Python – интерпретируемый язык, и по этой причине он не отличается высокой производительностью, которая необходима для сложных игр с гиперреалистичной графикой. Тем не менее есть несколько способов оптимизации, которые значительно ускоряют выполнение Python-кода, что позволяет писать на Питоне вполне приличные игры. Оптимизацию необязательно делать самостоятельно – в значительной степени эту задачу берут на себя специальные фреймворки и библиотеки. Фреймворков и библиотек для разработки игр на основе Python довольно много, вот самые популярные:

- Pygame
- PyKyra
- Pyglet
- Panda3D
- Kivy
- PyopenGL

Мы остановимся на самой популярной библиотеке, **Pygame** – она отлично подходит для начинающих разработчиков, и к тому же часто используется для быстрого прототипирования игр. На официальном сайте Pygame есть каталог игр, созданных с помощью библиотеки. Еще примеры игр на
Pygame не входит в стандартную поставку Python, для установки библиотеки выполните:

```
pip install pygame
```

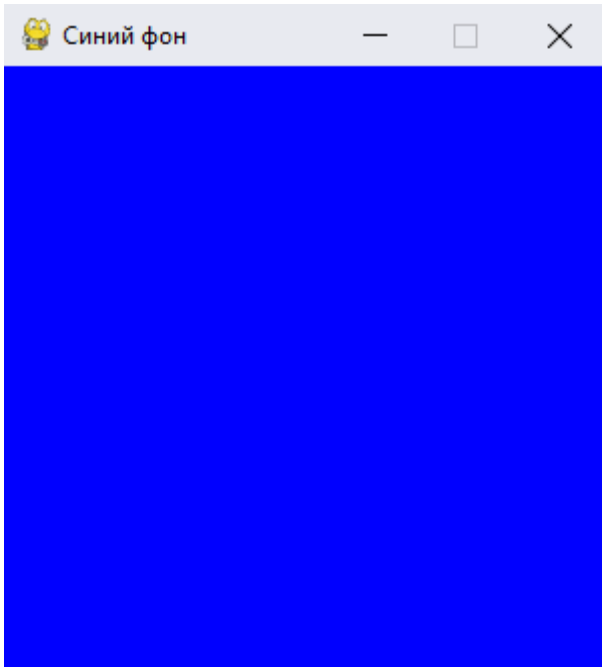
Рассмотрим базовые концепции Pygame – рисование, движение объектов, покадровую анимацию, обработку событий, обновление счетчика, обнаружение столкновения.

Окно и главный цикл приложения

Создание любого приложения на базе Pygame начинается с импорта и инициализации библиотеки. Затем нужно определить параметры окна, и по желанию – задать цвет (или изображение) фона:

```
import pygame

# инициализируем библиотеку Pygame
pygame.init()

# определяем размеры окна
window_size = (300, 300)

# задаем название окна
pygame.display.set_caption("Синий фон")

# создаем окно
screen = pygame.display.set_mode(window_size)

# задаем цвет фона
background_color = (0, 0, 255)  # синий

# заполняем фон заданным цветом
screen.fill(background_color)
```

```
# обновляем экран для отображения изменений
pygame.display.flip()

# показываем окно, пока пользователь не нажмет кнопку "Закрыть"
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```



Цикл **while True** играет роль главного цикла программы – в нем происходит отслеживание событий приложения и действий пользователя.
Функция **pygame.quit()** завершает работу приложения, и ее можно назвать противоположностью функции **pygame.init()**. Для завершения Python-процесса используется **exit()**, с той же целью можно использовать **sys.exit()**, но ее нужно импортировать в начале программы: import sys.
В качестве фона можно использовать изображение:

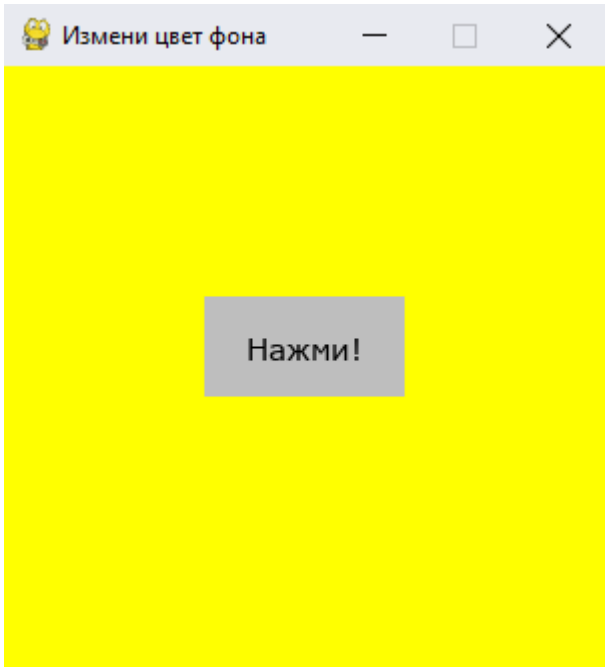
```
import pygame

pygame.init()

window_size = (400, 400)
screen = pygame.display.set_mode(window_size)
pygame.display.set_caption("Peter the Piglet")

# загружаем изображение
background_image = pygame.image.load("background.png")

# подгоняем масштаб под размер окна
background_image = pygame.transform.scale(background_image, window_size)

# накладываем изображение на поверхность
screen.blit(background_image, (0, 0))

pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            pygame.quit()
            exit()
```

Обработку событий (нажатий клавиш и кликов) в Pygame реализовать очень просто – благодаря встроенным функциям. Приведенный ниже код изменяет цвет фона после клика по кнопке. Обратите внимание, что в Pygame можно задавать цвет несколькими способами:

```
import pygame

pygame.init()
pygame.display.set_caption('Измени цвет фона')
window_surface = pygame.display.set_mode((300, 300))
background = pygame.Surface((300, 300))
background.fill(pygame.Color('#000000'))

color_list = [
    pygame.Color('#FF0000'),   # красный
    pygame.Color('#00FF00'),   # зеленый
    pygame.Color('#0000FF'),   # синий
    pygame.Color('#FFFF00'),   # желтый
    pygame.Color('#00FFFF'),   # бирюзовый
    pygame.Color('#FF00FF'),   # пурпурный
    pygame.Color('#FFFFFF')    # белый
]

current_color_index = 0

button_font = pygame.font.SysFont('Verdana', 15) # используем шрифт Verdana
button_text_color = pygame.Color("black")
button_color = pygame.Color("gray")
button_rect = pygame.Rect(100, 115, 100, 50)
button_text = button_font.render('Нажми!', True, button_text_color)

while True:
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        elif event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
            if button_rect.collidepoint(event.pos):
                current_color_index = (current_color_index + 1) % len(color_list)
                background.fill(color_list[current_color_index])

        window_surface.blit(background, (0, 0))
        pygame.draw.rect(window_surface, button_color, button_rect)
        button_rect_center = button_text.get_rect(center=button_rect.center)
        window_surface.blit(button_text, button_rect_center)
        pygame.display.update()
```



Как очевидно из приведенного выше примера, основной цикл Pygame приложения состоит из трех повторяющихся действий:

- Обработка событий (нажатий клавиш или кнопок).
- Обновление состояния.
- Отрисовка состояния на экране.

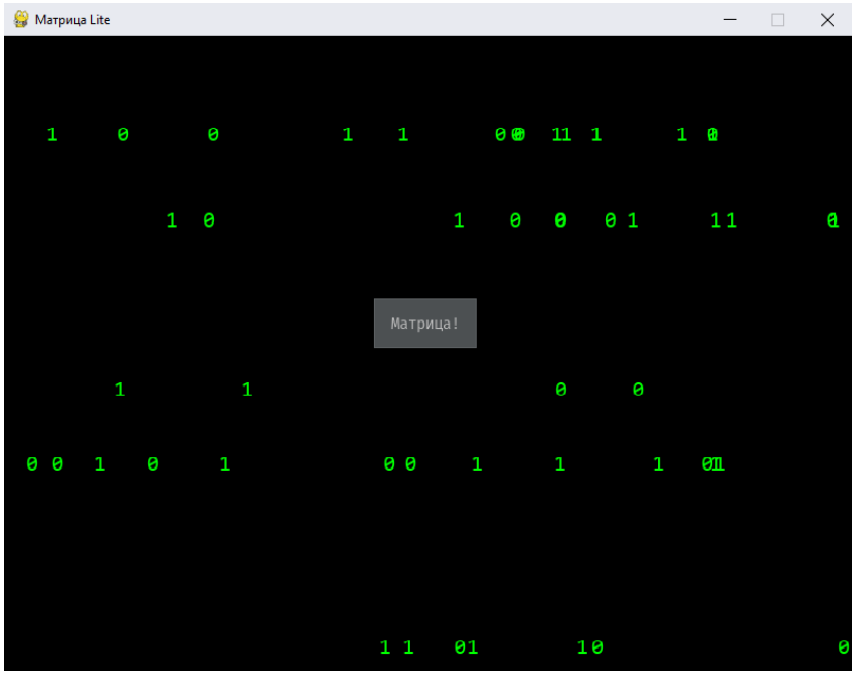
GUI для PyGame

Pygame позволяет легко и быстро интегрировать в проект многие нужные вещи – шрифты, звук, обработку событий, – однако не имеет встроенных виджетов для создания кнопок, лейблов, индикаторов выполнения и других подобных элементов интерфейса. Эту проблему разработчик должен решать либо самостоятельно (нарисовать прямоугольник, назначить ему функцию кнопки), либо с помощью дополнительных GUI-библиотек. Таких библиотек несколько, к самым популярным относятся:

- Pygame GUI
- Thorpy
- PGU

Вот простой пример использования **Pygame GUI** – зеленые нули и единицы падают вниз в стиле «Матрицы»:

```
import pygame
import pygame_gui
import random

window_size = (800, 600)
window = pygame.display.set_mode(window_size)
pygame.display.set_caption('Матрица Lite')
pygame.init()
gui_manager = pygame_gui.UIManager(window_size)

font = pygame.font.SysFont('Consolas', 20)
text_color = pygame.Color('green')
text_symbols = ['0', '1']
text_pos = [(random.randint(0, window_size[0]), 0) for i in range(50)]
text_speed = [(0, random.randint(1, 5)) for i in range(50)]
text_surface_list = []

button_size = (100, 50)
button_pos = (350, 250)
button_text = 'Матрица!'

button = pygame_gui.elements.UIButton(
    relative_rect=pygame.Rect(button_pos, button_size),
    text=button_text,
    manager=gui_manager
)

while True:
    time_delta = pygame.time.Clock().tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()

        if event.type == pygame_gui.UI_BUTTON_PRESSED:
            text_surface_list = []
            for i in range(50):
                text_symbol = random.choice(text_symbols)
                text_surface = font.render(text_symbol, True, text_color)
                text_surface_list.append(text_surface)

        gui_manager.process_events(event)

    gui_manager.update(time_delta)

    window.fill(pygame.Color('black'))

    for i in range(50):
        text_pos[i] = (text_pos[i][0], text_pos[i][1] + text_speed[i][1])
        if text_pos[i][1] > window_size[1]:
            text_pos[i] = (random.randint(0, window_size[0]), -20)
        if len(text_surface_list) > i:
            window.blit(text_surface_list[i], text_pos[i])

    gui_manager.draw_ui(window)
    pygame.display.update()
```



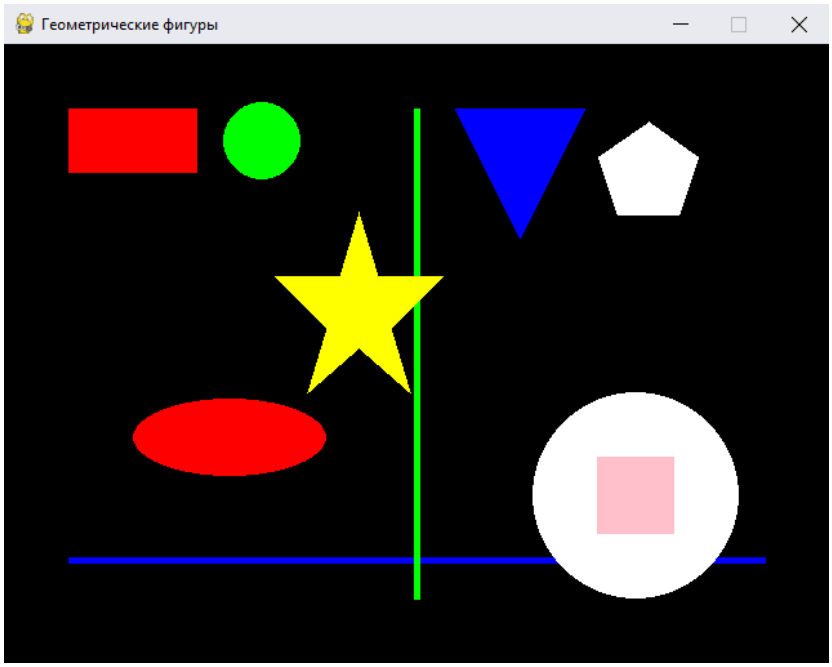
Рисование

В Pygame есть функции для простого рисования геометрических фигур – прямоугольников, окружностей, эллипсов, линий, многоугольников.
Вот пример:

```
import pygame
import math

pygame.init()
screen_width = 640
screen_height = 480
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Геометрические фигуры")

black = (0, 0, 0)
white = (255, 255, 255)
red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
yellow = (255, 255, 0)
pink = (255, 192, 203)

# рисуем прямоугольник
rect_x = 50
rect_y = 50
rect_width = 100
rect_height = 50
pygame.draw.rect(screen, red, (rect_x, rect_y, rect_width, rect_height))

# рисуем круг
circle_x = 200
circle_y = 75
circle_radius = 30
pygame.draw.circle(screen, green, (circle_x, circle_y), circle_radius)

# рисуем треугольник
triangle_x = 350
triangle_y = 50
```

```
triangle_width = 100
triangle_height = 100
triangle_points = [(triangle_x, triangle_y), (triangle_x + triangle_width,
triangle_y),
                   (triangle_x + triangle_width / 2, triangle_y +
triangle_height)]
pygame.draw.polygon(screen, blue, triangle_points)

# рисуем пятиугольник
pent_x = 500
pent_y = 100
radius = 40
sides = 5
pent_points = []
for i in range(sides):
    angle_deg = 360 * i / sides
    angle_rad = math.radians(angle_deg)
    x = pent_x + radius * math.sin(angle_rad)
    y = pent_y - radius * math.cos(angle_rad)
    pent_points.append((x, y))
pygame.draw.polygon(screen, white, pent_points)

# рисуем эллипс
ellipse_x = 100
ellipse_y = 275
ellipse_width = 150
ellipse_height = 60
pygame.draw.ellipse(screen, red, (ellipse_x, ellipse_y, ellipse_width,
ellipse_height))

# горизонтальная линия
horiz_line_y = 400
pygame.draw.line(screen, blue, (50, horiz_line_y), (590, horiz_line_y), 5)

# вертикальная линия
vert_line_x = 320
pygame.draw.line(screen, green, (vert_line_x, 50), (vert_line_x, 430), 5)

# рисуем желтую звезду
yellow_star_points = [(260 - 50, 250 - 70), (310 - 50, 250 - 70), (325 - 50, 200 -
70),
                      (340 - 50, 250 - 70), (390 - 50, 250 - 70), (350 - 50, 290 -
70),
                      (365 - 50, 340 - 70), (325 - 50, 305 - 70), (285 - 50, 340 -
70),
                      (300 - 50, 290 - 70)]
pygame.draw.polygon(screen, yellow, yellow_star_points)

# рисуем окружность с квадратом внутри
circle2_x = 490
circle2_y = 350
circle2_radius = 80
pygame.draw.circle(screen, white, (circle2_x, circle2_y), circle2_radius)
square_side = 60
square_x = circle2_x - square_side / 2
square_y = circle2_y - square_side / 2
pygame.draw.rect(screen, pink, (square_x, square_y, square_side, square_side))

pygame.display.update()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
```

```
            quit()
```



Анимация и обработка событий

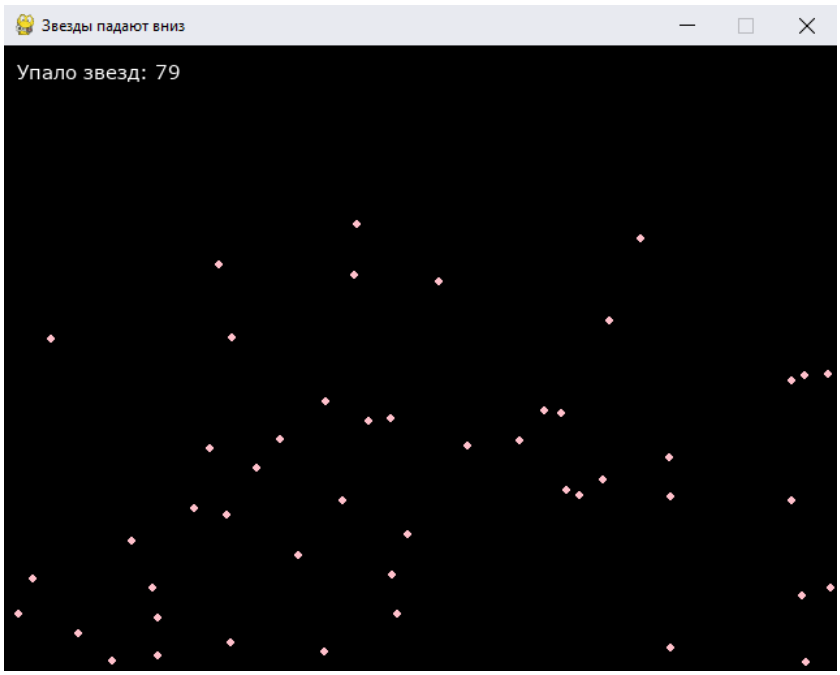
Выше, в примере с падающими символами в «Матрице», уже был показан принцип простейшей имитации движения, который заключается в последовательном изменении координат объекта и обновлении экрана с установленной частотой кадрарygame.time.Clock().tick(60). Усложним задачу – сделаем простую анимацию с падающими розовыми «звездами». Приложение будет поддерживать обработку двух событий:

- При клике мышью по экрану анимация останавливается.
- При нажатии клавиши Enter – возобновляется.

Кроме того, добавим счетчик упавших звезд. Готовый код выглядит так:

```
import pygame
import random

pygame.init()

screen_width = 640
screen_height = 480
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Звезды падают вниз")

black = (0, 0, 0)
white = (255, 255, 255)
pink = (255, 192, 203)

font = pygame.font.SysFont("Verdana", 15)

star_list = []
for i in range(50):
    x = random.randrange(screen_width)
    y = random.randrange(-200, -50)
```

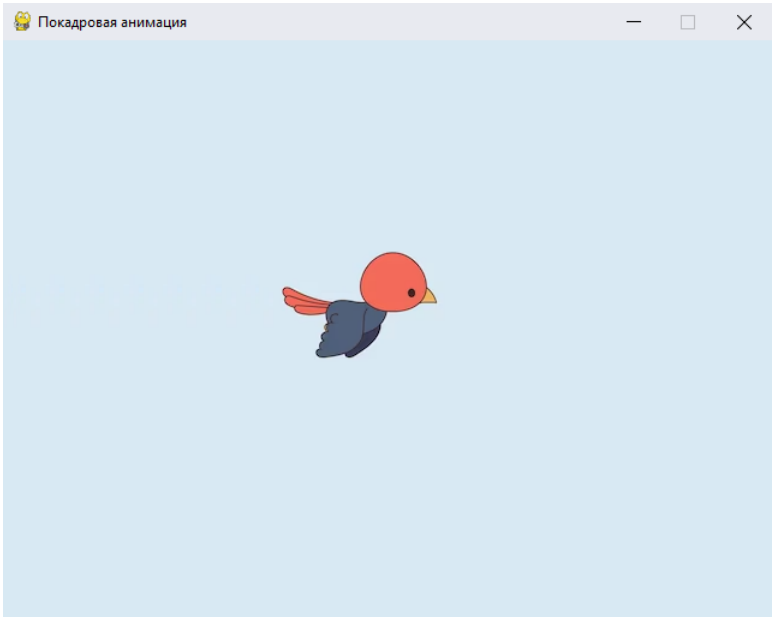
```
    speed = random.randrange(1, 5)
    star_list.append([x, y, speed])
score = 0

freeze = False # флаг для определения момента остановки

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        if event.type == pygame.MOUSEBUTTONDOWN: # останавливаем падение звезд по
клику
            freeze = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RETURN: # возобновляем движение вниз, если
нажат Enter
                freeze = False

    if not freeze: # если флаг не активен,
        # звезды падают вниз
        for star in star_list:
            star[1] += star[2]
            if star[1] > screen_height:
                star[0] = random.randrange(screen_width)
                star[1] = random.randrange(-200, -50)
                score += 1

    # рисуем звезды, выводим результаты подсчета
    screen.fill(black)
    for star in star_list:
        pygame.draw.circle(screen, pink, (star[0], star[1]), 3)
    score_text = font.render("Упало звезд: " + str(score), True, white)
    screen.blit(score_text, (10, 10))

    pygame.display.update()

    # устанавливаем частоту обновления экрана
    pygame.time.Clock().tick(60)
```



Покадровая анимация

Если у вас есть в запасе картинки с раскадровкой движения, превратить их в анимацию не составит труда:

```
import pygame

pygame.init()

window_size = (640, 480)
screen = pygame.display.set_mode(window_size)
color = (216, 233, 243)
screen.fill(color)
pygame.display.flip()
pygame.display.set_caption("Покадровая анимация")
clock = pygame.time.Clock()

# загружаем кадры
frame_images = []
for i in range(1, 9):
    frame_images.append(pygame.image.load(f"frame{i}.png"))

# параметры анимации
animation_length = len(frame_images)
animation_speed = 15  # кадры в секунду
current_frame_index = 0
animation_timer = 0
frame_position = [0, 0]

# вычисляем позицию для вывода кадров в зависимости от высоты окна
window_height = screen.get_height()
frame_height = frame_images[0].get_height()
frame_position[1] = int(window_height * 0.45) - int(frame_height / 2)

# запускаем основной цикл
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # обновление состояния
    time_delta = clock.tick(60) / 1000.0
    animation_timer += time_delta
    if animation_timer >= 1.0 / animation_speed:
        current_frame_index = (current_frame_index + 1) % animation_length
        animation_timer -= 1.0 / animation_speed

    frame_position[0] += 1  # сдвигаем кадр вправо

    # выводим кадры, обновляем экран
    current_frame = frame_images[current_frame_index]
    screen.blit(current_frame, frame_position)
    pygame.display.flip()

pygame.quit()
```

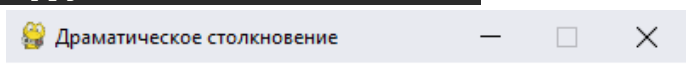


Столкновение объектов

В этом примере расстояние между объектами проверяется до тех пор, пока объекты не столкнутся. В момент столкновение движение прекращается, а цвет объектов – изменяется:

```
import pygame
import math

pygame.init()
screen_width = 400
screen_height = 480
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Драматическое столкновение")

# размеры и позиция окружности
circle_pos = [screen_width/2, 50]
circle_radius = 20

# размеры и позиция прямоугольника
rect_pos = [screen_width/2, screen_height-50]
rect_width = 100
rect_height = 50

# цвета окружности и прямоугольника
white = (255, 255, 255)
black = (0, 0, 0)
green = (0, 255, 0)
red = (255, 0, 0)

# скорость движения окружности
speed = 5

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()

    # окружность движется вниз
```

```
    circle_pos[1] += speed

    # проверяем (используя формулу расстояния),
    # столкнулась ли окружность с прямоугольником
    circle_x = circle_pos[0]
    circle_y = circle_pos[1]
    rect_x = rect_pos[0]
    rect_y = rect_pos[1]
    distance_x = abs(circle_x - rect_x)
    distance_y = abs(circle_y - rect_y)
    if distance_x <= (rect_width/2 + circle_radius) and distance_y <=
(rect_height/2 + circle_radius):
        circle_color = red # изменяем цвет фигур
        rect_color = green # в момент столкновения
    else:
        circle_color = green
        rect_color = black

    # рисуем окружность и прямоугольник на экране
    screen.fill(white)
    pygame.draw.circle(screen, circle_color, circle_pos, circle_radius)
    pygame.draw.rect(screen, rect_color, (rect_pos[0]-rect_width/2, rect_pos[1]-
rect_height/2, rect_width, rect_height))

    pygame.display.update()

    # останавливаем движение окружности, если она
    # столкнулась с прямоугольником
    if circle_pos[1] + circle_radius >= rect_pos[1] - rect_height/2:
        speed = 0

    # задаем частоту обновления экрана
    pygame.time.Clock().tick(60)
```

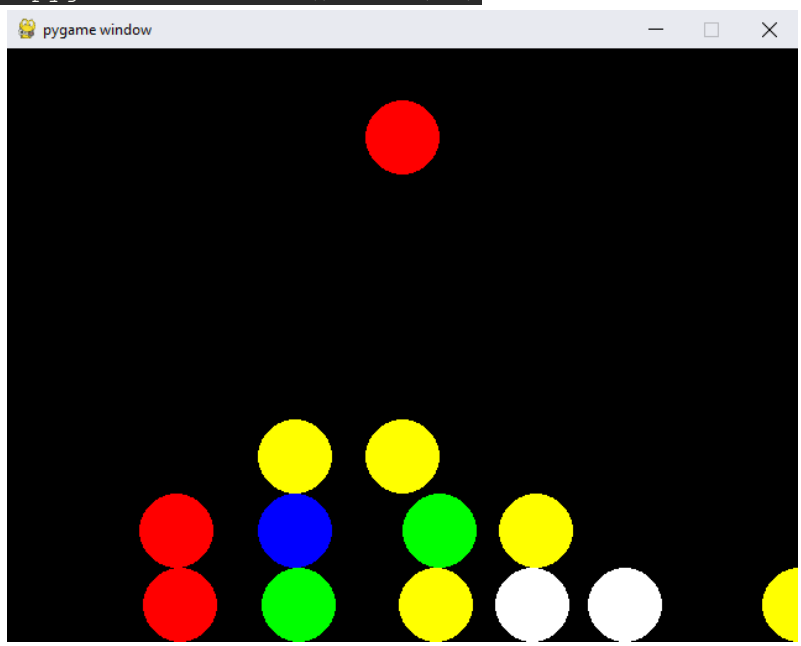


Управление движением объекта

Для управления движением (в нашем случае – с помощью клавиш ← и →) используются pygame.K_RIGHT (вправо) и pygame.K_LEFT (влево). В приведенном ниже примере передвижение падающих окружностей возможно только до момента приземления на дно игрового поля, или на предыдущие фигуры. Для упрощения вычисления факта столкновения фигур окружности вписываются в прямоугольники:

```
import pygame
import random

pygame.init()
screen_width = 640
screen_height = 480
screen = pygame.display.set_mode((screen_width, screen_height))

# цвета окружностей
white = (255, 255, 255)
red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
black = (0, 0, 0)
yellow = (255, 255, 0)

# цвет, скорость, начальная позиция окружности
circle_radius = 30
circle_speed = 3
circle_color = random.choice([red, green, blue, yellow, white])
circle_pos = [screen_width//2, -circle_radius]
circle_landed = False

# список приземлившихся окружностей и их позиций
landed_circles = []

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()

    # если окружность не приземлилась
    if not circle_landed:
        # меняем направление по нажатию клавиши
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            circle_pos[0] -= circle_speed
        if keys[pygame.K_RIGHT]:
            circle_pos[0] += circle_speed

        # проверяем, столкнулась ли окружность с другой приземлившейся окружностью
        for landed_circle in landed_circles:
            landed_rect = pygame.Rect(landed_circle[0]-circle_radius,
landed_circle[1]-circle_radius, circle_radius*2, circle_radius*2)
            falling_rect = pygame.Rect(circle_pos[0]-circle_radius, circle_pos[1]-
circle_radius, circle_radius*2, circle_radius*2)
            if landed_rect.colliderect(falling_rect):
                circle_landed = True
                collision_x = circle_pos[0]
                collision_y = landed_circle[1] - circle_radius*2
                landed_circles.append((collision_x, collision_y, circle_color))
                break
```

```
        # если окружность не столкнулась с другой приземлившейся окружностью
        if not circle_landed:
            # окружность движется вниз
            circle_pos[1] += circle_speed

            # проверяем, достигла ли окружность дна
            if circle_pos[1] + circle_radius > screen_height:
                circle_pos[1] = screen_height - circle_radius
                circle_landed = True
                # добавляем окружность и ее позицию в список приземлившихся
окружностей
                landed_circles.append((circle_pos[0], circle_pos[1],
circle_color))

    if circle_landed:
        # если окружность приземлилась, задаем параметры новой
        circle_pos = [screen_width//2, -circle_radius]
        circle_color = random.choice([red, green, blue, yellow, white])
        circle_landed = False

    # рисуем окружности
    screen.fill(black)
    for landed_circle in landed_circles:
        pygame.draw.circle(screen, landed_circle[2], (landed_circle[0],
landed_circle[1]), circle_radius)
    pygame.draw.circle(screen, circle_color, circle_pos, circle_radius)
    pygame.display.update()

    # частота обновления экрана
    pygame.time.Clock().tick(60)
```



Задания для выполнения

Все задания должны иметь графическую оболочку. Если в задаче сказано меньше/больше заданного числа – значит это параметр, который устанавливается пользователем как одна из настроек игры (приложения)

Задания с *** необязательны, но очень приветствуются)

Вариант 1

1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед.
2. Генерируется поле заданного размера, в котором в каждой клетке записываются числа по порядку. В одной клетке – одна цифра. Пользователь закрашивает пары соседних клеток если они имеют одинаковое значение или их сумма равна 10. Если между двумя клетками имеются закрашенные – они считаются соседними. Подсчитывается число ходов, победа достигается если количество не закрашенных клеток меньше заданного числа.
   *** Существует кнопка «Перестроить». При нажатии кнопки все закрашенные клетки удаляются, а оставшиеся перестраиваются по порядку. Игра продолжается пока не останется возможных ходов.
3. Разработать игру, в которой пользователь управляет шариком таким образом, чтобы он не столкнулся с препятствиями. В качестве препятствий выступают другие шарики, которые имеют случайный размер и цвет. Первые 10 секунд игры на экране присутствует 1 шарик, который перемещается по окну и отражается от стен. Каждые 10 секунд появляется еще 1 шарик. При столкновении двух шариков они отражаются друг от друга. Вести подсчет времени игры и количества шариков-противников на экране.

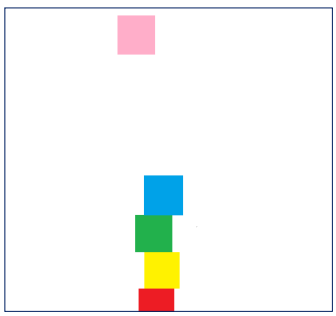
Вариант 2

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».
2. Разработать игру «Найди пару». Имеется набор парных картинок, расположенных на поле рубашкой кверху в случайном порядке. Пользователь поочередно переворачивает 2 из них. Если они одинаковые – остаются перевернутыми, если разные – снова скрываются.

3. Реализовать приложение «To do list». При запуске программы выводится окно, в котором содержится список задач. Можно отметить задачу выполненной, в этом случае она исчезает из списка, или добавить новую задачу. Каждая задача сохраняется как отдельная строка в текстовом файле.
   ***Выполненные задания можно отобразить по запросу.

Вариант 3

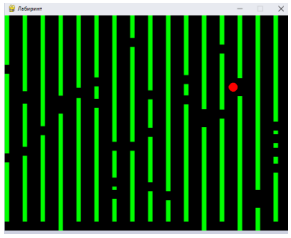
1. Реализовать игру викторину с несколькими вопросами и вариантами ответов. Реализовать вывод количества верных ответов относительно общего количества заданных вопросов. Вопросы должны выдаваться в случайном порядке без повторений.
2. Реализовать игру «Морской бой».
3. Реализовать игру «Башня». С «неба» падает квадрат случайного цвета. Управление влево\вправо осуществляется стрелками с клавиатуры. Необходимо приземлить квадрат так, чтобы перекрывать больше половины стороны нижестоящего. По достижении высоты башни 50% экрана изображение смещается вниз. Организовать подсчет количества этажей.

   ***со временем башня начинает колебаться как маятник. Каждые 10 поставленных квадратов увеличивается амплитуда и/или частота плавных (!!!) колебаний

Вариант 4

1. Реализовать игру крестики нолики. Ходы компьютера должны зависеть от предшествующих событий.
2. Реализовать игру «движущийся лабиринт». Должен генерироваться лабиринт из вертикальных линий со случайным количеством дверей. Линии (и соответственно «двери») смещаются вверх\вниз с разной скоростью. Чем быстрее движется линия – тем больше дверь. Игрок с помощью стрелок на клавиатуре должен пройти через лабиринт от одного конца поля до другого. Реализовать подсчет времени, затраченного на прохождение лабиринта.



*** Точка по горизонтали не перемещается, лабиринт движется по вертикали и горизонтали, со временем ускоряется. Подсчет времени – до проигрыша.

3. Разработать игру «Микробы». В круге «мензурке» заданного размера в случайном месте генерируется «Микроб» - кружок. При нажатии на него курсора, выполняется деление микроба, т.е. их становиться 2. Второй микроб появляется в случайном месте внутри «мензурки». Деление возможно до тех пор, пока позволяет площадь мензурки. Микробы не могут накладываться друг на друга.

Вариант 5

1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед
2. Реализовать игру «15». Поле заполняется в случайном порядке, осуществлять подсчет количества ходов.
3. Разработать игру, в которой поле-сетка в случайном порядке заполнена плитками шести различных цветов. Игрок может изменять цвет "заливки" на любой прилегающий к плитке цвет, начиная с плитки в верхнем левом углу поля, тем самым расширяя "отвоеванное" пространство. Игрок выигрывает, если он сможет окрасить поле в один цвет за определенное количество ходов
   *** Пользователь играет с компьютером. Побеждает тот, чья закрашенная площадь больше.

Вариант 6

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».
2. Разработать игру «Лабиринт». Прохождение необходимо выполнять курсором мыши. Если пользователь попадает курсором на ограничители, то игра считается проигранной.
3. Разработать игру «Гоночки». Есть двухполосная трасса, на которой сверху вниз движутся автомобили-прямоугольники. Игрок должен с помощью стрелок уклоняться от столкновений. Цвет прямоугольников-

автомобилей – случайный, интервал между ними должен быть таким, чтобы игрок имел возможность разминуться.
*** Движение – трехполосное, со временем скорость движения увеличивается

Вариант 7

1. Реализовать игру викторину с несколькими вопросами и вариантами ответов. Реализовать вывод количества верных ответов относительно общего количества заданных вопросов. Вопросы должны выдаваться в случайном порядке без повторений
2. Разработать игру, в которой поле-сетка в случайном порядке заполнена плитками шести различных цветов. Игрок может изменять цвет "заливки" на любой прилегающий к плитке цвет, начиная с плитки в верхнем левом углу поля, тем самым расширяя "отвоеванное" пространство. Игрок выигрывает, если он сможет окрасить поле в один цвет за определенное количество ходов.
3. В случайном месте окна случайным образом возникает произвольная геометрическая фигура. Игрок должен «поймать» ее путем нажатия ПКМ. Ведется подсчет пойманных и пропущенных фигур
   *** Способ «ловить» фигуру определяется как настройка игры. Вторым вариантом является перемещение курсора стрелками с клавиатуры. Проигрышем считается заданное число пропущенных подряд фигур

Вариант 8

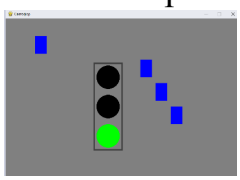
1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед
2. На поле произвольного размера два игрока, один из которых компьютер, играют фишками разных цветов, стараясь раньше соперника собрать из четырех своих фишек вертикаль, горизонталь или диагональ. Ходы компьютера должны зависеть от предшествующих событий.
   *** Количество фишек, которое необходимо собрать – произвольно, и является параметром, заданным игроком. Компьютер должен совершать ходы на основании предшествующих событий.
3. Разработать игру, в которой есть матрица 3х3 (9 клеток), в которой отсутствует одна картинка. Сделать так, чтобы пользователь мог, перемещая картинки собрать картину (и вставить последнюю деталь).

Вариант 9

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При

попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».

2. Разработать игру, в которой пользователь будет пытаться угадать цвет и/или (в зависимости от выбранного режима игры) номинал игральной карты. По достижении верного ответа выводиться изображение загаданной карты и количество попыток.
3. Напишите приложение-симулятор светофора. Когда горит зеленый свет – прямоугольники автомобили движутся вперед. Красный и желтый – останавливаются. Одновременно на экране может находиться не менее одного автомобиля в каждом из направлений движения и не более 5 в обоих направлениях.



*** Добавить пешеходный переход. Если на пешеходном переходе присутствует больше 3 человек – красный свет горит дольше – пока пешеходы идут через дорогу.

Вариант 10

1. Реализовать игру крестики нолики. Ходы компьютера должны зависеть от предшествующих событий
2. Разработать игру «Разукрашка». На поле генерируется заданное число геометрических фигур в случайных местах. При нажатии на замкнутую область выполняется заливка случайным цветом. Соседние области не должны закрашиваться одинаковыми цветами.
   *** Реализовать возможность загрузки в качестве разукрашки любого изображения, которое приводится к формату бинарного контурного рисунка.
3. Разработать игру «Лабиринт». Прохождение необходимо выполнять курсором мыши. Если пользователь попадает курсором на ограничители, то игра считается проигранной.

Вариант 11

1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед
2. Разработать игру «Вращающийся лабиринт». Имеется квадрат – лабиринт, пользователь стрелками осуществляет его вращение, чтобы загнать шарик в лунку.

3. Разработать игру, в которой пользователь будет пытаться угадать цвет и/или (в зависимости от выбранного режима игры) номинал игральной карты. По достижении верного ответа выводиться изображение загаданной карты и количество попыток.
   *** Карту загадывает как игрок, так и компьютер. Попытки угадать осуществляются поочерёдно.

Вариант 12

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».
2. Разработать игру «Микробы». В круге «мензурке» заданного размера в случайном месте генерируется «Микроб» - кружок. При нажатии на него курсора, выполняется деление микроба, т.е. их становиться 2. Второй микроб появляется в случайном месте внутри «мензурки». Деление возможно до тех пор, пока позволяет площадь мензурки. Микробы не могут накладываться друг на друга.
3. Реализовать игру «Поле чудес». Имеется ряд вопросов, один из которых задается игроку. Игрок пробует угадать букву. Если буква верная – дается еще попытка, если нет – ход переходит к двум виртуальным игрокам, которые пробуют отгадать букву. Победитель тот – кто угадал слово.
   *** добавить «вращение барабана», т.е. вес каждой отгаданной букве. Победителем считается тот, у кого больше баллов после того, как слово отгадано

Вариант 13

1. Реализовать игру викторину с несколькими вопросами и вариантами ответов. Реализовать вывод количества верных ответов относительно общего количества заданных вопросов. Вопросы должны выдаваться в случайном порядке без повторений
2. Реализовать игру в «Кости» с компьютером. Кубик бросается по очереди и выводиться изображение кубика. Побеждает тот, кто первый достигнет заданного числа.
   *** Каждый игрок загадывает со скольких бросков сможет достичь победы. Пользователь – путем ввода с клавиатуры, компьютер – случайным образом, но в диапазоне возможных значений. Диапазон возможных значений для заданного целевого числа отображается, чтобы пользователь также имел возможность ввести корректное значение.

3. На поле произвольного размера два игрока, один из которых компьютер, играют фишками разных цветов, стараясь раньше соперника собрать из четырех своих фишек вертикаль, горизонталь или диагональ. Ходы компьютера должны зависеть от предшествующих событий

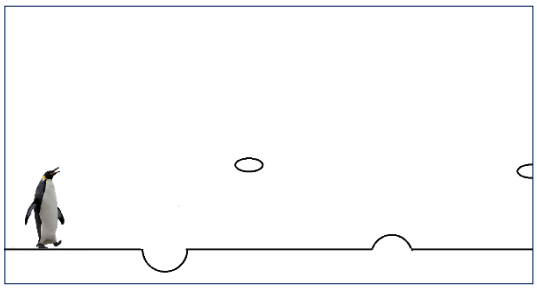
Вариант 14

1. Реализовать игру крестики нолики. Ходы компьютера должны зависеть от предшествующих событий
2. Реализовать симулятор снегопада. По экрану вниз движутся точки – снежинки. Вверху экрана выводиться счетчик количества снежинок. Каждые 100 снежинок уровень снега внизу окна увеличивается на 1 пиксель.
3. Создайте анимацию, в которой лиса преследует колобка. Колобок вращается вокруг оси.

*** колобком управляет игрок, на пути движения в случайном порядке могут возникать препятствия: камни и ямы, которые необходимо перепрыгивать управляя стрелками с клавиатуры.

Вариант 15

1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед
2. Разработать игру, в которой есть матрица 3x3 (9 клеток), в которой отсутствует одна картинка. Сделать так, чтобы пользователь мог, перемещая картинки собрать картину (и вставить последнюю деталь). *** Пользовать может иметь возможность загрузить свое изображение, определить размер матрицы, т.е. на сколько частей делить изображение.
3. Разработать игру «Бегущий пингвин». Пингвин перемещается вверх/вниз стрелками с клавиатуры. На движущейся «земле» случайным образом генерируются препятствия, которые нужно перепрыгнуть или пригнуться (поворот на 45^{o}). Частота препятствий ограничена, т.е. не должно быть случаев, когда пригнуться и подпрыгнуть нужно одновременно. Реализовать подсчет времени игры до первого поражения. Каждые 10 секунд скорость движения должна увеличиваться.



Вариант 16

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».
2. Разработать игру «Поймай фигуру». Сверху вниз движутся различные геометрические фигуры разных цветов. Вверху экрана каждые случайным образом генерируются задания, которые сменяются по мере выполнения, например: «Поймай 4 красных круга», «Поймай 2 розовых звезды», «Поймай 6 синих квадратов» и т.д. Количество, цвет и фигура – каждый раз случайны. Вести подсчет выполненных заданий. Ловить фигуры необходимо прямоугольником внизу экрана, управление которым влево/вправо осуществляется стрелками с клавиатуры.
3. Разработать игру «Гоночки». Есть двухполосная трасса, на которой сверху вниз движутся автомобили-прямоугольники. Игрок должен с помощью стрелок уклоняться от столкновений. Цвет прямоугольников-автомобилей – случайный, интервал между ними должен быть таким, чтобы игрок имел возможность разминуться
   *** Движение – четырех-полосное, игрок располагается посередине поля, автомобили-боты движутся в разных направлениях по случайным полосам, но никогда на одной полосе на встречу друг другу.

Вариант 17

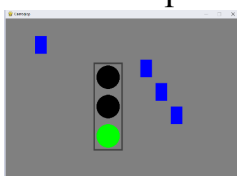
1. Реализовать игру викторину с несколькими вопросами и вариантами ответов. Реализовать вывод количества верных ответов относительно общего количества заданных вопросов. Вопросы должны выдаваться в случайном порядке без повторений
2. Разработать игру «Поймай фигуру». Сверху вниз движутся различные геометрические фигуры разных цветов. Вверху экрана случайным образом генерируются задания, которые сменяются по мере выполнения, например: «Поймай 4 красных круга», «Поймай 2 розовых звезды», «Поймай 6 синих квадратов» и т.д. Количество, цвет и фигура – каждый раз случайны. Вести подсчет выполненных заданий. Ловить фигуры – нажать на них курсором мыши.

*** реализовать корректный вывод предложения в зависимости от числительного

3. Генерируется поле заданного размера, в котором в каждой клетке записываются числа по порядку. В одной клетке – одна цифра. Пользователь закрашивает пары соседних клеток если они имеют одинаковое значение или их сумма равна 10. Если между двумя клетками имеются закрашенные – они считаются соседними. Подсчитывается число ходов, победа достигается если количество не закрашенных клеток меньше заданного числа

Вариант 18

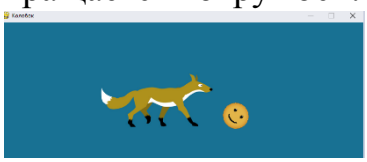
1. Реализовать игру крестики нолики. Ходы компьютера должны зависеть от предшествующих событий
2. Напишите приложение-симулятор светофора. Когда горит зеленый свет – прямоугольники автомобили движутся вперед. Красный и желтый – останавливаются. Одновременно на экране может находиться не менее одного автомобиля в каждом из направлений движения и не более 5 в обоих направлениях.



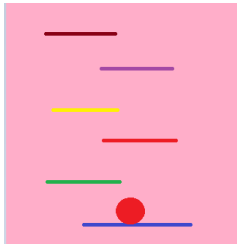
3. Реализовать игру «Змейка». Каждое съеденное яблоко должно менять цвет «змейки».
   *** Каждые 10 съеденных яблок увеличивают скорость движения змейки. Каждые 20 – добавляют на экран дополнительное яблоко – т.е. в один момент времени на экране находится сразу 2 (3,4,5 и т.д.) яблока

Вариант 19

1. Разработать игру «Камень, ножницы, бумага». В окне должны отображаться три кнопки с соответствующими изображениями, счетчики количества игр и побед
2. Создайте анимацию, в которой лиса преследует колобка. Колобок вращается вокруг оси.



3. Разработать игру «Лестница на небо». На экране появляются в случайных местах горизонтальные линии, движущиеся вниз. Игрок, красная точка, с помощью стрелок прыгает по ним вверх/влево/вправо. Линии сквозные. Реализовать счетчик времени игры.



***Добавить фон и виджет для игрока. Ускоряться со временем продвижения вверх

Вариант 20

1. Необходимо создать окно, которое будет содержать какое-либо послание пользователю, и одну кнопку с названием «Закрыть». При попытки навести курсор на кнопку снизу или сбоку– кнопка должна «убегать».
2. Напишите симулятор «Мерцающие звезды». Окружности сужаются и расширяются, имитируя мерцание. Скорость и «величина мерцания» для каждой звезды – различна.
   *** каждые 5 секунд 1 звезда исчезает, и одна появляется в другом месте. Если пользователь заметил одно из событий, и нажал ПКМ в окрестностях этой области, ему начисляются баллы: 2 – за исчезнувшую звезду, 1 – за появившуюся.
3. Реализовать игру «движущийся лабиринт». Должен генерироваться лабиринт из горизонтальных линий со случайным количеством дверей. Линии (и соответственно «двери») смещаются влево\вправо с разной скоростью. Игрок с помощью стрелок на клавиатуре должен пройти через лабиринт от одного верхней границы до нижней. Реализовать по[illegible], затраченного на прохождение лабиринта.