Project Progression

Alexander Phillips • Ethan Zafra

The Problem:

- **What: Gun Controllers are expensive.** We need a way to create a “cost effective” immersive gun controller for PC gaming that can detect where the user is aiming at on a screen
- **Why:** It’s important to create this system to create a reliable and immersive alternative to the gamepad
- **How:** Using a microbit, Accelerometer + gyroscope to track the gun.

The Context:

Similar Solutions:

- Sinden Lightgun & AimTrak Light Gun are available, but with high prices. Both cost around $130

- Bogatinov et al. developed a relatively cheap firearms simulator using a Microsoft Kinect, but were limited by the Kinect's ability to accurately portray coordinates from a long distance.

- Choi et al. used an infrared ray module with the wii remotes optical sensor, but could not predict wide-angle lens distortion.

The Sinden Lightgun®

AimTrak Light Gun®

Design Thinking

User’s POV:

Need a way to aim anywhere on the screen and have a way to refresh if inaccurate.

Has a trigger to Shoot

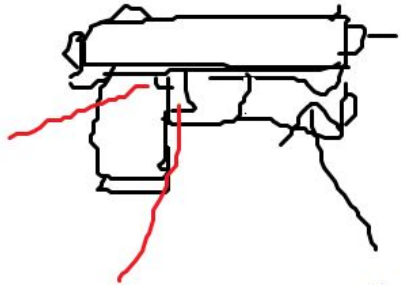
Way to reload if the game requires it

Feedback to give some of the feel of a gun

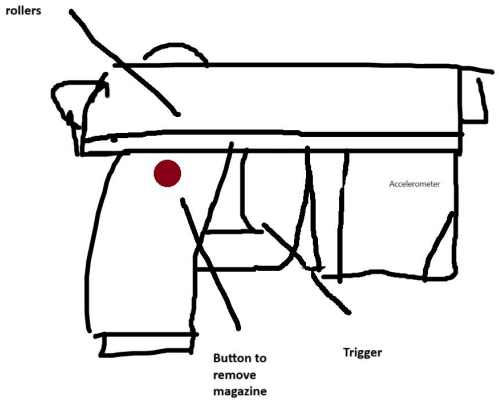
Need	Rating
Balanced Weight	4
Aim Accuracy	10
Feel/Feedback	7
Durability	6
Look	4
Responsiveness	5
Battery Life	1

WEIGHT	ACCURACY	FEEDBACK	RESISTANCE	REALISM	INPUT DELAY	DURATION
A	B	C	D	E	F	G

Initial Sketches and Prototypes



Accelerometer



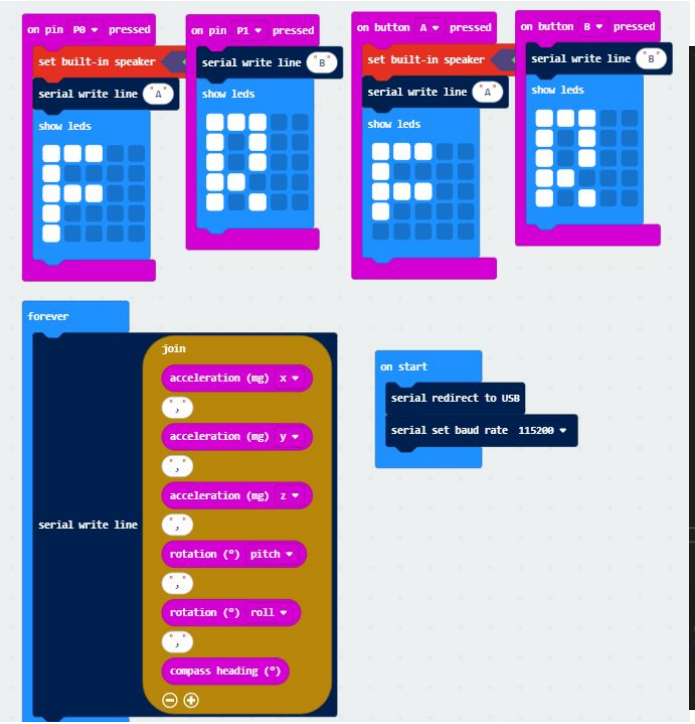
(3D SCAN)

Brainstorm Ideas

- **Valid:** Possibility of having a calibration section for accelerometer and gyroscope inside the gun
- **Promising:** Make a racking function for feedback
- **Valid:** Using infrared LEDs for the gun to track where its aiming
- **Valid:** With cameras and LED for the game to track where the gun is relative to the screen
- **Valid:** Take a picture of the screen and take in which colour hits
- **Promising:** Pistons to simulate the force made by a slide going back.

Electronics

Makecode:



Unity:

```
void OnMessageArrived(string msg)
{


    timeSinceLastCall = Time.realtimeSinceStartup - timeSinceStart;
    timeSinceStart = Time.realtimeSinceStartup;
    msg = msg.Trim(); // Remove any extra whitespace or newline characters
    Debug.Log("Message from micro:bit: " + msg);

    if (isFirst)
    {
        motion = Array.ConvertAll(msg.Split(','), float.Parse);
        startingBearing = motion[5];
    }


    if (msg == "A")
    {
        if(ammo > 0)
        {
            GameObject boolet = Instantiate(bulletPrefab,transform.position, transform.rotation);
            boolet.GetComponent<Rigidbody>().velocity = transform.forward * bulletSpeed;
        }
    }
    else if (msg == "B")
    {
        ammo = 10;
    }else if (msg != "")
    {
        motion = Array.ConvertAll(msg.Split(','), float.Parse);
        transform.rotation = Quaternion.Lerp(transform.rotation, Quaternion.Euler(motion[3], motion[5]+startingBearing, motion[4]), rotationSpeed * timeSinceLastCall);

        transform.position += new Vector3(motion[0] * speed * timeSinceLastCall,0,-motion[1] * speed * timeSinceLastCall);
    }
}
// Called on connect/disconnect events
0 references
void OnConnectionEvent(bool success)
{
    Debug.Log(success ? "Connected to micro:bit" : "Disconnected from micro:bit");
}
```

Unity Demo

fin