Circuit Playground TFT Gizmo Snow Globe

Created by Carter Nelson

https://learn.adafruit.com/circuit-playground-tft-gizmo-snow-globe

Last updated on 2024-06-03 02:57:17 PM EDT

Table of Contents

Overview

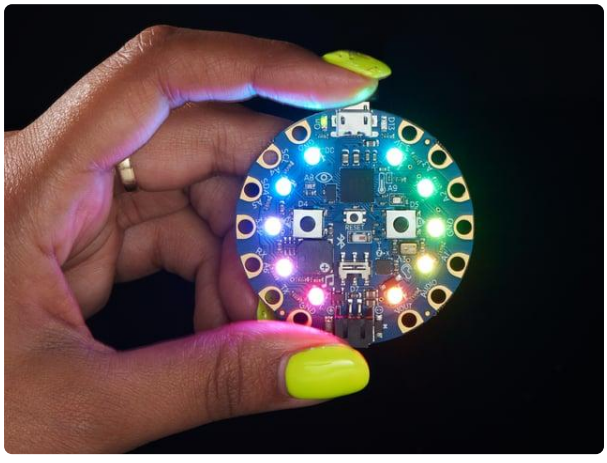
Ah, the classic snow globe. That clear orb full of mystery liquid, some sort of flake material, and typically a diorama depicting some scene. Shake it up and watch the snow fall.

In this guide we'll show you how to use a Circuit Playground Bluefruit (http://adafru.it/4333) along with a TFT Gizmo (http://adafru.it/4367) to create a digital version of a snow globe. We'll display customizable snow globe goodness on the TFT. You'll be able to set a custom background, create custom snow flakes, and of course shake to clear the snow globe and start it all over.

All coded in CircuitPython (https://adafru.it/EFq).

Let's get started.

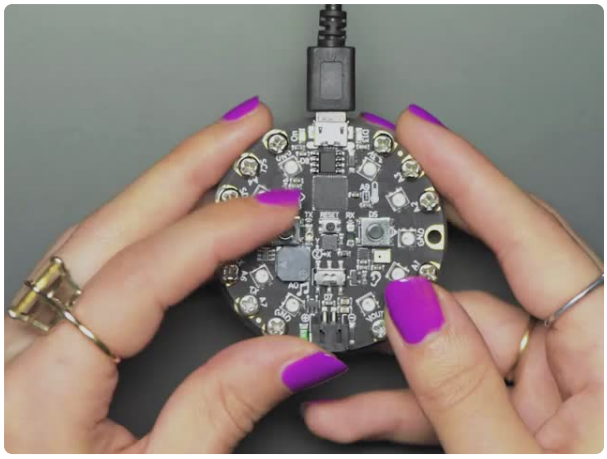
Due to memory constraints, this project will only run on the Circuit Playground Bluefruit.



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

DIY Ornament Kit - 6cm Diameter - Perfect for Circuit Playground

Have you put up with mainstream, uninspiring, low-tech tree ornaments for too long? This season why not deck the halls with codes of holly?This DIY Ornament...

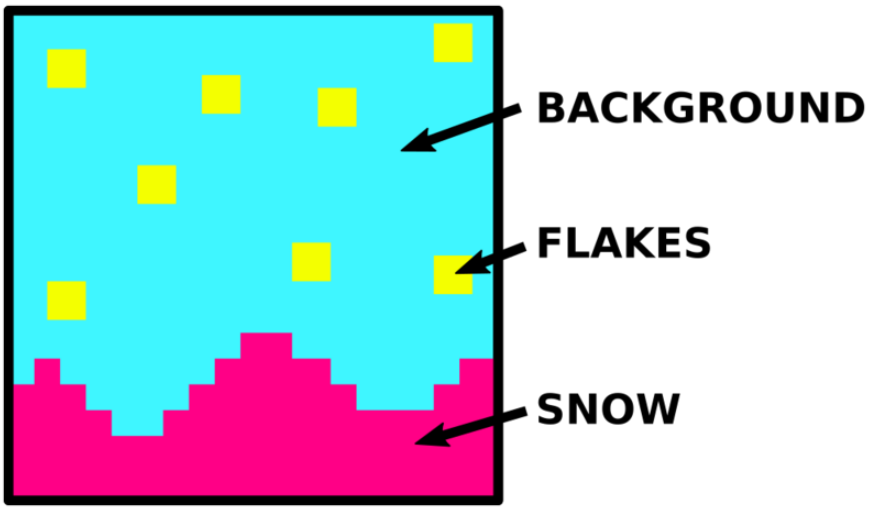
https://www.adafruit.com/product/4036

How It Works

There are three main components to the snow globe:

- The background image
- The flakes
- The snow on the ground

Something like this:



Let's talk about each of these.

Background

This one is pretty simple. It's just an image or a solid color. There's really nothing for it to do other than just sit there. So there's no more code associated with it other than loading it and showing it.

We'll pretty much just do what is described here (https://adafru.it/GB-) and then leave it alone.

Flakes

These are the pretty little snowflakes that come falling out of the sky. We will use little bitmaps to give them shape. We will also allow for a few different fall speeds, so they all don't just fall at the same rate. The other thing that needs to be done is check if they have hit the ground - the current snow level.

To actually make the flakes move, all that needs to be done is change the `y` value of their associated TileGrid. This is a CircuitPython displayio (https://adafru.it/GC0) feature. For more info see here (https://adafru.it/EFw) and here (https://adafru.it/GC1). So basically, each flake will be a separate TileGrid, and we will change the `y` value of each to make them fall.

Snow

This is the snow on the ground. At the beginning, there is no snow on the ground. As the flakes fall and hit the ground, snow will be added in the general area where the flake hit. So this level will grow as more and more flakes fall.

We'll use a displayio Bitmap (https://adafru.it/GC2) to represent the snow. The bitmap size will be the same as the screen size. It will start empty - totally transparent. To "add snow" we will just set individual pixels to the snow color.

Simple Snow Globe

Let's start with a simple version of the snowglobe. This version does not require any additional bitmap files. You can set a background image if you want. But if you don't have one and just want to get up and running, just leave it alone and it will use a solid color.

Here's the code:

```
# SPDX-FileCopyrightText: 2019 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from random import randrange
import board
import busio
import displayio
from adafruit_gizmo import tft_gizmo
import adafruit_imageload
import adafruit_lis3dh

#---| User Config |---------------
BACKGROUND = 0xAA0000              # specify color or background BMP file
NUM_FLAKES = 50                    # total number of snowflakes
SNOW_COLOR = 0xFFFFFF              # snow color
SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------

# Accelerometer setup
accelo_i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
accelo = adafruit_lis3dh.LIS3DH_I2C(accelo_i2c, address=0x19)

# Create the TFT Gizmo display
display = tft_gizmo.TFT_Gizmo()

# Load background image
try:
    bg_bitmap, bg_palette = adafruit_imageload.load(BACKGROUND,
                                                    bitmap=displayio.Bitmap,
                                                    palette=displayio.Palette)
# Or just use solid color
except (OSError, TypeError, AttributeError):
    BACKGROUND = BACKGROUND if isinstance(BACKGROUND, int) else 0x000000
    bg_bitmap = displayio.Bitmap(display.width, display.height, 1)
    bg_palette = displayio.Palette(1)
    bg_palette[0] = BACKGROUND
background = displayio.TileGrid(bg_bitmap, pixel_shader=bg_palette)

# Shared palette for snow bitmaps
palette = displayio.Palette(2)
palette[0] = 0xADAF00   # transparent color
palette[1] = SNOW_COLOR # snow color
palette.make_transparent(0)

# Snowflake setup
```

```
FLAKES = (
    0, 0, 0, 0,    0, 0, 0, 0,    1, 1, 1, 1,
    0, 0, 0, 0,    1, 1, 1, 0,    1, 1, 1, 1,
    0, 1, 1, 0,    1, 1, 1, 0,    1, 1, 1, 1,
    0, 1, 1, 0,    1, 1, 1, 0,    1, 1, 1, 1,
)
flake_sheet = displayio.Bitmap(12, 4, len(palette))
for i, value in enumerate(FLAKES):
    flake_sheet[i] = value
flake_pos = [0.0] * NUM_FLAKES
flakes = displayio.Group()
for _ in range(NUM_FLAKES):
    flakes.append(displayio.TileGrid(flake_sheet, pixel_shader=palette,
                                     width = 1,
                                     height = 1,
                                     tile_width = 4,
                                     tile_height = 4 ) )

# Snowfield setup
snow_depth = [display.height] * display.width
snow_bmp = displayio.Bitmap(display.width, display.height, len(palette))
snow = displayio.TileGrid(snow_bmp, pixel_shader=palette)

# Add everything to display
splash = displayio.Group()
splash.append(background)
splash.append(flakes)
splash.append(snow)
display.root_group = splash

def clear_the_snow():
    #pylint: disable=global-statement, redefined-outer-name
    global flakes, flake_pos, snow_depth
    display.auto_refresh = False
    for flake in flakes:
        # set to a random sprite
        flake[0] = randrange(0, 3)
        # set to a random x location
        flake.x = randrange(0, display.width)
    # set random y locations, off screen to start
    flake_pos = [-1.0*randrange(0, display.height) for _ in range(NUM_FLAKES)]
    # reset snow level
    snow_depth = [display.height] * display.width
    # and snow bitmap
    for i in range(display.width * display.height):
        snow_bmp[i] = 0
    display.auto_refresh = True

def add_snow(index, amount, steepness=2):
    location = []
    # local steepness check
    for x in range(index - amount, index + amount):
        add = False
        if x == 0:
            # check depth to right
            if snow_depth[x+1] - snow_depth[x] < steepness:
                add = True
        elif x == display.width - 1:
            # check depth to left
            if snow_depth[x-1] - snow_depth[x] < steepness:
                add = True
        elif 0 < x < display.width - 1:
            # check depth to left AND right
            if snow_depth[x-1] - snow_depth[x] < steepness and \
               snow_depth[x+1] - snow_depth[x] < steepness:
                add = True
        if add:
            location.append(x)
    # add where snow is not too steep
```

```
    for x in location:
        new_level = snow_depth[x] - 1
        if new_level >= 0:
            snow_depth[x] = new_level
            snow_bmp[x, new_level] = 1

while True:
    clear_the_snow()
    # loop until globe is full of snow
    while snow_depth.count(0) < display.width:
        # check for shake
        if accelo.shake(SHAKE_THRESHOLD, 5, 0):
            break
        # update snowflakes
        for i, flake in enumerate(flakes):
            # speed based on sprite index
            flake_pos[i] += 1 - flake[0] / 3
            # check if snowflake has hit the ground
            if int(flake_pos[i]) >= snow_depth[flake.x]:
                # add snow where it fell
                add_snow(flake.x, flake[0] + 2)
                # reset flake to top
                flake_pos[i] = 0
                # at a new x location
                flake.x = randrange(0, display.width)
            flake.y = int(flake_pos[i])
        display.refresh()
```

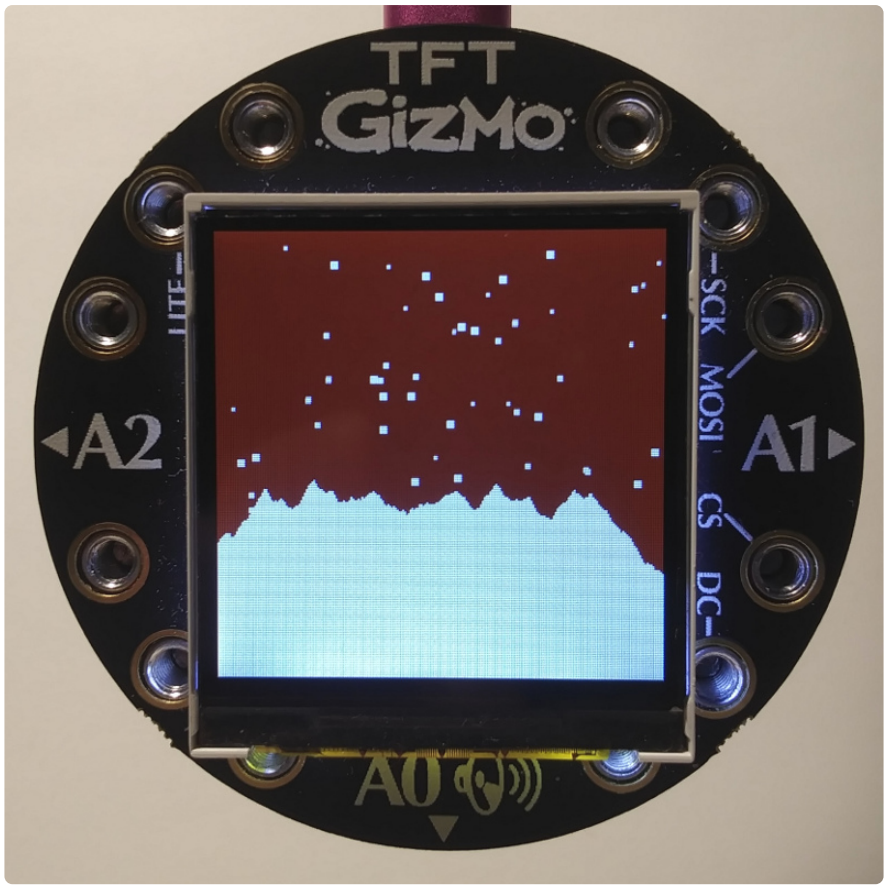
The features you can customize are grouped at the top of the code:

```
#---| User Config |---------------
BACKGROUND = 0xAA0000              # specify color or background BMP file
NUM_FLAKES = 50                    # total number of snowflakes
SNOW_COLOR = 0xFFFFFF              # snow color
SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------
```

The code comments describe what they do. Here's a summary:

- `BACKGROUND` - the background color **OR** image filename
- `NUM_FLAKES` - the total number of flakes to show
- `SNOW_COLOR` - the color of the snow
- `SHAKE_THRESHOLD` - shake-to-clear sensitivity

For now, just go ahead and try running it as is with the default values. You should get some nice white snow on a red background.



Change Colors

Now try changing the `BACKGROUND` and `SNOW_COLOR` values. Change those lines to something like this:

```
BACKGROUND = 0x00FF00               # specify color or background BMP file
NUM_FLAKES = 50                     # total number of snowflakes
SNOW_COLOR = 0xFF00FF               # snow color
SHAKE_THRESHOLD = 27                # shake sensitivity, lower=more sensitive
```

Hey! Yellow snow! On a green background.

Change Background

If you specify a bitmap file instead of a color for `BACKGROUND`, it will be loaded and used. For the TFT Gizmo, the file needs to be an indexed BMP file that is 240x240 in size. Here's one you can use to try this out:

blinka_dark.bmp

https://adafru.it/GC3

Download that file and copy it to your **CIRCUITPY** folder. Then change the background setting line to:

```
BACKGROUND = "/blinka_dark.bmp"              # specify color or background BMP file
```

And change the snow color back to white:

```
SNOW_COLOR = 0xFFFFFF              # snow color
```

Run that and you should end up with snow falling over a Blinka background.

Change Flakes

You can do a minor amount of flake shape customization. We'll provide a fancier version of the code next that will really let you do this. But if you want, you can play around with these lines of code:

```
FLAKES = (
    0, 0, 0, 0,    0, 0, 0, 0,    1, 1, 1, 1,
    0, 0, 0, 0,    1, 1, 1, 0,    1, 1, 1, 1,
    0, 1, 1, 0,    1, 1, 1, 0,    1, 1, 1, 1,
    0, 1, 1, 0,    1, 1, 1, 0,    1, 1, 1, 1,
)
```

See how there are 3 separate 4x4 grids? Those are the flakes, which are each 4 pixels wide by 4 pixels high. A `0` ends up being transparent. A `1` ends up being the snow color. So you can edit those if you want to change the flake shape. But with only 4x4 pixels to work with, it's pretty minimal customization.

Shake To Clear

Don't forget to also try shaking the board to "clear" the snow globe. This will erase all of the accumulated snow and reset all of the snowflakes back to the top. Go ahead - shake it like a polaroid!

Fancy Snow Globe

OK, let's get fancy with those flakes. Allowing for custom flakes is a perfect use of creating a "sprite sheet" to hold the flake shapes, as discussed here (https://adafru.it/GC4).

This comes at the cost of requiring you to actually create this sprite sheet. You'll also need to modify a few lines of code to provide some info about the sprite sheet layout. But with this feature, along with a settable background image, you can really have fun customizing your snow globe. We'll provide a few examples to get you started.

Here's the fancy version of the snow globe code:

```
# SPDX-FileCopyrightText: 2019 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from random import randrange
import board
import busio
import displayio
from adafruit_gizmo import tft_gizmo
import adafruit_imageload
import adafruit_lis3dh

#---| User Config |---------------
BACKGROUND = "/blinka_dark.bmp"    # specify color or background BMP file

NUM_FLAKES = 50                    # total number of snowflakes
FLAKE_SHEET = "/flakes_sheet.bmp"  # flake sprite sheet
FLAKE_WIDTH = 4                    # sprite width
FLAKE_HEIGHT = 4                   # sprite height
FLAKE_TRAN_COLOR = 0x000000        # transparency color

SNOW_COLOR = 0xFFFFFF              # snow color

SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------

# Accelerometer setup
accelo_i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
accelo = adafruit_lis3dh.LIS3DH_I2C(accelo_i2c, address=0x19)

# Create the TFT Gizmo display
display = tft_gizmo.TFT_Gizmo()

# Load background image
try:
    bg_bitmap, bg_palette = adafruit_imageload.load(BACKGROUND,
                                                    bitmap=displayio.Bitmap,
                                                    palette=displayio.Palette)
# Or just use solid color
except (OSError, TypeError, AttributeError):
    BACKGROUND = BACKGROUND if isinstance(BACKGROUND, int) else 0x000000
    bg_bitmap = displayio.Bitmap(display.width, display.height, 1)
    bg_palette = displayio.Palette(1)
    bg_palette[0] = BACKGROUND
background = displayio.TileGrid(bg_bitmap, pixel_shader=bg_palette)
```

```
# Snowflake setup
flake_bitmap, flake_palette = adafruit_imageload.load(FLAKE_SHEET,
                                                      bitmap=displayio.Bitmap,
                                                      palette=displayio.Palette)
if FLAKE_TRAN_COLOR is not None:
    for i, color in enumerate(flake_palette):
        if color == FLAKE_TRAN_COLOR:
            flake_palette.make_transparent(i)
            break
NUM_SPRITES = flake_bitmap.width // FLAKE_WIDTH * flake_bitmap.height //
FLAKE_HEIGHT
flake_pos = [0.0] * NUM_FLAKES
flakes = displayio.Group()
for _ in range(NUM_FLAKES):
    flakes.append(displayio.TileGrid(flake_bitmap, pixel_shader=flake_palette,
                                     width = 1,
                                     height = 1,
                                     tile_width = FLAKE_WIDTH,
                                     tile_height = FLAKE_HEIGHT,
                                     x = randrange(0, display.width),
                                     default_tile=randrange(0, NUM_SPRITES)))

# Snowfield setup
snow_depth = [display.height] * display.width
snow_palette = displayio.Palette(2)
snow_palette[0] = 0xADAF00   # transparent color
snow_palette[1] = SNOW_COLOR # snow color
snow_palette.make_transparent(0)
snow_bitmap = displayio.Bitmap(display.width, display.height, len(snow_palette))
snow = displayio.TileGrid(snow_bitmap, pixel_shader=snow_palette)

# Add everything to display
splash = displayio.Group()
splash.append(background)
splash.append(flakes)
splash.append(snow)
display.root_group = splash

def clear_the_snow():
    #pylint: disable=global-statement, redefined-outer-name
    global flakes, flake_pos, snow_depth
    display.auto_refresh = False
    for flake in flakes:
        # set to a random sprite
        flake[0] = randrange(0, NUM_SPRITES)
        # set to a random x location
        flake.x = randrange(0, display.width)
    # set random y locations, off screen to start
    flake_pos = [-1.0*randrange(0, display.height) for _ in range(NUM_FLAKES)]
    # reset snow level
    snow_depth = [display.height] * display.width
    # and snow bitmap
    for i in range(display.width*display.height):
        snow_bitmap[i] = 0
    display.auto_refresh = True

def add_snow(index, amount, steepness=2):
    location = []
    # local steepness check
    for x in range(index - amount, index + amount):
        add = False
        if x == 0:
            # check depth to right
            if snow_depth[x+1] - snow_depth[x] < steepness:
                add = True
        elif x == display.width - 1:
            # check depth to left
            if snow_depth[x-1] - snow_depth[x] < steepness:
```

```
                add = True
        elif 0 < x < display.width - 1:
            # check depth to left AND right
            if snow_depth[x-1] - snow_depth[x] < steepness and \
               snow_depth[x+1] - snow_depth[x] < steepness:
                add = True
        if add:
            location.append(x)
    # add where snow is not too steep
    for x in location:
        new_level = snow_depth[x] - 1
        if new_level >= 0:
            snow_depth[x] = new_level
            snow_bitmap[x, new_level] = 1

while True:
    clear_the_snow()
    # loop until globe is full of snow
    while snow_depth.count(0) < display.width:
        # check for shake
        if accelo.shake(SHAKE_THRESHOLD, 5, 0):
            break
        # update snowflakes
        for i, flake in enumerate(flakes):
            # speed based on sprite index
            flake_pos[i] += 1 - flake[0] / NUM_SPRITES
            # check if snowflake has hit the ground
            if flake_pos[i] >= snow_depth[flake.x]:
                # add snow where it fell
                add_snow(flake.x, FLAKE_WIDTH)
                # reset flake to top
                flake_pos[i] = 0
                # at a new x location
                flake.x = randrange(0, display.width)
            flake.y = int(flake_pos[i])
        display.refresh()
```

As you can see, there are more items in the customization section:

```
#---| User Config |---------------
BACKGROUND = "/blinka_dark.bmp"    # specify color or background BMP file

NUM_FLAKES = 50                    # total number of snowflakes
FLAKE_SHEET = "/flakes_sheet.bmp"  # flake sprite sheet
FLAKE_WIDTH = 4                    # sprite width
FLAKE_HEIGHT = 4                   # sprite height
FLAKE_TRAN_COLOR = 0x000000        # transparency color

SNOW_COLOR = 0xFFFFFF              # snow color

SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------
```

Most of these are the same as the simple version in the previous section. The new ones are for specifying a flake sprite sheet. They are:

- `FLAKE_SHEET` - the BMP file containing the flake sprites
- `FLAKE_WIDTH` - the width of each sprite
- `FLAKE_HEIGHT` - the height of each sprite
- `FLAKE_TRAN_COLOR` - the color to use as transparency

For `FLAKE_TRAN_COLOR`, it's best to just pick something simple when creating your sprite sheet bitmap. Don't use one of the colors in the flake itself, since then it wouldn't show up.

The two that are probably the most confusing are `FLAKE_WIDTH` and `FLAKE_HEIGHT`. You can think of these as the **width** and **height** of each of your flakes. These should be the same for each flake. It's not the total width and height of your flake sheet - just the sprites themselves. For a more in depth discussion see here (https://adafru.it/GC1) and here (https://adafru.it/GC4).

Background Image

To start out, the code is reusing the background image from the previous section of the guide. Download it from the link there. Be sure to have the file **blinka_dark.bmp** in your **CIRCUITPY** folder.

Simple Snowglobe Redux

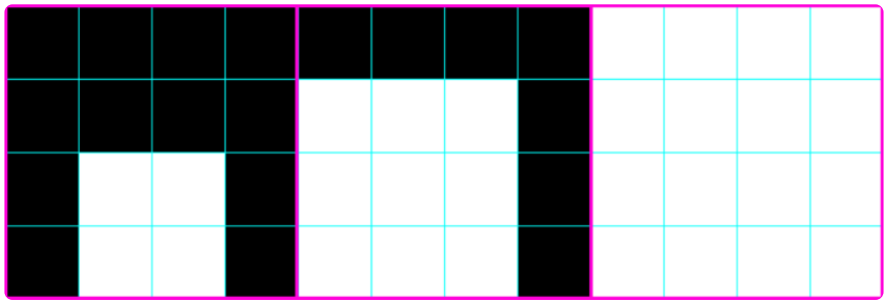
Let's start by simply recreating the same flakes from the previous section, but this time by using a sprite sheet. Here's the file:

flakes_sheet.bmp

https://adafru.it/GC5

Download that and copy it to your **CIRCUITPY** folder.

If you open it in an image viewer, you won't see much since it's so small. But zoomed in, it looks something like this:



The border and grids have been added for reference. They aren't part of the actual file. You can see how there are 3 different 4x4 bitmaps. We need to tell the code this, which is what these lines do:

```
FLAKE_WIDTH = 4                     # sprite width
FLAKE_HEIGHT = 4                    # sprite height
```

With the background (**blinka_dark.bmp**) and flake sprite sheet (**flake_sheet.bmp**) files copied to your **CIRCUITPY** folder, you can try running the code above. You should get the same as before - a Blinka background with simple white snow flakes.

Now let's try something different.

Who Watches The...Squids?

This example better shows how you can customize the snow globe. Hmmm. What else can we make fall from the sky. How about squids? Sure. And with a background that matches the reference. Grab these two files:

watchmen_bg.bmp

https://adafru.it/GC6

squid_sheet_16.bmp

https://adafru.it/GC7

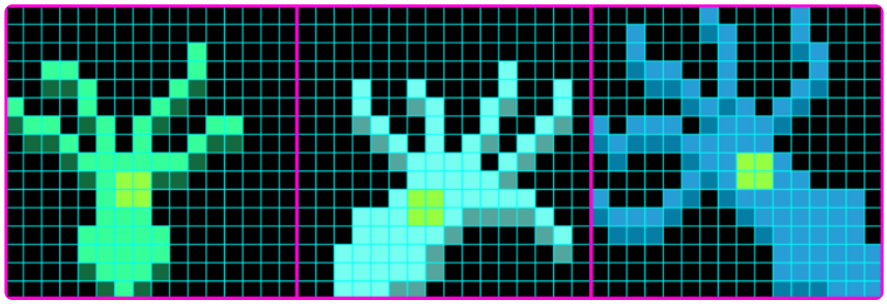
and save to your **CIRCUITPY** folder.

Then, change the customization settings to this:

```
#---| User Config |---------------
BACKGROUND = "/watchmen_bg.bmp"    # specify color or background BMP file

NUM_FLAKES = 20                    # total number of snowflakes
FLAKE_SHEET = "/squid_sheet_16.bmp"  # flake sprite sheet
FLAKE_WIDTH = 16                   # sprite width
FLAKE_HEIGHT = 16                  # sprite height
FLAKE_TRAN_COLOR = 0x000000        # transparency color

SNOW_COLOR = 0x279ED5              # snow color

SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------
```

The background image is just another BMP file. Other than that, nothing new.

The sprite sheet is pretty different though. The flakes are larger - 16 x 16. They also contain multiple colors. Zoomed in, it looks something like this:



The total number of flakes is reduced a bit, since these are larger flakes. And to make the snow match the squids, it is set to a blue-ish color.

With everything copied over and the other changes in place, run the code again and you should get:

Falling squids!

Have Fun!

Have fun with this flake customization feature. Like maybe have cat and dog flakes? Or maybe frogs? Falling hearts might be nice. Or just fancier snow flakes. Up to you.

What else can fall from the sky? Here's one more for you. But you'll have to load these up to see...

wg_bg.bmp

https://adafru.it/GC8

wg_sheet.bmp

https://adafru.it/GC9

```
#---| User Config |---------------
BACKGROUND = "/wg_bg.bmp"    # specify color or background BMP file

NUM_FLAKES = 20                    # total number of snowflakes
FLAKE_SHEET = "/wg_sheet.bmp"  # flake sprite sheet
FLAKE_WIDTH = 20                   # sprite width
FLAKE_HEIGHT = 20                  # sprite height
FLAKE_TRAN_COLOR = 0x000000        # transparency color
```

```
SNOW_COLOR = 0xFF00FF              # snow color

SHAKE_THRESHOLD = 27               # shake sensitivity, lower=more sensitive
#---| User Config |---------------
```

More Details

Making Sprite Sheets

The main key is to save your files as indexed bitmap image files. Then they can be loaded using the CircuitPython Image Load (https://adafru.it/EFL) library. Exactly how you do this will depend on what software you are using. The sprite sheets in this guide were created using GIMP (https://adafru.it/GCa). The general process went something like:

- Create new image width x height pixels
- Use 1 pixel pencil tool to draw sprites
- Save as .xcf file for future edits
- When ready to export:
  - Image -> Mode -> Indexed
  - File -> Export As...
  - specify filename with .bmp extension

A similar export method can be used for the background images.

Flake Fall Speed

So how were different flake fall speeds implemented? Well, pretty simply. The speed is based on the flake index. It comes down to this one line of code (from the fancy version):

```
flake_pos[i] += 1 - flake[0] / NUM_SPRITES
```

The first flake falls the fastest. The last flake falls the slowest. That's why the various flake examples arranged the flakes from smallest to largest.

Since the `y` location is an integer value, a separate float value is used to allow for floating point math. That is what gets stored in `flake_pos`. This is simply changed to an integer when it comes time to set the flake position:

```
flake.y = int(flake_pos[i])
```

A fancier way to do this might be to allow for setting custom fall speeds for each flake. But that would require an additional storage mechanism and even more work to manually set up for each flake sprite sheet.

Another idea might be to somehow "weigh" the flake sprite, like how many non-transparent pixels it contains. And then base fall speed on that.

Animated Flakes?

Sure. Why not? That would be pretty cool. This could be done. Maybe in a future version.

Snow Accumulation

It's pretty easy to know when a flake hits the ground. But then what? How do we "add" snow to the currently accumulated snow. The simplest would be to just blindly add a few pixels around where the flake fell. However, this can lead to a "spikey" profile to the snow, which doesn't look very natural.

The current version of the code tries to deal with this, but in a pretty simple way. It does a local steepness check based on the surrounding snow and will only add pixels if it is currently not too steep. This works for the most part, but does lead to somewhat unnatural looking edge effects.

It'd be nice if this were fancier. Another maybe-in-the-future mod. Maybe an avalanche simulator?

Layering in displayio

This project is a good example of how multiple `TileGrids` and `Groups` can be used to create layered effects and animation. The background image and the snow on the ground are just a single `TileGrid` that fills the entire display. The flakes are more interesting. Each flake is a `TileGrid` and is only as big as the flake itself. These are all added to the same `Group`. All three of these elements, background (`TileGrid`), snow (`TileGrid`), and flakes (`Group` of `TileGrid`s), are then added to the main `Group` which is shown on the display. As such, the ordering matters in these lines of code:

```
# Add everything to display
splash = displayio.Group()
splash.append(background)
splash.append(flakes)
splash.append(snow)
display.show(splash)
```