Д. В. Коршикова, Е. В. Купчина

АЛГОРИТМ МАСШТАБИРОВАНИЯ ГРАФИЧЕСКИХ ЭЛЕМЕНТОВ: ПРИНЦИПЫ И ПРИМЕРЫ

В данной статье будут рассмотрены основные принципы алгоритмов масштабирования и приведен пример модификации кода, реализующего такой алгоритм.

Введение

Масштабирование графических элементов – это процесс изменения размеров объектов в рамках графического интерфейса или изображения. Этот процесс часто используется в разработке программного обеспечения, 3D-моделировании и в играх для адаптации элементов под разные размеры экранов и разрешения.

I. Принципы масштабирования

Основные принципы масштабирования включают:

1. Аффинные преобразования: Масштабирование может быть представлено аффинными преобразованиями, которые включают изменения размеров, вращения и перенос объектов.

2. Пропорциональность: Пропорциональное масштабирование сохраняет соотношения сторон объекта. Это важно для предотвращения искажений.

3. Центр масштабирования: Масштабирование может происходить относительно различных точек, таких как центр объекта, угол или произвольная точка.

4. Растрирование и векторная графика: Для растровой графики масштабирование может повлечь за собой потерю качества, в то время как векторные изображения могут масштабироваться без потери четкости.

II. Пример кода

Рассмотрим пример на Python с использованием библиотеки Pygame, которая позволяет работать с графическими элементами. В этом примере создается простой графический элемент (прямоугольник) и реализуется его масштабирование. Сначала инициализируется библиотека Pygame и создается окно заданного размера. Устанавливаются начальные размеры и цвет прямоугольника.

```
pygame.init()
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode
((screen_width, screen_height))
rectangle_color = (0, 128, 255)
rectangle_width = 100
rectangle_height = 50
scale_factor = 1.0
```

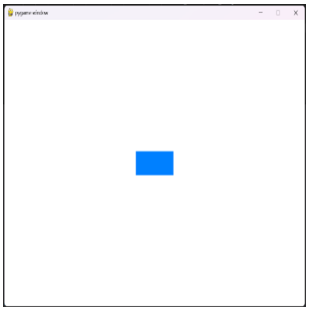
Главный цикл программы. В этом цикле обрабатываются события, такие как нажатие клавиш.

```
while True:
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
sys.exit()
if event.type == pygame.KEYDOWN:
if event.key == pygame.K_UP:
scale_factor += 0.1
elif event.key == pygame.K_DOWN:
scale_factor = max(0.1, scale_factor - 0.1)
```

Далее происходит масштабирование. При нажатии клавиш «вверх» и «вниз» изменяется коэффициент масштабирования и рассчитывается новые размеры прямоугольника. Очищается экран и рисуется новый масштабированный прямоугольник.

```
new_width = int(rectangle_width * scale_factor)
new_height = int(rectangle_height * scale_factor)
screen.fill((255, 255, 255))
pygame.draw.rect(screen, rectangle_color,
(screen_width // 2 - new_width // 2,
screen_height // 2 - new_height // 2,
new_width, new_height))
pygame.display.flip()
```

III. Пример работы кода



IV. Изменения в коде

Для реализации плавного масштабирования вместо скачкообразного потребуется изменить подход к обработке коэффициента масштабирования во время обновления экрана. Вместо того чтобы изменять коэффициент масштабирования сразу на фиксированное значение при нажатии клавиш, можно использовать интерполяцию для плавного изменения масштаба. Необходимо добавить переменную для хранения желаемого масштаба и постепенно увеличивать или уменьшать текущий масштаб, чтобы достичь желаемого значения.

1. Переменные currentscale - текущее значение масштабирования, которое будет плавно изменяться и targetscale - желаемое значение масштабирования, которое будет устанавливаться по нажатиям клавиш.

2. Скорость масштабирования: scalespeed - определяет, насколько быстро будет происходить плавное изменение масштаба.

3. Логика изменения масштаба: после нажатия клавиши «вверх» и «вниз» targetscale обновляется, и постепенно приводит currentscale к targetscale, добавляя или вычитая scalespeed.

Ниже приведена обновленная часть кода с плавным масштабированием:

```
if current_scale < target_scale:
current_scale += scale_speed
if current_scale > target_scale:
current_scale = target_scale
elif current_scale > target_scale:
current_scale -= scale_speed
if current_scale < target_scale:
current_scale = target_scale
```

Эти изменения обеспечат плавный переход между размерами прямоугольника, улучшая пользовательский опыт.

V. Пример кода с изменениями

VI. Выводы

Масштабирование графических элементов – важный аспект разработки пользовательских интерфейсов и визуализации данных. Понимание принципов аффинных преобразований и пропорциональности позволит разработчикам создавать более адаптивные и интуитивно понятные приложения. Приведенный выше пример демонстрирует реализацию масштабирования в Pygame, но принципы могут быть применены к другим графическим библиотекам и фреймворкам.

1. Васильев, Алексей Программирование на Python в примерах и задачах / А. Васильев – 2022. – С. 538-596.
2. Голд, Майкл Создание видеоигр с помощью PyGame / М. Голд – 2023. – С. 24-55.

Коршикова Дарья Валерьевна, магистрант кафедры информационных технологий автоматизированных систем БГУИР, korshikova@bsuirl.by.

Купчина Екатерина Валерьевна, магистрант кафедры информационных технологий автоматизированных систем БГУИР, e.kupchina@bsuirl.by.

Научный руководитель: Трофимович Алексей Фёдорович, старший преподаватель, заместитель декана ФИТУ, trofimaf@bsuir.by.