Fireworks:

```
from adafruit_circuitplayground import cp
import time
import random

num_pixels = cp.pixels.n

cp.pixels.brightness = 1


def theater_chase(color1 = 0x0000FF, color2 = 0xFFFFFF, delay = 0.1):
    cp.pixels.brightness = 1
    index = 0
    while True:
        for i in range(num_pixels):
            if (i + index) % 3 == 0:
                cp.pixels[i] = color1
            else:
                cp.pixels[i] = color2
        cp.pixels.show()
        index+= 1
        time.sleep(0.01)

while True:
    theater_chase(0xFF0000, 0x00FF00)
    theater_chase()
```

Sparkle:

```
def sparkle(color= 0x000000,color2= 0xFF00FF, delay = 0.01):
    for i in range(30):
        cp.pixels.fill(color)
        cp.pixels[random.randint(0, 9)]= color2
        time.sleep(delay)
    cp.pixels.fill(0)
```

Fade:

```
def fade_out() :
    brightness = 1.0
    change_by = 0.1
    cp.pixels.fill(0x250000)
    while brightness > 0:
        brightness -= change_by
        cp.pixels.brightness = brightness
        time.sleep(0.1)

def fade_in(color = 0x250025):
    brightness=0.0
    change_by = 0.1
    cp.pixels.fill(color)
    while brightness < 1:
        brightness += change_by
        cp.pixels.brightness = brightness
        time.sleep(0.1)
```

Slow fill:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio

num_pixels = 10
# tuple containing colors
colors = (0xff0000, 0x0000ff, 0xff00ff, 0x000f0f, 0x00ff00, 0x0f000f)

def slow_fill(color):
	for i in range(num_pixels):
		cp.pixels[i] = color
		cp.pixels.show()
		time.sleep(0.1)
cp.pixels.fill(0)
while True:
	for color in colors:
		slow_fill(color)
```

Capacitive Touch demo:

```
from adafruit_circuitplayground import cp
import time
while True:
    cp.pixels.fill(0)
    if cp.touch_A1:
        cp.pixels[1] = 0x080000
    elif cp.touch_A2:
        cp.pixels[2] = 0x080000
    elif cp.touch_A3:
        cp.pixels[3] = 0x080000
```

analog:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio

pot = analogio.AnalogIn(board.A1)
while True:
	print(int(pot.value / 65535 * 10))
	time.sleep(0.1)
```

Change brightness:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio

cp.pixels.fill(0xf041f)
pot = analogio.AnalogIn(board.A1)
while True:
    norm_data = pot.value / 65535
    cp.pixels.brightness = norm_data
    cp.pixels.show()
    time.sleep(0.1)
```

Dictionary:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio

named_colors = {
    'red': 0xFF0000,
    'green': 0x00FF00,
    'blue': 0x0000FF
}
while True:
    for color in named_colors:
        for i in range(5):
            cp.pixels[i] = named_colors['red']
            time.sleep(0.2)
```

Loop:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio


colors = (0xff0000, 0x0000ff, 0xff00ff, 0x000f0f, 0x00ff00, 0x0f000f)
def loop_test(color):
    for i in range(0, 10, 4):
        cp.pixels[i] = color
        cp.pixels.show()
        time.sleep(0.1)
cp.pixels.fill(0)

while True:
    for color in colors:
        loop_test(color)
```

Simon game:

```
from adafruit_circuitplayground import cp
import time
import board
import analogio

named_colors = {
    'red': 0xFF0000,
    'green': 0x00FF00,
    'blue': 0x0000FF,
    'yellow': 0xFF9000
}
while True:
    for color in named_colors:
        for i in range(5):
            cp.pixels[i] = named_colors['yellow']
            time.sleep(0.2)
```

Dice

```
import board
import time
import displayio
import terminalio
from adafruit_display_text import label
from adafruit_display_shapes.circle import Circle
import die
from digitalio import DigitalInOut
from gamepadshift import GamePadShift
pad = GamePadShift(DigitalInOut(board.BUTTON_CLOCK),
DigitalInOut(board.BUTTON_OUT), DigitalInOut(board.BUTTON_LATCH))

# Use the built-in Display object
display = board.DISPLAY

# Make the display context
splash = displayio.Group()
display.show(splash)

# Make a background color fill
color_bitmap = displayio.Bitmap(160, 128, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(color_bitmap, x=0, y=0, pixel_shader=color_palette)
splash.append(bg_sprite)

# Add text to the screen
text = "Hello World!"
font = terminalio.FONT
color = 0x0000FF
text_area = label.Label(font, text=text, color=color)
text_area.x = 20
text_area.y = 40
splash.append(text_area)
text_area.hidden = True


# Add a circle to the screen
circle1 = Circle(79, 63, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle1)
circle1.hidden = True
circle2 = Circle(26, 105, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle2)
```

```
circle2.hidden = True
circle3 = Circle(26, 21, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle3)
circle3.hidden = True
circle4 = Circle(79, 21, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle4)
circle4.hidden = True
circle5 = Circle(132, 21, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle5)
circle5.hidden = True
circle6 = Circle(79, 105, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle6)
circle6.hidden = True
circle7 = Circle(132, 105, 10, fill=0xDD00FF, outline=0x000000)
splash.append(circle7)
circle7.hidden = True

def roll():
    circle1.hidden = True
    circle2.hidden = True
    circle3.hidden = True
    circle4.hidden = True
    circle5.hidden = True
    circle6.hidden = True
    circle7.hidden = True
    die1 = die.Die(6)
    roll = die1.roll()
    print(roll)
    if roll %2 == 1:
        circle1.hidden = False
    if roll >1:
        circle2.hidden = False
        circle5.hidden = False
    if roll >3:
        circle3.hidden = False
        circle7.hidden = False
    if roll == 6:
        circle4.hidden = False
        circle6.hidden = False
```

```
# while loop needed to see the output
while True:

    pressed = pad.get_pressed()
    if pressed & 2 > 0:
        roll()
    time.sleep(0.3)
```

Rescale:

```
def rescale(input_min = 0, input_max = 65535, output_min = 0, output_max = 180):
        input = (input_max-input_min)
        output = (output_max-output_min)
```