A Day and Night in the Garden of Flowers

JouskaChrysalism IMGUR
Majestic_science INST
Sleek-turle GIT

0 ABSTRACT

This simulation shows the day and night cycles of the cute community of flowers. They dance and exist all day long while they soak up the sun. Each flower is loaded with random parameters to generate random load builds, then they just exist during runtime.

KEYWORDS

Python – scripting language used as a interpreter for c++,

Polar coordinates – a coordinate pair (theta , R), used to graph unit circles with angle specified and length from origin as the axis.

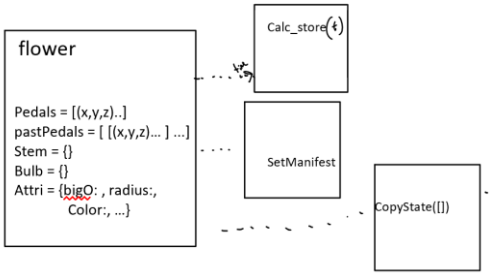
Pickling – the act of flatting data on the heap to a parse able language as a text file byte-stream.

1 INTRODUCTION

In this system, randomly generated flowers are created. They all have attributes like size, frequencies, colour. These all go into the equation f(theta) = A sin(theta) + constant. This parameterization creates a flower like graph and does most of the computing the entire program. So then during its runtime, these factors affect the pedals of the flower. Meanwhile there is a subtle animation of the light source of the Sun that cycles day and night. When the sun comes up, the light emerges and the cycles continue :)

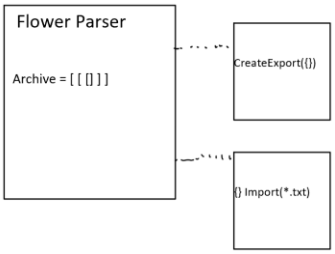
2 KEY ENTITIES

2.1.1 Flower



The flower is pretty self-maintaining. Each chunk for time you use the calc store module to iterate and calculate each new state of the pedals, either that OR you can use copy state and load in an existing array of points into Floweras well. after all flowers are computed you can use set manifest to update the drawing. Or you do it immediately after calc_store but that may be less efficient.

2.1.2 Flower Pickler



This picklerwill be only used at the beginning or end of the program. Once you have existing data you can load that into the file or at the end you send in the array of flowers to be flattened for later.

3 ARCHITECTURE FLOW OF EVENTS

3.1 BUILDING

3.1.1 With initial file

In this inherited behavior of the program, there is a byte file already located in the directory that contains a previously created load build and lifecycle. In this file it contains all the dynamic data pertaining to the pedals , in this way, All of the complexity of calculating new points is removed from the system and hence allows for even more vast animations. Hopefully it might even aid readability.

Anyways, all the points are gathered in as static memory on the stack, then just sent to the flowers during the game loop dictating their exact positions. Each chunk of time , the flowers are sent to the next state. According to this static memory

3.1.2 build new on runtime

In this case a random number generator fills the flowers with initial condition attributes such as the radius or others, like a complexity can be specified to make the flowers more or less precise. Also in this case, during the game loop, each chunk of time, more cpu speed is dedicated to calculating the next state of the flowers.

3.2 DYNAMIC PLOTS

3.2.1 What factors are changing?

Each chunk of time, every pixel in the flower pedals is changed. Whether the read from the static array from the file, or they are calculated using just that one equation calcFlower(), which works with the polar coordinate system, that is the only piece of the flower that is changing.

As well as the flower though, the sun is moving, this is what changes day and night. Some activity also changes during the day

3.3 ENVIROMENT

3.3.1 What factors are NOT changing?

In this simulation, there are static objects that do not change and do not require processing. The stems and buds of all the flowers are background and do not change. As well as that there is the grass and soil underneath that remain there as well.

4 COMPLEXITY…

4.1.1 Runtime

Runtime at the least complexity is just which is calculated each step in the loop. Num flower pieces * num flower

4.1.2 Storage

Storage requires all the data calculated each step of the runtime. So it has all that times all the chunks.