**МГТУ им. Н.Э. Баумана**

**КОМПЬЮТЕРНАЯ ГРАФИКА**

**Лабораторный практикум №7**

**по теме: « Реализация алгоритма отсечения отрезка регулярным отсекателем»**

***Студент: Нгуен Фыок Санг***

***Группa: ИУ7И-46Б***

***Работу проверил:***

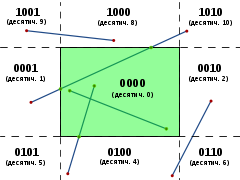
2020

Цель работы: изучение и программная реализация алгоритма отсечения отрезка.

Алгоритм отсечения (простой, Сазерленда-Коэна, деления отрезка пополам) выбирается в соответствии с номером студента в списке группы.

1. Простой алгоритм отсечения отрезка:

* Входные данные:
  + Отсекатель: прямоугольник со сторонами, параллельными координатным осям [left, right, top, bottom]
  + два конца отрезка: P1(X1, Y1), P2(X2, Y2)
* Выход:
  + R1, R2: R1R2 - это часть отрезки P1P2, расположенная полностью внутри откатателя

Для определения принадлежности точки одной из девяти областей, на которые разбивается плоскость продолжениями ребер отсекателя, вводится четырехразрядный (битовый) код:

* T1 =1, если точка лежит левее окна, и 0 в противном случае
* T2 =1, если точка лежит правее окна, и 0 в противном случае
* T3 =1, если точка лежит ниже окна, и 0 в противном случае
* T4 =1, если точка лежит выше окна, и 0 в противном случае
* **Точка лежит внутри отсекателя, если все биты ее кода – нулевые**
* **Отрезок будет полностью видимым, если коды обоих его концов нулевые.**
* Если побитное логическое произведение кодов концов отрезка не равно нулю, то отрезок является полностью невидимым
* **Уравнение P1P2: Y = m(X – X0) + Y0**

**X = (Y – Y0) / m + X0, m = (Y 2 – Y1)/ (X2 – X1)**

**для горизонтальных: m = 0**

**для вертикальных: m = 1030(большое число)**

* Если отрезок является полностью видимым, конец.
* Если отрезок не является полностью видимым, то далее ищутся точки пересечения отрезка с границами отсекателя. При этом сначала анализируется возможность пересечения отрезка с очередной границей отсекателя.
* Если пересечение возможно, то находятся координаты точки пересечения, после чего проводится анализ ее корректности. Под корректностью понимается принадлежность найденной точки ребру отсекателя, а не его продолжению. В случае корректности найденного пересечения точка заносится в результат и ищется вторая точка пересечения отрезка с границами отсекателя.
* После нахождения двух точек пересечения визуализируется полученный результат. Если же пересечение не является корректным, то выполненные вычисления оказываются фактически напрасными и в этом случае ищется пересечение со следующей стороной отсекателя (рис.3.6.3). Если не было найдено ни одного корректного пересечения, то отрезок является невидимым.

1. Ввод координат отсекателя Xл, Xп, Yн, Yв.
2. Ввод координат концов отрезка P1(X1,Y1), P2(X2,Y2).
3. Вычисление кодов концов отрезка T1, T2. Вычисление сумм кодов концов отрезка ,
4. Установка признака видимости отрезка pr =1 (pr =1 - отрезок видимый; pr = -1 - отрезок невидимый).
5. Задание начального значения тангенса угла наклона отрезка m=1030 (большое число, вначале предполагается, что отрезок вертикальный).
6. Проверка полной видимости отрезка: если (S1=0)&(S 2=0)=истина, то отрезок видимый; выполнение в этом случае следующих действий: занесение в результат координат концов отрезка R1=P1 , R2=P2 и переход к п. 31.
7. Вычисление логического произведения кодов концов отрезка .
8. Проверка тривиальной невидимости отрезка: если PL≠ 0, то отрезок невидим. В этом случае установка признака pr= -1 и переход к п. 31.
9. Проверка видимости первого конца отрезка: если S1=0 (первый конец видим), то выполнение следующих действий: R1=P1 (занесение этой вершины в результат), Q=P2 (занесение координат другой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.
10. Проверка видимости второго конца отрезка: если S2=0 (второй конец видим), то выполнение следующих действий: R1= P2 (занесение этой вершины в результат), Q=P1 (занесение координат другой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.
11. Установка начального значения шага отсечения i=0.
12. Вычисление текущего номера шага отсечения i=i+1.
13. Проверка завершения процедуры отсечения: если i>2, то переход к п.31.
14. Занесение в рабочую переменную Q координат i-ой вершины Q=Pi.
15. Определение расположения отрезка: если X2=X1 (отрезок вертикальный), то переход к п.23 (не может быть пересечения с левой и правой границами отсекателя).
16. Вычисление тангенса угла наклона отрезка m=(Y2-Y1)/(X 2-X1).
17. Проверка возможности пересечения с левой границей отсекателя: если Qx>Xл (пересечения нет), то переход к п.20.
18. Вычисление ординаты точки пересечения отрезка с левой границей отсекателя: Yр=m(Xл - Qx)+Qy .
19. Проверка корректности найденного пересечения: если (Yр ≥Yн)&(Yp ≤Yв)= истина (пересечение корректное), то выполнение следующих действий: Ri.x =Xл, Ri..y =Yр (занесение полученных координат в результат), переход к п.12.
20. Проверка возможности пересечения отрезка с правой границей отсекателя: если Qx < Yп (пересечения нет), то переход к п.23.
21. Вычисление ординаты точки пересечения с правой границей: Yр=m(Xп - Qx)+Qy .
22. Проверка корректности найденного пересечения: если (Yр ≥Yн)& (Yр ≤Yв)= истина (пересечение корректно), то выполнение следующих действий Ri.x =Xп, ( занесение полученных координат в результат ), переход к п.12.
23. Проверка горизонтальности отрезка: если m=0, то переход к п.30.
24. Проверка возможности пересечения с верхней границей отсекателя: если Qy < Yн (пересечения нет), то переход к п.27.
25. Вычисление абсциссы точки пересечения с верхней границей: Xр=(Yв - Qy)/m+Qx.
26. Проверка корректности найденного пересечения: если (Xр≥ Xл)&(Xр≤ Xп)= =истина (пересечение корректно), то выполнение следующих действий: Ri.x=Xр; Ri.y=Yв (занесение полученных координат в результат); переход к п. 12.
27. Проверка возможности пересечения с нижней границей отсекателя: если Qx>Yн (пересечения нет), то переход к п. 30 (вершина невидима и ни одно пересечение не является корректным, следовательно, отрезок невидим).
28. Вычисление абсциссы точки пересечения с нижней границей: Xр=(Yн - Qy)/m+Qx.
29. Проверка корректности найденного пересечения: (Xр≥ Xл)&(Xр≤ Xп)= =истина (пересечение корректно), то выполнение следующих действий: Ri.x=Xр; Ri.y=Yв (занесение полученных координат в результат); переход к п. 12.
30. Установка признака видимости pr= -1 (отрезок невидим полностью, так как ни одно пересечение не оказалось корректным).
31. Проверка признака видимости: если pr=1, то вычерчивание отрезка R1R2.
32. Конец

def clipping(win):

buffer = win.scene.itemAt(now, QTransform()).rect()

win.w = { 'left': buffer.left(), 'right': buffer.right(), 'top': buffer.bottom(), 'bottom': buffer.top() }

for line in win.lines:

res = line\_clipping(win.w, line)

if res[0] == 1:# отрезок является целиком или частично видимым

R = res[1]

R1 = R[0]

R2 = R[1]

win.pen.setColor(new\_color)

win.scene.addLine(R1[0], R1[1], R2[0], R2[1], win.pen)

win.pen.setColor(old\_color)

def line\_clipping(w, line):

p1 = line[0]

p2 = line[1]

#Вычисление кодов концов отрезка T1, T2

T1 = point\_code(p1, w)

T2 = point\_code(p2, w)

# Вычисление сумм кодов

S1 = sum\_code(T1)

S2 = sum\_code(T2)

#Установка признака видимости отрезка

pr = 1

m = 1e30

R = [p1, p2]

# Проверка полной видимости отрезка

if (S1 == 0 and S2 == 0):

R[0] = p1

R[1] = p2

return [pr, R]

# Вычисление логического произведения кодов

PL = mult\_code(T1, T2)

# отрезок невидим

if (PL != 0):

pr = -1

return [pr,]

# Проверка видимости первого конца отрезка

if (S1 == 0):

R[0] = p1

i = 0

# Проверка видимости второго конца отрезка

elif (S2 == 0):

R[0] = p2

line[0], line[1] = line[1], line[0]

i = 0

else:

i = -1

#Начальное значение: -1

while i < 1:

i += 1

Q = line[i]

# отрезок вертикальный не может быть

# пересечения с левой и правой границами отсекателя

if (p1[0] != p2[0]):

m = (p1[1] - p2[1]) \* 1.0 / (p1[0] - p2[0])

# Проверка возможности пересечения с левой границей отсекателя

if (Q[0] <= w['left']):

y = m \* (w['left'] - Q[0]) + Q[1]

# анализ корректности

if (y >= w['bottom'] and y <= w['top']):

R[i] = [w['left'], y]

continue

# Проверка возможности пересечения с правой границей отсекателя

if (Q[0] >= w['right']):

y = m \* (w['right'] - Q[0]) + Q[1]

# анализ корректности

if (y >= w['bottom'] and y <= w['top']):

R[i] = [w['right'], y]

continue

# Проверка горизонтальности отрезка

if (p1[1] != p2[1]):

` # Проверка возможности пересечения с верхней границей отсекателя

if (Q[1] >= w['top']):

x = (w['top'] - Q[1]) / m + Q[0]

# анализ корректности

if (x >= w['left'] and x <= w['right']):

R[i] = [x, w['top']]

continue

# Проверка возможности пересечения с нижней границей отсекателя

if (Q[1] <= w['bottom']):

x = (w['bottom'] - Q[1]) / m + Q[0]

\

if (x >= w['left'] and x <= w['right']):

R[i] = [x, w['bottom']]

continue

pr = -1

return [pr, ]

#endwhile

return [pr, R]

