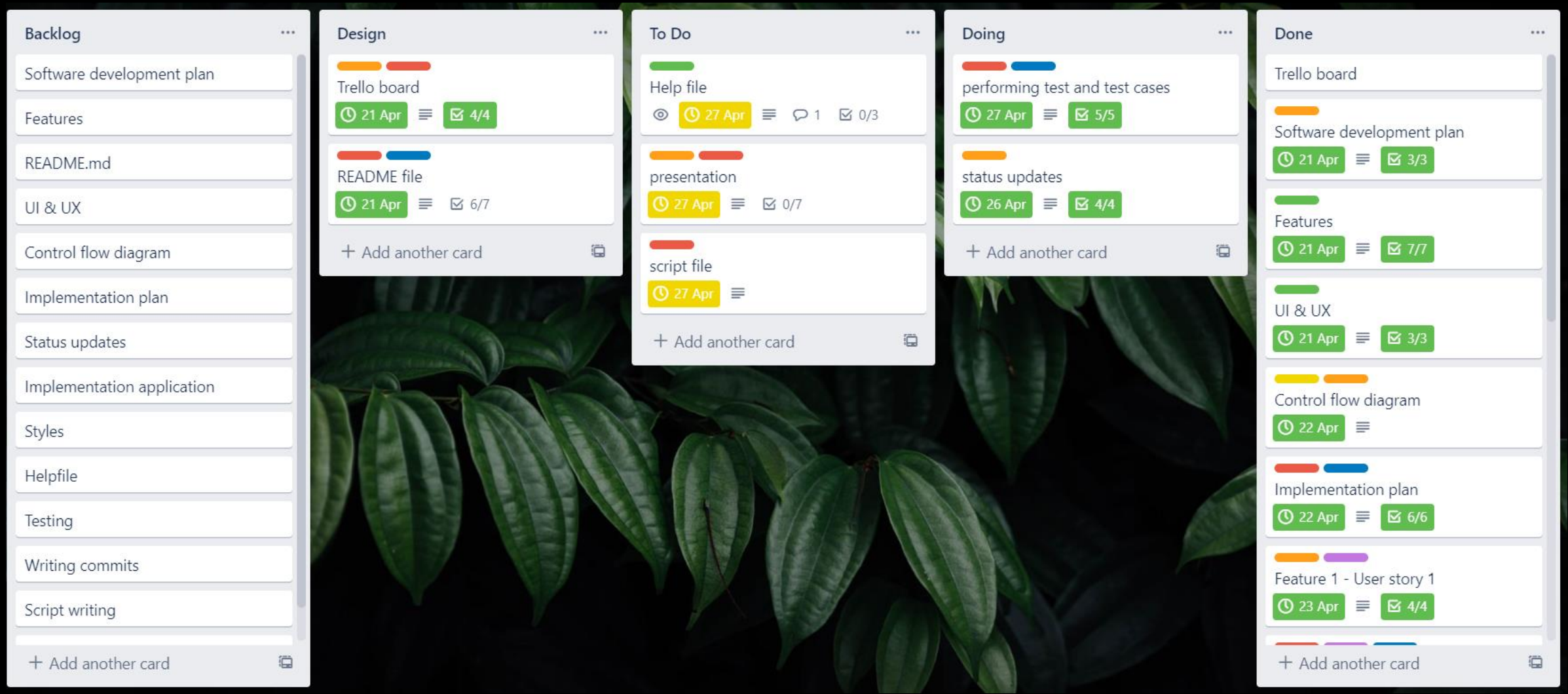
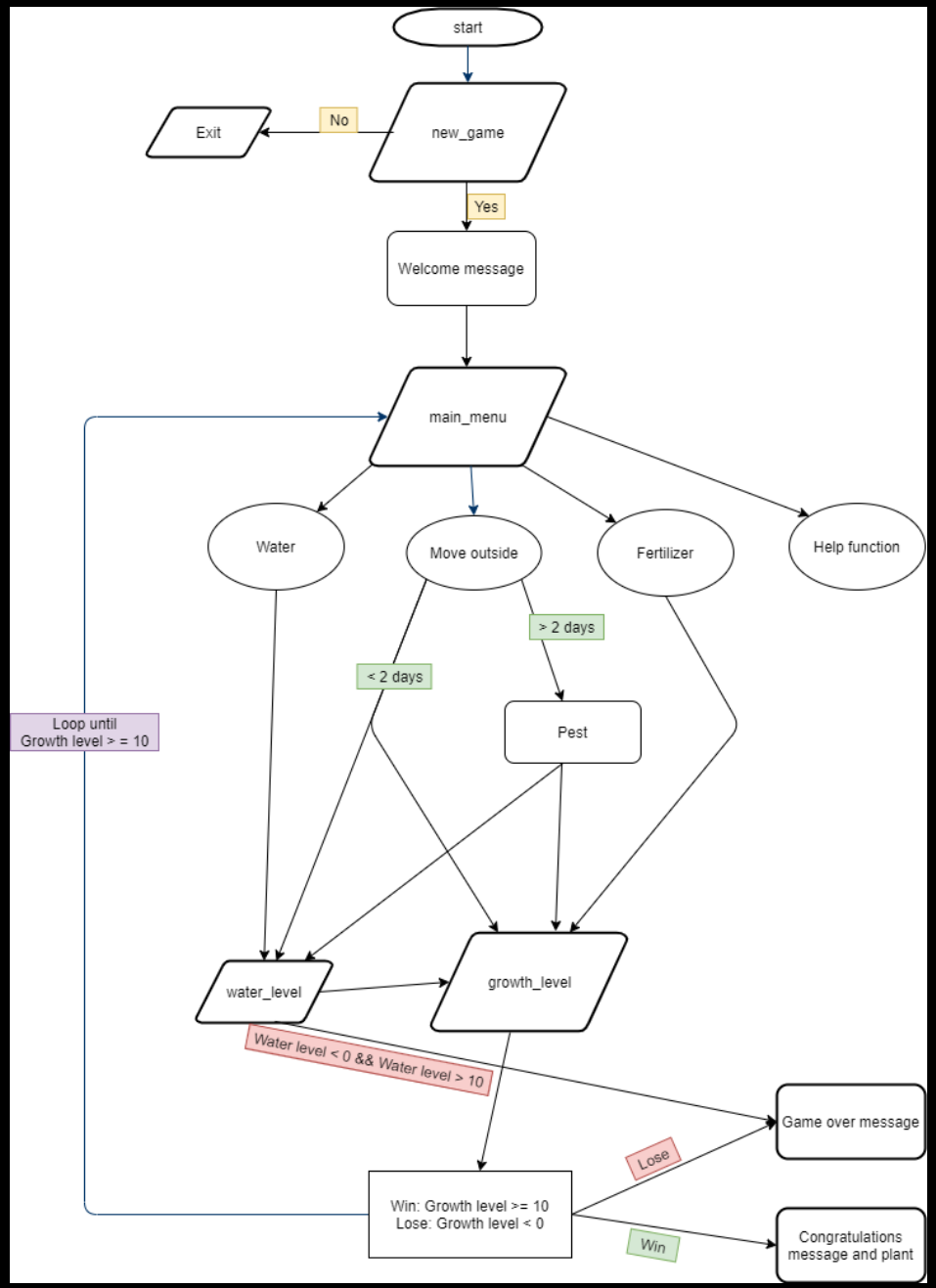
The virtual plant game created using Ruby

Trello board



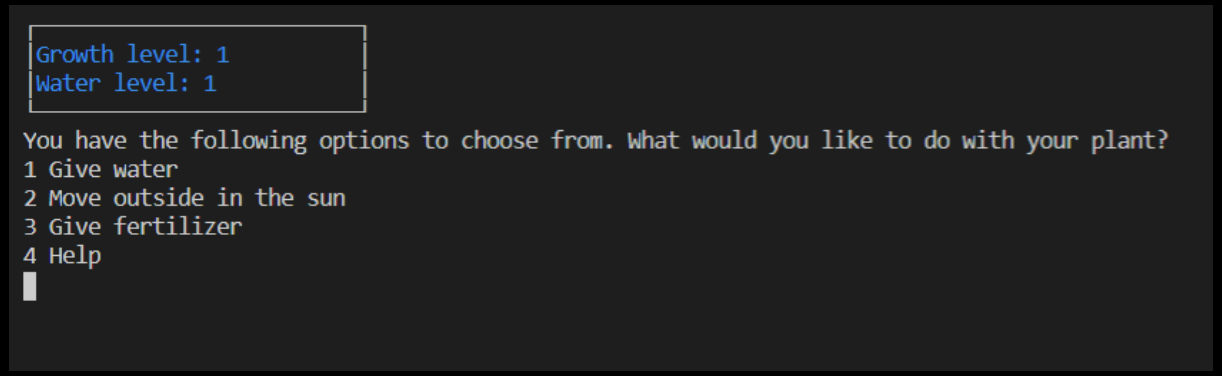
Control flow

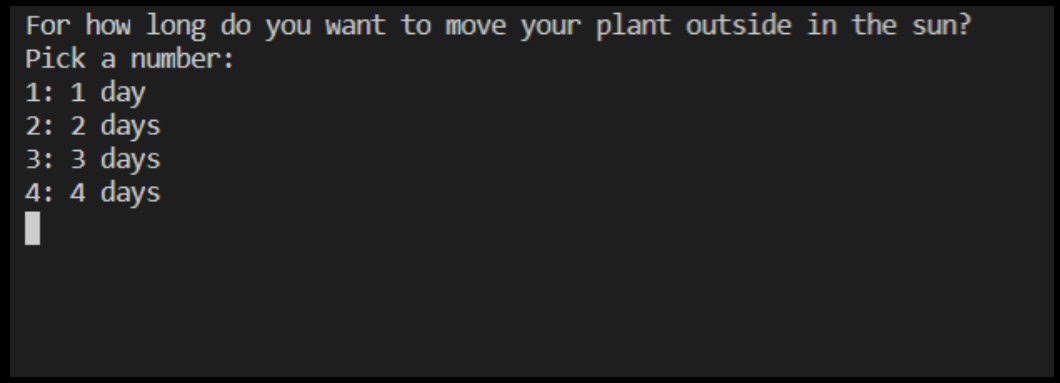


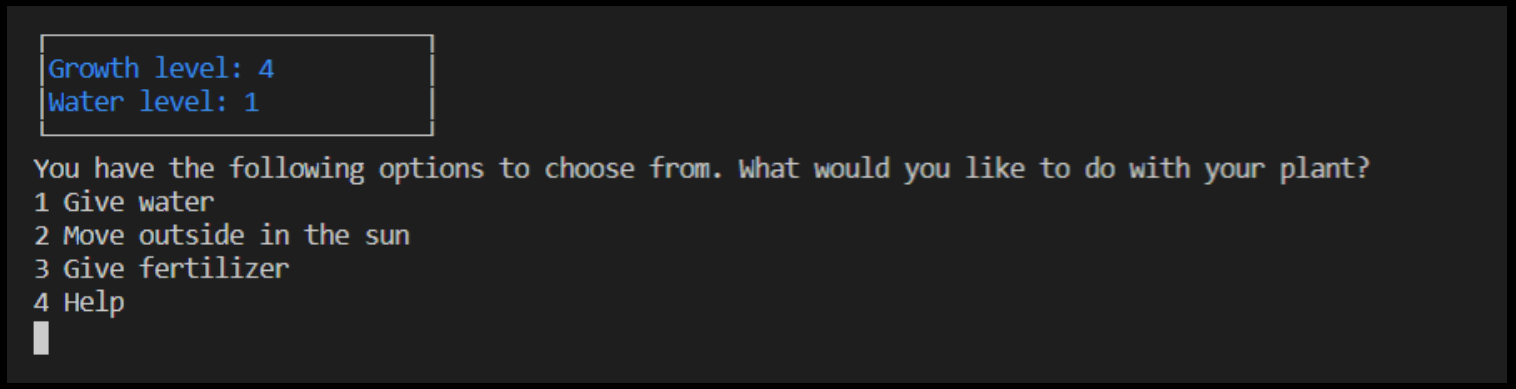
Walk through

- The game will begin with a Welcome message displayed to the user using a Gem
- When a new game is started the user will be introduced to their new seedling with instructions to take care of the plant
- Game objective – take care of your plant until it grows a flower
- Game over when the plant dies

- The plant will have a growth level and water level which is displayed to the user throughout the game – and updated accordingly
- Giving the plant water will increase the water level
- The user will be asked how long they want to move their plant in the sun for
- Water level will decrease according to the corresponding number of days the plant has to spend in the sun and growth level will increase
- Fertilizer = Boolean, increases growth level the most
- Can only be used once





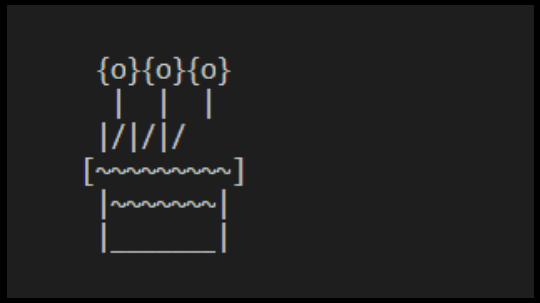


Code snippet from main menu

```ruby
system("clear")
print TTY::Box.frame "Growth level: #{@growth_level}\nWater level: #{@water_level}".colorize(:light_blue)
# gem tty box is used to display the growth_level and water_level to user throughout the game
puts "You have the following options to choose from. What would you like to do with your plant?\n1 Give water\n2 Move outside in the sun
# main menu options
# will loop through up to game_over when user either wins or loses
response = gets.strip.to_i
    if response == 1
        if @water_level < 10
            system("clear")
            types("Well done, you have given your plant some water.")
            # user will be able to give plant water whilst water_level is less than 10
            sleep(2)
            @water_level += 2
            @growth_level += 1
            # water_level increase with 2 each time water is given
            # growth_level increase with 1 each time water is given
            if @water_level == 9
                types("Slow down and check your water level, you dont want your plant to drown!")
                # warning message to user that the water_level is high
                # because the plant will die when water_level goes over 10
                sleep(3)
            end
            if @water_level > 10
                end_game_lose = Game_over.new()
                end_game_lose.lose("over watering.")
                # calls lose method in Game_over class when water_level more than 10
                # gives string as death reason of plant in variable
                @growth_level = -1
                #sets growth_level to minus 1 which will cause loop to break and game to quit
            end
```

Win/lose game

- User will win the game when growth level reaches 10 where after a ASCII picture of the plant with a flower will be printed to the screen
- User will lose when the plant dies
- Game over text will be displayed to the screen using a gem along with the reason why the plant died




```ruby
def lose(reason)
    # method defining lose with a message and GameOver text
    system("clear")
    begin
    types("Oh no! It looks like your plant has died due to ".colorize(:red) + reason.colorize(:red))
    rescue NameError
        puts "Check your method variable name."
    end
    sleep(2)
    system("clear")
    font = TTY::Font.new(:starwars)
    puts font.write("GameOver!").colorize(:red)
    # prints GameOver text to screen using gem
end
```

Test cases

- Decided to use the test unit on my growth and water levels for one on the test cases
- Assert_compare
- Basically just checks that when new plant is created, the growth and water levels are more than 0

```
class Assert_compare_test < Test::Unit::TestCase
    # test that the water and growth level of the plant is more than 0 when new plant is created
    def test_growth_level
        new_plant = Plant_pet.new(1,1)
        # test will pass
        # test will fail when (1,0)
        assert_compare(0, "<", new_plant.growth_level)
    end

    def test_water_level
        new_plant = Plant_pet.new(1,1)
        # test will pass
        # test will fail when (0,1)
        assert_compare(0, "<", new_plant.water_level)
    end
end
```

Questions?