УДК 004

Разработка игры 2048 на фреймворкеKivy

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
студент

Лучанинов Дмитрий Васильевич
Приамурский государственный университет имени Шолом-Алейхема
старший преподаватель кафедры информационных систем, математики и методик преподавания

Аннотация
В данной статье продемонстрирован процесс создания игры 2048 с помощью фреймворкаKivyи языка программирования Python.
Ключевыеслова: Python, Kivy.

Game development framework, 2048 by Kivy

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Luchaninov Dmitry Vasilyevich
Sholom-Aleichem Priamursky State University
Senior lecturer of the Department of Information Systems, Mathematics and teaching methods

Abstract
This article demonstrated the process of creating the game in 2048 with the help of Kivy framework and the Python programming language.
Keywords: Python, Kivy.

Фреймворк Kivy достаточно недавно появился на рынке и принес с собой много интересных вещей. Это возможность делать один код для всех устройств, Windows, linux, Mac, Android, iOS и взаимодействие с низкоуровневыми датчиками [1].

Целью данной статьи является создание игры 2048 на фреймворкеKivy.

Для ознакомления с языком программирования Python можно познакомится с различными исследованиями. Г.С.Сейдаметов продемонстрировал особенности использования языка программирования python в подготовке будущих инженеров-программистов [2]. Э.А.Усеинов продемонстрировал использование объектно-ориентированного программирования в рамках дисциплины «язык программирования python»

[3]. Д.В.Шкодырев продемонстрировал использование языка программирования python и системы компьютерной алгебры sympy на факультативных занятиях по теории чисел»[4]. В.В.Бурков и И.В.Бантыш показали использование языка программирования python для решения задач математического моделирования электромеханических систем [5].

Скрипт начинается с импорта нужных библиотек.

```
   from __future__ import division
   import random

   fromkivy.animation import Animation
   fromkivy.app import App
   fromkivy.core.window import Window, Keyboard
   fromkivy.graphics import Color, BorderImage
   fromkivy.properties import ListProperty, NumericProperty
   fromkivy.uix.widget import Widget
   fromkivy.utils import get_color_from_hex
   fromkivy.vectorimportVector
```

Дальше идут настройки цвета фона, клавиш управления.

```
   spacing = 15
   colors = (
     'eee4da', 'ede0c8', 'f2b179', 'f59563',
     'f67c5f', 'f65e3b', 'edcf72', 'edcc61',
  'edc850', 'edc53f', 'edc22e')
   tile_colors = {2 ** i: color for i, color in
enumerate(colors, start=1)}
   key_vectors = {
Keyboard.keycodes['up']: (0, 1),
Keyboard.keycodes['right']: (1, 0),
Keyboard.keycodes['down']: (0, -1),
Keyboard.keycodes['left']: (-1, 0),
}
```

Класс `Tile` содержит в себе настройки иконок игрового поля.

```
class Tile(Widget):
font_size = NumericProperty(24)
number = NumericProperty(2)
color = ListProperty(get_color_from_hex(tile_colors[2]))
number_color = ListProperty(get_color_from_hex('776e65'))
def __init__(self, number=2, **kwargs):
super(Tile, self).__init__(**kwargs)
self.font_size = 0.5 * self.width
self.number = number
self.update_colors()
defupdate_colors(self):
self.color = get_color_from_hex(tile_colors[self.number])
ifself.number> 4:
```

```
self.number_color = get_color_from_hex('f9f6f2')
def resize(self, pos, size):
self.pos = pos
self.size = size
self.font_size = 0.5 * self.width
```

Функция `all_cells` добавляет ячейки на игровое поле.

```
def all_cells(flip_x=False, flip_y=False):
for x in (reversed(range(4)) if flip_x else range(4)):
for y in (reversed(range(4)) if flip_y else range(4)):
yield (x, y)
```

Класс `Board` отвечает за построение, управление и логику игровых фишек.

```
class Board(Widget):
    b = None
moving = False
def __init__(self, **kwargs):
super(Board, self).__init__(**kwargs)
self.resize()
def reset(self):
self.b = [[None for i in range(4)] for j in range(4)]
self.new_tile()
self.new_tile()
defnew_tile(self, *args):
empty_cells = [(x, y) for x, y in all_cells()
ifself.b[x][y] is None]
if not empty_cells:
print('Game over (tentative: no cells)')
return
x, y = random.choice(empty_cells)
tile = Tile(pos=self.cell_pos(x, y),
size=self.cell_size)
self.b[x][y] = tile
self.add_widget(tile)
iflen(empty_cells) == 1 and self.is_deadlocked():
print('Game over (board is deadlocked)')
self.moving = False
defis_deadlocked(self):
for x, y in all_cells():
ifself.b[x][y] is None:
return False
number = self.b[x][y].number
ifself.can_combine(x + 1, y, number) or \
self.can_combine(x, y + 1, number):
return False
return True
def move(self, dir_x, dir_y):
ifself.moving:
return
```

```
dir_x = int(dir_x)
dir_y = int(dir_y)
forboard_x, board_y in all_cells(dir_x> 0, dir_y> 0):
tile = self.b[board_x][board_y]
if not tile:
continue
x, y = board_x, board_y
whileself.can_move(x + dir_x, y + dir_y):
self.b[x][y] = None
                x += dir_x
                y += dir_y
self.b[x][y] = tile
ifself.can_combine(x + dir_x, y + dir_y, tile.number):
self.b[x][y] = None
                x += dir_x
                y += dir_y
self.remove_widget(self.b[x][y])
self.b[x][y] = tile
tile.number *= 2
if (tile.number == 2048):
print('You win the game')
tile.update_colors()
if x == board_x and y == board_y:
continue  # nothing has happened
anim = Animation(pos=self.cell_pos(x, y),
duration=0.25, transition='linear')
if not self.moving:
anim.on_complete = self.new_tile
self.moving = True
anim.start(tile)
defvalid_cell(self, board_x, board_y):
return (board_x>= 0 and board_y>= 0 and
board_x<= 3 and board_y<= 3)
defcan_move(self, board_x, board_y):
return (self.valid_cell(board_x, board_y) and
self.b[board_x][board_y] is None)
defcan_combine(self, board_x, board_y, number):
return (self.valid_cell(board_x, board_y) and
self.b[board_x][board_y] is not None and
self.b[board_x][board_y].number == number)
defcell_pos(self, board_x, board_y):
return  (self.x  +  board_x  *  (self.cell_size[0]  +  spacing)  +
   spacing,
self.y + board_y * (self.cell_size[1] + spacing) + spacing)
def resize(self, *args):
self.cell_size = (0.25 * (self.width - 5 * spacing), ) * 2
self.canvas.before.clear()
withself.canvas.before:
BorderImage(pos=self.pos, size=self.size, source='board.png')
Color(*get_color_from_hex('ccc0b4'))
forboard_x, board_y in all_cells():
BorderImage(pos=self.cell_pos(board_x, board_y),
size=self.cell_size, source='cell.png')
```

```
if not self.b:
return
forboard_x, board_y in all_cells():
tile = self.b[board_x][board_y]
if tile:
tile.resize(pos=self.cell_pos(board_x, board_y),
size=self.cell_size)
on_pos = resize
on_size = resize
defon_key_down(self, window, key, *args):
if key in key_vectors:
self.move(*key_vectors[key])
defon_touch_up(self, touch):
        v = Vector(touch.pos) - Vector(touch.opos)
ifv.length() < 20:
return
if abs(v.x) > abs(v.y):
v.y = 0
else:
v.x = 0
self.move(*v.normalize())
```

Класс запуска игры.

```
classGameApp(App):
defon_start(self):
board = self.root.ids.board
board.reset()
Window.bind(on_key_down=board.on_key_down)
```

Создаем экземпляр класса и вызываем метод run.

```
if __name__ == '__main__':
Window.clearcolor = get_color_from_hex('faf8ef')
GameApp().run()
```

Результат можно посмотреть на рис. 1 и рис. 2.

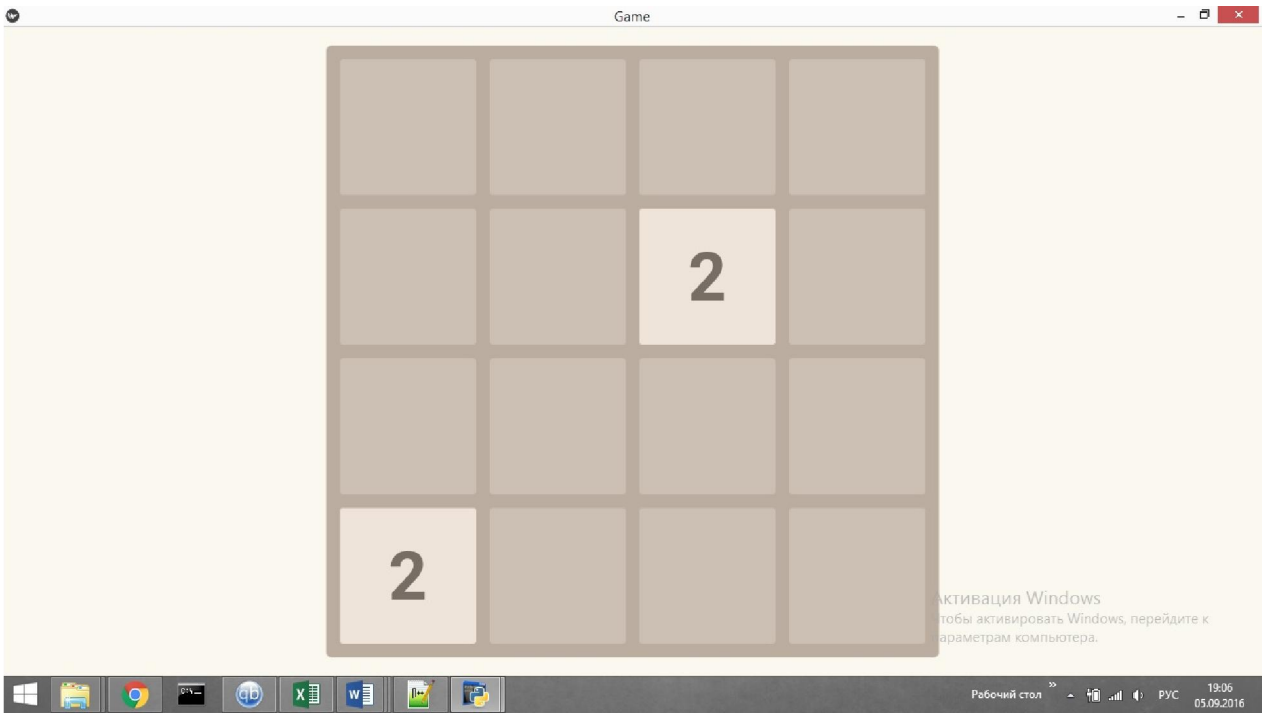

Рисунок 1

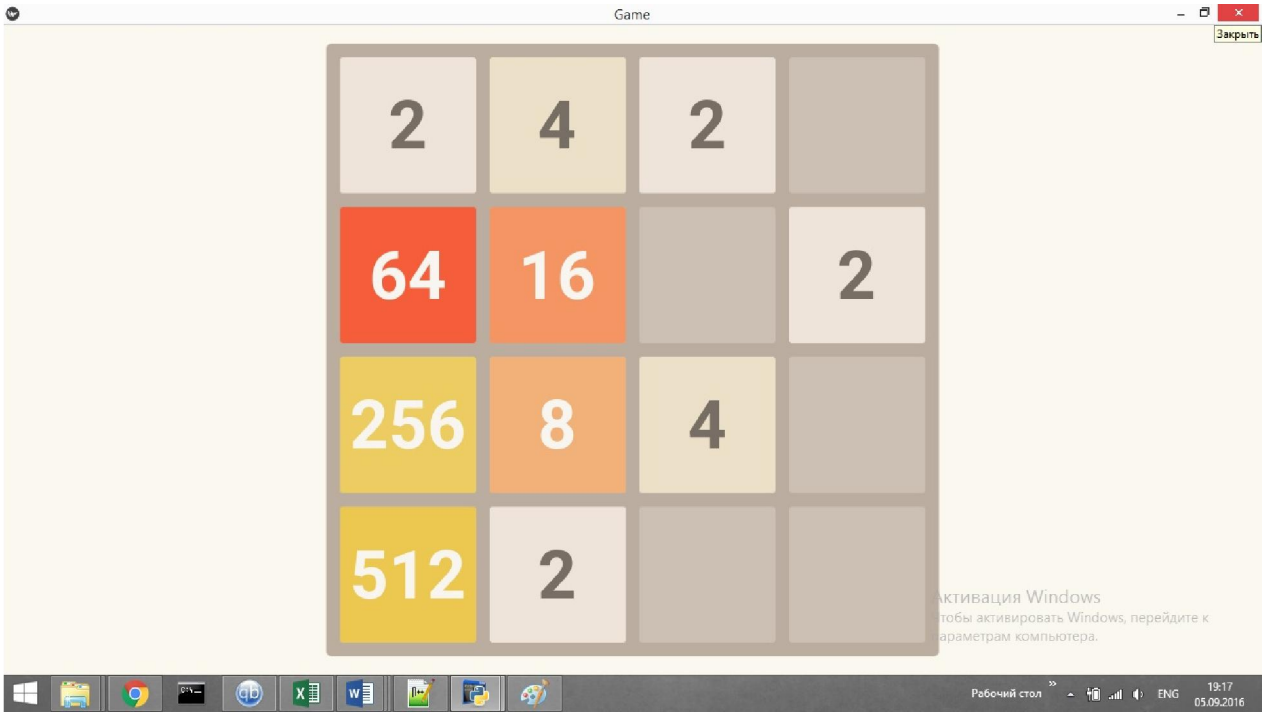

Рисунок 2

Вывод: Разработана игра 2048 на языке программирования Pythonc использованием фреймворка Kivy.

Библиографический список

1. Фреймворк Kivy. [Электронный ресурс]. URL: https://kivy.org(дата обращения: 5.09.2016)
2. Сейдаметов Г. С. Особенности использования языка программирования

python в подготовке будущих инженеров-программистов // В сборнике: INTERNATIONAL SCIENTIFIC REVIEW Издательство: Олимп (Иваново) С. 50-51

3. Усеинов Э.А. Объектно-ориентированное программирование в рамках дисциплины «язык программирования python» // В сборнике: Ученые записки крымского инженерно-педагогического университета Издательство: Государственное бюджетное образовательное учреждение высшего образования Республики Крым "Крымский инженерно-педагогический университет" (Симферополь) С. 157-160
4. Шкодырев Д.В. Использование языка программирования python 3 и системы компьютерной алгебры sympyна факультативных занятиях по теории чисел // В сборнике: Математическое образование в школе и вузе: теория и практика (mathedu-2015) материалы V Международной научно-практической конференции. Отв. ред. Н.В. Тимербаева. 2015. С. 287-288
5. Бурков В. В., Бантыш И. В. Использование языка программирования python для решения задач математического моделирования электромеханических систем // В сборнике: Современные проблемы автоматизации и управления в энергетике и машиностроении сборник научных трудов международной научно-практической конференции. Цикл «Автоматизация и управления» кафедры «Технологии машиностроения» (ТМС) Пензенского государственного технологического университета. 2015. С. 226-234