Постулат. 2022. №1 ISSN 2414-4487
УДК 004.624

Реализация песочных часов на языке программирования Python

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс примитивных песочных часов с реальной физикой объектов. Для создания используется язык программирования Python и его стандартная библиотека. Созданное приложение служит наглядным пояснением как можно взаимодействовать с объектами в рамках реальных движений физических объектов.
Ключевые слова: Python, анимация

Implementing an hourglass in the Python programming language

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of a primitive hourglass with real physics of objects. For creation, the Python programming language and its standard library are used. The created application serves as a visual explanation of how you can interact with objects within the framework of real movements of physical objects.
Keywords: Python, animation

Эта программа визуализации имеет грубый физический движок, который имитирует песок, падающий через маленькое отверстие песочных часов. Песок скапливается в нижней половине песочных часов; затем песочные часы переворачиваются, и процесс повторяется.

Цель исследования – написать приложение имитации работы песочных часов на языке программирования Python.

Ранее этим вопросом интересовались Т.С. Мизяев, Н.В. Бужинская развивали тему «Разработка развивающей игры на языке программирования python» [1] в которой описывается, что в настоящее время язык Python приобретает все большую популярность за счет универсальности своих функциональных возможностей. Этот язык может применяться как для работы с базами данных, так и разработки сайтов и других программных продуктов. В данной статье рассмотрен способ создания развивающей игры на языке программирования Python с использованием библиотеки Pygame. Д.Ф. Ахметгалеева, Г.Р. Галиаскарова с темой «Создание игры-змейки на языке программирования python "гомер ест пончики"» [2], а подробнее про

то как создать с помощью библиотеки tkinter, игру змейку на языке программирования Python. Мой выбор пал на персонажа Гомера, который поедает пончики. В качестве головы змеи - голова Гомера, в качестве яблок - пончики, которые так любит персонаж. В процессе игры, с каждым съеденным пончиком, персонаж удлиняется и скорость игры увеличивается. Цель игры - съестькак можно больше пончиков. С.А. Ханфенова, А.М. Узденова опубликовали статью «Изучений базовых понятий языка python на примере создания простой игры» [3] в статье рассмотрена методика изучения базовых структур популярного языка программирования Python, таких как: инструкции, модули, блоки, операторы. Изучение через решение конкретных практических задач, как правило, более эффективно, поэтому рассматривается написание игры «Угадай число». Код данной игры содержит все вышеперечисленные базовые понятия языка программирования.

Программа песочных часов реализует элементарный физический движок. Физический движок — это программное обеспечение, которое имитирует физические объекты, падающие под действием гравитации, сталкивающиеся друг с другом и движущиеся в соответствии с законами физики. Физические движки, используются в видеоиграх, компьютерной анимации и научных симуляциях. Каждая «песчинка» проверяет, пусто ли пространство под ней, и перемещается вниз, если это так. В противном случае проверяет, может ли она двигаться вниз и влево или вниз и вправо.

```
import random
import sys
import time

import bext

pause_span = 0.2
fall_break = 50
width_end = 79
height_end = 25
X = 0
Y = 1
sand = chr(9617)
bar = chr(9608)
clock = set()

for i in range(18, 37):
    clock.add((i, 1))
    clock.add((i, 23))

for i in range(1, 5):
    clock.add((18, i))
    clock.add((36, i))
    clock.add((18, i + 19))
    clock.add((36, i + 19))

for i in range(8):
    clock.add((19 + i, 5 + i))
    clock.add((35 - i, 5 + i))
    clock.add((25 - i, 13 + i))
    clock.add((29 + i, 13 + i))
```

```
init_bar = set()
for y in range(8):
    for x in range(19 + y, 36 - y):
        init_bar.add((x, y + 4))


def main():
    bext.fg('yellow')
    bext.clear()
    bext.goto(0, 0)
    print('Ctrl-C to quit.', end='')
    for wall in clock:
        bext.goto(wall[X], wall[Y])
        print(bar, end='')

    while True:
        allSand = list(init_bar)
        for sand in allSand:
            bext.goto(sand[X], sand[Y])
            print(sand, end='')

        runHourglassSimulation(allSand)


def runHourglassSimulation(allSand):
    while True:
        random.shuffle(allSand)
        sandMovedOnThisStep = False

        for i, sand in enumerate(allSand):
            if sand[Y] == height_end - 1:
                continue

            noSandBelow = (sand[X], sand[Y] + 1) not in allSand
            noWallBelow = (sand[X], sand[Y] + 1) not in clock
            canFallDown = noSandBelow and noWallBelow

            if canFallDown:
                bext.goto(sand[X], sand[Y])
                print(' ', end='')
                bext.goto(sand[X], sand[Y] + 1)
                print(sand, end='')
                allSand[i] = (sand[X], sand[Y] + 1)
                sandMovedOnThisStep = True
            else:
                belowLeft = (sand[X] - 1, sand[Y] + 1)
                noSandBelowLeft = belowLeft not in allSand
                noWallBelowLeft = belowLeft not in clock
                left = (sand[X] - 1, sand[Y])
                noWallLeft = left not in clock
                notOnLeftEdge = sand[X] > 0
                canFallLeft = (noSandBelowLeft and noWallBelowLeft and
noWallLeft and notOnLeftEdge)
                belowRight = (sand[X] + 1, sand[Y] + 1)
                noSandBelowRight = belowRight not in allSand
                noWallBelowRight = belowRight not in clock
                right = (sand[X] + 1, sand[Y])
                noWallRight = right not in clock
                notOnRightEdge = sand[X] < width_end - 1
                canFallRight = (noSandBelowRight and noWallBelowRight and
noWallRight and notOnRightEdge)
                fallingDirection = None
```

```
                if canFallLeft and not canFallRight:
                    fallingDirection = -1
                elif not canFallLeft and canFallRight:
                    fallingDirection = 1
                elif canFallLeft and canFallRight:
                    fallingDirection = random.choice((-1, 1))

                if random.random() * 100 <= fall_break:
                    belowTwoLeft = (sand[X] - 2, sand[Y] + 1)
                    noSandBelowTwoLeft = belowTwoLeft not in allSand
                    noWallBelowTwoLeft = belowTwoLeft not in clock
                    notOnSecondToLeftEdge = sand[X] > 1
                    canFallTwoLeft = (
                            canFallLeft and noSandBelowTwoLeft and
noWallBelowTwoLeft and notOnSecondToLeftEdge)
                    belowTwoRight = (sand[X] + 2, sand[Y] + 1)
                    noSandBelowTwoRight = belowTwoRight not in allSand
                    noWallBelowTwoRight = belowTwoRight not in clock
                    notOnSecondToRightEdge = sand[X] < width_end - 2
                    canFallTwoRight = (
                            canFallRight and noSandBelowTwoRight and
noWallBelowTwoRight and notOnSecondToRightEdge)
                    if canFallTwoLeft and not canFallTwoRight:
                        fallingDirection = -2
                    elif not canFallTwoLeft and canFallTwoRight:
                        fallingDirection = 2
                    elif canFallTwoLeft and canFallTwoRight:
                        fallingDirection = random.choice((-2, 2))

                if fallingDirection == None:
                    continue

                bext.goto(sand[X], sand[Y])
                print(' ', end='')
                bext.goto(sand[X] + fallingDirection, sand[Y] + 1)
                print(sand, end='')
                allSand[i] = (sand[X] + fallingDirection, sand[Y] + 1)
                sandMovedOnThisStep = True

        sys.stdout.flush()
        time.sleep(pause_span)
        if not sandMovedOnThisStep:
            time.sleep(2)
            for sand in allSand:
                bext.goto(sand[X], sand[Y])
                print(' ', end='')
            break


if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        sys.exit()
```

Результат работы кода можно увидеть на рисунке 1.

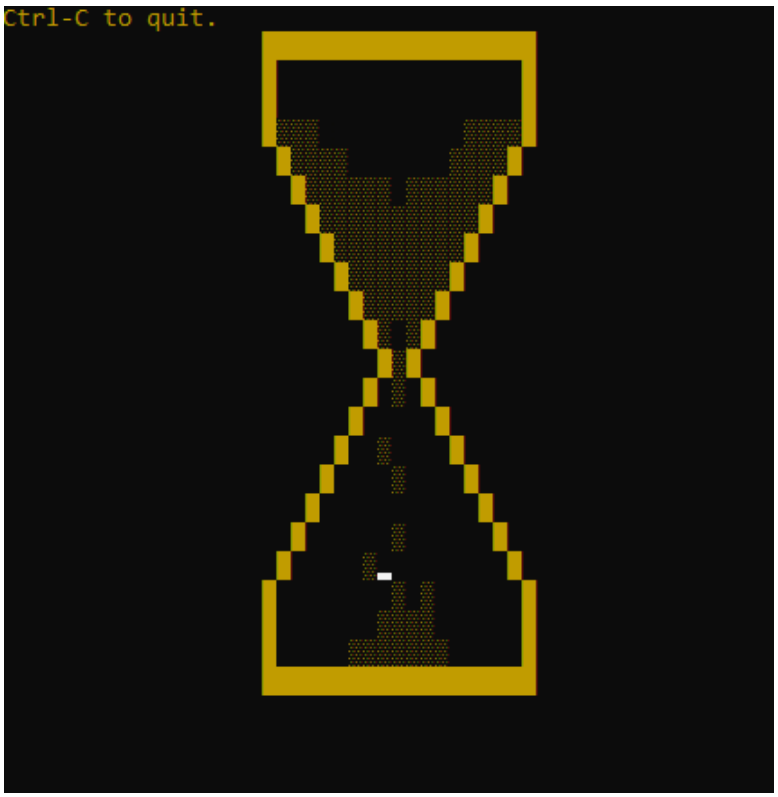

Рис. 1 Симуляция падающего песка

Вывод

В этой статье было приведено описание приложения, имитирующего реальную физику объектов на примере песочных часов. Благодаря этому приложению можно разобраться в работе анимации на языке программирования Pythonи создании симуляции объектов.

Библиографический список

1. Мизяев Т.С., Бужинская Н.В. Разработка развивающей игры на языке программирования python //. В сборнике: Стимулирование инновационного развития общества в стратегическом периоде. сборник статей Международной научно-практической конференции. 2018. С. 42-45. URL: https://www.elibrary.ru/item.asp?id=32657770 (Дата обращения: 05.01.2022)
2. Ахметгалеева Д.Ф., Галиаскарова Г.Р. Создание игры-змейки на языке программирования python "гомер ест пончики" // Colloquium-journal. 2019. № 17-2 (41). С. 24-27. URL: https://www.elibrary.ru/item.asp?id=39246704 (Дата обращения: 05.01.2022)
3. Ханфенова С.А., Узденова А.М. Изучений базовых понятий языка python на примере создания простой игры // В сборнике: Молодежь. Наука. Образование. Сборник научных трудов по материалам конкурса научных работ "Студент-исследователь". Карачаевск, 2020. С. 199-204. URL: https://www.elibrary.ru/item.asp?id=44463179 (Дата обращения: 05.01.2022)