 Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

                        Факультет      Информационных технологий и управления

                        Кафедра         Интеллектуальных информационных технологий

**ОТЧЁТ**

по дисциплине “Графический интерфейс интеллектуальных систем”

Лабораторная работа №3

Выполнил:

Кончатов С.Н., гр. 221703

Проверил:

Сальников Д.А.

Минск 2024

Задание: разработать элементарный графический редактор, реализующий построение параметрических кривых, используя форму Эрмита, форму Безье и B-сплайн. Выбор метода задается из пункта меню и доступен через панель инструментов «Кривые». В редакторе должен быть предусмотрен режим корректировки опорных точек и состыковки сегментов. В программной реализации необходимо реализовать базовые функции матричных вычислений.

Ход работы

Для создания программы использовалась библиотека python PyGame.

Код реализации Эрмитовой интерполяции

class Herm\_inter(Curve\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def find\_points(self,points):

        # Находим значение шага

        d=0

        for i in range(len(points)-1):d+=min(abs(points[i][0]-points[i+1][0]),abs(points[i][1]-points[i+1][1]))

        try:d=1/d

        except:d=1

        t=d

        # Начальная точка и матрица Gnx

        point\_o=[points[0]]

        gnx=[points[0],points[2],[points[1][0]-points[0][0],points[1][1]-points[0][1]],\

             [points[3][0]-points[2][0],points[3][1]-points[2][1]]]

        # Нахождение точек

        while t<=1:

            point=np.matmul(np.array([[t\*\*3,t\*\*2,t,1]]),np.matmul(MN,np.array(gnx)))

            # Построение отрезка

            Brz\_alg().draw\_line([[round(point\_o[0][0]),round(point\_o[0][1])],[round(point[0][0]),round(point[0][1])]],lambda:True)

            point\_o=point

            t+=d

    def draw\_line(self,points,check):

        # Проверка количества заданных точек

        if len(points)<3:Brz\_alg().draw\_line(points,lambda:True)

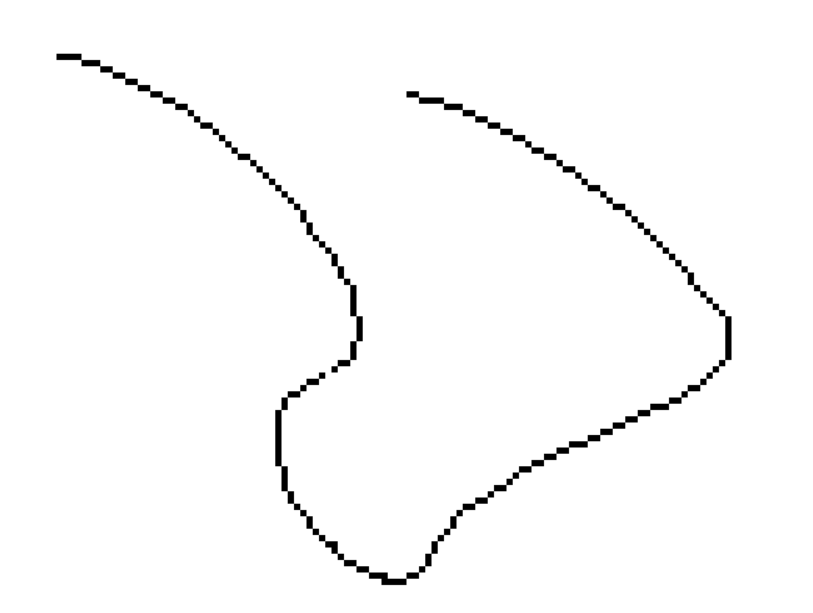
        elif len(points)==3:self.find\_points(points+[[2\*points[-1][0]-points[-2][0],2\*points[-1][1]-points[-2][1]]])

        elif len(points)==4:self.find\_points(points)

        else:

            self.find\_points(points[:4])

            self.draw\_line(points[2:],check)



Код реализации форм Безье

class Bez\_form(Curve\_Alg):

    # Режим рисования

    def draw(self,draw\_click):

        return super().draw(draw\_click)

    def find\_points(self,points,mb):

        # Находим значение шага

        d=0

        for i in range(len(points)-1):d+=min(abs(points[i][0]-points[i+1][0]),abs(points[i][1]-points[i+1][1]))

        try:d=1/d

        except:d=1

        t=d

        # Начальная точка

        point\_o=[points[0]]

        # Нахождение промежуточных точек

        while t<=1:

            point=np.matmul(np.array([[t\*\*n for n in range(len(points)-1,-1,-1)]]),np.matmul(mb,np.array([point for point in points])))

            # Построение отрезка

            Brz\_alg().draw\_line([[round(point\_o[0][0]),round(point\_o[0][1])],[round(point[0][0]),round(point[0][1])]],lambda:True)

            point\_o=point

            t+=d

        Brz\_alg().draw\_line([[round(point\_o[0][0]),round(point\_o[0][1])],[round(points[-1][0]),round(points[-1][1])]],lambda:True)

    def draw\_line(self,points,check):

        # Проверка количества заданных точек

        if len(points)<3:Brz\_alg().draw\_line(points,lambda:True)

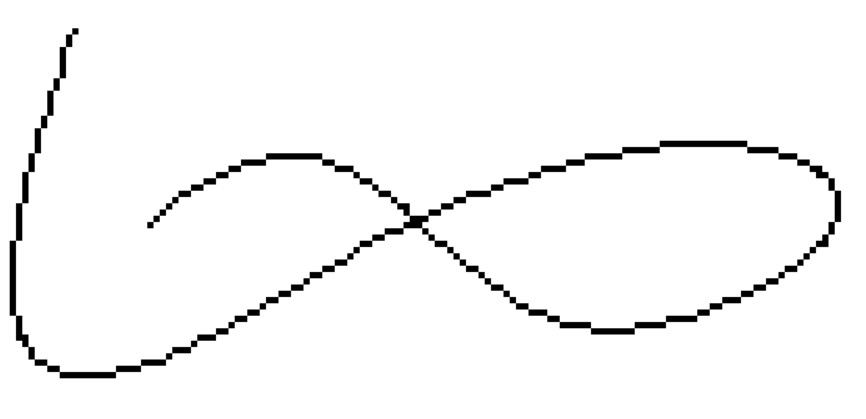
        elif len(points)==3:self.find\_points(points,MB3)

        elif len(points)==4:self.find\_points(points,MB4)

        else:

            self.find\_points(points[:4],MB4)

            self.draw\_line([points[3]]+[[2\*points[3][0]-points[2][0],2\*points[3][1]-points[2][1]]]+points[4:],check)



Код реализации Б-сплайна

class B\_spl(Curve\_Alg):

    # Режим рисования

    def draw(self, draw\_click):

        return super().draw(draw\_click)

    def find\_points(self,points,point\_o):

        # Нахождение шага

        d=0

        for i in range(len(points)-1):d+=min(abs(points[i][0]-points[i+1][0]),abs(points[i][1]-points[i+1][1]))

        try:d=1/d

        except:d=1

        t=0

        # Нахождение точек

        while t<=1:

            point=np.matmul(np.array([[t\*\*3,t\*\*2,t,1]]),\

                    np.matmul(MS,np.array([point for point in points])))/6

            # Построение отрезка

            Brz\_alg().draw\_line([[round(point\_o[0][0]),round(point\_o[0][1])],[round(point[0][0]),round(point[0][1])]],lambda:True)

            point\_o=point

            t+=d

        return point\_o

    def draw\_line(self,points,check):

        # Задано 2 точки -- построение отрезка

        if len(points)<3:Brz\_alg().draw\_line(points,lambda:True)

        # Задано 3 точки -- обавление точки

        elif len(points)==3:self.find\_points([points[0],points[1],points[2],\

                                              [2\*points[2][0]-points[1][0],2\*points[2][1]-points[1][1]]],[points[0]])

        # Заданно 4 и более

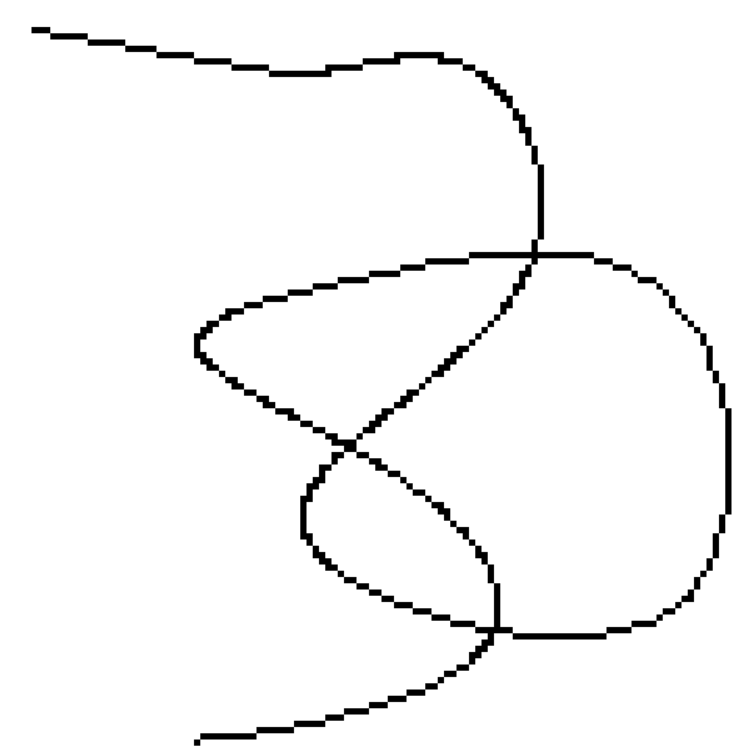
        else:

            point\_o=[points[0]]

            # Построение секции

            for i in range(len(points)-3):

                point\_o=self.find\_points([point for point in points[i:i+4]],point\_o)



Вывод

В ходе данной лабораторной работы были получены знания об алгоритмах построении кривых, данные знания были применены на практике и реализованы в среде Python.