Outer Billiard Report

Evan Huynh

September 11, 2017

Abstract

This report gives a briefly look about our code and progress in the "Outer Billiard" project.

Contents

1 Introduction

This report gives a briefly look about our code and progress in the "Outer Billiard" project.

2 Background theory

Two points are symmetrical if there correspond to the same distance of a middle point but are in the opposite direction.

3 Approaching to the problem

According to the background theory provided, there is one easy way to determine the position of its symmetrical point over a middle point by calculating the directional vector of the initial point to the middle point, then by translating it with that directional vector, we obtain the position of the symmetrical point that we are needed to find.

Take one example, let assume there exists point $K\ (x_K, y_K)$. Let $L\ (x_L, y_L)$ be the middle point. The co-ordinate of the symmetrical point of K over L is $M\ (x_M, y_M)$ can be calculate as $x_M = x_L - x_K$, $y_M = y_L - y_K$. In order word, we have translated point B with vector $\overrightarrow{LK}$.

This concept can be demonstrate with a function (reflectPoint) in the appendix.

One such issue exist in this problem is to ascertain which corner of the $\triangle ABC$ to draw the next symmetrical point. As we have from the hypothesis, the billiard travel across the first corner, then the second, then the third, and repeat the process. Since the ball travels and marks the location before changing its direction, the number of marks determine which direction it goes when changing the direction.

This can be solved using modularity. Let n be the marked location, including the initial location $K\ (x_K, y_K)$. if $n \equiv 1(\mod 3)$, then balls goes over A. If $n \equiv 2(\mod 3)$, then it goes over B. If it is neither, then it goes over C. This strategy works as drawing n segments require at least $n+1$ points, and since the drawing process repeats over three corner of the equilateral triangle, the modularity of 3 of marked points determines the next directions the ball moves.

4 Demonstrations

Here

5 Discussion

Here

6 Appendix

6.1 Mathematica

6.1.1 Function

```
1 (*
2 Create a point of reflection over a middle point
3 Usage: reflectPoint[{outPoint},{middlePoint}]}
4 *)
5 reflectPoint[outPoint_?ListQ,middlePoint_?ListQ]:=(
6 xMove = -(outPoint[[1]]-middlePoint[[1]]);
7 yMove = -(outPoint[[2]]-middlePoint[[2]]);
8 {middlePoint[[1]]+xMove,middlePoint[[2]]+yMove}
9 )
```

6.1.2 Main program

```
1 (*------------BEGIN--------------*)
2 Manipulate[
3 (*----------------*)
4 (*Create the first equilateral triangle*)
5 triangle1=Polygon[CirclePoints[3]];
6 xA=0;yA=1; pointA={xA,yA}; textA= Text["A",{0.1,1}];
7 xB=-(Sqrt[3]/2);yB=-(1/2); pointB={xB,yB}; textB =
    ↪ Text ["B",{-0.92,-0.54}];
8 xC=Sqrt[3]/2; yC=-(1/2); pointC={xC,yC}; textC =
    ↪ Text["C",{0.92,-0.54}];
9 xK;yK; pointK={xK,yK}; textK=Text["K",{xK+0.1,yK
    ↪ +0.1}];
10 pointList={pointK};
11
12 (*Add everything to the plot2 initially*)
13 plot2={EdgeForm[Directive[Thick,Blue]],Directive[
    ↪ White],triangle1,Directive[Black],Point[pointA
    ↪ ],Point[pointB],Point[pointC],textA,textB,
    ↪ textC,PointSize[0.02],Point[pointK],textK};
14 (*Done creating a equilateral triangle*)
15
16 (*Add point to list*)
17 doCtimes; doCtimes=Floor[doCtimes];
18 Do[
19 If[Mod[Length[pointList],3]==1,
20 pointList=AppendTo[pointList,reflectPoint[Last[
    ↪ pointList],pointA]];,
```

```
21  If  [Mod[Length[pointList],3]==2,
22  pointList=AppendTo[pointList,reflectPoint[Last[
       ↪ pointList],pointB]];,
23  pointList=AppendTo[pointList,reflectPoint[Last[
       ↪ pointList],pointC]];
24  ]
25  ]
26  ,doCtimes];
27  plot3={Blue,Point[pointList],Black,Line[pointList]};
28  plot4={plot2,plot3};
29
30  (*Export  the  result*)
31  plot4={plot2,plot3};
32  Show[Graphics[plot4],Axes-> True,AxesStyle->Black]
33  (*----------------*)
34
35  ,{{xK,2,"x-coordinate"},-5,5},{{yK,2,"y-coordinate"
       ↪ },-5,5},{{doCtimes,3,"Number␣of␣movements"
       ↪ },0,10}]
36  (*--------------END----------------*)
```