Screen Saver

Creative Coding Assignment 02

Inspiration

Nowadays, many game console company use minimalism design. They tend to use simple lines or graphics, such as the four console buttons pattern commonly used in PlayStations design.

In this screen saver Assignment, I choose gaming as the theme, and I will make it with a minimalism design. In addition to this, a practical function of displaying time and a smooth repeating animation is driven by noise() will be added.

▲ PlayStation gaming style Wallpaper

Artistic & Creative Related Dicisions

This Screen Saver is divided into four parts.

(1) First of all, the first one is the button patterns generated randomly on the screen's background.
(2) The second is the animation of two waves on the top and bottom of the screen driven by noise().
(3) The third is the button particle movement animation controlled by noise and translate function at the centre of the screen.
(4) And the last is the clock in the centre of the screen.

▲ A02 Screen Saver screenshot

Style

In order to create a Minimalism style screen saver, I started with colors and shape of objects. First of all, for the color part, I mainly use grayscale tones. Most of the spaces are mainly in black and grey color and only some of particle objects have hue colors.

Secondly, objects do not have complicated shapes, there are simple lines and geometric figures only.

Functions

Secondly, for the animation, except for the emitter, all of the animations have trajectories rather than random activities, which makes the animation look very smooth and matches the Minimalism style of the screen saver.

Furthermore, the screen saver has added the function of displaying date and time, which makes it beautiful and practical.

Implementation

As mentioned in the previous part, the screen saver is mainly divided into four parts. They are background, waves, particle objects and clock. For better control, I created three classes and a data type.

Background

The first part is the background of the screen saver. To do that, I created four PShape methods to create shapes, then load them into the PShape fields in the constructor.

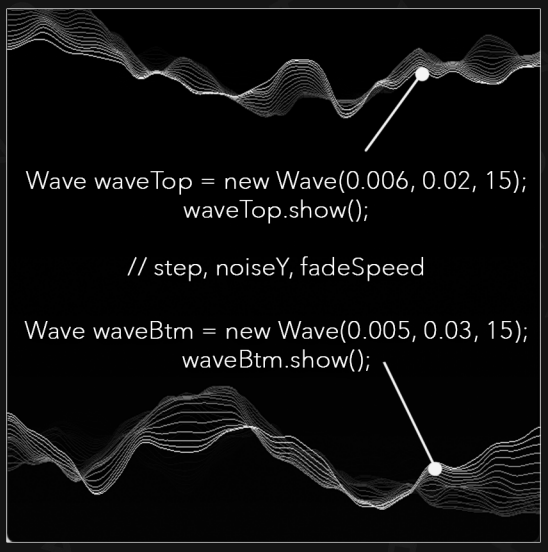
In this class, I wrote a method to draw the background by inputting the background pattern's size, the thickness of the shape, and the number of objects. This function is driven by random(). First, a random seed is set, and then the pattern is averagely generated on the screen.

▲ Background

Waves

The second part is the two waves above and below the screen. A wave object is created whose constructor receives three parameters to change the wave animation's smoothness and the speed at which the animation fades out. Then the wave animation can be displayed on the screen through the show() method of the class.

This animation mainly created by using PGraphics and 2d noise. In PGraphic, the data generated by noise() is displayed with a line to form a wave pattern. In noise X, step is input as the variable of influence. In noise Y, because I wanted to show a frontal rather than side-viewing wave-shaped picture, I created a global variable noiseY and increased it in each loop. The linear increase creates a wave that continuously faces the screen direction. Then I add a rectangle with transparency and cover the entire screen range to PGraphic to create a gradual wave animation effect.



▲ Waves

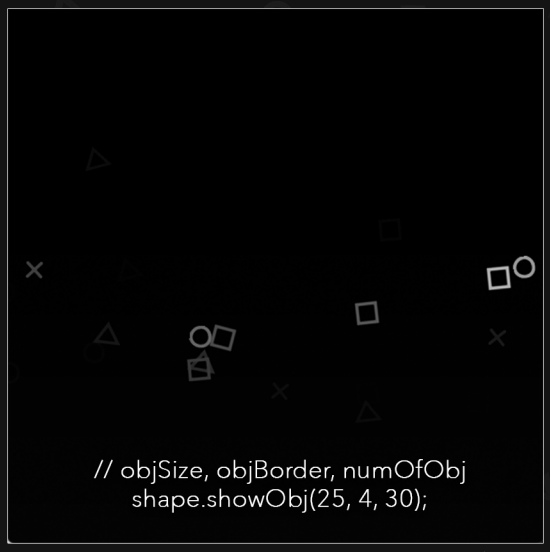
A mask function is then used to make the PGraphics transparent so that it is arbitrarily covered on the top of the screen to create a fading effect. To achieve the upper and lower waves, I first input different initial values when creating a new instance so that they would not repeat the same wave pattern. Then use the functions of pushMatrix and translate to place them at the top and bottom of the screen.

Particles

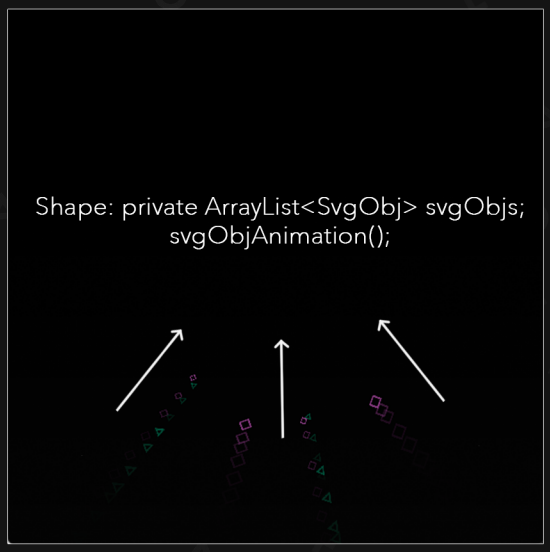
For the particle object, there are two types of it.

The first is inspired by example 7 in lecture 8, which moves from the left to the right of the screen. First, a random seed is set, so that the following particles will move with a trajectory. Then I use variables speed and frameCount and divide them by the number of width + 100, so that the particles can be played smoothly out of the screen without noticing. Then the pushMatrix() and translate() methods are used to achieve rotation when the object moves. To chose a shape to show, a variable - choice is created to select a number from 1-4 randomly, and then cooperate with the switch to make it randomly select the particle pattern and display it on the screen.

The second is to create an object and load one of the four svg files downloaded on the Internet and use the choice variable and switch statement used in the previous particle effect. Randomly assign a coordinate when creating an instance. Then an ArrayList is created in the Shape class to store these objects, and display them on the screen through the showSvgObj() method, and then use the methods of translate and pushMatrix to control their moving direction and let them disappear after moving to the center of the screen, and then restart Appears at the edge of the screen.



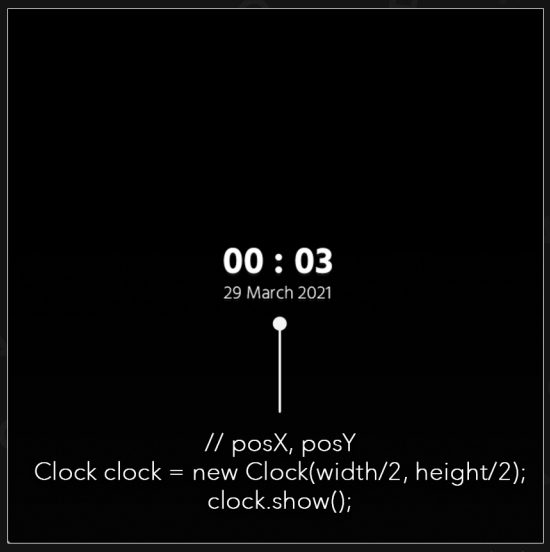
▲ Particle V1



▲ Particle V2

Clock

For the clock part, I created a Clock class to handle the clock's property and functions. When creating a new instance of it, the constructor needs the input of the clock's coordinates. Then get the relevant data through the hour(), minute(), month(), day(), and year() variables of processing, and convert the data such as month() from an int into String in English by a switch statement, and then display it on the screen as text.



▲ Clock