

Projectverdediging Informaticawerktuigen

I-systemen

Ruben Anseeuw en Quinten Steeland

KU Leuven Kulak

Academiejaar 2021 – 2022

Overzicht

- ① From Jsonfile to drawing
- ② Command line arguments
- ③ Random config and progress bar
- ④ History, backups and cache
- ⑤ Docker and Git
- ⑥ Testing and difficulties

Overzicht

- 1 From Jsonfile to drawing
- 2 Command line arguments
- 3 Random config and progress bar
- 4 History, backups and cache
- 5 Docker and Git
- 6 Testing and difficulties

Input

- ▶ getConfigName
- ▶ readJsonFile
- ▶ getConfig

Generate l-system

```
def generateLSystem(axiom, rules, translations, iterations):  
  
    currentString = axiom  
  
    print("Generating l-system...")  
  
    for i in range(iterations):  
        newList = []  
  
        for item in currentString:  
            if item in rules.keys():  
                newList.append(rules[item])  
  
            elif item in translations.keys():  
                newList.append(item)  
  
            else:  
                newList.append(item)  
  
        currentString = ''.join(newList)  
  
    return currentString
```

Drawing L-system

```
def runTurtle(lSystem, translations, showDrawProcess, useProgressbar, iterations):  
  
    if len(lSystem) < 1024:  
        print("Iterating over string:", lSystem)  
    else:  
        print(f"Iterating over long string (length: {len(lSystem)}).")  
  
    screen, turt = turtleInitiate(showDrawProcess, iterations)  
  
    screen, turtlePosition = turtleRunMainLoop(screen, turt, lSystem, translations, useProgressbar)  
  
    return screen, turtlePosition  
  
def turtleRunMainLoop(screen, turt, lSystem, translations, useProgressbar):  
  
    storage = []  
    turtlePosition = 0,0,0,0  
  
    if (len(lSystem) > 500000 and useProgressbar is None) or useProgressbar:  
        for chara in progressbar(lSystem, " ", 100):  
            turtlePosition = turtleRunInstructions(screen, turt, lSystem, translations, storage, turtlePosition, chara)  
    else:  
        for chara in lSystem:  
            turtlePosition = turtleRunInstructions(screen, turt, lSystem, translations, storage, turtlePosition, chara)  
  
    turtle.update()  
  
    return screen, turtlePosition
```

Push and pop

- ▶ stores position, angle and color in storage
- ▶ gets data from storage

```
def turtlePush(screen, turt, storage):  
  
    try:  
        storage += [(turt.pos(), turt.heading(), turt.color()[1])]  
    except:  
        storage = [(turt.pos(), turt.heading(), turt.color()[1])]  
  
def turtlePop(screen, turt, storage):  
  
    try:  
        turt.penup()  
        turt.goto(storage[-1][0])  
        turt.setheading(storage[-1][1])  
        turt.pencolor(storage[-1][2])  
        storage.pop(-1)  
        turt.pendown()  
    except:  
        print("Error: you can't pop more than you push")  
        exit(0)
```

Overzicht

- 1 From Jsonfile to drawing
- 2 Command line arguments
- 3 Random config and progress bar
- 4 History, backups and cache
- 5 Docker and Git
- 6 Testing and difficulties

Help message

All commandline options:

-h or --help	Displays help.
-c or --config <config filename>	Used to give the config file name to the program.
-i or --iterations [amount]	Used to give the amount of iterations to the program.
-e or --export <filename>	Exports the turtle drawing to an eps file.
-s or --output_svg	Convert the eps image from eps to svg.
-sd or --show_draw_process	Use this flag to see the turtle move.
-up or --use_progress_bar	Use this flag if you always want a progressbar.
-np or --no_progress_bar	Use this flag if you want no progressbar (e.g. for speed).
-ca or --close_after_drawing	Exit immediately after drawing.
-rc or --random_config <settings filename>	Generate a random l-System.
-uw or --use_website	Generate a website with the latest lSystem drawing.

Export

```
def exportImage(screen, exportImageName, turtlePosition, configFilename, iterations):

    if exportImageName == "":
        exportImageName = datetime.datetime.now().isoformat(sep="T", timespec='seconds') + "_" + configFilename.rsplit("/", 1)[-1][:-5] + "_" + str(iterations)
    if exportImageName[-4:] != ".eps":
        exportImageName += ".eps"

    path = './Images/'
    completeName = os.path.join(path, exportImageName)
    Cwidth = abs(turtlePosition[2] - turtlePosition[0])*20
    Cheight = abs(turtlePosition[3] - turtlePosition[1])*20
    screen.getcanvas().postscript(x__=_turtlePosition[0]-10, y__=_turtlePosition[3]-10, width=_Cwidth, height=_Cheight, _file=_completeName)
```

Eps to Svg

```
def convertEpsToSvg(forWebsite=True):

    path = "./Images"
    image = getMostRecentFile(path)

    subprocess.run(["ps2pdf", "-dEPSCrop", image, image[:-3]+"pdf"])
    subprocess.run(["pdf2svg", image[:-3]+"pdf", image[:-3]+"svg"])

    if forWebsite:
        subprocess.run(["cp", image[:-3]+"svg", "./Images/mostRecent.svg"])

def getMostRecentFile(path):

    files = os.listdir(path)
    paths = [os.path.join(path, basename) for basename in files]
    return max(paths, key=os.path.getctime)
```

Help message

All commandline options:

-h or --help	Displays help.
-c or --config <config filename>	Used to give the config file name to the program.
-i or --iterations [amount]	Used to give the amount of iterations to the program.
-e or --export <filename>	Exports the turtle drawing to an eps file.
-s or --output_svg	Convert the eps image from eps to svg.
-sd or --show_draw_process	Use this flag to see the turtle move.
-up or --use_progress_bar	Use this flag if you always want a progressbar.
-np or --no_progress_bar	Use this flag if you want no progressbar (e.g. for speed).
-ca or --close_after_drawing	Exit immediately after drawing.
-rc or --random_config <settings filename>	Generate a random l-System.
-uw or --use_website	Generate a website with the latest lSystem drawing.

Use website

```
def updateWebsite():
    """
    function that sets up and runs a website displaying the most recent l-system
    """
    app = Flask(__name__)
    file1 = open("./History/History.txt", "r")
    for line in file1:
        None
    listline = line.split("\t")

    imageopen= open("./Images/mostRecent.svg", 'r')
    imagescript = ""
    for line in imageopen:
        imagescript += line

    @app.route("/")
    def table():
        return "<!DOCTYPE html><html><head><meta charset='UTF-8' /><meta name='<!--"

app.run(host='0.0.0.0', debug=False)
```

Overzicht

- 1 From Jsonfile to drawing
- 2 Command line arguments
- 3 Random config and progress bar**
- 4 History, backups and cache
- 5 Docker and Git
- 6 Testing and difficulties

Random config

```
if settings.get("variables") is None:
    availableVariables = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y"]
else:
    availableVariables = settings.get("variables")

if settings.get("constants") is None:
    availableConstants = ["+", "-", "*", "/", "!", "@", "#", "$", "%", "^", "&", "*", "=", "<", ">", "/", "\\", " ", "?", "{", "}", "(", ")", "[", "]", "<tab>"
else:
    availableConstants = settings.get("constants")

if settings.get("instructions") is None:
    instructions = ["angle", "draw", "forward", "color", "nop"]
else:
    instructions = settings.get("instructions")

if settings.get("instructionsOdds") is None:
    instructionsOdds = [1/len(instructions)]*len(instructions)
else:
    instructionsOdds = settings.get("instructionsOdds")

if settings.get("colors") is None or settings.get("colors").lower() == "random":
    useColors = "random"
if settings.get("colors").lower() == "list":
    useColors = getAllValidColors()
else:
    useColors = settings.get("colors")

if settings.get("usePushPop"):
    pushPopOdds = settings.get("usePushPopOdds")
```

Random config

```
def generateRule(selectedVariables, selectedConstants, variableVsConstantOdds, ruleLenght):  
  
    rule = []  
    while rule == []:  
        for i in range(ruleLenght):  
            if uniform(0, 1) <= variableVsConstantOdds:  
                pick = randint(0, len(selectedVariables)-1)  
                rule += selectedVariables[pick]  
            else:  
                pick = randint(0, len(selectedConstants)-1)  
                rule += selectedConstants[pick]  
  
        if rule != []:  
            count = 0  
            while count < len(rule):  
                if rule[count] == "[":  
                    pick = randint(count+1, len(rule))  
                    rule.insert(pick, "]")  
                count += 1  
  
            strRule = "".join(rule)  
            while "[" in strRule:  
                strRule = strRule.replace("[", "")  
  
    return strRule
```


Random config

```
def generateTranslation(instructions, instructionsOdds, useColors, amountMove, amountAngle, variableVsConstantOdds, lenght, chara=None):
    translation = []
    oddsList = addUpOdds(list(instructionsOdds))
    while translation == []:
        for i in range(lenght):

            if chara == "[" and uniform(0, lenght) <= 1/lenght and "push" not in translation:
                translation.append("push")
                i += 1
            elif chara == "]" and uniform(0, lenght) <= 1/lenght and "pop" not in translation:
                translation.append("pop")
                i += 1
            else:
                tran = chooseInstruction(instructions, oddsList)
                if tran != "nop":
                    translation.append(tran)
                else:
                    if "nop" not in translation:
                        translation.append(tran)
                    else:
                        i -= 1

            if tran in ["color", "draw", "angle", "forward"]:
```

Random config

```
if tran == "color":
    if useColors == "random":
        nextTran = generateRandomColor()
    else:
        nextTran = pickAmountOfItemsFromList(useColors, 1, True)

elif tran == "draw":
    nextTran = round(uniform(amountMove.get("min"), amountMove.get("max")), 3)

elif tran == "angle":
    nextTran = round(uniform(amountAngle.get("min"), amountAngle.get("max")), 3)

elif tran == "forward":
    nextTran = round(uniform(amountMove.get("min"), amountMove.get("max")), 3)

translation.append(nextTran)
i += 1

if chara == "[" and "push" not in translation:
    translation.append("push")

elif chara == "]" and "pop" not in translation:
    translation.append("pop")

return translation
```

Progress bar

```
def progressbar(it, prefix="", size=100, file=sys.stdout):
    count = len(it)

    def show(j):
        x = int(size*j/count)
        file.write("%s[%s%s] %i/%i %i%%s\r" % (prefix, "█"*x, "░"*(size-x), j, count, x, "%"))
        file.flush()

    show(0)
    for i, item in enumerate(it):
        yield item
        show(i+1)
    file.write("\n")
    file.flush()
```

Overzicht

- 1 From Jsonfile to drawing
- 2 Command line arguments
- 3 Random config and progress bar
- 4 History, backups and cache**
- 5 Docker and Git
- 6 Testing and difficulties

Backup

- ▶ -1 = Backup and clear current
- ▶ 0 = clear current
- ▶ number of restore-file

```
try:
    chosenIndex = int(input())
except ValueError:
    print("Not a number.")
    exit(0)

if chosenIndex < -1:
    print("Must be greater than -2.")
    exit(0)

elif chosenIndex > len(allFiles):
    print("That file does not exist.")

if chosenIndex > 0:

    targetFile = allFiles[chosenIndex-1]
    with open(f"./History_backups/{targetFile}_" + str(chosenIndex) + ".txt") as readFile:
        fileContent = readFile.readlines()

    with open("./History/History.txt", "w") as writeFile:
        writeFile.writelines(fileContent)

elif chosenIndex == 0:
    clearHistoryFile()

elif chosenIndex == -1:
    subprocess.run(["./makeBackup.sh"])
    clearHistoryFile()
```

Overzicht

- 1 From Jsonfile to drawing
- 2 Command line arguments
- 3 Random config and progress bar
- 4 History, backups and cache
- 5 Docker and Git**
- 6 Testing and difficulties

Docker

```
FROM ubuntu:20.04

ENV TZ=Europe/Brussels

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

RUN apt-get update

RUN apt-get -y install python3 python3-pip python3-tk python3-markupsafe ghostscript pdf2svg

COPY requirements.txt /l-systems/


















WORKDIR /l-systems

RUN cat requirements.txt

RUN pip install -r requirements.txt

COPY . /l-systems

ENTRYPOINT ["python3", "lSystem.py"]
```

 .github/workflows	Update tests.yml
 Examples	Merge pull request #33 from Quinten-Steelnd-KuLeuven/Dev
 History	Update readme and about info
 History_backups	Update readme and about info
 Images	Remove unnecesairy file
 Random_configs	Add examples
 Unit_tests	Update readme and about info
 .dockerignore	Fix docker history
 .gitignore	Move history file
 Dockerfile	Fix docker history
 History.txt	Move history file
 README.md	Merge pull request #33 from Quinten-Steelnd-KuLeuven/Dev
 Website.py.txt	Move history file
 crontabSetup.sh	split main file & random config integration
 generateRandomConfig.py	Bugfix
 ISystem.py	Cleanup
 loop.py	Add examples

Overzicht

- ① From Jsonfile to drawing
- ② Command line arguments
- ③ Random config and progress bar
- ④ History, backups and cache
- ⑤ Docker and Git
- ⑥ Testing and difficulties

Testing

```
def test_two_1():
    assert lSystem_test_two(1) == "F+F-F-FF+F+F-F+F-F-FF+F+F-F+F-F-FF+F+F-F+F-F-FF+F+F-F"

def test_two_2():
    assert lSystem_test_two(2) == "F+F-F-FF+F+F-F+F-F-FF+F+F-F+F-F-FF+F+F-F+F-F-FF+F+F-F"

# == test 3 ==
def test_three_1():
    assert lSystem_test_three(0) == "A"

def test_three_2():
    assert lSystem_test_three(1) == "AB"

def test_three_3():
    assert lSystem_test_three(2) == "ABA"

def test_three_4():
    assert lSystem_test_three(3) == "ABAAB"

def test_three_5():
    assert lSystem_test_three(4) == "ABAABABA"

def test_three_6():
    assert lSystem_test_three(5) == "ABAABABAABAAB"
```

Difficulties

- ▶ crontab

