GD 50 (Flappy Bird).

initialize: love.load()
render: love.draw()
resizable: love.resize()

Objectives:
- backgrounds
- procedural generation of pipes
- bird controls
- implementation of rules
- scoring system
- sounds.

function
↳ love.draw()
push:start()
:
push.finish()
end.

AABB - axis aligned bounding box.

Parallax scroll: an illusion of depth in a 2D distance.
E.g. you are in your car and see the objects near you pass by very quickly while the objects far from you move slowly.

local - ensures the variable cannot be used across files.

Procedural generation of pipes.
Random Y position when spawned
Pipes will scroll at a certain speed, that would mean making use of a scroll variable and dt to update x position
Ensuring we don't have constantly increasing pipes
1. we don't need more than the number of pipes that fit on screen
2. storing all of the data for the pipes would require a lot of memory.

Solution
1. Pipes table 2. spawn Timer

Line of Action:

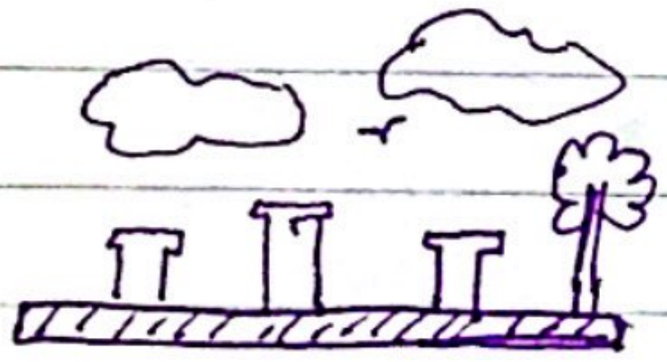
create a table

upon every pipe being spawned, insert it

when the pipe leaves the screen, remove it.

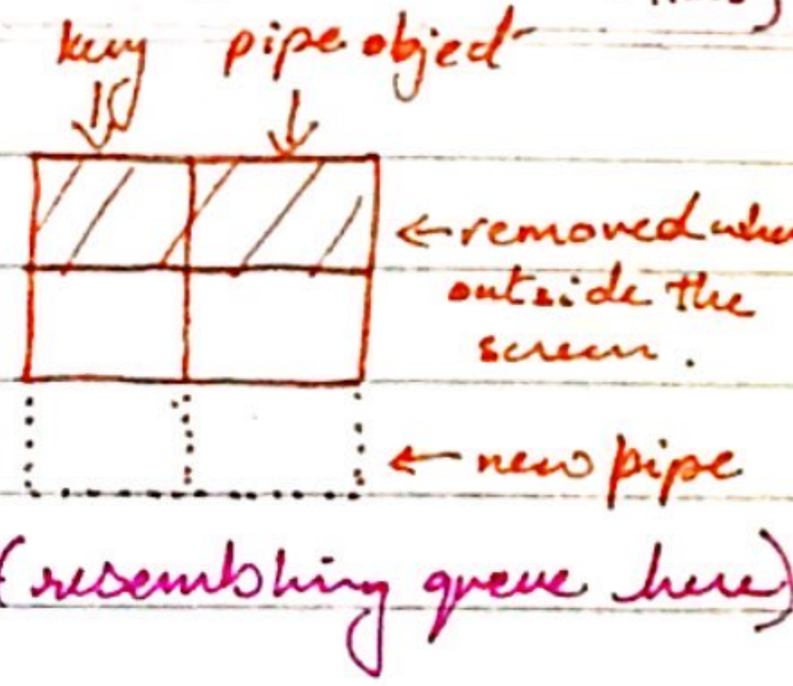
```
for k, pipe in pairs(pipes) do
    pipe:update(dt)

    if pipe.x < -pipe.width then
        table.remove(pipes, k)
    end
end
```

Draw order: (make it look like pipes are coming from ground)



1. Background
2. Pipes
3. Ground

(think of tables like linked lists)



(resembling queue here)

```
for k, pipe in pairs(pipes) do
    pipe:render()
end
```

Q. When to spawn a pipe?

A. We can deduce the time it takes for a pipe to leave the screen, on the basis of that we can decide it when to spawn

```
if $spawnTimer > 2 then
    table.insert(pipes, Pipe())
    spawnTimer = 0
end.
```

Adding the second pipe

(as pipes exist in pairs with certain distances)

Time for a little abstraction.

We want the second pipe to make use of the initial pipe instead of being independent.

- We will remove the pipes tables and create a new table which includes both top and bottom pipes.