1. 检查按钮状态

您可以使用 ButtonInput<MouseButton> 资源来检查特定鼠标按钮的状态：

- 使用 .pressed(…) / .released(…) 检查按钮是否被按住，只要按钮处于相应状态，这些方法每帧都会返回 true。
- 使用 .just_pressed(…) / .just_released(…) 检测实际的按下/释放，这些方法仅在按下/释放发生的帧更新时返回 true。

2. 运行条件

另一种工作流程是为您的系统添加运行条件，使它们仅在发生适当输入时运行。

强烈建议您编写自己的运行条件，以便您可以检查任何您想要的内容，支持可配置的绑定等。

对于原型设计，Bevy 提供了一些内置的运行条件：

```
toggle_color.run_if(input_just_pressed(MouseButton::Left)),
move_camera.run_if(input_pressed(MouseButton::Middle)),
```

3. 鼠标滚动 / 滚轮

要检测滚动输入，请使用 MouseWheel 事件：

```
fn scroll_camera(
    mut camera: Query<&mut Transform, With<Camera>>,
    mut wheel_reader: EventReader<MouseWheel>,
) {
    let Ok(mut camera_transform) = camera.get_single_mut()
else {
        return;
    };
    for wheel in wheel_reader.read() {
        let unit = if let MouseScrollUnit::Line = wheel.unit
{
            35.
        } else {
            1.
        };
        camera_transform.translation += Vec3::new(0.,
wheel.y * unit, 0.);
    }
}
```

MouseScrollUnit 枚举很重要：它告诉您滚动输入的类型。Line 适用于具有固定步长的硬件，如桌面鼠标上的滚轮。Pixel 适用于具有平滑（细粒度）滚动的硬件，如笔记本电脑触控板。

您可能需要对这些进行不同的处理（使用不同的灵敏度设置），以在两种类型的硬件上提供良好的体验。

注意：Line 单位不保证具有整数值/步长！至少 macOS 在操作系统级别对滚动进行非线性缩放/加速，这意味着即使使用带有固定步长滚轮的常规 PC 鼠标，您的应用程序也会获得奇怪的行数值。

4. 鼠标运动

如果您不关心鼠标光标的确切位置，而只是想查看鼠标从帧到帧移动了多少，请使用此选项。这对于控制 3D 摄像机等非常有用。

使用 MouseMotion 事件。每当鼠标移动时，您将获得一个带有增量的事件。

```
fn move_camera(
    mut camera: Query<&mut Transform, With<Camera>>,
    mut motion_reader: EventReader<MouseMotion>,
) {
    let Ok(mut camera_transform) = camera.get_single_mut()
else {
        return;
    };
    for motion in motion_reader.read() {
        camera_transform.translation += Vec3::new(-
motion.delta.x, motion.delta.y, 0.);
    }
}
```

5. 光标位置

如果您想准确跟踪指针/光标的位置，请使用此选项。这对于在游戏或 UI 中点击和悬停在对象上非常有用。

您可以从相应的窗口获取鼠标指针的当前坐标（如果鼠标当前在该窗口内）：

```
fn toggle_color(
    camera: Query<(&Camera, &GlobalTransform)>,
    window: Query<&Window>,
    squares: Query<(&Handle<ColorMaterial>,
&GlobalTransform)>,
    mut materials: ResMut<Assets<ColorMaterial>>,
) {
    let (camera, camera_transform) = camera.single();
    let Some(cursor_position) =
window.single().cursor_position() else {
        return;
    };
    let Some(cursor_position) =
camera.viewport_to_world_2d(camera_transform,
cursor_position)
    else {
        return;
    };
    for (square_color_handle, square_transform) in &squares
{
        let translation = square_transform.translation();
        if cursor_position.x > translation.x - 40.
            && cursor_position.x < translation.x + 40.
            && cursor_position.y > translation.y - 40.
            && cursor_position.y < translation.y + 40.
        {
            let Some(material) =
materials.get_mut(square_color_handle) else {
                continue;
            };
            let hsva = Hsva::from(material.color);
            material.color =
Color::Hsva(hsva.with_hue((hsva.hue + 180.) % 360.));
        }
    }
}
```

要检测指针移动时，请使用 CursorMoved 事件获取更新的坐标：

```
fn cursor_circle(
    camera: Query<(&Camera, &GlobalTransform)>,
    mut gizmo: Gizmos,
    mut moved_reader: EventReader<CursorMoved>,
) {
    let Ok((camera, camera_transform)) = camera.get_single()
else {
        return;
    };
    for moved in moved_reader.read() {
        let Some(point) =
camera.viewport_to_world_2d(camera_transform,
moved.position) else {
            continue;
        };
        gizmo.circle_2d(point, 20., Color::WHITE);
    }
}
```

请注意，您只能获取窗口内鼠标的位置；您无法获取整个操作系统桌面/整个屏幕上的鼠标全局位置。

您获得的坐标位于“窗口空间”。它们表示窗口像素，原点是窗口的左上角。