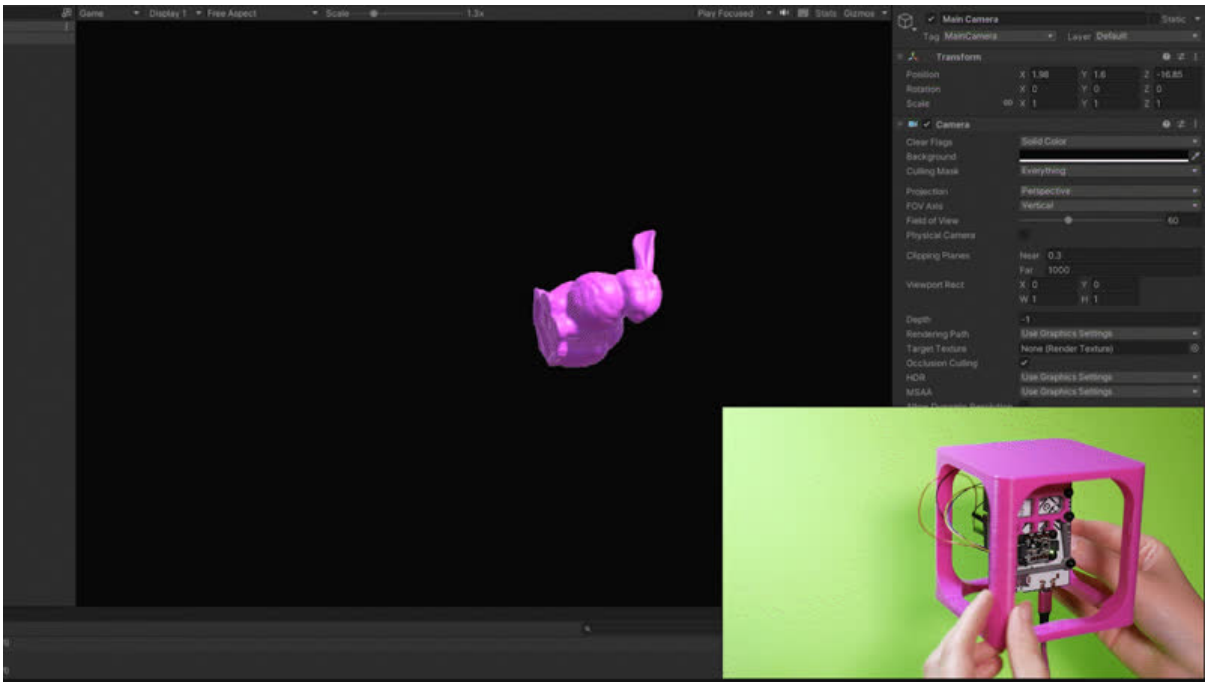
Controlling Objects in Unity with a 9 DoF Sensor and Arduino

Created by Liz Clark

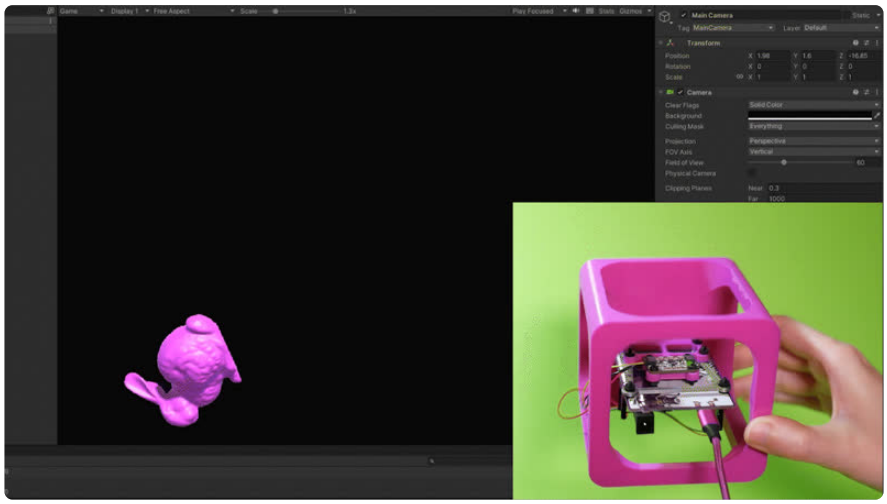


https://learn.adafruit.com/controlling-objects-in-unity-with-arduino

Last updated on 2024-06-03 03:40:55 PM EDT

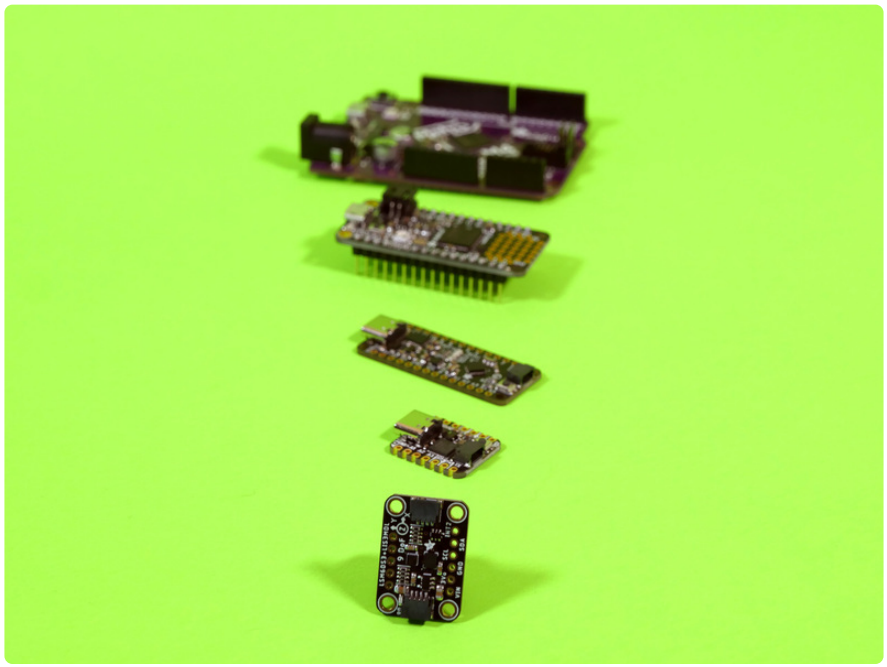
Table of Contents

Overview



Unity (https://adafru.it/10Fs) is one of the most widely used game engines for developing video games. In this guide, you'll learn how to integrate your Arduino based hardware projects with your Unity projects to create interactive experiences.

Data from a 9 DoF sensor is sent over a serial connection from the board to a C# script in Unity. The C# script is set up to affect Unity objects with the incoming data.



Calibrating Your 9 DoF Sensor

The Adafruit LSM6DS3TR-C + LIS3MDL board has an accelerometer, gyroscope and magnetometer onboard, making it a 9 Degrees of Freedom (DoF) sensor. You can calibrate the various sensors for the most accurate movement measurements

possible using the How to Fuse Motion Sensor Data into AHRS Orientation Learn Guide (https://adafru.it/LAr).

By following that guide, your chosen Arduino board stores the calibration in non-volatile memory so that it can be accessed going forward. Unity expects Quaternion values for rotation, so that calibration method is the most useful.

How to Fuse Motion Sensor Data into AHRS Orientation (Euler/ Quaternions)

https://adafru.it/LAr

There is Arduino code included for using the LSM6DS3TR-C + LIS3MDL sensor without calibration. It takes the accelerometer readings and uses the Arduino `map_range()` function to map the values between `-180` and `180` for Euler readings. It won't be as accurate as a fully calibrated sensor, but it will demonstrate how to input data into Unity.

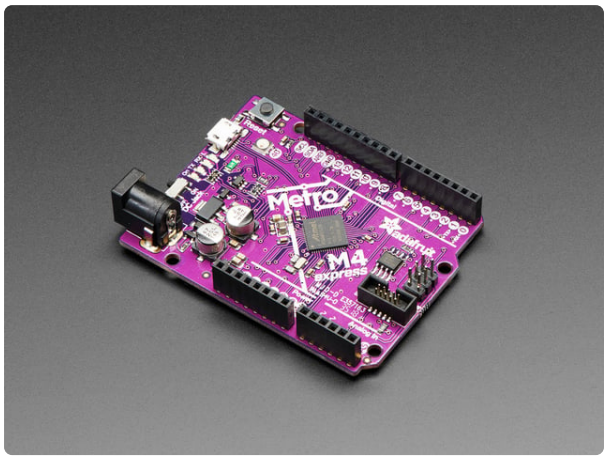
Prerequisite Guides

Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU

https://adafru.it/10Ft

Parts

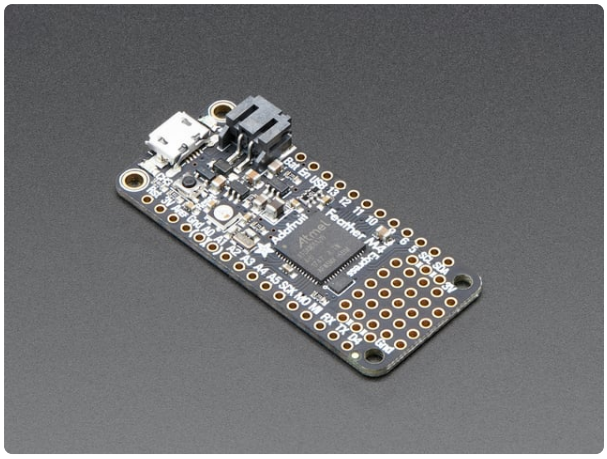
An M0 or M4 board works best for this project, especially if you're using a stored calibration for the sensor.



Adafruit Metro M4 feat. Microchip ATSAMD51

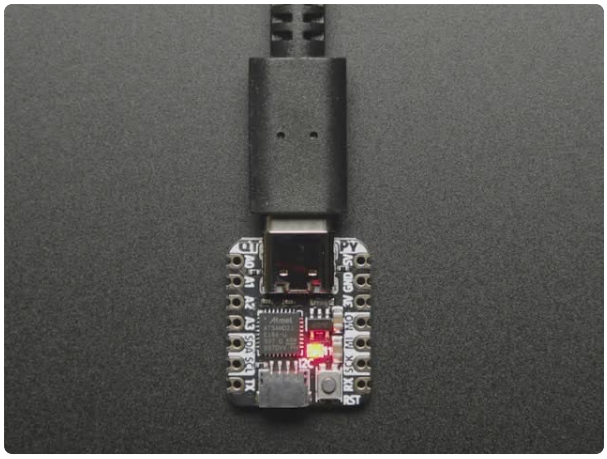
Are you ready? Really ready? Cause here comes the fastest, most powerful Metro ever. The Adafruit Metro M4 featuring the Microchip ATSAMD51. This...

https://www.adafruit.com/product/3382



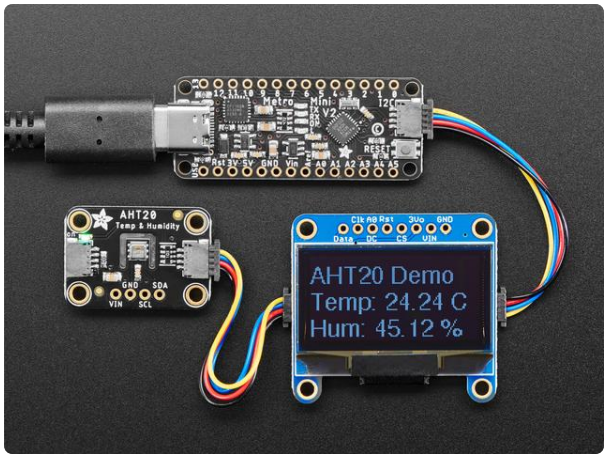
Adafruit Feather M4 Express - Featuring ATSAMD51

It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...
https://www.adafruit.com/product/3857



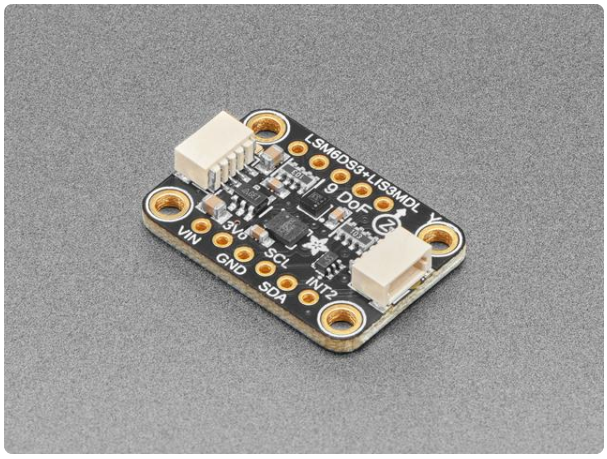
Adafruit QT Py - SAMD21 Dev Board with STEMMA QT

What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with our favorite lil chip, the SAMD21 (as made famous in our GEMMA M0 and Trinket M0 boards).This time it...
https://www.adafruit.com/product/4600



Adafruit Metro Mini 328 V2 - Arduino-Compatible - 5V 16MHz

Now available here!One of our star development boards is...
https://www.adafruit.com/product/5597



Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU

Add high-quality motion, direction, and orientation sensing to your Arduino project with this all-in-one 9 Degree of Freedom (9-DoF) sensor with sensors from ST. This little...
https://www.adafruit.com/product/5543

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

1 x USB A to USB C Cable — https://www.adafruit.com/product/5153

Pink and Purple Woven USB A to USB C Cable - 1 meter long

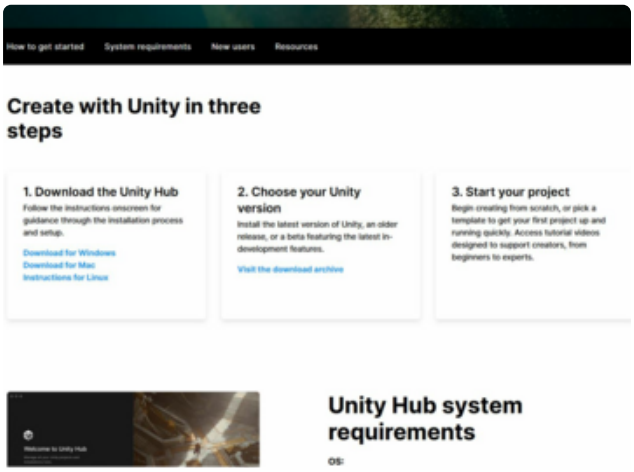
---

1 x USB A to micro B Cable — https://www.adafruit.com/product/4111

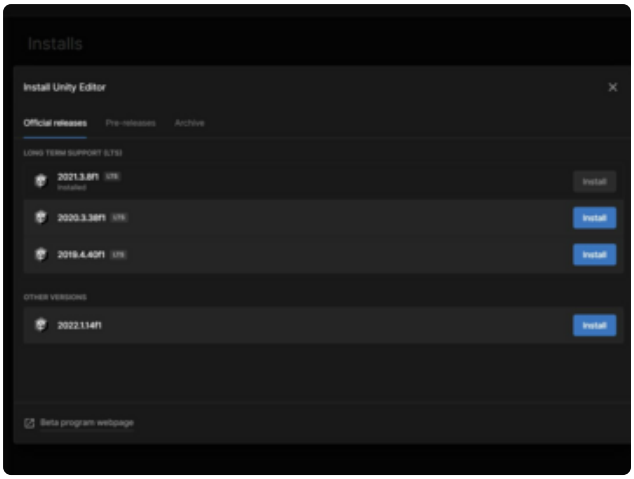
Fully Reversible Pink/Purple USB A to micro B Cable - 1m long

---

Install and Setup Unity with Visual Studio

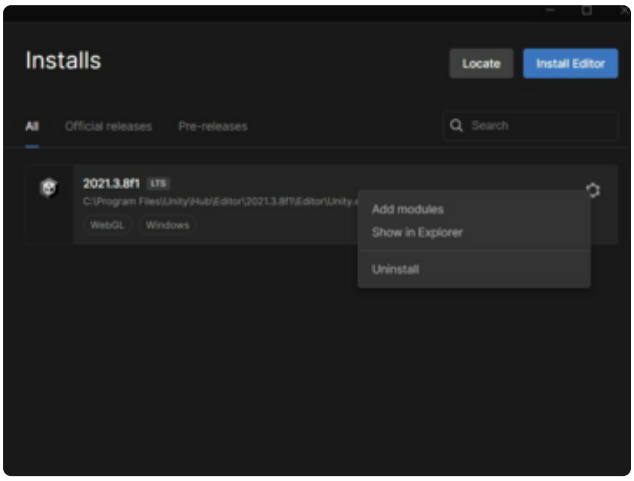


Navigate to the Downloads page on the Unity website (https://adafru.it/10Fu) and download Unity Hub for your operating system.

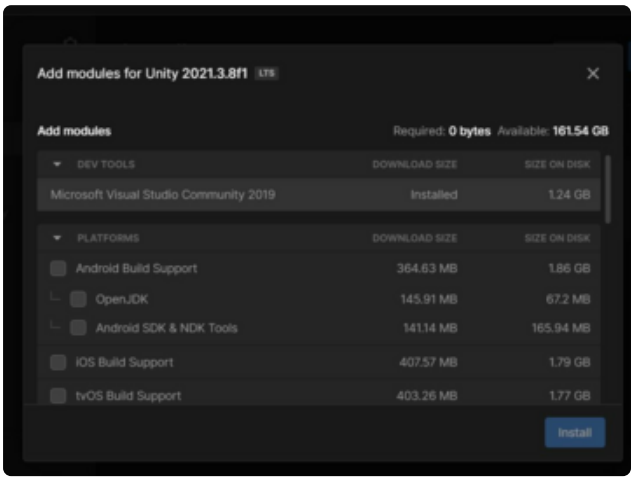


After Unity Hub installs, open it and select the **most recent Long Term Support version** of Unity to install.

Install Microsoft Visual Studio Community 2019

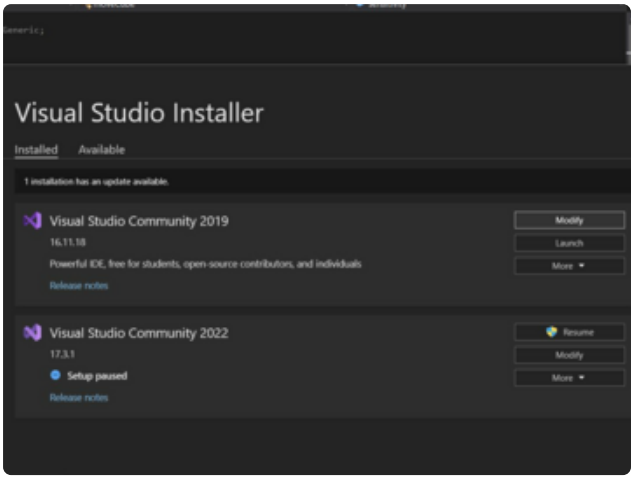


Go to your Unity installation in Unity Hub and click on the **settings cog wheel**. Then, click on **Add modules** to view the available modules for download.

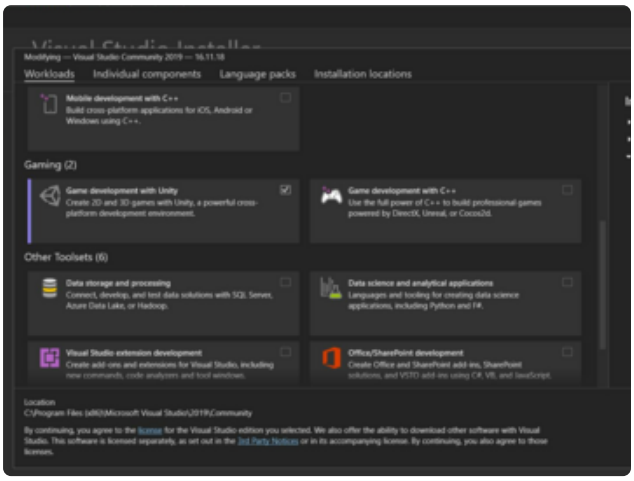


Under Dev Tools, **check off** Microsoft Visual Studio Community 2019 and then click **Install**. This will install Visual Studio, which you'll use to edit C# scripts in your Unity program.

Install Unity Workloads for Visual Studio 2019

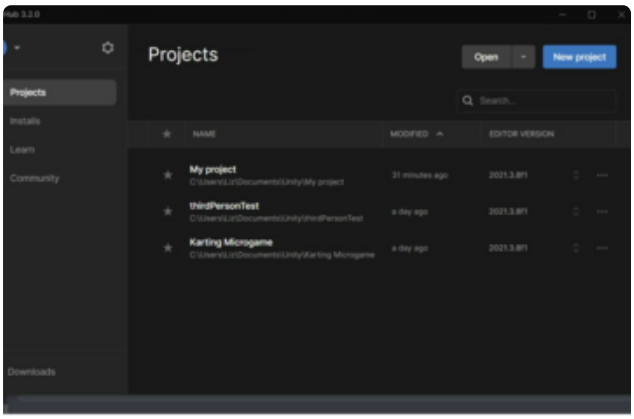


After installing Visual Studio, go to the Visual Studio Installer window and click on **Modify** next to the Visual Studio 2019 instance.

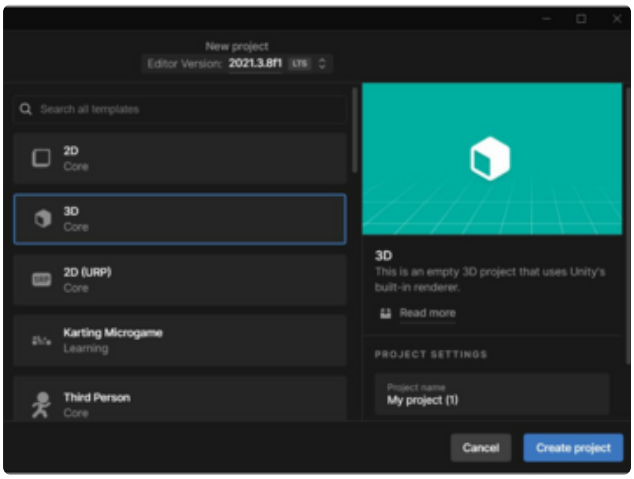


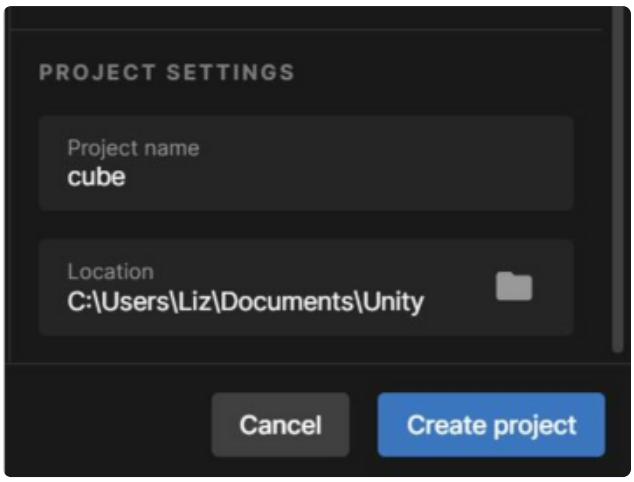
Under **Workloads**, navigate to the **Gaming** section. Check off the **Game development with Unity** option. Then, click **Install**.

Import a 3D Object Into Unity

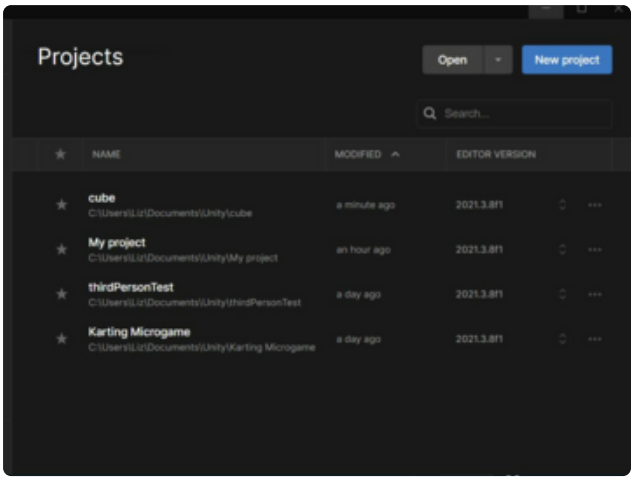


In the **Projects tab** of Unity Hub, click on the blue **New project** button.



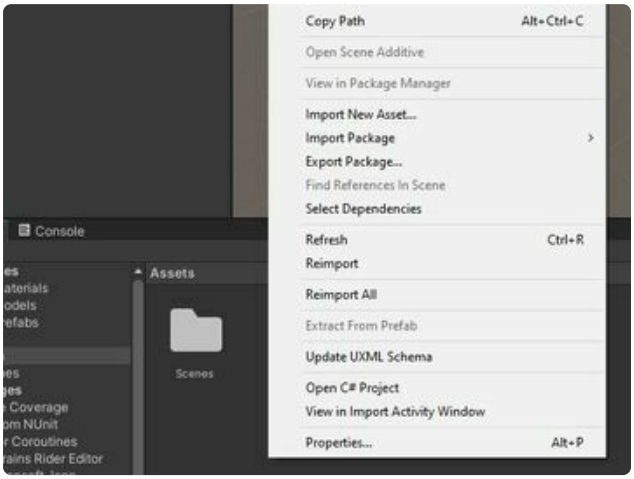


Select the **3D Core** project template. Then, **rename** your project and **select the project's storage location**. Click the blue **Create project** button to finish.



Click on **your project** to open it.

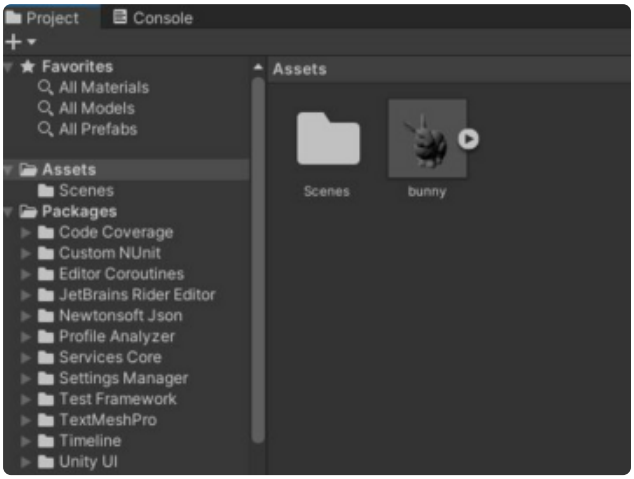
Import the 3D File

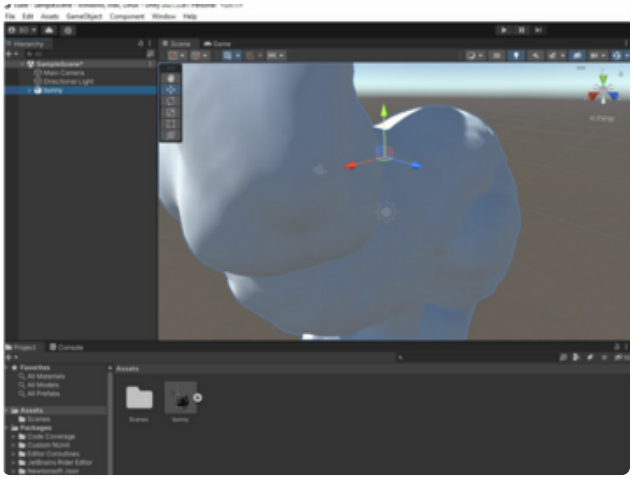


Unity has support for a few different 3D file formats (https://adafru.it/10Fv), including .OBJ and .DXF files.

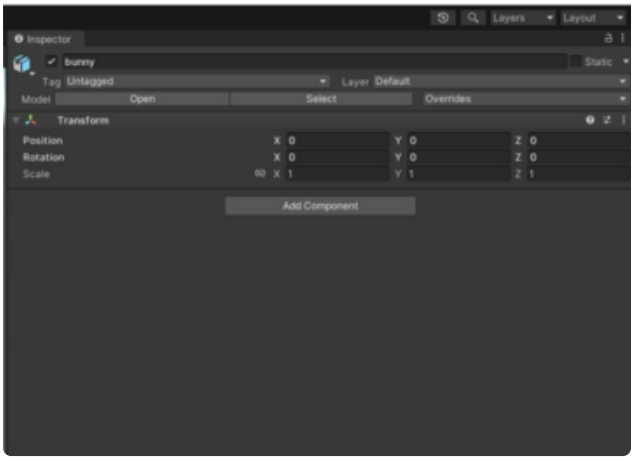
Right-click in the **Assets window** and click **Import New Asset...**

A file directory window will open. Navigate to your 3D file and click **Import**. Your 3D file will appear in the Assets window.



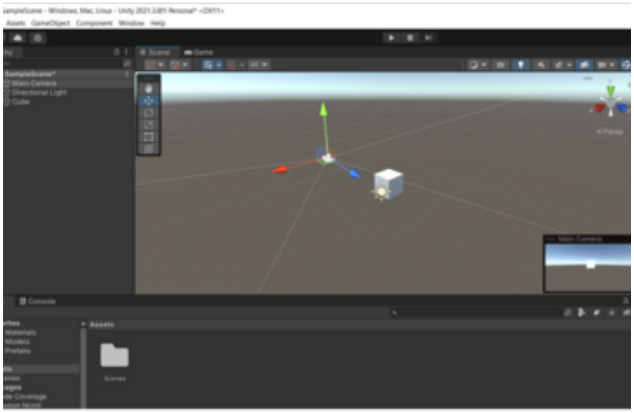


Drag and drop the 3D file from the **Assets window** into the **Hierarchy window**. You'll see your 3D object appear in the Scene window.



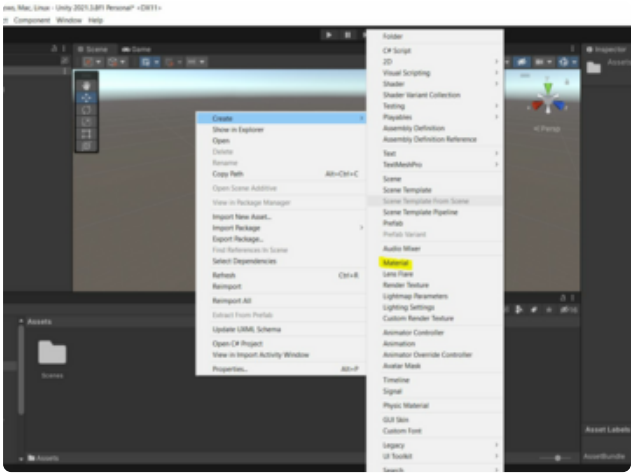
You can change the object's position, rotation and size with the Transform controls in its Inspector window.

Position the Camera

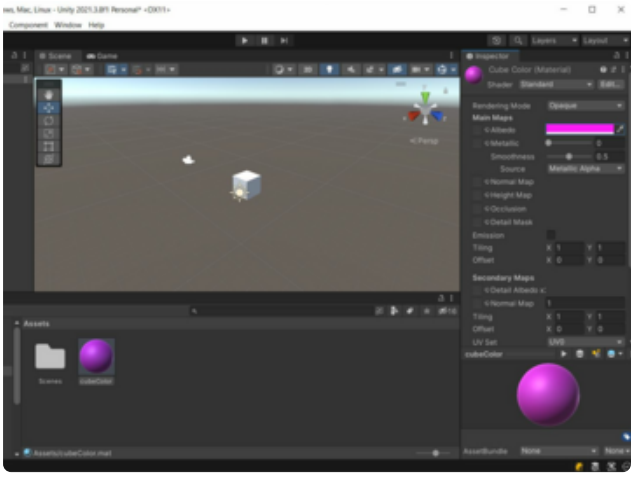


Select the **Main Camera** in the **Hierarchy window** and adjust its view using the **movement** and **rotation tools** to center the Scene. This affects your view while in gameplay mode.

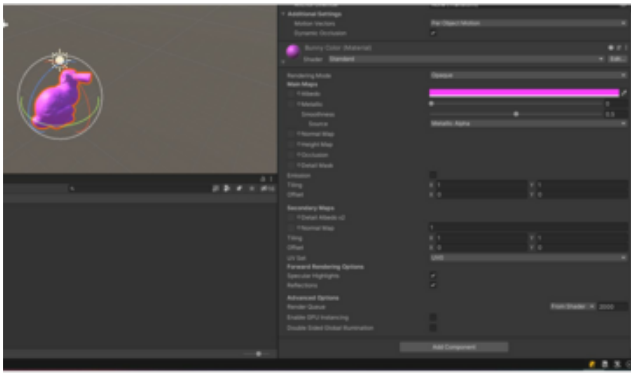
Change the Object's Color



Right-click in the **Assets window**. Select **Create** and **Material** to create a new color mask.



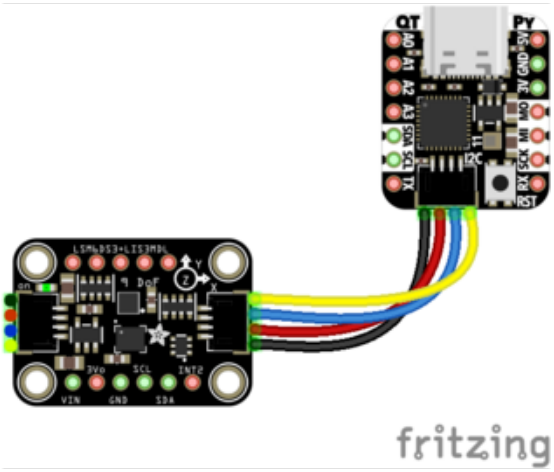
Click in the **Albedo** window to change the color of the Material. Other attributes of the Material can be edited here as well.

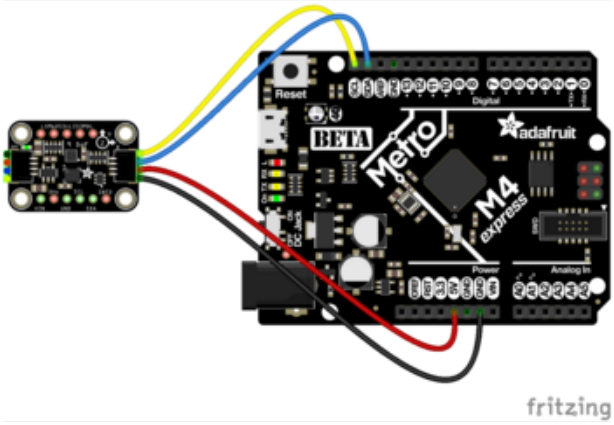


Drag the Material from the **Assets window** to the **object** in the **Hierarchy window**. The object will change appearance and you can edit the Material settings in the object's Inspector window.

Arduino Code

The LSM6DS3TR-C + LIS3MDL is plugged into the Arduino board's STEMMA port, if available, with a STEMMA QT cable. Otherwise, it can be wired to the board's pins.

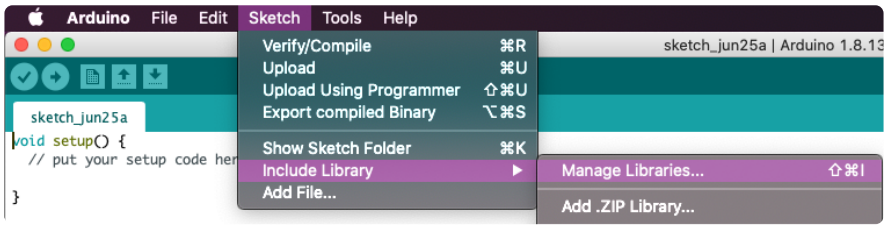




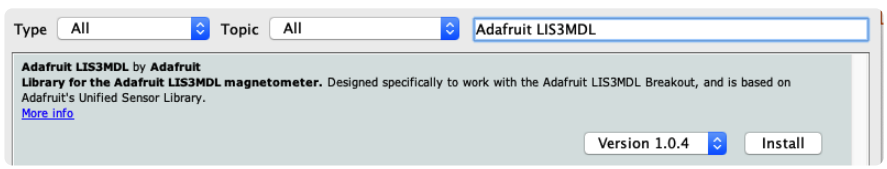
Board 5V to **sensor VIN (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

Library Installation

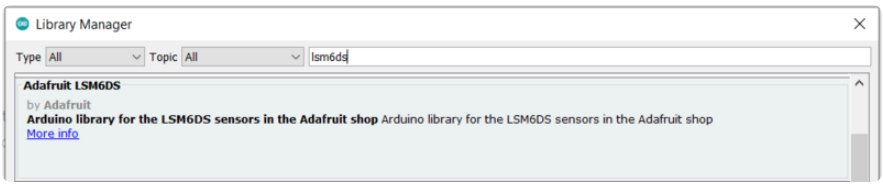
You can install the **Adafruit LIS3MDL** library and the **Adafruit LSM6DS** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit LIS3MDL**, and select the **Adafruit LIS3MDL** library:



Follow the same process for the **Adafruit LSM6DS** library.



If asked about dependencies, click "Install all".

If the "Dependencies" window does not come up, then you already have the dependencies installed.

Arduino Code - No Calibration

```
// SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_LSM6DS3TRC.h>

int x;
int y;
int z;

Adafruit_LSM6DS3TRC lsm6ds;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  if (!lsm6ds.begin_I2C()) {
    while (1) {
      delay(10);
    }
  }
  lsm6ds.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
  lsm6ds.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS);

  lsm6ds.setAccelDataRate(LSM6DS_RATE_104_HZ);
  lsm6ds.setGyroDataRate(LSM6DS_RATE_104_HZ);

}

void loop() {
  // put your main code here, to run repeatedly:

  sensors_event_t accel;
  sensors_event_t gyro;
```

```
  sensors_event_t temp;
  lsm6ds.getEvent(&accel, &gyro, &temp);

  x = map(accel.acceleration.x, -12, 11, 180, -180);
  y = map(accel.acceleration.y, -19, 20, -180, 180);
  z = map(accel.acceleration.z, -17, 15, 180, -180);

  Serial.print("|");
  Serial.print(x);
  Serial.print("|");
  Serial.print(y);
  Serial.print("|");
  Serial.print(z);
  Serial.println();
  delay(50);
}
```

Arduino Code - Calibrated

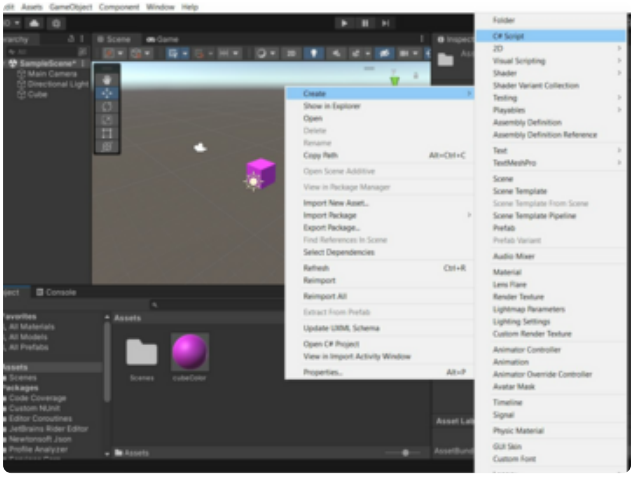
For this code to work, you need to follow the calibration steps outlined in the How to Fuse Motion Sensor Data into AHRS Orientation Learn Guide (https://adafru.it/LAr).

Make sure you calibrate your sensor before trying to run this code!

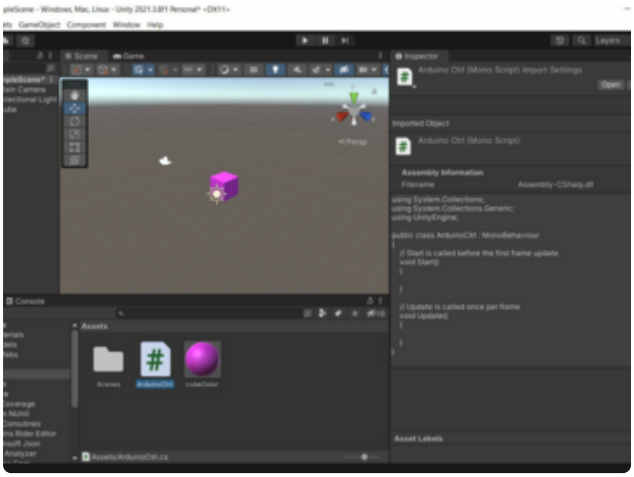
```
// SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_LSM6DS3TRC.h>
#include <Adafruit_AHRS.h>
#include <Adafruit_Sensor_Calibration.h>
#include <Adafruit_LIS3MDL.h>

Adafruit_Sensor *accelerometer, *gyroscope, *magnetometer;

Adafruit_LIS3MDL lis3mdl;
Adafruit_LSM6DS3TRC lsm6ds;

//Adafruit_NXPSensorFusion filter; // slowest
//Adafruit_Madgwick filter;  // faster than NXP
Adafruit_Mahony filter;

#if defined(ADAFRUIT_SENSOR_CALIBRATION_USE_EEPROM)
  Adafruit_Sensor_Calibration_EEPROM cal;
#else
  Adafruit_Sensor_Calibration_SDFat cal;
#endif

#define FILTER_UPDATE_RATE_HZ 100
#define PRINT_EVERY_N_UPDATES 10

uint32_t timestamp;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  if (!lsm6ds.begin_I2C()) {
    while (1) {
      delay(10);
    }
  }
```

```
  accelerometer = lsm6ds.getAccelerometerSensor();
  gyroscope = lsm6ds.getGyroSensor();
  magnetometer = &lis3mdl;

  lsm6ds.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
  lsm6ds.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS);
  lis3mdl.setRange(LIS3MDL_RANGE_4_GAUSS);

  lsm6ds.setAccelDataRate(LSM6DS_RATE_104_HZ);
  lsm6ds.setGyroDataRate(LSM6DS_RATE_104_HZ);
  lis3mdl.setDataRate(LIS3MDL_DATARATE_1000_HZ);
  lis3mdl.setPerformanceMode(LIS3MDL_MEDIUMMODE);
  lis3mdl.setOperationMode(LIS3MDL_CONTINUOUSMODE);

  filter.begin(FILTER_UPDATE_RATE_HZ);

}

void loop() {

  float roll, pitch, heading;
  float gx, gy, gz;
  static uint8_t counter = 0;

  sensors_event_t accel, gyro, mag;
  accelerometer->getEvent(&accel);
  gyroscope->getEvent(&gyro);
  magnetometer->getEvent(&mag);

  cal.calibrate(mag);
  cal.calibrate(accel);
  cal.calibrate(gyro);
  gx = gyro.gyro.x * SENSORS_RADS_TO_DPS;
  gy = gyro.gyro.y * SENSORS_RADS_TO_DPS;
  gz = gyro.gyro.z * SENSORS_RADS_TO_DPS;

  filter.update(gx, gy, gz,
                accel.acceleration.x, accel.acceleration.y, accel.acceleration.z,
                mag.magnetic.x, mag.magnetic.y, mag.magnetic.z);
  if (counter++ <= PRINT_EVERY_N_UPDATES) {
    return;
  }
  counter = 0;

  roll = filter.getRoll();
  pitch = filter.getPitch();
  heading = filter.getYaw();
  Serial.print("|");
  Serial.print(heading);
  Serial.print("|");
  Serial.print(pitch);
  Serial.print("|");
  Serial.print(roll);
  Serial.println();

}
```

Unity C# Script

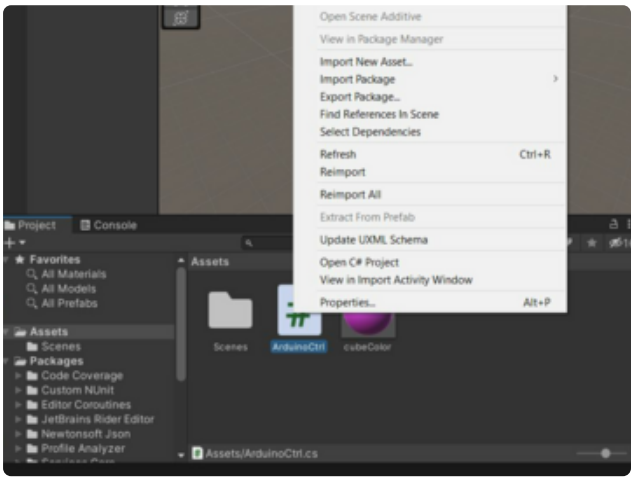


In the **Asset window**, **right-click** and select **Create**, followed by **C# Script**.

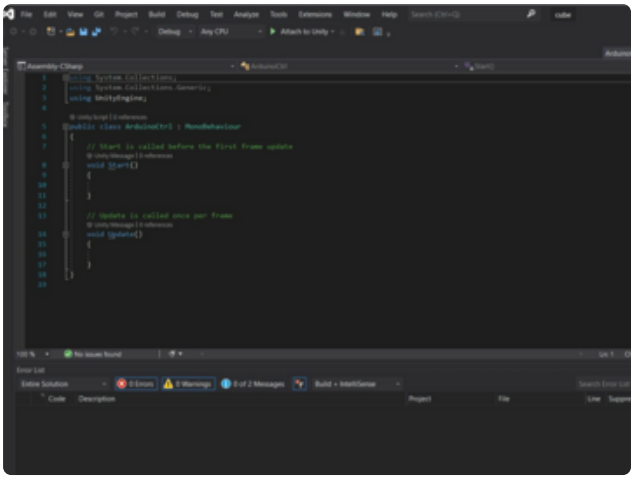


Rename the new C# Script **"ArduinoCtrl"**.

Edit the C# Script



Right-click the **ArduinoCtrl** C# Script and select **Open C# Project**. This opens the script in Visual Studio so that you can edit the code.



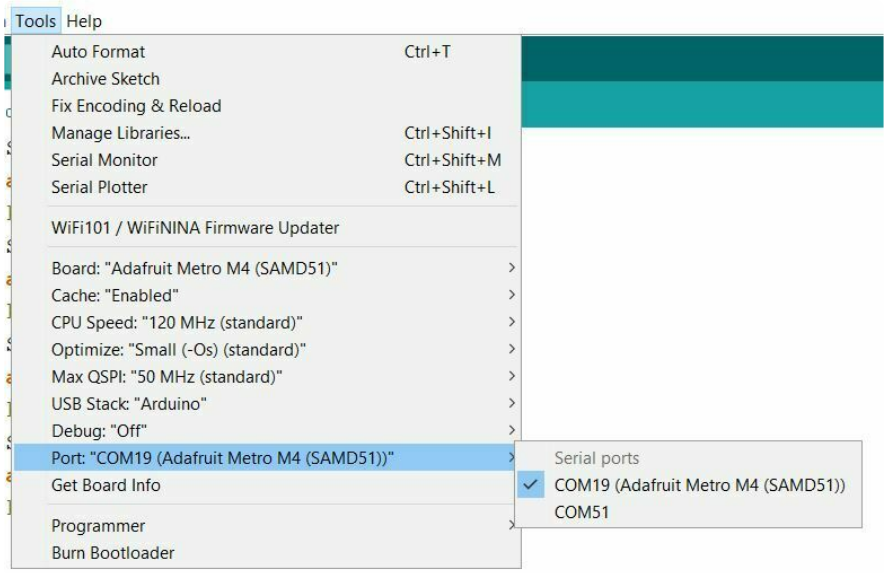
In the Visual Studio window, copy and paste the C# code from below into the **ArduinoCtrl** file. Then, **Save** the file.

```
// SPDX-FileCopyrightText: 2022 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

using System.Collections;
using System.Collections.Generic;
using System.IO.Ports;
using UnityEngine;
using UnityEngine.UI;

public class arduinoCtrl : MonoBehaviour
{
    // replace with your board's COM port
    SerialPort stream = new SerialPort("COM52", 9600);

    public Transform t;

    void Start()
    {
        stream.Open();
    }

    void Update()
    {
        Vector3 lastData = Vector3.zero;

        string UnSplitData = stream.ReadLine();
        print(UnSplitData);
        string[] SplitData = UnSplitData.Split('|');

        float AccX = float.Parse(SplitData[1]);
        float AccY = float.Parse(SplitData[2]);
        float AccZ = float.Parse(SplitData[3]);

        lastData = new Vector3(AccX, AccY, AccZ);

        t.transform.rotation = Quaternion.Slerp(t.transform.rotation,
Quaternion.Euler(lastData), Time.deltaTime * 2f);

    }
}
```

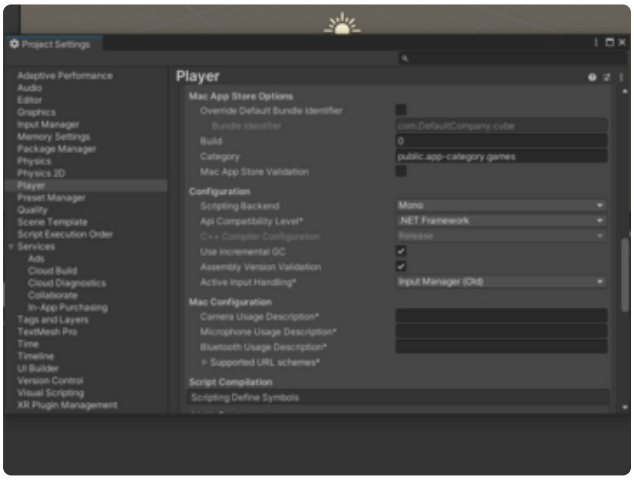
Update the Port Name

Update the `SerialPort` object with your Arduino board's port name. You can find this information in the Arduino IDE under Tools - Port.

```
stream = new SerialPort("COM19", 9600);
```

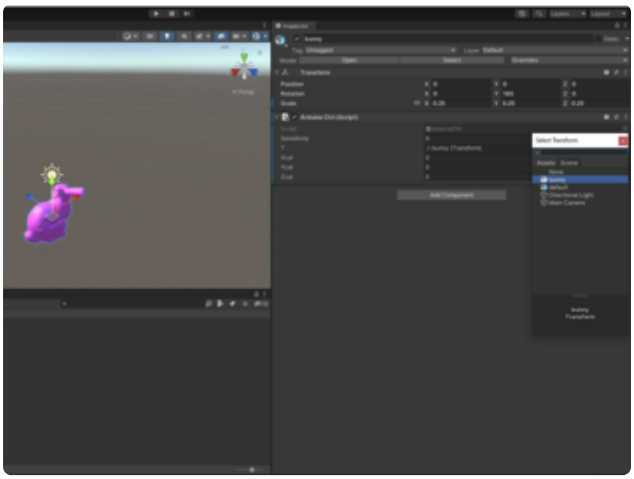


Update the API Compatibility Level



In the main Unity project window, click on **Edit** and then **Project Settings...** Click on **Player** and scroll down to the **Configuration settings** under Other Settings. Change the **Api Compatibility Level*** to **.NET Framework**. This allows all of the expected Unity Visual Studio libraries to work as expected in the C# script.

Apply the C# Script to the 3D Object



In the **Asset window**, **drag the ArduinoCtrl file** to the **3D object** in the **Hierarchy window**. You'll see the **ArduinoCtrl** script appear in the object's Inspector window.

In the **T box**, click on and **select your 3D object** from the list. This applies the script to the object to control its rotational movement.

How the Two Scripts Work Together

The Arduino code, in both the calibrated and uncalibrated version, send X, Y and Z coordinates from the 9 DoF sensor over serial. `|` are used as data delimiters and a line break is added after the Z data point.

```
//uncalibrated code loop

lsm6ds.getEvent(&amp;accel, &amp;gyro, &amp;temp);

x = map(accel.acceleration.x, -12, 11, 180, -180);
y = map(accel.acceleration.y, -19, 20, -180, 180);
z = map(accel.acceleration.z, -17, 15, 180, -180);

Serial.print("|");
Serial.print(x);
Serial.print("|");
Serial.print(y);
Serial.print("|");
Serial.print(z);
Serial.println();
delay(50);
```

```
//calibrated code loop

roll = filter.getRoll();
pitch = filter.getPitch();
heading = filter.getYaw();
Serial.print("|");
Serial.print(heading);
Serial.print("|");
Serial.print(pitch);
Serial.print("|");
Serial.print(roll);
Serial.println();
```

The C# Script

In the C# script, a serial port is opened and the incoming baud rate is set. You will change the serial port to match your board's port. `t` represents the object that will be rotated in Unity. By making it a `public` object, it will appear in Unity as an editable value.

```
SerialPort stream = new SerialPort("COM52", 9600);

public Transform t;
```

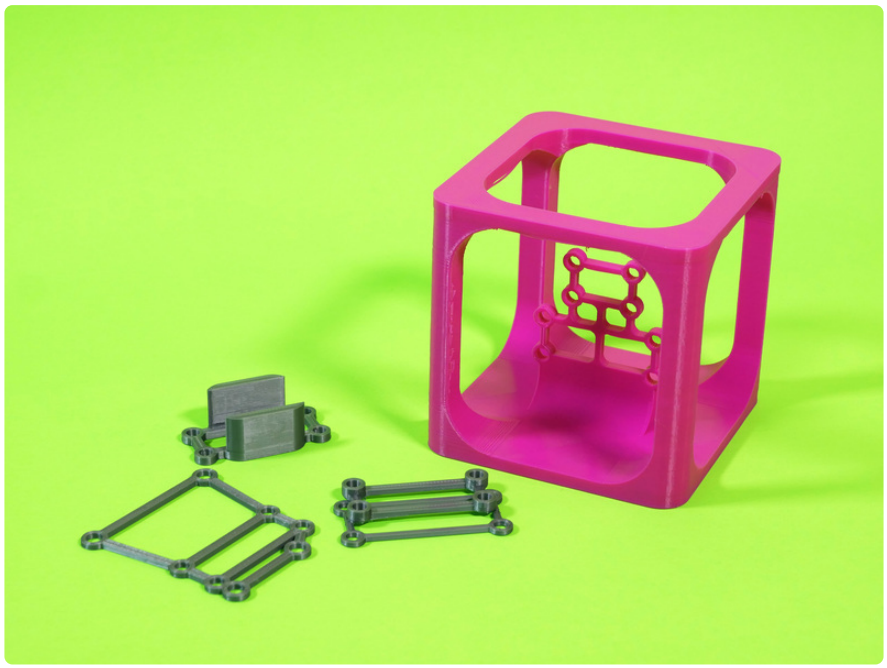
In the loop, the incoming serial data is parsed using the `|` delimiter. The X, Y and Z values are assigned to individual variables and converted to `float`s.

```
string UnSplitData = stream.ReadLine();
print(UnSplitData);
string[] SplitData = UnSplitData.Split('|');

float AccX = float.Parse(SplitData[1]);
float AccY = float.Parse(SplitData[2]);
float AccZ = float.Parse(SplitData[3]);
```

The X, Y and Z data are packed into a `Vector3` object. Finally, the data is sent to the Unity and applied to the destination 3D object using a `rotation` with `Quaternion.Slerp()` to apply the X, Y and Z coordinates as Euler measurements.

```
lastData = new Vector3(AccX, AccY, AccZ);

t.transform.rotation = Quaternion.Slerp(t.transform.rotation,
Quaternion.Euler(lastData), Time.deltaTime * 2f);
```

Optional 3D Printing



You can use the 9 DoF sensor without an enclosure, but mounting it in an enclosure will give you steadier readings and a better idea of the physical orientation of the sensor. It may be assembled in a 3D printed cube, described below. By default, the cube has mounting holes for a Metro Mini V2. However, there are mounting plates for Metro, Feather and QT Py format boards as well.

The STL files can be downloaded directly here or from Thingiverse.

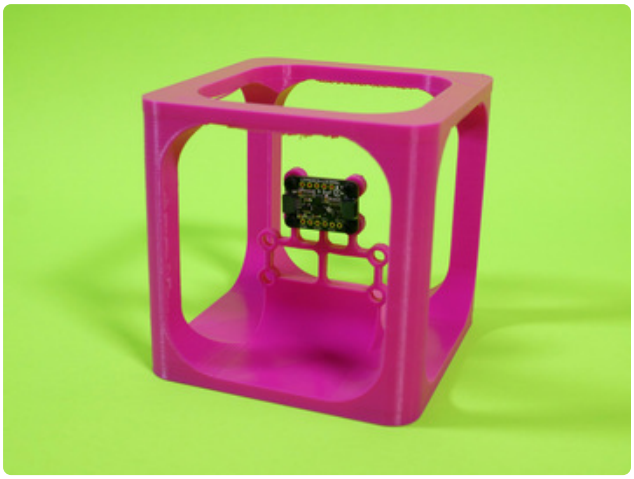
unityCube.zip

https://adafru.it/10Fw

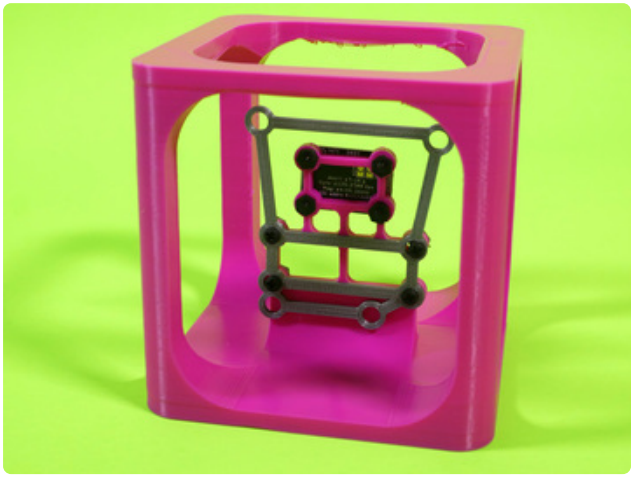
Thingiverse download

https://adafru.it/10Fx

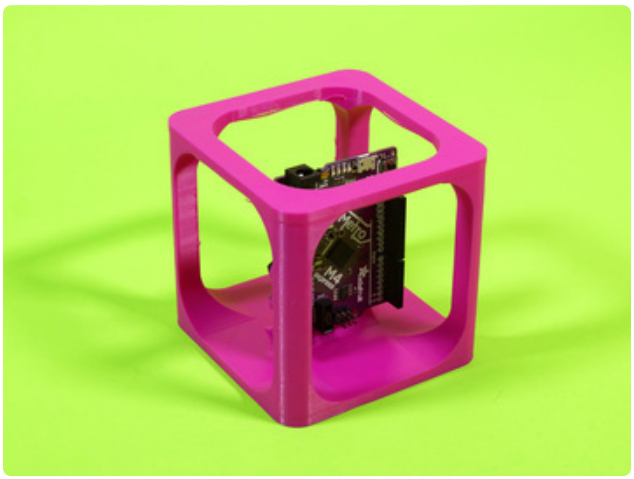
Assembly



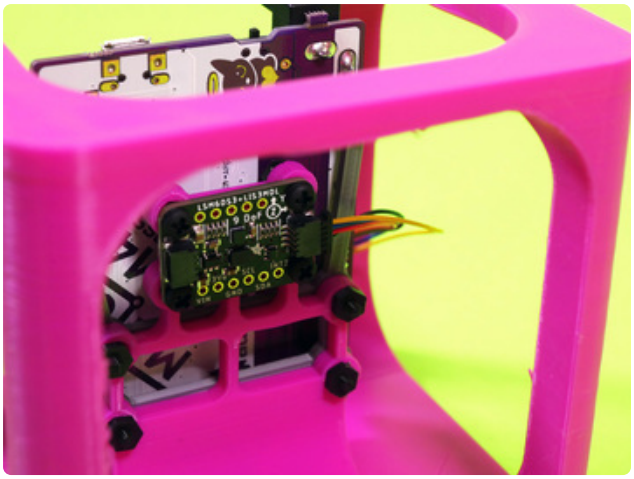
Mount the 9 DoF sensor board to the cube using M2.5 screws and nuts.



If you are using a mounting plate, mount the plate to the cube using the Metro Mini's mounting holes with M2.5 screws and nuts.



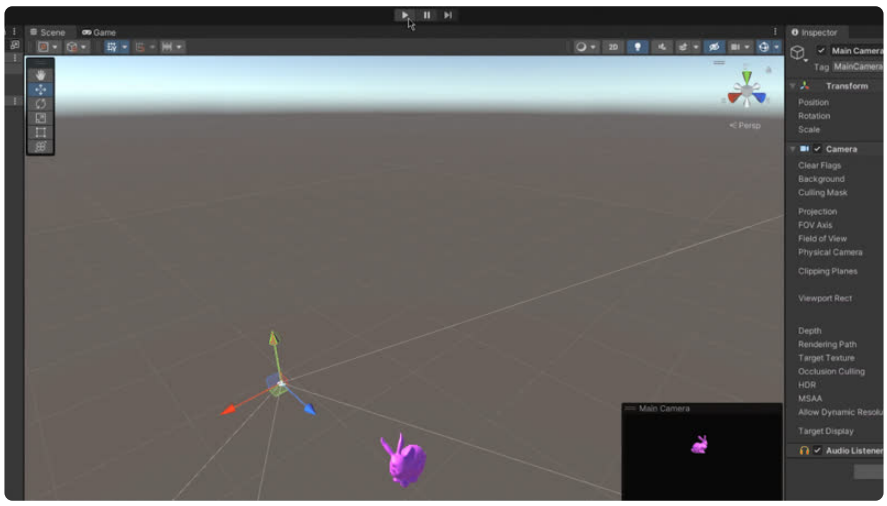
Mount your board using M2.5 screws and nuts.



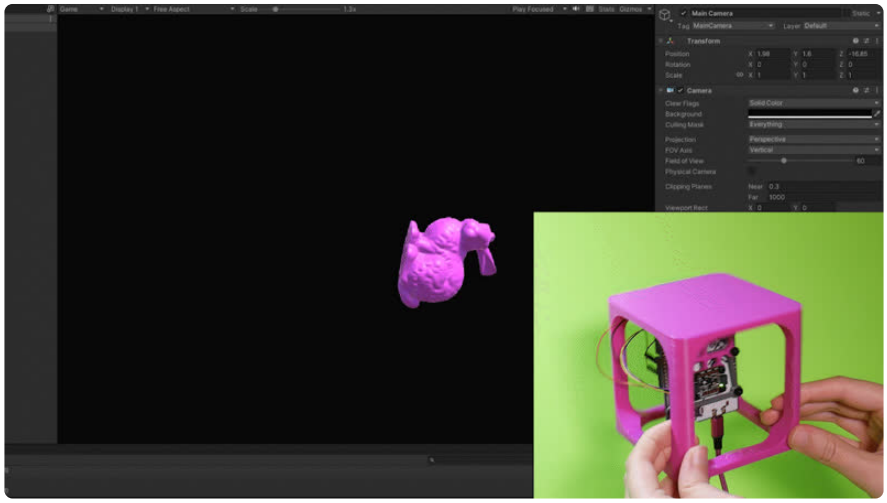
Plug the 9 DoF STEMMA board into the Arduino board using a STEMMA QT cable.

Control the Object with the 9 DoF Sensor

Upload the sketch to your chosen board using the Arduino IDE. Then, press the **Play button** at the top of the Unity window to enter Gameplay mode.



Twist and turn the 9 DoF sensor around and you should see your 3D object spin around the Scene window in unison with the board.



Going Further

You could map the incoming sensor data to build a controller for a game built in Unity. You could also experiment with different sensors to control objects in different ways. For example, you could use a light sensor to change the lighting of a scene depending on the ambient light in your space.