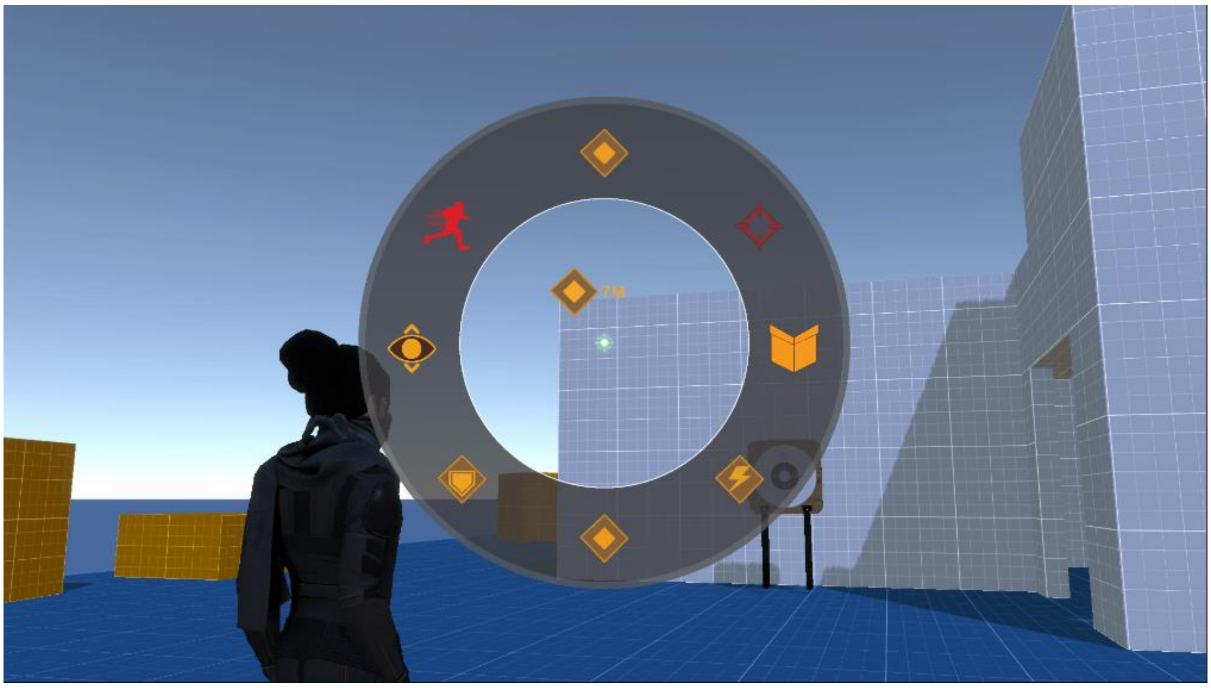
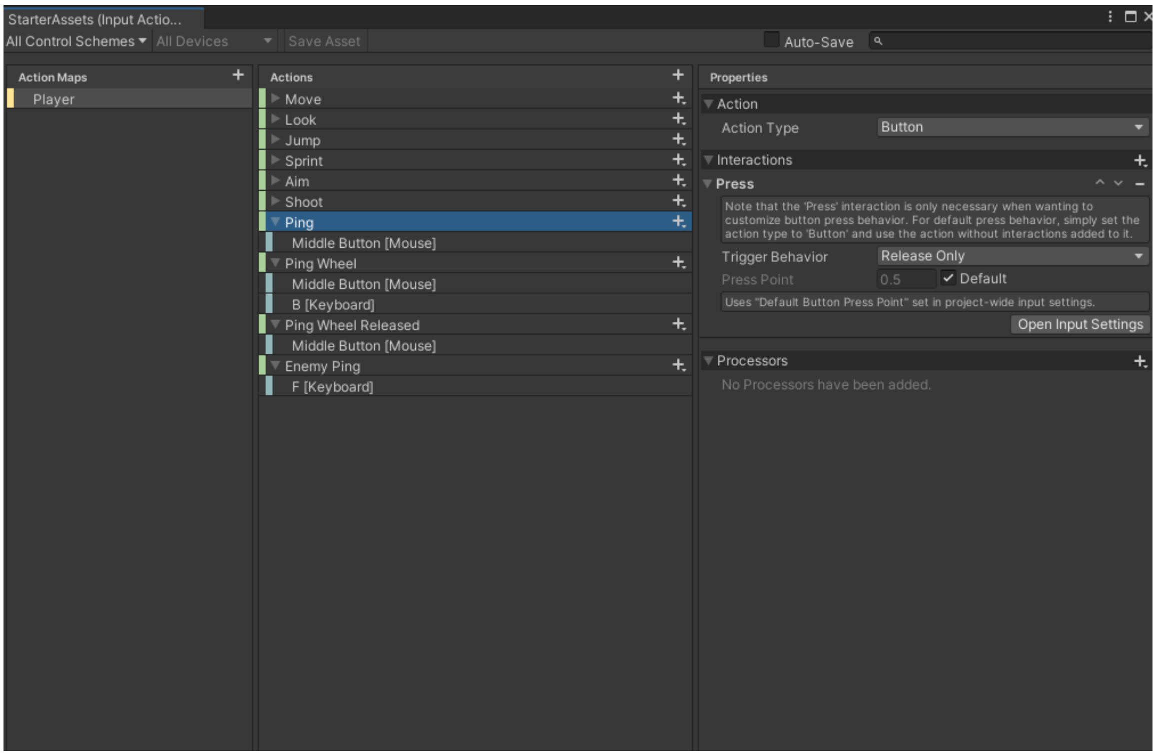


Figure 8 Ping Wheel Implementation

Code:

New Input System

Setup (Player Input Action)



StarterAssets (Input Actio...

All Control Schemes ▾ All Devices ▾ Save Asset Auto-Save

Action Maps
- Player

Actions
- ▶ Move
- ▶ Look
- ▶ Jump
- ▶ Sprint
- ▶ Aim
- ▶ Shoot
- ▼ Ping
  - Middle Button [Mouse]
- ▼ Ping Wheel
  - Middle Button [Mouse]
  - B [Keyboard]
- ▼ Ping Wheel Released
  - Middle Button [Mouse]
- ▼ Enemy Ping
  - F [Keyboard]

Properties

▼ Action

Action Type: Button

▼ Interactions

▼ Press

Note that the 'Press' interaction is only necessary when wanting to customize button press behavior. For default press behavior, simply set the action type to 'Button' and use the action without interactions added to it.

Trigger Behavior: Release Only

Press Point: 0.5 ✓ Default

Uses "Default Button Press Point" set in project-wide input settings.

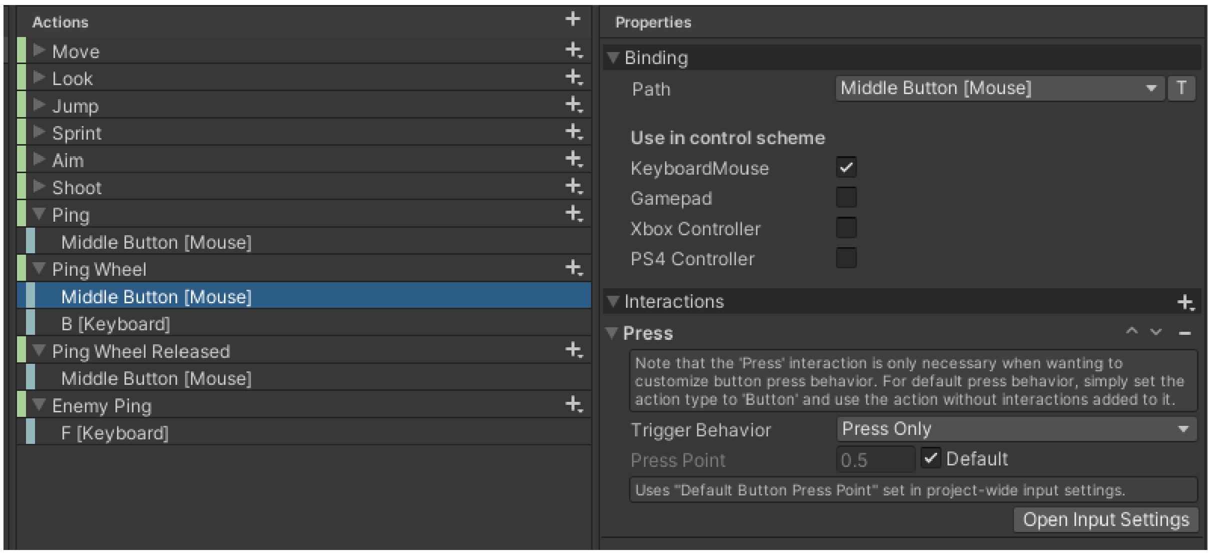
Open Input Settings

▼ Processors

No Processors have been added.

Figure 9 Interactions and Bindings

These are all the actions setup in the input action asset with the respective bindings(buttons) for pinging. The user can ping using the middle mouse button and the enemy ping can be triggered with 'F' on the keyboard.



Actions
- ▶ Move
- ▶ Look
- ▶ Jump
- ▶ Sprint
- ▶ Aim
- ▶ Shoot
- ▼ Ping
  - Middle Button [Mouse]
- ▼ Ping Wheel
  - Middle Button [Mouse]
  - B [Keyboard]
- ▼ Ping Wheel Released
  - Middle Button [Mouse]
- ▼ Enemy Ping
  - F [Keyboard]

Properties

▼ Binding

Path: Middle Button [Mouse]

Use in control scheme
- KeyboardMouse ✓
- Gamepad
- Xbox Controller
- PS4 Controller

▼ Interactions

▼ Press

Note that the 'Press' interaction is only necessary when wanting to customize button press behavior. For default press behavior, simply set the action type to 'Button' and use the action without interactions added to it.

Trigger Behavior: Press Only

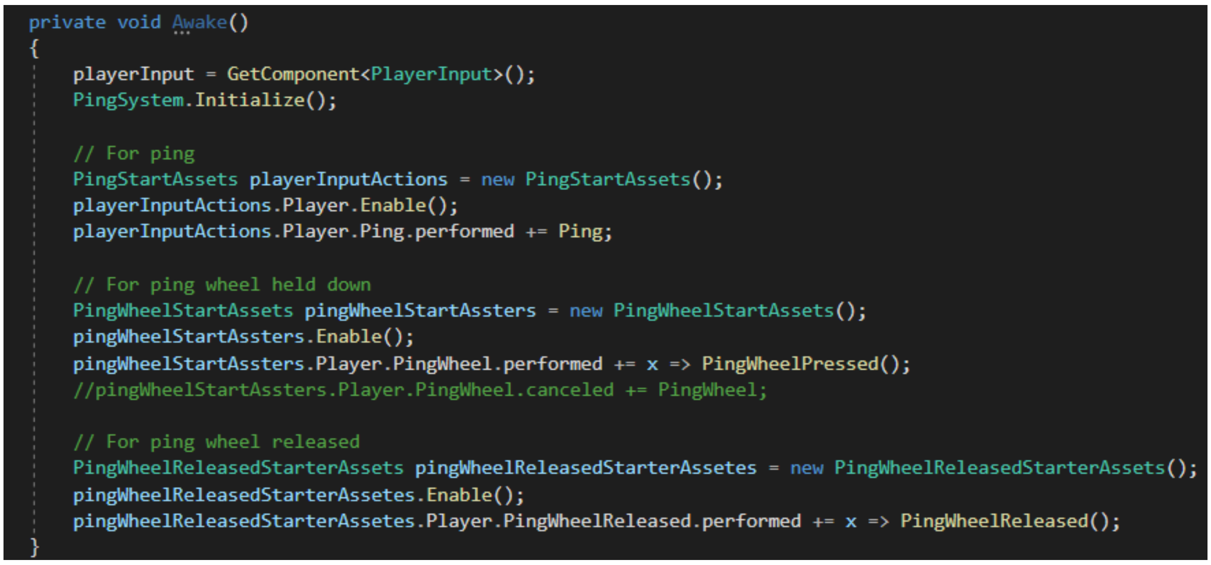
Press Point: 0.5 ✓ Default

Uses "Default Button Press Point" set in project-wide input settings.

Open Input Settings

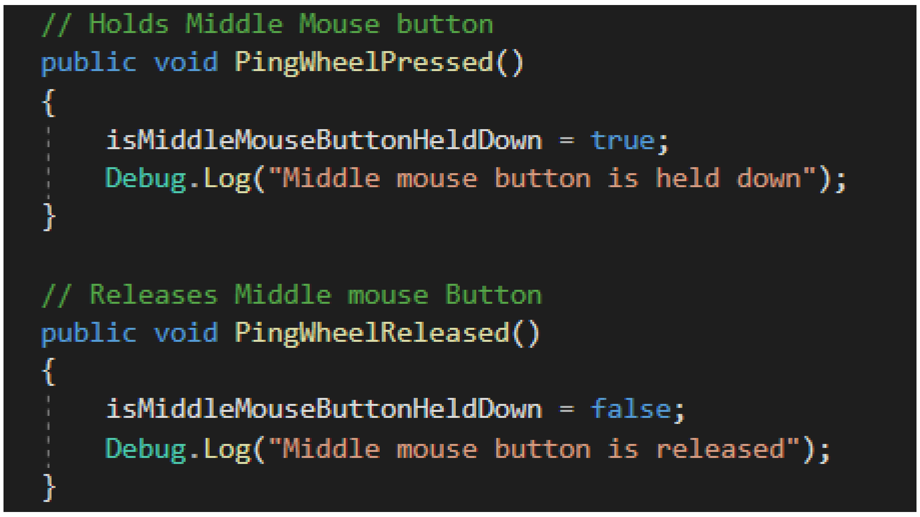
Figure 10 Interaction Setup

The ping wheel is set to "press only" in the interaction tabs and for the normal pinging it is set to "release only". This is done to trigger the ping wheel when it is pressed and held. While the general can only be spawned when the button is released.

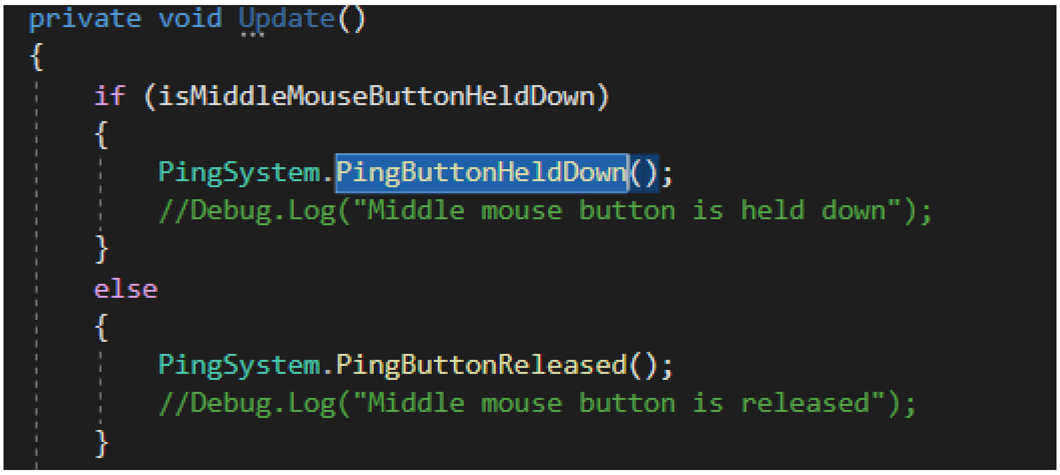


```
private void Awake()
{
    playerInput = GetComponent<PlayerInput>();
    PingSystem.Initialize();

    // For ping
    PingStartAssets playerInputActions = new PingStartAssets();
    playerInputActions.Player.Enable();
    playerInputActions.Player.Ping.performed += Ping;

    // For ping wheel held down
    PingWheelStartAssets pingWheelStartAssters = new PingWheelStartAssets();
    pingWheelStartAssters.Enable();
    pingWheelStartAssters.Player.PingWheel.performed += x => PingWheelPressed();
    //pingWheelStartAssters.Player.PingWheel.canceled += PingWheel;

    // For ping wheel released
    PingWheelReleasedStarterAssets pingWheelReleasedStarterAssetes = new PingWheelReleasedStarterAssets();
    pingWheelReleasedStarterAssetes.Enable();
    pingWheelReleasedStarterAssetes.Player.PingWheelReleased.performed += x => PingWheelReleased();
}
```

To use the new input system in code, this is how you would initialize it awake. I then have a function to get some Booleans true. These functions are bonded to the specific middle mouse button pressed.



```
// Holds Middle Mouse button
public void PingWheelPressed()
{
    isMiddleMouseButtonHeldDown = true;
    Debug.Log("Middle mouse button is held down");
}

// Releases Middle mouse Button
public void PingWheelReleased()
{
    isMiddleMouseButtonHeldDown = false;
    Debug.Log("Middle mouse button is released");
}
```



```
private void Update()
{
    if (isMiddleMouseButtonHeldDown)
    {
        PingSystem.PingButtonHeldDown();
        //Debug.Log("Middle mouse button is held down");
    }
    else
    {
        PingSystem.PingButtonReleased();
        //Debug.Log("Middle mouse button is released");
    }
```

As I need to check if it is being held every frame, I do some condition checks on update and when isMiddleMouseButtonHeldDown returns true the ping wheel shows up and the user can ping based on the ping wheel.

Sources:

samyam. (2021). *YouTube*. Retrieved November 30, 2021, from https://www.youtube.com/watch?v=m5WsmlEOFiA

Unity. (2021). *YouTube*. Retrieved November 30, 2021, from https://www.youtube.com/watch?v=4QuPlKzdq14

Zyger. (2021). *YouTube*. Retrieved November 30, 2021, from https://www.youtube.com/watch?v=MRdQyrpflB4