TFT Gizmo Turtle
Created by Ruiz Brothers

https://learn.adafruit.com/tft-gizmo-turtle

Last updated on 2023-09-30 12:40:04 PM EDT
## Table of Contents

### Overview
- Turtle GFX Bluefruit
- Fridge Magnet
- TFT Gizmo
- Prerequisite Guides
- Parts

### CircuitPython on Circuit Playground Bluefruit
- Install or Update CircuitPython

### Turtle Graphics on Gizmo
- Libraries
- The Mu Editor
- Turtle Graphics Code
- Square
- Asterisk
- Circle
- Circle Petals
- Star
- Rainbow Benzene
- Parabolas
- Sierpinski Triangle
- Hilbert Curve
- Koch Snowflake
- Christmas Tree
- Snowflakes
- Where to Find More

### 3D Printing
- 3D Parts
- CAD Assembly
- Design Source Files
- Slicing Parts
- Install Magnets

### Assembly
- Installing Circuit Playground Bluefruit and TFT Gizmo
- Install the Battery
- Installing the Circuit Playground Bluefruit
- Secure Hardware
- Plug in Battery
- Turning It Off
- Install PCB to Turtle
- Install a lens for TFT Gizmo Display
- Install Turtle Shell
- Snaps Fit the Turtle Shell
- Accessible USB
- TFT Gizmo Turtle
Overview

Turtle GFX Bluefruit
3D print and program your own turtle graphics turtle with CircuitPython! Use the TFT Gizmo and Circuit Playground Bluefruit to display graphics and images.

This project uses the Turtle Graphics library for Adafruit’s CircuitPython. This was inspired by the Logo programming language. With CircuitPython, you can program your own turtle graphics and display them on Adafruit’s TFT Gizmo.

Fridge Magnet
The case is 3D printed and designed to house the components. The parts were designed to snap fit so the assembly is pretty easy to put together.

It’s got little neodymium magnets on the bottom of the flippers. You can stick it to your refrigerator or any metal surface.

TFT Gizmo
The TFT Gizmo is specifically designed to work with the Circuit Playground Express and Circuit Playground Bluefruit. The TFT Gizmo features a 240x240 IPS screen for displaying graphics and images. The two boards are bolted together using included hardware.
Prerequisite Guides

Read through the guides linked below for more information on Circuit Playground Bluefruit, CircuitPython, and the Turtle Graphics Library.

- Introducing Circuit Playground Bluefruit
- Turtle Graphics for CircuitPython
- Circuit Python Essentials Guide

Parts

Components used to build this project.

Circuit Playground Bluefruit
TFT Gizmo
350mAh battery
40mm convex plastic lens
Fully reversible USB cable
Neodymium Magnets

Circuit Playground Bluefruit - Bluetooth Low Energy
Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...
https://www.adafruit.com/product/4333
Circuit Playground TFT Gizmo - Bolt-on Display + Audio Amplifier
Extend and expand your Circuit Playground projects with a bolt on TFT Gizmo that lets you add a lovely color display in a sturdy and reliable fashion. This PCB looks just like a round...
https://www.adafruit.com/product/4367

Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4237

Convex Plastic Lens with Edge
The eyes have it! Add this little lens to make a big expression with our Spooky Eyes demo for microcontrollers or Raspberry Pi. These are plastic lenses, with brilliant clarity and a...
https://www.adafruit.com/product/3917

Fully Reversible Pink/Purple USB A to micro B Cable - 1m long
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...
https://www.adafruit.com/product/4111
CircuitPython on Circuit Playground Bluefruit

Install or Update CircuitPython

Follow this quick step-by-step to install or update CircuitPython on your Circuit Playground Bluefruit.

Download the latest version of CircuitPython for this board via circuitpython.org

Click the link above and download the latest UF2 file

Download and save it to your Desktop (or wherever is handy)
Plug your Circuit Playground Bluefruit into your computer using a known-good data-capable USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the small Reset button in the middle of the CPB (indicated by the red arrow in the image). The ten NeoPixel LEDs will all turn red, and then will all turn green. If they turn all red and stay red, check the USB cable, try another USB port, etc. The little red LED next to the USB connector will pulse red - this is ok!

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

(If double-clicking doesn't do it, try a single-click!)
You will see a new disk drive appear called CPLAYBTBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to CPLAYBTBOOT.

The LEDs will turn red. Then, the CPLAYBTBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

©Adafruit Industries
Turtle Graphics on Gizmo

Libraries
Now we'll install the libraries that we need to use Turtle graphics on the Circuit Playground Bluefruit with TFT Gizmo.

Click this link to go to the circuitpython.org Libraries page. Download the latest version of the Bundle library .zip file that matches the version of CircuitPython you're using on the board.

Uncompress the .zip file and then copy the following directory and three library .mpy files to the lib directory of the CIRCUITPY drive:

- adafruit_bus_device
- adafruit_logging.mpy
- adafruit_st7789.mpy
- adafruit_turtle.mpy

The Mu Editor
Adafruit recommends using the free program Mu to edit your CircuitPython programs and save them on your Circuit Playground Bluefruit. You can use any text editor, but Mu has some handy features.

See this page on the Circuit Playground Bluefruit guide on the steps used to install Mu.
Turtle Graphics Code

Let's start of with a simple bit of code. It will draw a square on the screen.

Copy the code below and then paste it into a new document in Mu. Save it to the CIR CUITPY drive as code.py.

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
# # SPDX-License-Identifier: MIT

#Turtle Gizmo Square
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |=======================================

turtle.pendown()

turtle.forward(80)
turtle.right(90)

turtle.forward(80)
turtle.right(90)

turtle.forward(80)
turtle.right(90)

turtle.forward(80)
turtle.right(90)
```
Square
After the setup code imports libraries and configures some pins for the display, the main body of the code runs. Watch the screen on your TFT Gizmo as the board restarts and runs the code.

First, it sets the pen down with the `turtle.pendown()` command. (Later, you'll learn to lift the pen with `turtle.penup()` so you don't always draw a line then the turtle moves!)

The turtle starts out at the origin point -- \(0,0\) on the x- and y-axis respectively. It then moves forward 80 pixels, drawing a line as it goes using the `turtle.forward(80)` command. This draws the top of the square.

Next, the turtle turns right using the `turtle.right(90)` command.

This same pair of commands -- `forward()` and `right()` -- run three more times in order to complete the square.
Since there are some repetitive steps in there, we can tidy up the code by iterating through a loop four times like this:

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
# SPDX-License-Identifier: MIT

# Turtle Gizmo Square Loop
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import turtle

displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
            backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |========================================

turtle.pendown()
for _ in range(4):
    turtle.forward(80)
    turtle.right(90)
while True:
    pass
```

Try out that code -- it's much shorter, but you'll see the same square drawn on the screen!
Now that we can loop through a command many times, let's add another two commands -- `turtle.back()` and `turtle.left()`

We can move forward and backward to create a line, shift the angle 18 degrees to the left, and do it all again. After twenty iterations of this we have a nice star pattern!

With a small change to the code (see the comment on line 18) we can turn this into an iris pattern!

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

#Turtle Gizmo Asterix
#==| Turtle Gizmo Setup start |=======================================================================================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import turtle

displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                 backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |=======================================================================================================

turtle.pendown()
for _ in range(20):
    turtle.forward(80)
    turtle.back(80) #try changing this to 70 for an iris effect
turtle.left(18)

while True:
    pass
```
The built-in command `turtle.circle()` makes it very simple to draw a triangle.
No, just kidding! It draws a circle!

Try out the example here, and then use some different numbers for the circle radius instead of the 118 here.

```python
turtle.penup()
turtle.right(90)
turtle.forward(118)
turtle.left(90)
turtle.pendown()
turtle.circle(118)  # radius of the circle
```
while True:
    pass

Circle Petals

Let's get a little fancy. We will now introduce a few more techniques, including the `turtle.pencolor()` command to change the color of the line that our turtle draws, as well as iterating through multiple loops with changing variables.

This will allow us to change the size of a series of five circles, and repeat the pattern four times.

```
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT
#
# Turtle Circle Petals
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                 backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
```
colors = [Color.YELLOW, Color.GREEN]
for _ in range(4):
    for i in range (5):
        turtle.pencolor(colors[i % 2])
        turtle.pendown()
        turtle.circle(60 - (i*10) )
        turtle.penup()
        turtle.right(90)

while True:
    pass

---

Star
Here's a super stylish, 1970's inspired star! Note how the length of each forward command is increased as the value of \( i \) goes up during each of the 26 iterations of the loop.

---

# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Turtle Gizmo Star Fancy
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio

©Adafruit Industries
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |=========================================
turtle.pendown()
turtle.pencolor(Color.BLUE)
for i in range(26):
    turtle.fd(i*10)
    turtle.rt(144)
while True:
    pass

Rainbow Benzene
Here's a classic, beautiful design. This hexagonal shape twists with each iteration, and every side of each perceived hexagon is a different color of the rainbow!
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                   backlight_PIN=board.A3, rotation=180)
turtle = turtle(display)

benzsize = min(display.width, display.height) * 0.5

print("Turtle time! Lets draw a rainbow benzene")

colors = (Color.RED, Color.ORANGE, Color.YELLOW, Color.GREEN, Color.BLUE,
          Color.PURPLE)

turtle.pendown()
start = turtle.pos()

for x in range(benzsize):
    turtle.pencolor(colors[x%6])
    turtle.forward(x)
    turtle.left(59)

while True:
    pass
Parabolas
This lovely parabolic curve set creates the illusion of curvature. It is in fact made up entirely of straight line segments, just as you would with a piece of string art and some pins pushed into a canvas.

It also introduces the `turtle.dot()` command, which drops down a dot of your desired radius wherever the turtle is. In this case, the turtle is moved with `turtle.goto()` This command allows the turtle to instantly head to a coordinate point on the screen rather than needing explicit angle rotations specified along with a "forward" command.

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Turtle Gizmo Parabolic Jack
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                 backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |========================================

print("Draw parabolas using straight line segments!")

def vert(x, y, size):
    turtle.goto(x, y)
    turtle.dot(size)

turtle.penup()
turtle.pencolor(Color.GREEN)

vert(0, 0, 7)
vert(0, 100, 7)
vert(100, 0, 7)
vert(0, -100, 7)
vert(-100, 0, 7)

x_quad=[10, 10, -10, -10]
y_quad=[10, -10, -10, 10]

for q in range(4):
    for i in range(0,11):
        x_from = 0
```
```python
y_from = (10-i) * y_quad[q]
x_to = i * x_quad[q]
y_to = 0
turtle.penup()
turtle.goto(x_from,y_from)
turtle.pendown()
turtle.goto(x_to,y_to)
turtle.home()

while True:
    pass
```

Sierpinski Triangle

Time to get fractal! These are recursive, equilateral triangles as described by Waclaw Sierpinski in 1915! Read more about them here.(

---

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT
#
# Turtle Gizmo Sierpinski Triangle
#===| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
```
from adafruit_turtle import turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80, backlight_pin=board.A3, rotation=180)
turtle = turtle(display)

# ==| Turtle Gizmo Setup end |=================================================

def getMid(p1, p2):
    return ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2) # find midpoint

def triangle(points, depth):
    turtle.penup()
    turtle.goto(points[0][0], points[0][1])
turtle.pendown()
    turtle.goto(points[1][0], points[1][1])
turtle.goto(points[2][0], points[2][1])
turtle.goto(points[0][0], points[0][1])

    if depth > 0:
        triangle([points[0],
            getMid(points[0], points[1]),
            getMid(points[0], points[2])],
            depth-1)
        triangle([points[1],
            getMid(points[0], points[1]),
            getMid(points[1], points[2])],
            depth-1)
        triangle([points[2],
            getMid(points[2], points[1]),
            getMid(points[0], points[2])],
            depth-1)

big = min(display.width/2, display.height/2)
little = big / 1.4
seed_points = [ [-big, -little], [0, big], [big, -little] ] # size of triangle
triangle(seed_points, 4)

while True:
    pass
Hilbert Curve
Here's another interesting fractal pattern, called a Hilbert curve.

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
# SPDX-License-Identifier: MIT

# Turtle Gizmo Hilbert
### Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
    backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
### Turtle Gizmo Setup end |========================================

def hilbert2(step, rule, angle, depth, t):
    if depth > 0:
        a = lambda: hilbert2(step, "a", angle, depth - 1, t)
        b = lambda: hilbert2(step, "b", angle, depth - 1, t)
        left = lambda: t.left(angle)
        right = lambda: t.right(angle)
        forward = lambda: t.forward(step)
        if rule == "a":
            left()
            b()
            forward()
            right()
            a()
            forward()
            a()
            right()
            forward()
        b()
        left()
        if rule == "b":
            right()
            a()
            forward()
            left()
            b()
            forward()
            b()
            left()
            forward()
```

©Adafruit Industries
turtle.penup()
turtle.goto(-108, -108)
turtle.pendown()
turtle.pencolor(Color.PURPLE)
hilbertZ(7, "a", 90, 5, turtle)

while True:
    pass

Koch Snowflake
This is a third generation fractal Koch snowflake.

# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
# # SPDX-License-Identifier: MIT

# Turtle Gizmo Koch Snowflake
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
backlight_pin=board.A3, rotation=180)
turtle = turtle(display)

#==| Turtle Gizmo Setup end |=========================================

def f(side_length, depth, generation):
    if depth == 0:
        side = turtle.forward(side_length)
    else:
        side = lambda: f(side_length / 3, depth - 1, generation + 1)
        side()
        turtle.left(60)
        side()
        turtle.right(120)
        side()
        turtle.left(60)
        side()

turtle.penup()
turtle.goto(-99, 56)
turtle.pendown()

num_generations = 3
top_side = lambda: f(218, num_generations, 0)
top_side()
turtle.right(120)
top_side()
turtle.right(120)
top_side()

while True:
    pass
Christmas Tree
This lovely Christmas tree was adapted from this program written by Keith Randall.

```python
# SPDX-FileCopyrightText: 2019 John Edgar Park for Adafruit Industries
# SPDX-License-Identifier: MIT

# Turtle Gizmo Christmas Tree
#==| Turtle Gizmo Setup start |========================================
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import Color, turtle
displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                   backlight_pin=board.A3, rotation=180)
turtle = turtle(display)
#==| Turtle Gizmo Setup end |========================================

# Fractal Christmas Tree:
# https://codegolf.stackexchange.com/questions/15860/make-a-scalable-christmas-tree
# by Keith Randall
n = 42  # input value for scaling the tree. note: ornaments don't scale
turtle.goto(0, -20)

#star
turtle.left(90)
turtle.forward(3*n)
turtle.pencolor(Color.YELLOW)
turtle.left(126)
turtle.pendown()
for _ in range(5):
    turtle.forward(n/5)
    turtle.right(144)
    turtle.forward(n/5)
    turtle.left(72)
    turtle.right(126)

#tree
turtle.pencolor(Color.GREEN)
turtle.back(n*4.8)

def tree(d, s):
    if d <= 0:
        return
    turtle.forward(s)
    tree(d-1, s*.8)
    turtle.right(120)
    tree(d-3, s*.5)
    turtle.right(120)
```

©Adafruit Industries
tree(d-3, s*.5)
turtle.right(120)
turtle.back(s)
turtle.pendown()
turtle.pencolor(Color.GREEN)
tree(15, n)
turtle.back(n/2)

#ornaments
def ornament(x, y):
    turtle.penup()
    turtle.goto(x, y)
    turtle.pencolor(Color.RED)
    turtle.pendown()
    turtle.dot(7)
    turtle.penup()

orn_pnts=[(5, 60), (-7, 40), (10, 20), (-15, 0), (25, -20),
          (-27, -30), (7, -33), (40, -60), (-9, -63),
          (-50, -88), (62, -97)]

for j in range(len(orn_pnts)):
    ornament(orn_pnts[j][0], orn_pnts[j][1])

turtle.penup()
turtle.goto(0, -120)
while True:
    pass

Snowflakes

This example will draw lots of beautiful snowflake-style patterns! If you want to get technical about the number of sides in your flakes, make some adjustments to this line:

draw_flake(randint(5, 8)) #adjust number of arms here
import time
from random import randint
import board
import busio
import displayio
from adafruit_st7789 import ST7789
from adafruit_turtle import turtle

displayio.release_displays()
spi = busio.SPI(board.SCL, MOSI=board.SDA)
display_bus = displayio.FourWire(spi, command=board.TX, chip_select=board.RX)
display = ST7789(display_bus, width=240, height=240, rowstart=80,
                  backlight_pin=board.A3, rotation=180)
turtle = turtle(display)

def draw_arm():
    turtle.pendown()
    for angle, length in arm_data:
        turtle.forward(length)
        turtle.left(angle)
        turtle.forward(length)
        turtle.backward(length)
        turtle.right(2*angle)
        turtle.forward(length)
        turtle.backward(length)
        turtle.left(angle)
    turtle.penup()

def draw_flake(arms):
    turtle.penup()
    turtle.home()
    turtle.clear()
    angle = 0
delta_angle = 360 // arms
    for _ in range(arms):
        turtle.home()
        turtle.setheading(angle)
        draw_arm()
        angle += delta_angle
    turtle.penup()
    turtle.home()

while True:
    arm_data = [(randint(30, 80), randint(10, 40)) for _ in range(5)]
    draw_flake(randint(5, 8)) #adjust number of arms here
    time.sleep(5)
Where to Find More

You can find much more information about turtle graphics as well as example scripts through an Internet search of "turtle python graphics examples" or some variation. Since turtle graphics are mainly in terms of moving and turning they can usually be converted to CircuitPython and the adafruit_turtle library.

Check out these example sites:

- Michael0x2a Examples
- Geeks for Geeks Examples
- Vivax Solutions Examples
3D Printing

3D Parts
STL files for 3D printing are oriented to print "as-is" on FDM style machines. Original design source may be downloaded using the links below.

CAD Assembly
This animation shows how the parts snap are put together. The TFT Gizmo is bolted to the Circuit Playground Bluefruit with hardware screws. The PCBs are fitted on top of the turtle body while the turtle shell snap fits on top. A plastic convex lens can be used to magnify the display.

Download CAD files
Download STL files

Design Source Files
The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, SAT and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from our Adafruit CAD parts GitHub Repo ().
Slicing Parts

The parts were sliced using CURA using the slice settings below. The parts were 3D printed using PLA filament.

- PLA filament 220c extruder
- 0.2 layer height
- 10% gyroid infill
- 60mm/s print speed
- 60c heated bed

Install Magnets

Neodymium magnets can be press fitted into the bottoms of the flippers. If the magnets are too tight to fit into, a deburring tool can be used to loosen up the tolerance.

Magnet Dimensions: 1/4" dia. x 1/8" thick

The small slits can be used to pop out the magnets. Use the tip of a small flathead screwdriver to pry them out of the holes.

Assembly

Installing Circuit Playground Bluefruit and TFT Gizmo

Use the included hardware to secure the TFT Gizmo to the Circuit Playground Bluefruit. Optionally use a 350mah battery to make this a portable project.
Install the Battery
Place the battery on top of the TFT Gizmo and orient it so the battery cable is facing the pico connector.

Installing the Circuit Playground Bluefruit
Orient the Circuit Playground Bluefruit so the USB port is facing up with the 12o'clock label on the TFT Gizmo.

Secure Hardware
Place the CPB over the TFT Gizmo. Use a screwdriver to install the included screws. Tightly fasten all 12 screws onto the pads.
Plug in Battery
Pull out the battery cable from in between the CPB and TFT Gizmo. Then, plug it into the battery port.

Turning It Off
The CPB will power on once the battery is connected. To turn off the circuit, you’ll need to disconnect the battery.

Install PCB to Turtle
Orient the PCBs for the USB port is facing the bottom rear. Place the PCBs over the body of the turtle. Press down to snap fit onto the built in standoffs on the turtle.
Install a lens for TFT Gizmo Display
Optionally use a plastic convex lens to magnify the display. Place it in the center of the screen. The lens is not glued to the parts, it is simply held in place by the turtle shell.

Install Turtle Shell
Place the turtle shell over the display. The opening in the center of the shell will keep the lens in place.

Snaps Fit the Turtle Shell
Hold the assembly together and turn the turtle over on its side. Press fit the two parts together so the snaps are fitted over the nubs on the inside of the shell.
Accessible USB
The USB port is accessible on the bottom rear of the turtle. The rest of the components on the Circuit Playground Bluefruit are also accessible.

TFT Gizmo Turtle
And there you have it, that's how to assemble our 3D Printed Turtle. This was a cute project to work on and I hope it inspires you to try out Adafruit’s CircuitPython.

There's tons of other great projects you can make. Check out the BLE image transfer guide () for beaming pictures to the TFT Gizmo using a mobile device!