به نام خدا

پروژه برنامه نویسی کامپیوتر

Definite Integral

Convex Hull

گروه ۱

علی بهاری

محمدرضا اسماعیلی

علی پورعلی صفت

استاد

جناب آقای دکتر تقدس

دی ماه ۱۳۹۶

فهرست

Definite Integral

مقدمه

همانطور که می دانید هدف از انتگرال بدست آوردن مساحت زیر نمودار شکل های مختلف است و ما در برنامه نویسی نمیتوانیم از روش هایی که در علم ریاضیات مانند روش جز به جز استفاده می کنیم بهره ببریم پس باید از روش های ابتدایی استفاده کنیم که یکی از این روش ها این است که ما نمودار خود را به مستطیل هایی به عرض بسیار کوچک(دقت محاسبات) و ارتفاع مقدار تابع افراض کنیم تا بتوانیم مساحت زیر نمودار را با تقریب خوبی بدست آوریم.

متغیر ها:

x1 و y1 و x2 و y2 : نقاطی که برای رسم نمودار از آن استفاده شده است.

i : شمارنده که در حلقه ها از آن استفاده شده است.

sum : متغیری که برای جمع کردن مساحت های قسمت های زیر نمودار از آن استفاده شده است.

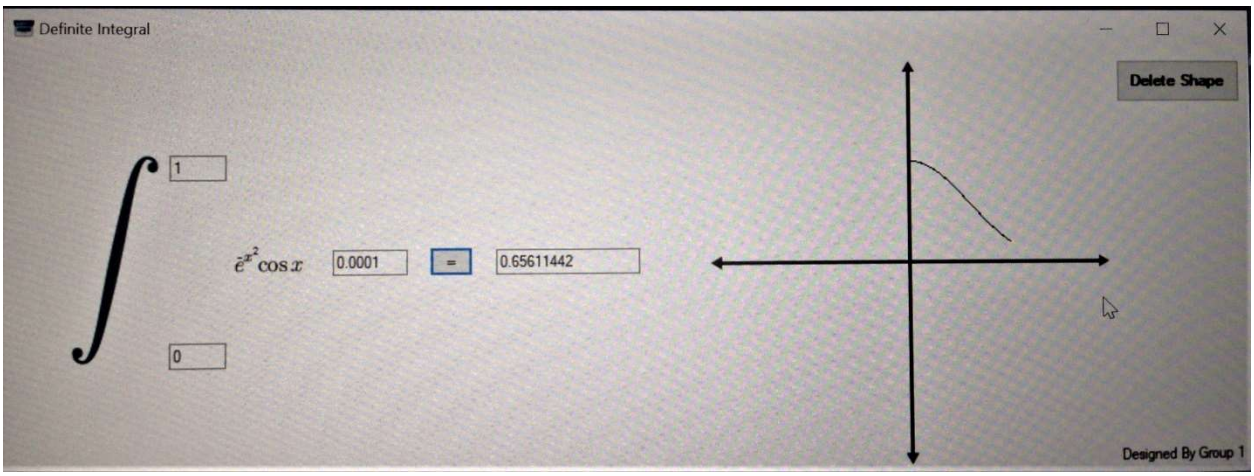
t : متغیری که برای زمان برعکس کردن سر و ته بازه از آن استفاده شده است.

definiteintegral : متغیری که تمام ویژگی انتگرال در آن وجود دارد مثل سر و ته و طول بازه

الگوریتم :

الگوریتم این برنامه به آن صورت است که ما عرض مستطیل ها را که همان دقت ما می باشد(بدیهی است که هرچه این عدد کوچکتر باشد دقت بیشتر است) به همراه بازه ای را که میخواهیم مساحت زیر نمودار را محاسبه کنیم از کاربر می گیریم و سپس ما این بازه را به بازه هایی به طول مقداری که از کاربر در مرحله یک گرفتیم تقسیم می کنیم و سپس نقاط ابتدایی بازه ها را به عنوان عرض مستطیل در آن بازه در نظر میگیریم سپس با جمع این مساحت ها انتگرال تابع را با دقتی خوب بدست می آوریم. فقط به این نکته توجه می شود که با برعکس دادن سر و ته بازه مقدار آن باید منفی مقدار عادی شود و در صورت برابر بودن سر و ته بازه مقدار انتگرال صفر می شود در قسمت رسم نمودار نیز ما با داشتن مختصات تک تک نقاط توانستیم با رسم خط بین نقاط شکل تقریبی تابع را به دست آوریم.

نمونه اجرایی



Convex Hull

مقدمه

یافتن چند ضلعی محدبی که تمام نقاط فرضی درون و یا روی آن باشند و در کل محیط مینیمم داشته باشد یکی از قدیمی ترین مسائل در هندسه محاسباتی است که از زمان مطرح شدن آن تا به امروز الگوریتم های مختلفی برای حل آن بوسیله کامپیوتر ارائه شده است این چندضلعی convex hull نام دارد که در این جا ۳ نوع از الگوریتم ها را برای یافتن آن به اختصار توضیح می دهیم. همچنین در قسمت دوم مسئله به یافتن وضعیت یک نقطه فرضی نسبت به چندضلعی می پردازیم که برای حل آن نیز ۳ روش مختلف در این جا ارائه شده است هدف این گزارش کار ارائه راه حلی برای حل این مسئله است، که در عین این که ساده و قابل فهم باشد زمان اجرایی آن نیز تا حد امکان کم باشد.

الگوریتم های یافتن **Convex Hull** :

۱. **Gift wrapping** یا **Jarvis march**

این روش با یافتن x و y مینیمم که قطعا بر روی Convex Hull قرار دارد و با قرار دادن آن به عنوان نقطه شروع با استفاده از ضرب خارجی خطی که تمام نقاط در سمت چپ آن قرار دارند را پیدا می کند و با ادامه این روند شکل مورد نظر را رسم می کند زمان اجرای این برنامه از رابطه O(n*h) به دست می آید که وابسته به خروجی برنامه است.

۲.**Quick Hull**

این روش دو نقطه ای که x مینیمم و ماکسیمم دارند را پیدا کرده و خطی را بین آن دو رسم می کند حال نقطه ای که بیشترین فصله از این خط را دارد پیدا کرده و این نقطه قطعا بر روی convex hull قرار دارد حال این سه نقطه بر روی مثلثی قرار دارند که ما الگوریتم را برای یک خط از آن اجرا کرده ایم حال با ادامه این روند بر روی دو ضلع دیگر مثلث و ردر ادامه مثلث های جدید شکل مورد نظر پیدا می شود زمان اجرای این برنامه از رابطه o(n^2) که از نسبت به بقیه الگوریتم ها زمان اجرایی بیشتری دارد.

۳.**Graham Scan**

این روش که توسط گروه انتخاب شده است و در اینجا به شرح آن می پردازیم با مرتب سازی نقاط بر اساس زاویه ای که نسبت به افق می سازد کار می کند روش ارائه شده در این گزارش کار با الگوریتم اصلی تفاوت هایی دارد به این صورت که ما با استفاده از روش هایی الگوریتم را ساده تر و قابل فهم تر کرده ایم زمان اجرایی این برنامه از رابطه o(n*log n) به دست می آید

الگوریتم های یافتن وضعیت نقطه :

۱. Ray casting

در این روش از نقطه داده شده خطی به سمت چندضلعی رسم می کنیم اگر تعداد دفعاتی که خط شکل را قطع کرد فرد بود یعنی نقطه داخل شکل قرار دارد و اگر زوج بود یعنی نقطه بیرون شکل قرار دارد مشکل این الگوریتم عدم کارایی آن برای زمانی است که نقطه روی چندضلعی قرار دارد.

۲.Winding number

در این روش از نقطه داده شده به راس های چندضلعی پاره خط هایی رسم می کنیم اگر مجموع زوایای بین این پاره خط ها به ۳۶۰ شد یعنی نقطه درون شکل قرار دارد و در غیر این صورت بیرون شکل قرار دارد برای نقطه روی چندضلعی آن نیز شروطی باید اجرا شود تا جواب نهایی صحیح به دست آید در نتیجه این روش نسبت به روش قبل بهتر است.

۳. Cross product

این روش که توسط گروه انتخاب شده است ضرب خارجی ضلع و خطی که نقطه مورد نظر را به راس وصل می کند را محاسبه می کند و با توجه به علامت آن نتیجه گیری می کند این روش نسبت به دو الگوریتم دیگر کامل تر و سریع تر است.

گزارش کار:

متغیرها:

در ابتدا به تعریف تمام متغیرهای استفاده شده در برنامه می پردازیم.

pt : نقطه ای که توسط کاربر به برنامه داده می شود.

k : شمارنده ای که برای قسمت روی خط افتادن نقاط استفاده شده است.

i : متغیری که به عنوان شمارنده در حلقه های کد از آن استفاده شده است.

j : متغیری که برای یافتن شماره نقطه اولیه از آن استفاده شده است.

z : متغیری که برای یافتن شماره نقطه دیگر ضلع convex hull (نقطه ثانویه) از آن استفاده می شود.

ymax : متغیری که بیشترین مقدار y بین نقاط در آن قرار داده می شود.

a : آرایه ای که تمام x های نقاط در آن قرار می گیرد.

b : آرایه ای که تمام y های نقاط در آن قرار داده می شود.

n : لیستی از شیب خطی های ایجاد شده است که برای فهمیدن آن که آیا نقاط روی یک خط هستند از آن استفاده شده است.

c : لیستی از x های راس های convex hull است که برای حل قسمت دوم مسئله از آن استفاده شده است.

d : لیستی از y های راس های convex hull است که برای حل قسمت دوم مسئله از آن استفاده شده است.

startingpoint : نقطه اولیه است که الگوریتم از آن شروع می شود.

secondpoint : نقطه ای که بعد از هر بار اجرای الگوریتم خط از startingpoint به آن وصل می شود.

alfa : زاویه ای که هر خط با افق می سازد.

alfamin : کمترین زاویه ای که با افق ساخته می شود.

lastalfa : زاویه ی مرحله قبل الگوریتم که برای فهمیدن دور از آن استفاده می شود.

m : شیب خط هر خط که برای فهمیدن زاویه با افق از آن استفاده می شود.

pts : آرایه ای تشکیل شده از دو نقطه که ضلع convex hull را می سازند و از این متغیر برای رسم این ضلع در دستور drawline استفاده می شود.

positionpoint : نقطه ای که کاربر برای فهمیدن وضعیت آن نسبت به convex hull به برنامه می دهد.

cross : ضرب خارجی ضلع و خطی که بین یک راس و positionpoint رسم می شود که با استفاده از مقدار آن وضعیت نقطه مشخص می شود.

negative : شمارنده ای که در صورت منفی شدن cross به مقدار آن اضافه می شود.

positive : شمارنده ای که در صورت مثبت شدن cross به مقدار آن اضافه می شود.

zero : شمارنده ای که در صورت صفر شدن cross به مقدار آن اضافه می شود.

الگوریتم :

الگوریتم استفاده شده در این گزارش کار الگوریتم graham scan است که اولین بار توسط دونالد گراهام در سال ۱۹۷۲ ارائه شده است. با توجه به پیچیدگی های بخش هایی از این الگوریتم تلاش ما بر این بود که کد را تاحد امکان ساده کنیم حال به شرح الگوریتم برنامه می پردازیم:

در قسمت اول الگوریتم ما به دنبال نقطه ای با y مینیمم می گردیم در صورتی که کم ترین مقدار y در چند نقطه پدیدار شد سپس ما نقطه ای با x بیشتر را انتخاب می کنیم یعنی در کل ما به دنبال نقطه ای با کمترین y و بیشترین x هستیم و آن را نقطه آغازی قرار می دهیم.

حال باید به این نکته توجه شود که در فضای ویندوز فرم محور مثبت y ها به سمت پایین است پس در نتیجه ما به دنبال نقطه ای می گردیم که بیشترین y را داشته باشد در کد این عمل به این صورت انجام شده است که ابتدا ما تمام x های نقاط را در آرایه a و تمام y های نقاط را در آرایه b قرار داده ایم و هر دفعه در حلقه مقدار y ای که از بقیه بیشتر می شود را در متغیر ymax قرار می دهیم حال نکته مهم این است که ما بدانیم ymax برای کدام نقطه است که در اینجا برای فهمیدن شماره آن در لیست از متغیر j استفاده کرده ایم حال در صورتی که چند نقطه با ymax داشته باشیم نقطه با x بیشتر به عنوان نقطه آغازین انتخاب می شود. حال نقطه آغازی را در متغیر startingpoint قرار می دهیم

حال با پیدا شدن startingpoint به قسمت دوم الگوریتم می رویم

درقسمت دوم الگوریتم به دنبال نقطه ای می گردیم که کم ترین تغییر زاویه نسبت به افق را داشته باشد چون در این صورت تمام نقاط در سمت چپ پاره خط بین startingpoint واین نقطه قرار می گیرند و در این صورت با تکرار این عمل برای کل شکل convex hull رسم خواهد شد.

دراین قسمت کد باید کل این روش تکرار شود پس آن را در یک حلقه کلی قرار می دهیم. برای به دست آوردن زاویه نسبت به افق به شیب خط بین startingpoint و نقطه بعدی نیازمندیم که بااستفاده از فرمول زیر آن را به دست آورده و در متغیر m قرار می دهیم.

$$m = \frac{y2 - y1}{x2 - x1}$$

حال با استفاده از فرمول زیر زاویه نسبت به افق را به دست می آوریم فقط باید به این نکته توجه شود که زاویه بر حسب رادیان به دست می آید که با ضرب آن در ۱۸۰ و تقسیم آن بر عدد پی آن را به درجه تبدیل کرده ایم.

$$\theta = \tan^{-1}(|m|)$$

به دلیل آن که خروجی تابع $\tan^{-1}$ بین π/2 و π/2 – از شروط برای مشخص شدن صحیح زاویه نسبت به افق استفاده کرده ایم به این صورت که اگر شرط اول برقرا بود یعنی خط در ربع چهارم قرار دارد و زاویه آن در حقیقت θ -360 است اگر شرط دوم برقرار بود یعنی خط درربع دوم قرار دارد و زاویه آن در حقیقت θ -180 است بقیه شروط نیز به همین ترتیب برقرار هستند فقط نکته مهم آن است که وقتی در ربع اول است ممکن است که شکل یک دور زده باشد و در نتیجه alfa θ +360 باید باشد که شرط چهارم این را بیان می کند ما با قرار دادن lastalfa به عنوان زاویه قبل گفته ایم که اگر در ربع سوم و چهارم بود یعنی شکل یک دور کامل زده است و زوایا از این به بعد در ربع اول باید با ۳۶۰ جمع شود

سپس با پیدا کردن کمترین زاویه با همان روش یافتن y مینیمم secondpoint نیز شناخته می شود و باید پاره خطی بین firstpoint و secondpointرسم کنیم که همان ضلع convex hull خواهد بود حال برای ادامه کار secondpoint تبدیل به startingpoint جدید ما می شود و دوباره به یافتن secondpoint جدید می پردازیم حال این الگوریتم تا زمانی ادامه پیدا می کند که startingpoint ما دوباره همان نقطه با y وx ماکسیمم شود و در نهایت convex hull رسم خواهد شد.

قسمت دوم مسئله:

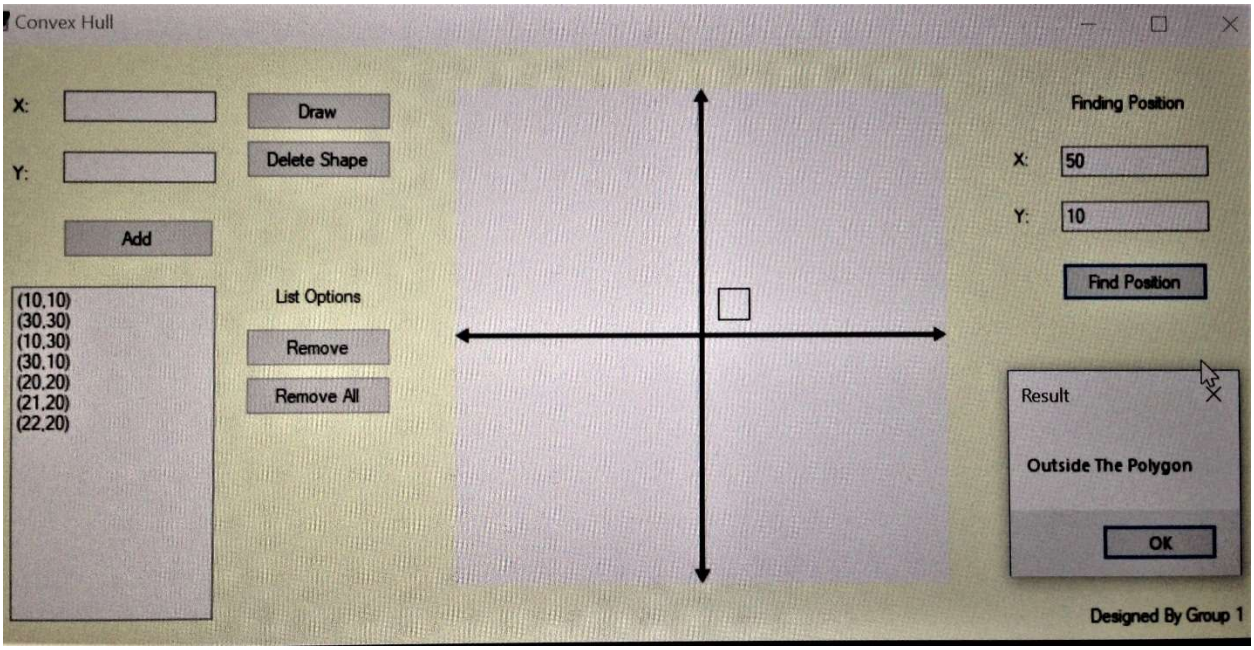
در این قسمت به یافتن وضعیت نقطه نسبت به convex hull می پردازیم در این قسمت باید به این نکته توجه شود که الگوریتم winding number با تابع $\tan^{-1}$ کارایی ندارد زیرا در این راه ما با به دست آوردن زوایای بین نقطه مورد نظر و رئوس و جمع آن ها باید به مجموع ۳۶۰ درجه برسیم ولی چون $\tan^{-1}$ بین π/2 و π/2 – است مجموع زوایا به ۳۶۰ نمی رسد و استفاده از این تابع بی فایده است بهترین کار این است که از تابع $\cos^{-1}\theta$ استفاده کنیم که بین صفر و π قرار دارد که برای به دست آوردن آن نیز از ضرب داخلی استفاده می کنیمحال الگوریتم ضرب خارجی که در کد از آن استفاده شده است را مورد بررسی قرار می دهیم.

در این روش این طور عمل می کنیم که با توجه به این که ما اضلاع convex hull را داریم از نقطه داده شده به رئوس بردارهایی فرضی رسم می کنیم و هر دفعه ضرب خارجی اضلاع را به ترتیب در این بردار های فرضی به دست می آوریم مقدار ضرب خارجی که برای استفاده راحتر از آن در متغیر cross قرار داده ایم برای ما کاربردی ندارد و علامت cross برای حل سوال از اهمیت برخوردار است به این صورت که اگر علامت تمامی cross ها ثابت بماند و تغییر نکند یعنی همه مثبت باشند و یا همه منفی باشند نقطه در convex hull قرار دارد و اگر تغییر علامت دهد بیرون شکل قرار دارد ما با تعریف متغیر های negative برای cross های منفی و positive برای cross های مثبت به حل مسئله پرداخته ایم حال

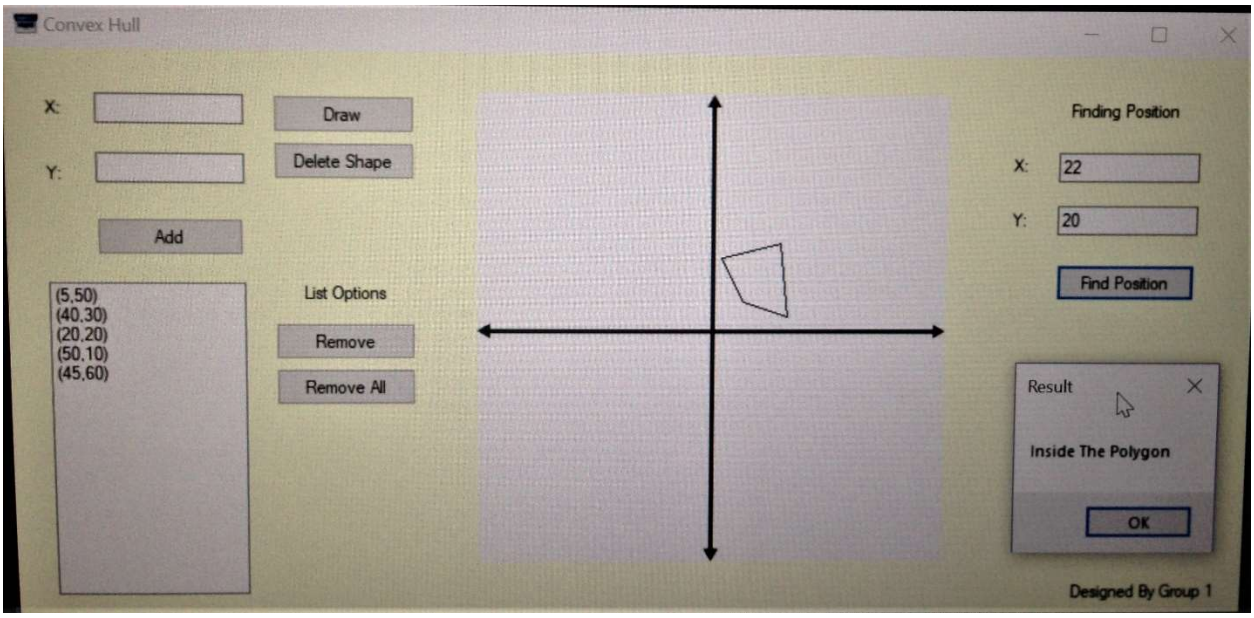
اگر مقدار ضرب خارجی صفر شود روی شکل قرار دارد، به این نیز باید توجه شود که تمام خطوطی به صورت بردار هستند و تا بینهایت امتداد دارند به همین دلیل اگر ما نقطه ای در امتداد اضلاع شکل بگذاریم مقدار cross صفر شده و برنامه وضعیت را اشتباه نشان می دهد ولی ما با تعریف متغیر zero برای cross های صفر و استفاده از شروطی این مشکل را برطرف کرده ایم.

ما به ازای هر مقدار منفی به negative ، به ازای هر مقدار مثبت به positive و به ازای هر صفر به zero اضافه خواهیم کرد حال اگر negative و positive هر دو مثبت بودند یعنی تغییر علامت داشته و نقطه مورد نظر بیرون است اگر negative و zero مثبت بودند یعنی نقطه روی شکل قرار دارد چون هم مقدار صفر دارد و هم مقدار منفی و در غیر این دو صورت نقطه درون شکل قرار دارد.

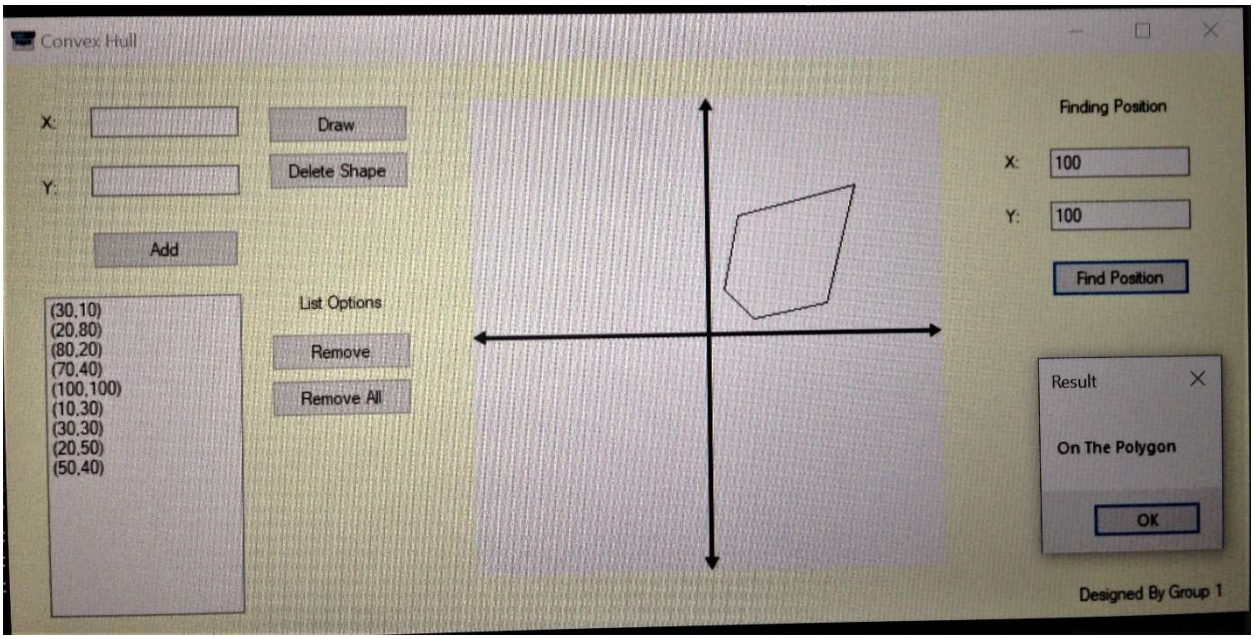
نمونه های اجرا شده کد:



نمونه ا



نمونه ۲



نمونه ۳

نتایج :

برای به دست آوردن زاویه بین دو بردار در کامپیوتر بهتر است از تابع ArcCos استفاده شود زیرا به خاطر محدوده این تابع تبدیلات برای فهمیدن زاویه نسبت به افق کم تر و راحت تر می شود.

در ضرب خارجی باید به این نکته توجه شود که بردار هایی که در هم ضرب خارجی می شوند در واقع خط هستند و محدودیتی ندارند.

منابع :

https://en.wikipedia.org/wiki/Convex_hull_algorithms

https://en.wikipedia.org/wiki/Graham_scan

/https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan

https://en.wikipedia.org/wiki/Point_in_polygon

https://stackoverflow.com/questions/217578/how-can-i-determine-whether-a-2d-point-is-within-a-polygon

https://www.codeproject.com/Articles/1210225/Fast-and-improved-D-Convex-Hull-algorithm-and-its

http://codeforces.com/blog/entry/7342