МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«23» 05 _____ 2024г.

ИНТЕРАКТИВНОСТЬ В PROCESSING

Методические указания
по выполнению лабораторной работы
по дисциплине Инженерная и компьютерная графика
для студентов направления подготовки
09.03.01«Информатика и вычислительная техника»

Курск 2024 г.

УДК 621.37(075)
Составители: М.В. Бобырь, С.А. Кулабухов

Рецензент
Доцент кафедры программной инженерии,
кандидат технических наук
Т.Н. Конаныхина

Интерактивность в Processing: методические указания по выполнению лабораторной работы по дисциплине Инженерная и компьютерная графика / Юго-Зап. гос. ун-т; сост.: М.В. Бобырь, С.А. Кулабухов. – Курск, 2024. – 16 с.: ил. 1, табл. 1. – Библиограф.: с. 16.

Рассмотрены базовые понятия компьютерной графики на основе программирования в Processing. Описан механизм создания интерактивных объектов. Приведены задания для самостоятельного выполнения.

Предназначены для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать 23.05.24 . Формат 60x84 1/16.
Усл.печ.л. 0,7 Уч.-изд.л. 0,6. Тираж 20 экз. Заказ 427 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

ИНТЕРАКТИВНОСТЬ В PROCESSING.

1. Цель работы

Понять основы компьютерной графики и сложные графические объекты в среде разработки Processing. Разобрать механизм создания интерактивных графических форм.

2. Общие сведения

Несмотря на то, что Processing – язык простой, необходимо строить программу по относительно чёткой структуре.

Первое требование – любой скетч должен содержать две функции – void setup() и void draw(). Если принимать за отправную точку стандартный синтаксис языка C, то void setup() находится в main() до вечного цикла, а void draw() – в этом самом вечном цикле.

Второе – в функции setup() должны содержаться функции инициализации, список которых вы видите ниже:

size(x,y) – задаёт размер активного окна в пикселях

stroke(color) – задаёт цвет линий

background(color) – задаёт цвет фона активного окна

Понятно, что существует ещё множество функций инициализации, которые можно использовать, ну или можно сократить инит до минимума, оставив лишь функцию size(), а цвета будут установлены на стандартные.

Все функции отрисовки должны помещаться в функцию draw(). Она будет выполняться вечно (до закрытия программы) и с её помощью обновляется буфер экрана.

3. Изучение функции draw() и setup()

Запустите программный код в Processing:

Код, записанный в блоке draw(), выполняется сверху вниз, а затем повторяется до тех пор, пока вы не выйдете из программы, нажав кнопку Stop или закрыв окно. Функция setup() запускается однократно при запуске программы

```
void setup()
{
 println("I’m starting");
}
void draw()
{
 println("I’m running");
}
```

Переменные можно задавать выше функций draw и setup

```
int x = 280;
int y = 50;
int diameter = 50;

void setup() {
        size(480, 120);
        smooth();
        fill(102);
}

void draw() {
        background(204);
        ellipse(x, y, diameter, diameter);
}
```

Например, для перемещения курсора мыши вдоль экрана необходимо использовать следующий код:

```
void setup()
{
```

```
        size(580, 220);
        fill(0, 122);
        smooth();
        noStroke();
}
 void draw()
{
        ellipse(mouseX, mouseY, 12, 12);
 }
```

Чтобы убрать шлейф следующий за мышкой, можно воспользоваться следующим приемом: чтобы обновлять экран перед появлением нового круга, примените функцию background() в начале функции draw(), перед рисованием фигуры

```
 void setup()
{
        size(580, 220);
        fill(0, 122);
        smooth();
        noStroke();
 }

 void draw()
 {
        background(204);
        ellipse(mouseX, mouseY, 12, 12);
}
```

Для рисования непрерывных линий используются функции pmouseX и pmouseY . Они сохраняют позицию мыши из предыдущего кадра и обновляются при каждом запуске блока команд из draw(), как функции mouseX и mouseY.

```
 void setup()
{
        size(580, 220);
        strokeWeight(4);
        smooth();
```

```
        stroke(0, 102);
 }

 void draw() {
        line(mouseX, mouseY, pmouseX, pmouseY);
}
```

Переменные pmouseX и pmouseY используются для вычисления скорости перемещения мыши. Для этого нужно измерить расстояние между текущим и предыдущим положением мыши. Медленно двигающаяся мышь пройдет небольшое расстояние, с увеличением скорости расстояние возрастет. В примере функция dist() определяет скорость движения мыши и устанавливает толщину проводимой линии.

```
 void setup()
{
        size(580, 220);
        smooth();
        stroke(0, 122);
 }

 void draw()
{
        float weight = dist(mouseX, mouseY, pmouseX, pmouseY);
        strokeWeight(weight);
        line(mouseX, mouseY, pmouseX, pmouseY);
}
```

В предыдущем примере координаты мыши непосредственно определяют положение круга на экране. Но иногда необходимо сделать линию более плавной несмотря на неровную траекторию курсора. Эта техника называется *easing* (улучшение). Для нее нужны две величины - текущая и следующая (Рис. 1).

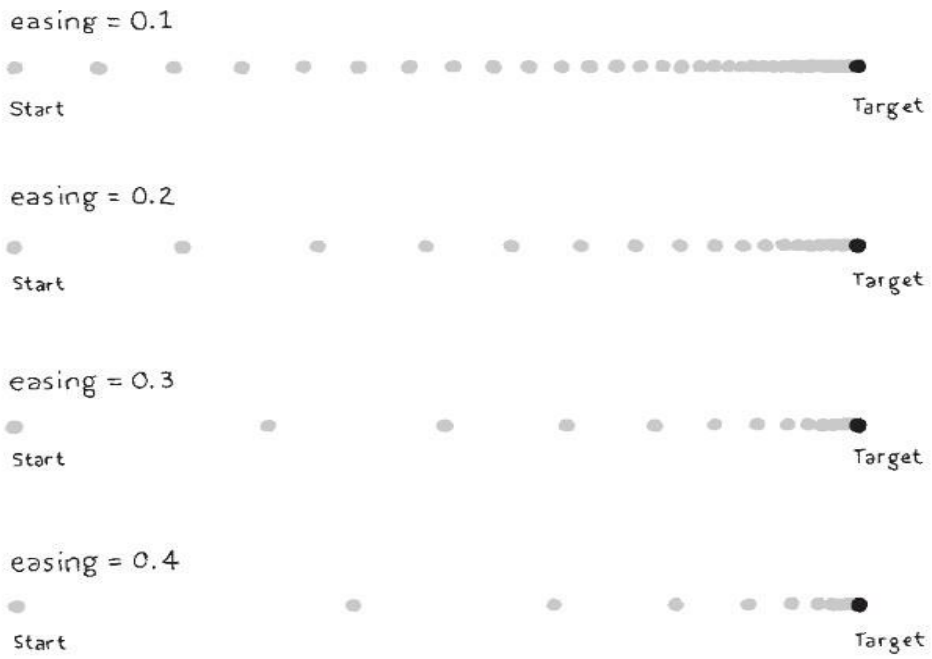

Рисунок 1

С каждым шагом текущее значение приближается к следующему:

```
float x;
float easing = 0.01;
float diameter = 12;

void setup()
{
        size(320, 150);
        smooth();
}

void draw()
{
        float targetX = mouseX;
        x += (targetX - x) * easing;
        ellipse(x, 40, 12, 12);
        println(targetX + " : " + x);
}
```

Значение переменной x всегда стремится к *target*X. Скорость, с которой x стремится к *target*X устанавливается переменной *easing* в

диапазоне от 0 до 1. Невысокое значение easing вызывает большую инерционность, чем высокое. При easing равной 1 задержки не будет. Когда вы запустите пример, обе величины будут выводиться на консоль функцией *println*(). Заметьте, что когда вы двигаете мышь, эти числа сильно различаются, но когда мышь неподвижна, они постепенно уравниваются.

Код начинается с x +=. Вычисляется разность между следующим и текущим положением круга, затем умножается на переменную easing и прибавляется к x для приближения к следующему значению

Также может использоваться и такая техника

```
float x;
float y;
float px;
float py;
float easing = 0.05;

void setup() {
        size(480, 120);
        smooth();
        stroke(0, 102);
}
void draw() {
        float targetX = mouseX;
        x += (targetX - x) * easing;
        float targetY = mouseY;
        y += (targetY - y) * easing;
        float weight = dist(x, y, px, py);
        strokeWeight(weight);
        line(x, y, px, py);
        py = y;
        px = x;
}
```

4. Преобразование чисел.

Переменная mouseX находится в пределах от 0 до ширины окна, но, возможно, вы захотите изменить диапазон значений переменной mouseX. Для этого вам потребуется разделить mouseX на некоторую величину, а затем прибавить или вычесть другую величину для реализации сдвига диапазона влево или вправо

```
 void setup()
{
        size(440, 220);
        strokeWeight(9);
        smooth();
 }

 void draw()
{
        background(204);
        stroke(255);
        line(220, 110, mouseX, mouseY); // Белая линия
        stroke(0);
        float mx = mouseX/2 + 60;
        line(220, 110, mx, mouseY); // Черная линия
}
```

Более удобный способ сделать это - функция map(). Она преобразует диапазон значений переменной. Первый параметр - переменная, второй и третий - минимальное и максимальное значение переменной, четвертое и пятое - желаемое минимальное и максимальное значение переменной. Все вычисления скрыты в функции map().

```
 void setup()
{
        size(440, 220);
```

```
        strokeWeight(9);
        smooth();
 }

 void draw()
{
            background(204);
        stroke(255);
        line(220, 110, mouseX, mouseY); // Белая линия
        stroke(0);
        float mx = map(mouseX, 0, width, 110, 350);
        line(220, 110, mx, mouseY); // Черная линия
 }
```

5. Работа с мышью

Переменная mousePressed принимает различные значения в зависимости от того, нажата кнопка мыши или нет. В переменной mousePressed сохраняются данные типа boolean; это значит, что она принимает два значения: истина (true) и ложь (false). Когда нажата кнопка мыши, mousePressed принимает значение истина.

Переменная mousePressed используется вместе с оператором if чтобы определить, когда должна запускаться строка кода.

```
 void setup()
{
        size(240, 120);
        smooth();
        strokeWeight(30);
 }

 void draw()
{
        background(204);
        stroke(102);
        line(40, 0, 70, height);
```

```
        if (mousePressed == true)
 {
                stroke(0);
        }
        line(0, 70, width, 50);
}
```

Функцию if() можно использовать совместно с оператором else()

```
void setup() {
  size(240, 120);
  smooth();
  strokeWeight(30);
}
void draw()
 {
  background(204);
  stroke(102);
  line(40, 0, 70, height);
  if (mousePressed) {
        stroke(0);
  }
   else {
        stroke(255);
  }
  line(0, 70, width, 50);
}
```

Переменная mouseButton принимает три значения: LEFT, CENTER и RIGHT. Используйте оператор сравнения ==, чтобы определить, какая из кнопок нажата.

```
  void setup() {
        size(120, 120);
        smooth();
        strokeWeight(30);
  }
```

```
void draw() {
        background(204);
        stroke(102);
        line(40, 0, 70, height);
        if (mousePressed) {
                if (mouseButton == LEFT) {
    stroke(255);        // Если правая кнопка нажата, то линия белая
                }
                else {
    stroke(0);          // Если правая кнопка нажата,    то линия
черная
                }
        line(0, 70, width, 50);
        }
}
```

Оператор if может быть использован вместе в переменными mouseX и mouseY для задания положения курсора в окне. Программа в этом примере определяет, где находится курсор - справа или слева от линии, а затем двигает линию в направлении курсора

```
float x;
int offset = 10;

void setup() {
        size(240, 120);
        smooth();
        x = width/2;
}

void draw() {
        background(204);
        if (mouseX > x) {
                x += 0.5;
                offset = -10;
        }
```

```
        if (mouseX < x) {
            x -= 0.5;
            offset = 10;
        }
        line(x, 0, x, height);
        line(mouseX, mouseY, mouseX + offset, mouseY - 10);
        line(mouseX, mouseY, mouseX + offset, mouseY + 10);
        line(mouseX, mouseY, mouseX + offset*3, mouseY);
}
```

Программа для увеличения границы круга, до тех пор пока в нем находится курсор мыши

```
int x = 120;
int y = 60;
int radius = 12;

void setup() {
    size(240, 120);
    smooth();
    ellipseMode(RADIUS);
}

void draw() {
    background(204);
    float d = dist(mouseX, mouseY, x, y);
    if (d < radius) {
        radius++;
        fill(0);
    }
    else {
        fill(255);
    }
    ellipse(x, y, radius, radius);
}
```

6. Задания для самостоятельного выполнения

1. Запустите программу Processing. Выполните приведенные в описании хода работы примеры.

2. Постройте программу динамического изменения размеров двумерной фигуры. Задание взять из таблицы 1 согласно варианту.

№ вар	Наименование фигуры	Действие по нажатию на правую кнопку мыши	Действие по нажатию на левую кнопку мыши
1	Треугольник	Увеличение размеров	Уменьшение размеров
2	Квадрат	Уменьшение размеров	Увеличение размеров
3	Четырехугольник	Перемещение вправо	Перемещение влево
4	Прямоугольная трапеция	Перемещение вверх	Перемещение вниз
5	Круг	Увеличение размеров	Уменьшение размеров
6	Треугольник	Уменьшение размеров	Увеличение размеров
7	Квадрат	Перемещение вправо	Перемещение влево
8	Четырехугольник	Перемещение вверх	Перемещение вниз
9	Прямоугольная трапеция	Увеличение размеров	Уменьшение размеров
10	Круг	Уменьшение размеров	Увеличение размеров
11	Треугольник	Перемещение вправо	Перемещение влево
12	Квадрат	Перемещение вверх	Перемещение вниз
13	Четырехугольник	Увеличение размеров	Уменьшение размеров
14	Прямоугольная трапеция	Уменьшение размеров	Увеличение размеров

15	Круг	Перемещение вправо	Перемещение влево
16	Треугольник	Перемещение вверх	Перемещение вниз
17	Квадрат	Увеличение размеров	Уменьшение размеров
18	Четырехугольник	Уменьшение размеров	Увеличение размеров
19	Прямоугольная трапеция	Перемещение вправо	Перемещение влево
20	Круг	Перемещение вверх	Перемещение вниз
21	Треугольник	Увеличение размеров	Уменьшение размеров
22	Квадрат	Уменьшение размеров	Увеличение размеров
23	Четырехугольник	Перемещение вправо	Перемещение влево
24	Прямоугольная трапеция	Перемещение вверх	Перемещение вниз
25	Круг	Увеличение размеров	Уменьшение размеров

7. Контрольные вопросы

1. Для чего используется функция draw()?
2. Какие типы данных используются в Processing?
3. Расскажите о способах взаимодействия с мышью?
4. Зачем нужна процедура setup()?
5. Что такое интерактивность и как она реализована в Processing?

8. Содержание отчёта

Отчёт должен содержать:

1) титульный лист;
2) наименование работы и цель исследований;

3) описание хода выполнения задания;

4) изображение построенной фигуры и код программы для ее построения.

9. Библиографический список

1. Кейси Риз и Бен Фрай «Учимся программировать вместе с Processing» перевод с английского Издательская группа BHV, 2012. - 194 с., ил.