Ubiquitous and Mobile Computing
Introduction to Unity

Aaron Haim, Ashish Gurung,
Aaron Alphonsus, Priyanka Benachamardi

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

What is Unity?

- A cross-platform game engine developed by Unity Technologies.
- Originally made exclusively for Apple devices, received Android support in 2010 with Unity 3.
- Unity 5 made it universally accessible to more than 25 platforms
- Latest Version, 2020.1.12f1

Specific Problems

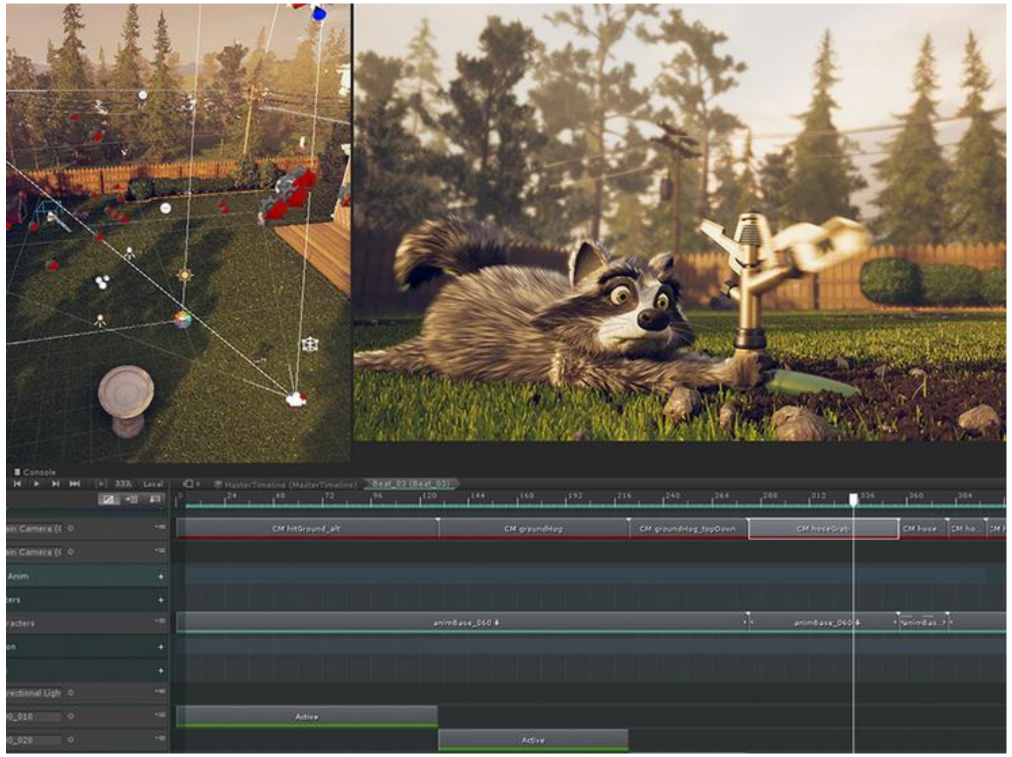
- Game developers had to design and develop their own game engines
  - id Tech 4 was available to the public, but it was technically limited
    - Limited in design, provided no multi-core support by default
    - Reliance on high-end hardware components
  - Unreal Engine was also available, but limited
    - Made specifically for 3D support and offers no benefits compared to other 2D engines
    - Development Kit was restricted to engine licensing until November 2009
- Unity was made to make 2D and 3D interactions more accessible

Use Cases and Examples

- Game Development
  - 2D (Among Us), 3D (Subway Surfers), AR (Pokémon Go), VR (Superhot)

Use Cases and Examples

- Television/Film Development
  - Sonder, Mr. Carton, Giant Bear, Sherman Animation Project



Use Cases and Examples

- Automotive, Transportation, & Manufacturing
  - Design and test virtual cars (Autoliv)

Normal
X-Ray

Use Cases and Examples

- Architecture, Engineering, & Construction
  - Bridges between buildings (SHoP Architects)

Effects

- 2D Effects
  - Sprite mapping, World Renderer
- 3D Effects
  - Mipmaps, Screen Space Ambient Occlusion, Full-Screen Post-Processing
- Other Effects
  - Support for Multithreaded processes, Low-Level Plugin Interface

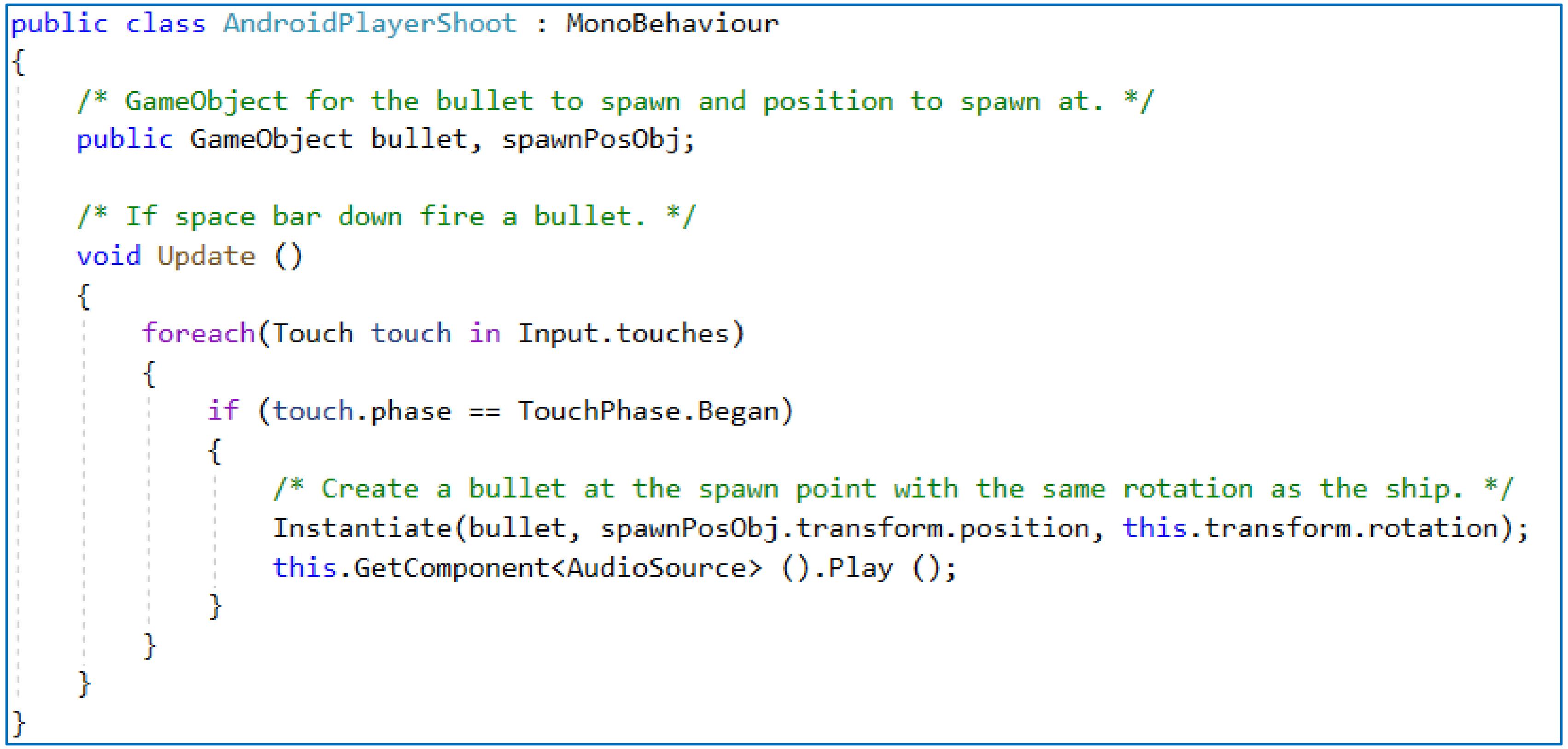
How It Works

- Uses a scripting API in C# combined with Drag-and-Drop Object Creation
  - 2-Stage Language Translator
    - Converts specified code into C++
    - Converts the C++ to its target platform
  - Interface API
    - Project is pre-compiled for usage with target platform
    - Uses a simple handler to grab inputs and display output
  - Managed Code Stripping
    - Faster build time with less compilation and conversion
    - Includes scripts, plugins, and .NET frameworks

How does it work with Android?

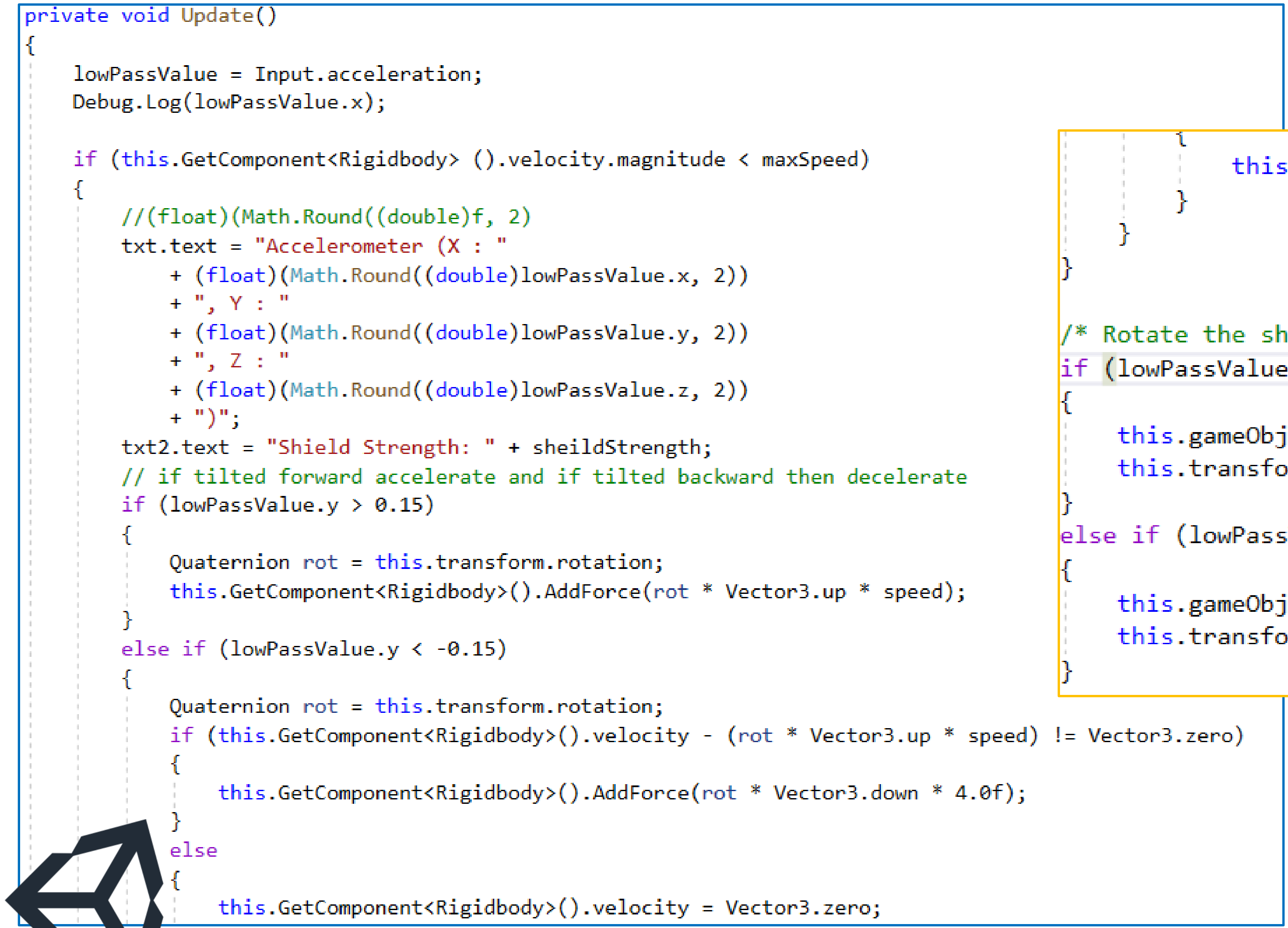
- Unity builds an Android app
- Includes .NET bytecode interpreter in native code; based on Mono
- During runtime, the interpreter executes the bytecodes

Code Snippet(Touch Behavior)

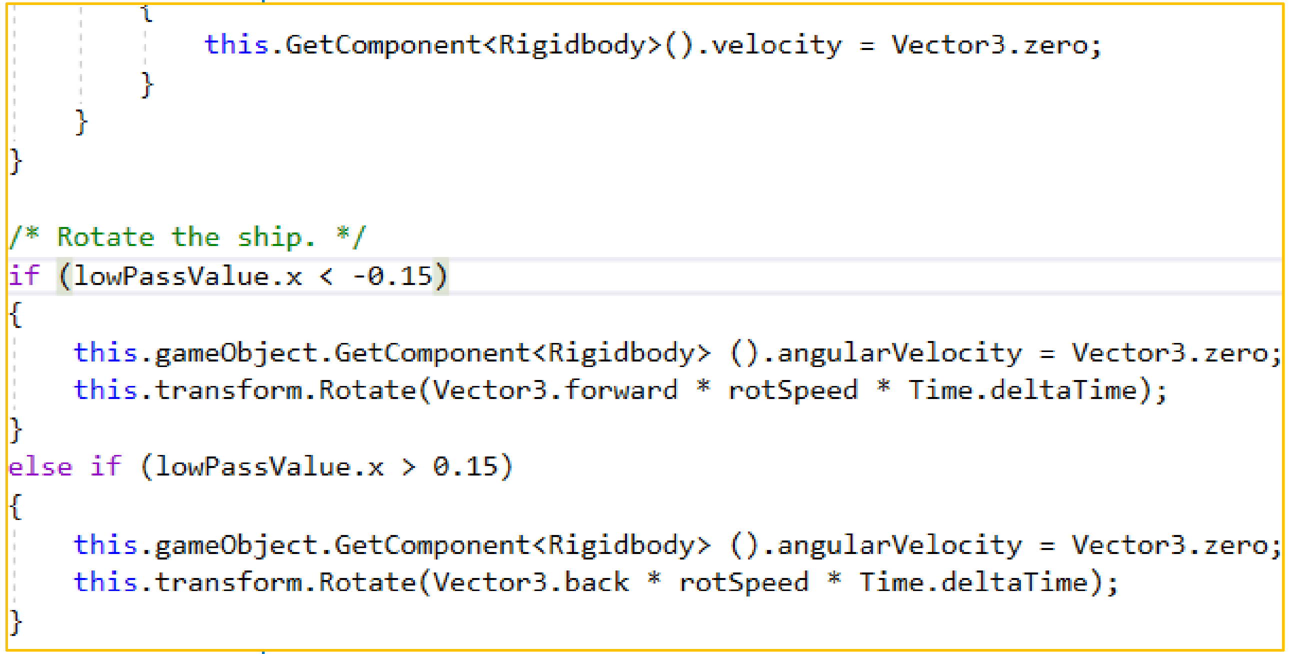


```
public class AndroidPlayerShoot : MonoBehaviour
{
    /* GameObject for the bullet to spawn and position to spawn at. */
    public GameObject bullet, spawnPosObj;

    /* If space bar down fire a bullet. */
    void Update ()
    {
        foreach(Touch touch in Input.touches)
        {
            if (touch.phase == TouchPhase.Began)
            {
                /* Create a bullet at the spawn point with the same rotation as the ship. */
                Instantiate(bullet, spawnPosObj.transform.position, this.transform.rotation);
                this.GetComponent<AudioSource> ().Play ();
            }
        }
    }
}
```

Code Snippet(Text View and Accelerometer)

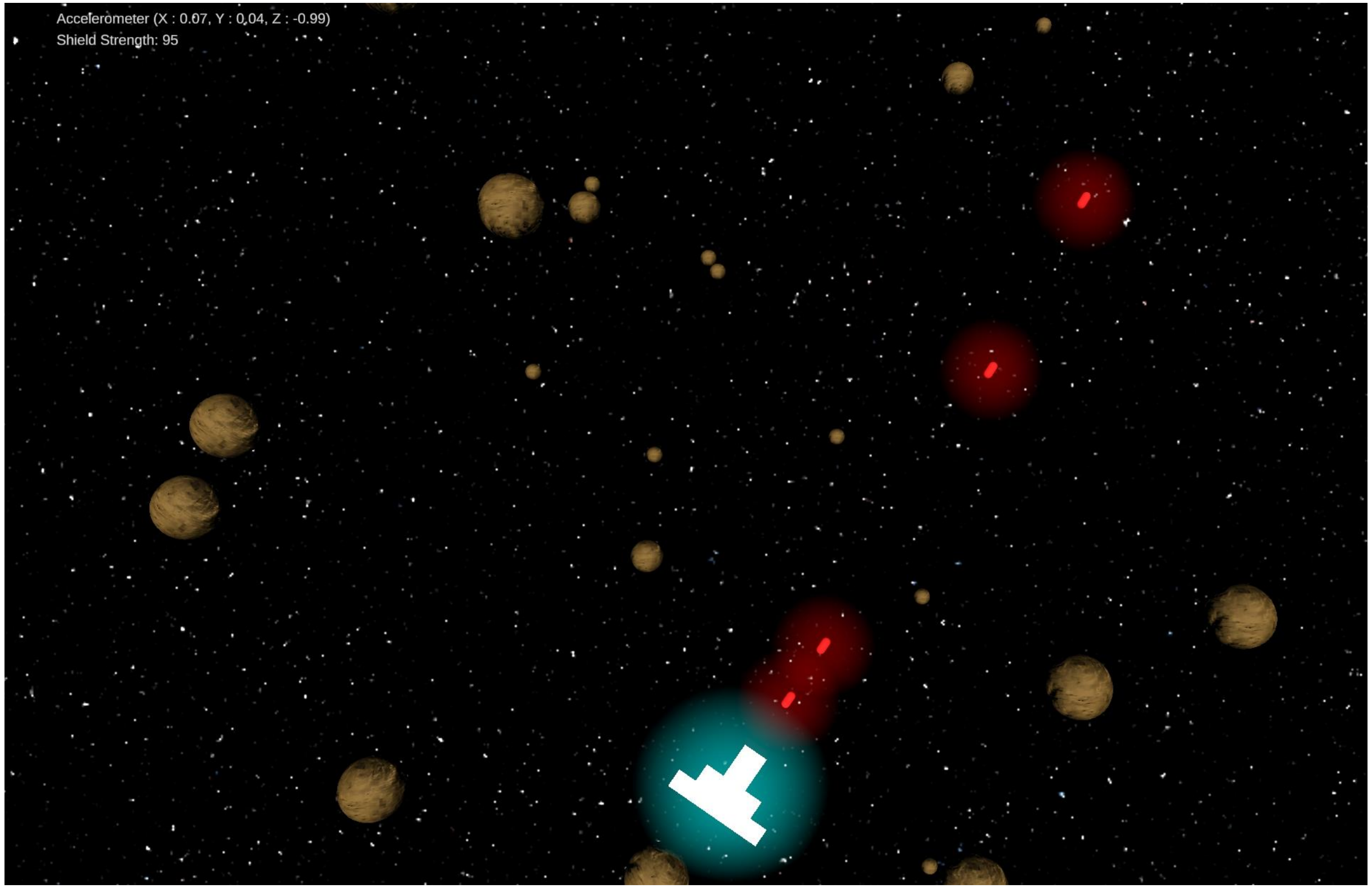


```
private void Update()
{
    lowPassValue = Input.acceleration;
    Debug.Log(lowPassValue.x);

    if (this.GetComponent<Rigidbody> ().velocity.magnitude < maxSpeed)
    {
        //(float)(Math.Round((double)f, 2)
        txt.text = "Accelerometer (X : "
            + (float)(Math.Round((double)lowPassValue.x, 2))
            + ", Y : "
            + (float)(Math.Round((double)lowPassValue.y, 2))
            + ", Z : "
            + (float)(Math.Round((double)lowPassValue.z, 2))
            + ")";
        txt2.text = "Shield Strength: " + sheildStrength;
        // if tilted forward accelerate and if tilted backward then decelerate
        if (lowPassValue.y > 0.15)
        {
            Quaternion rot = this.transform.rotation;
            this.GetComponent<Rigidbody>().AddForce(rot * Vector3.up * speed);
        }
        else if (lowPassValue.y < -0.15)
        {
            Quaternion rot = this.transform.rotation;
            if (this.GetComponent<Rigidbody>().velocity - (rot * Vector3.up * speed) != Vector3.zero)
            {
                this.GetComponent<Rigidbody>().AddForce(rot * Vector3.down * 4.0f);
            }
            else
            {
                this.GetComponent<Rigidbody>().velocity = Vector3.zero;
```



```
            {
                this.GetComponent<Rigidbody>().velocity = Vector3.zero;
            }
        }
}

/* Rotate the ship. */
if (lowPassValue.x < -0.15)
{
    this.gameObject.GetComponent<Rigidbody> ().angularVelocity = Vector3.zero;
    this.transform.Rotate(Vector3.forward * rotSpeed * Time.deltaTime);
}
else if (lowPassValue.x > 0.15)
{
    this.gameObject.GetComponent<Rigidbody> ().angularVelocity = Vector3.zero;
    this.transform.Rotate(Vector3.back * rotSpeed * Time.deltaTime);
}
```

Screenshot of the game



QUESTIONS

References

- https://unity.com/
- https://unity.com/solutions/film/animation
- https://unity.com/case-study/autoliv
- https://unity.com/case-study/shop-architects
- https://docs.unity3d.com/Manual/NativePluginInterface.html
- https://docs.unity3d.com/2019.1/Documentation/Manual/ManagedCodeStripping.html
- https://www.mono-project.com/
- https://docs.unity3d.com/530/Documentation/Manual/android.html
- https://docs.unity3d.com/560/Documentation/Manual/MobileInput.html