دانشگاه آزاد اسلامی واحد تهران جنوب

دانشکده فنی و مهندسی

برنامه ترسیم و پرکردن چند ضلعی

گرافیک کامپیوتری

دانشجو:

عباس اللهیاری (۸۹۱۱۳۴۰۲۴۲)

استاد درس:

خانم آزاده طباطبائی

بهار ۱۳۹۱


بسم الله الرحمن الرحیم

چکیده

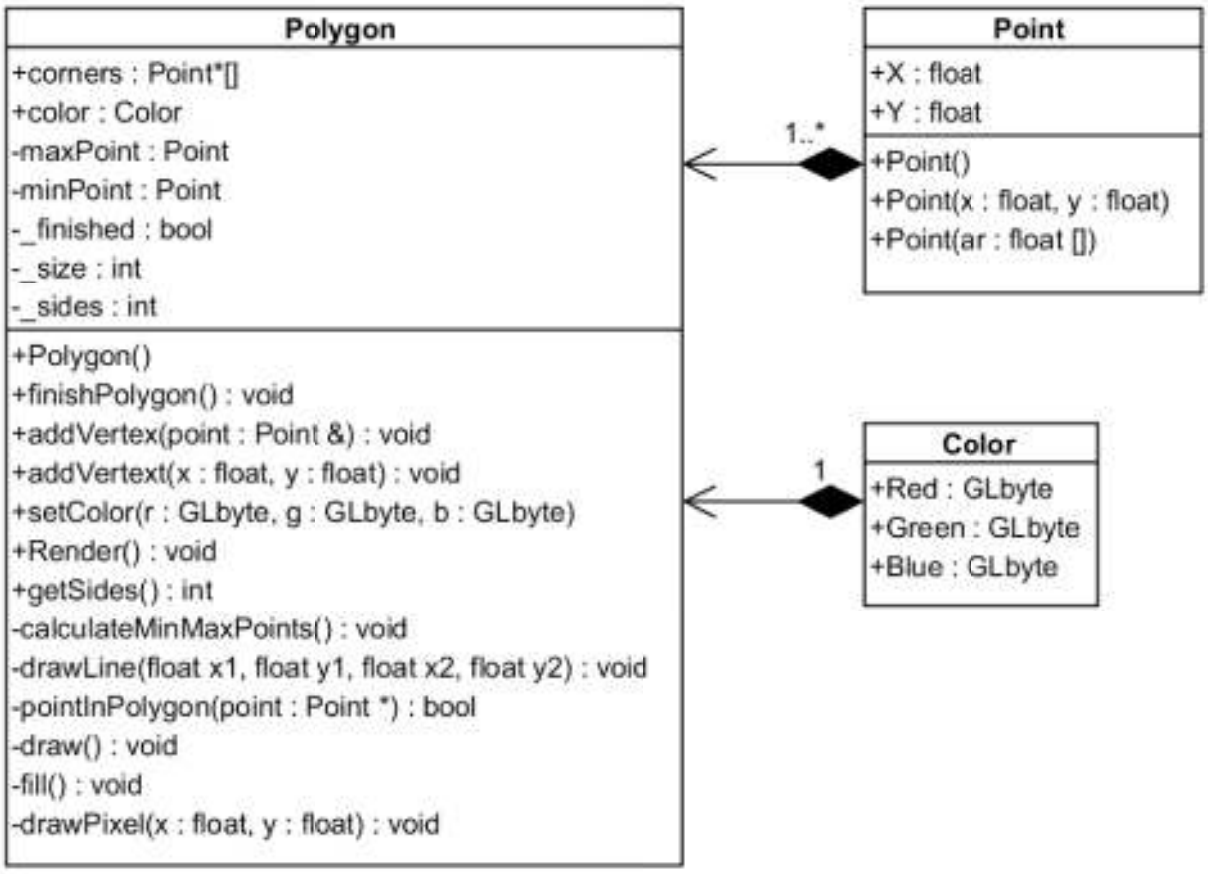
یک برنامه با دریافت نقطه از ورودی یک چند ضلعی رسم کرده و آن را پر می کند.

فهرست مطالب

فهرست اشکال

فهرست جداول

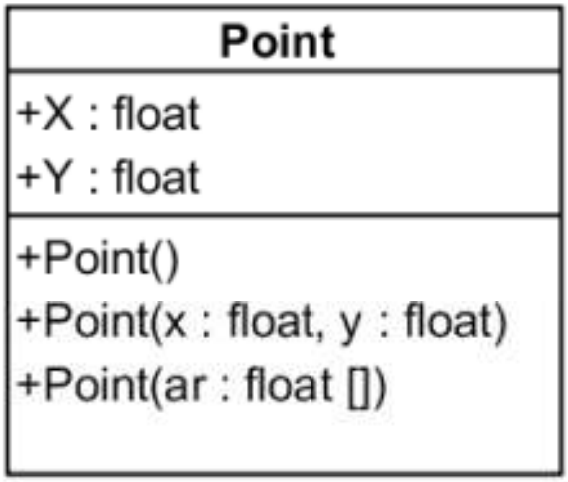
۱. معماری برنامه



شکل ۱-۱: کلاس دیاگرام برنامه

به طور کلی معماری برنامه شامل یک کلاس اصلی Polygon است که این امکان را می‌دهد تا یک چند ضلعی را توصیف و رسم کنیم. که برای صحولت کار یک کلاس Point برای مشخص کردن نقاط موجود در چند ضلعی، و همچنین یک استراکت Color برای مشخص کردن رنگ چند ضلعی، تعریف شده است.

۱,۱. کلاس Point



شکل ۲-۱: کلاس دیاگرام Point

فایل ها: Polygon.h

این کلاس برای ترسیم راحت‌تر نقطه در صفحه و افزایش خوانایی برنامه ایجاد شده است.

۱,۱,۱. فیلدها

نام فیلد	نوع داده ای	سطح دسترسی	شرح
X	float	public	نگهداری مقدار مختصات X
Y	float	public	نگهداری مقدار مختصات Y

جدول ۱-۱: فیلد های Point

۱,۱,۲. متدها

نام متد	پارامترهای ورودی	نوع برگشتی	سطح دسترسی	شرح
Point	-	آبجکتِ Point	public	X=0, Y=0
Point	float, float	آبجکتِ Point	public	دریافت مقادیر X, Y
Point	float[2]	آبجکتِ Point	public	دریافت آرایه شامل مقادیر X و Y

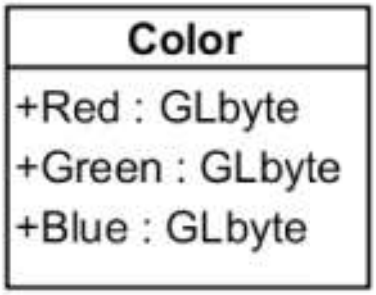
جدول ۲-۱: متد های Point

۱,۱,۳. پیاده‌سازی

```
Class Point{
public:
	Point(){
		X=0;
		Y=0;
	}
	Point(float x, float y){
		X=x;
		Y=y;
	}
	Point(float  ar[2]){
		X=ar[0];
```

```
                Y=ar[1];
            }
            float X;
            float Y;
        };
```

Color struct ۱,۲.



شکل ۳-۱: Color struct

فایل ها: Polygon.h

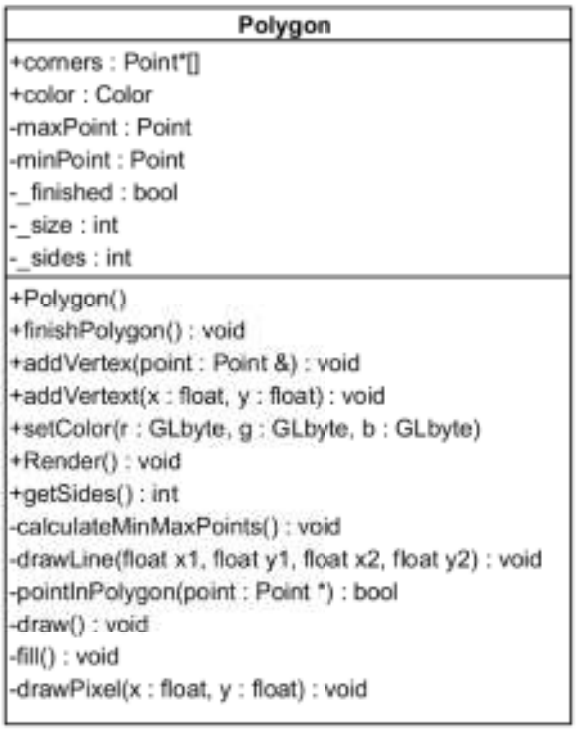
این struct بسیار ساده می‌باشد و فقط ۳ فیلد برای مقدار رنگ هر چند ضلعی دارد. و درون تعریف کلاس Polygon تعریف می شود. پیاده‌سازی این struct در بخش پیاده‌سازی کلاس Polygon می آید.

فیلدها ۱,۲,۱.

نام فیلد	نوع داده ای	سطح دسترسی	شرح
Red	GLbyte	public	رنگ قرمز
Green	GLbyte	public	رنگ سبز
Blue	GLbyte	public	رنگ آبی

جدول ۳-۱: فیلد های Color

Polygon .۱,۳



شکل ۴-۱: کلاس دیاگرام Polygon

فایل ها: Polygon.h , Polygon.cpp

این کلاس عضو اصلی برنامه می‌باشد و نماینده مجموعه چند ضلعی ها است.

۱,۳,۱. فیلد ها

نام فیلد	نوع داده ای	سطح دسترسی	شرح
Color	Color	public	رنگ چندضلعی
corners	vector<Point*>	public	گوشه‌های چندضلعی
_finished	bool	private	وضعیت تمام شدن ترسیم چند ضلعی
_sides	int	private	تعداد اضلاع
maxPoint	Point	private	مختصات بزرگترین نقطه مورد نیاز برای رسم چند ضلعی
minPoint	Point	private	مختصات کوچکترین نقطه مورد نیاز برای رسم چند ضلعی

جدول ۴-۱: فیلد های Polygon

۱,۳,۲. متد ها

نام متد	پارامترهای ورودی	نوع برگشتی	سطح دسترسی	شرح
Polygon	-	آبجکتِ Polygon	public	متد سازنده
addVertex	Point &	void	public	دریافت مقادیر X, Y
addVertex	float,float	void	public	دریافت آرایه شامل مقادیر X و Y
setColor	GLbyte, GLbyte, GLbyte	void	public	تعیین رنگ
Render	-	void	public	نمایش چند ضلعی بر صفحه
getSides	-	int	public	بازگشت تعداد گوشه ها
finishPolygon	-	void	public	بستن چند ضلعی
calculateMinMaxPoint	-	void	private	محاسبه بزرگترین و کوچک‌ترین نقطه
drawLine	float,float,float,float	void	private	ترسیم خط
pointInPolygon	Point *	bool	private	برسی می‌کند که نقطه درون چند ضلعی است یا خیر
drawPixel	float,float	void	private	ترسیم یک نقطه
draw	-	void	private	رسم چند ضلعی
fill	-	void	private	پر کردن چندضلعی

جدول ۵-۱: متدهای Polygon

۱,۳,۳. پیاده‌سازی

۱,۳,۳,۱. تعریف کلاس Polygon

```
class Polygon
{
public:
	struct  Color{
		GLbyte red;
		GLbyte green;
		GLbyte blue;
	} color;

	Polygon(void);

	~Polygon(void);
	std::vector<Point*> corners;
	void finishPolygon();



	void addVertex(Point & point);
	void addVertex(float x,float y);
	void setColor(GLbyte r, GLbyte g, GLbyte b);
	void Render();
	inline int getSides();

private:
```

```
		bool _finished;
		Point maxPoint;
		Point minPoint;
		int _sides;
		void calculateMinMaxPoints();
		void drawLine(float x1,float y1,float x2,float y2);
		bool pointInPolygon(Point * point);
		void draw();
		void fill();
		inline void drawPixel(float x,float y);

	};
```

۱,۳,۴. متد سازنده Polygon

در این متد فقط مقدار _finished را برابر false قرار می دهیم.

```
Polygon::Polygon(void)
{
	_finished=false;
}
```

۱,۳,۵. متد getSides

این متد درواقع یک Property است که تعداد اضلاع را بوسیله تعداد عناصر موجود در corners بدست می آورد.

```
int Polygon::getSides(){
	return _sides=corners.size();
}
```

۱,۳,۶. متد calculateMinMaxPoint

این الگوریتم محدوده‌ای که در آن چندضلعی رسم می‌شود را مشخص می کند، تا بعد برای پر کردن آن فقط نقاطی که درون این محدوده هستند، برسی شوند.

```
void Polygon::calculateMinMaxPoints(){
```

اولین نقطه موجود به عنوان کوچکترین نقطه و آخرین نقطه به عنوان بزرگترین نقطه در نظر گرفته می شود

```
	minPoint=*corners[0];

	maxPoint=*corners[getSides()-1];
```

طبق یک الگوریتم ساده که تمام اضلاع را پیماش می‌کند بزرگترین نقطه و کوچک‌ترین معین می شود.

در این قسمت مقدار x نقطه minPoint را برابر کوچکترین x موجود در چند ضلعی قرار می دهد. و مقدار x نقطه maxPoint را برابر بزرگترین x موجود در چند ضلعی.

```
	for(int i=0;i<_sides;i++){

		if(corners[i]->X > maxPoint.X)

			maxPoint.X=corners[i]->X;

		else if (corners[i]->X < minPoint.X)

			minPoint.X=corners[i]->X;
```

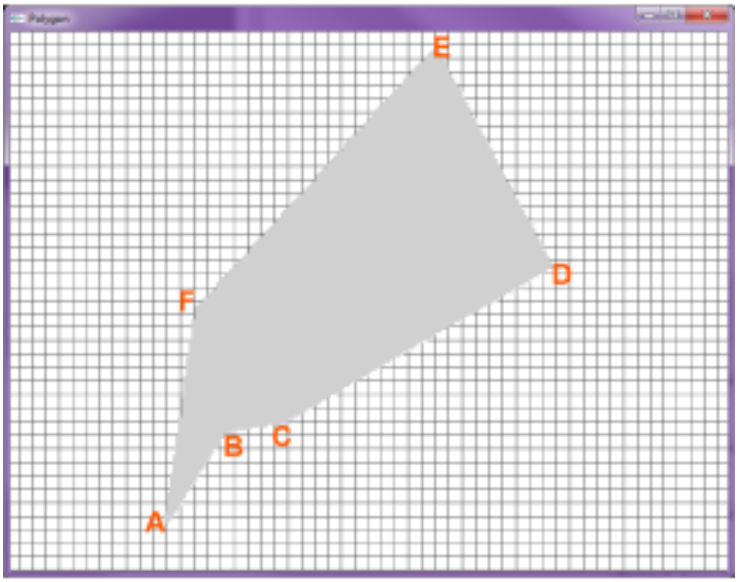
در این قسمت مقدار y نقطه minPoint را برابر کوچکترین y موجود در چند ضلعی قرار می دهد. و مقدار y نقطه maxPoint را برابر بزرگترین y موجود در چند ضلعی.

```
		if(corners[i]->Y > maxPoint.Y)

			maxPoint.Y= corners[i]->Y;

		else if (corners[i]->Y < minPoint.Y)

			minPoint.Y= corners[i]->Y;

	}
}
```

به عنوان مثال در شکل ۵-۱

این الگوریتم از نقطه E که دارای بزرگترین مقدار Y است Y را گرفته، و از نقط D که دارای بزرگترین مقدار X است، X را می‌گیرد و با استفاده از آن‌ها maxPoint را مقدار دهی می کند.

به طور مشابه، نقطه A که هم دارای کمترین X و هم کمترین Y است نقطه minPoint را ایجاد می دهد.



شکل ۵-۱: نقاط min و max

۱,۳,۷. متد setColor

مقادیر ورودی را به فیلد های مربوطه در فیلد color نسبت می دهد.

```
void Polygon::setColor(GLbyte r,GLbyte g, GLbyte b){

	color.red=r;

	color.green=g;

	color.blue=b;

}
```

۱,۳,۸. متد های addVertex

این متد دارای دو نسخه می‌باشد

نسخه‌ای که Point می‌گیرد و متد calculateMinMaxPoints را صدا می زند.

```
void Polygon::addVertex(Point & point){

	corners.push_back(&point);

	calculateMinMaxPoints();

}
```

نسخه دوم آنکه مقادیر x و y را به ترتیب می گیرد، یک نمونه از Point ساخته و آن را به متد addVertex با ورودی Point می فرستد.

```
void Polygon::addVertex(float x,float y){

	Point *point=new Point(x,y);

	addVertex(*point);

}
```

۱,۳,۹. متد finishPolygon

این متد چند ضلعی را با عوض کردن مقدار متغیر _finished اعلام می‌کند

```
void Polygon::finishPolygon(){

	_finished=true;

}
```

۱,۳,۱۰. متد drawLine

این متد یک خط رسم می کند، که در ابتدا شیب خط را محاسبه می‌کند، اگر شیب خط بین ۰ و ۱ باشد از الگوریتم برزنهام، و اگر مقداری غیر از این باشد از الگوریتم DDA برای رسم خط استفاده می کند.

```
void Polygon::drawLine(float x1,float y1, float x2, float y2)
{
	float m= fabs((y2-y1)/(x2-x1));
	if( 0 < m < 1 ) {
		//Bresenham....
	}
	else
	{
		//DDA...
	}
}
```

۱,۳,۱۱. متد pointInPolygon

این متد برسی می‌کند که آیا نقطه ورودی در داخل چندضلعی است یا خارج از آن.

الگوریتم استفاده شده به شرح زیر است

۱. مقدار oddNodes را در ابتدا مساوی false قرار می دهیم.

۲. به ازای تمام گوشه‌ها این مراحل را انجام می دهیم.

۱. هر ضلع را یک پاره خط در نظر می گیریم.

۲. P_y که ارتفاع(Y) نقطه انتخاب است باید بین دو نقطه شروع و پایان پاره خط مورد نظر باشد.

۳. اگر شرط ۲ درست باشد، با فرمول شکل ۶-۱ محاسبه می کنیم، که آیا اگر از نقطه مورد نظر به طور افقی خطی رسم کنیم، پاره خط مورد نظر را قطع می‌کنید یا خیر.

$$P_i = P_{start_X} + \frac{P_Y - P_{start_Y}}{P_{end_Y} - P_{start_Y}} \times \left(P_{end_X} - P_{start_X} \right)$$

شکل ۶-۱: فرمول محاسبه برخورد خط

۴. اگر نقطه تلاقی در راست P_x قرار بگیرد، یعنی شرط $P_x < P_i$ بر قرار شود، یک برخورد حساب می‌شود و مقدار oddNodes تغییر می کند.

۵. این کار با ازای تمام گوشه‌ها تکرار می شود

۶. در آخر مقدار oddNodes برگشت داده می شود، اگر true باشد، یعنی نقطه درون چند ضلعی است و اگر false باشد یعنی نقطه خارج از چند ضلعی است.

به طور کلی این الگوریتم از محل نقطه انتخابی، خطوط افقی رسم می کند، اگر این خط در سمت راست نقطه مورد نظر، پار خط مورد نظر را قطع کند، مقدار oddNodes عوض می شود.

```
bool Polygon::pointInPolygon(Point * p) {

      bool  oddNodes=false;
```

```
	for (int start=0,  end = _sides-1; start< _sides ; start++) {
```

برای خوانایی بیشتر دو متغیر تعریف کرده، Ps نقطه ابتدایی پاره خطی که برسی با توجه با آن انجام می شود، و Pe نقطه پایانی آن پاره خط است.

```
		Point Ps(corners[start]->X,corners[start]->Y);

		Point Pe(corners[end]->X , corners[end]->Y);
```

در اول برسی می‌کنیم که آیا نقطه مورد نظر در بین پاره خط ها قرار می‌گیرد (پاره خط را قطع کند) هم در جهت عقربه های ساعت، هم بر خلاف عقربه های ساعت. این کار از اینکه الگوریتم از روی یکی از رئوس پاره خط را رسم کند، جلوگیری می کند.

```
		bool isYBetweenLineCW= Pe.Y <  p->Y && Ps.Y >= p->Y;

		bool isYBetweenLineCCW= Ps.Y < p->Y &&  Pe.Y >= p->Y;

		if(isYBetweenLineCW || isYBetweenLineCCW){

			float Pi=Ps.X + (((p->Y - Ps.Y) / (Pe.Y - Ps.Y)) * (Pe.X - Ps.X));
```

اگر نقطه، پاره خط مورد نظر را قطع کرد،

بعد از این مرحله به شرط دوم می‌رسیم که توسط این فرمول شکل۶-۱ تعیین می‌شود.

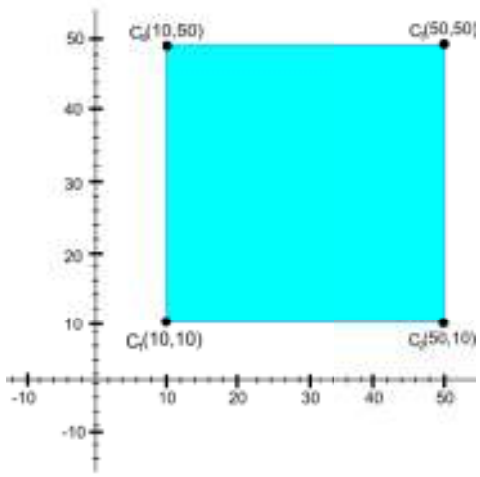
```
			if(p->X < Pi ){

				oddNodes=!oddNodes;

			}

		}

		end=start;

	}
```

در آخر نمونه ورودی به متد را پاک می کنیم. و مقدار oddNodes را بر می گردانیم.

```
	delete p;

	return oddNodes;

}
```

۱,۳,۱۱,۱. مثال

شکل ۷-۱ را فرض کنید.



شکل ۷-۱: یک مربع

به عنوان مثال اگر به دنبال نقطه (۲۰،۲۰) در این چند ضلعی بگردیم. از آنجایی که چهار ضلع دارد، چهار بار حلقه اجرا می شود.

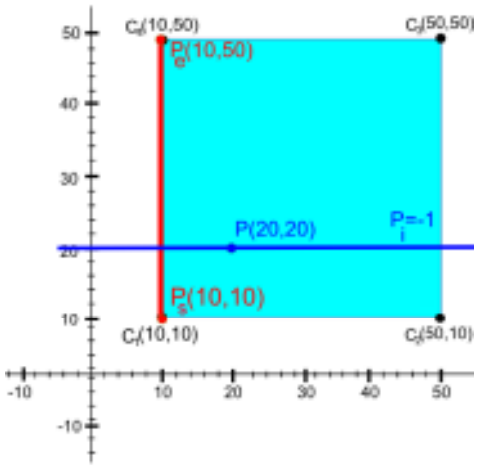
۱. شکل ۸-۱: چون پاره خط در سمت چپ، قرار دارد شرط $P_x < P_i$ برقرار نمی‌شود پس به عنوان یک برخورد حساب نمی شود.

۲. شکل ۹-۱: در اینجا، ارتفاع P_y بیشتر از ارتفاع پاره خط است، پس شرط اول برقرار نشده و پاره خط قطع نمی شود.

۳. شکل ۱۰-۱: در این شکل، هر دو شرط برقرار هستند، پس مقدار oddNodes، برابر true می شود.

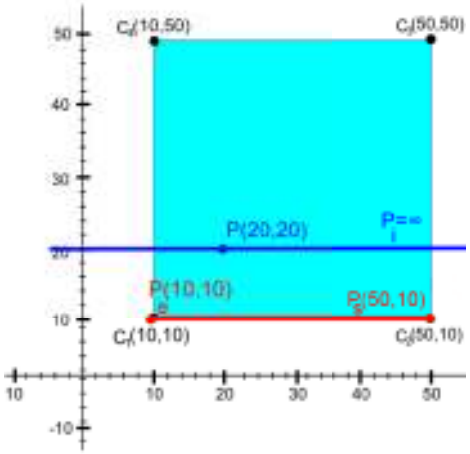
۴. شکل ۱۱-۱: شرط دوم برقرار نمی باشد، پس برخوردی صورت نمی گیرد.

۵. در آخر از آنجایی که oddNodes، هم چنان true است، و تنها یک پاره خط قطع شده است، نقطه در چند ضلعی وجود دارد.



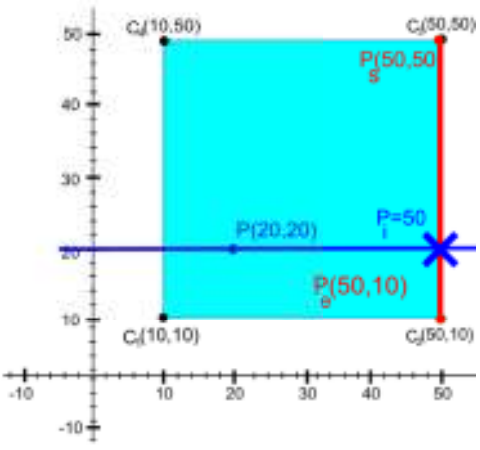
شکل ۸-۱: محاسبه PIP نقطه (۲۰،۲۰)

پاره خط اول



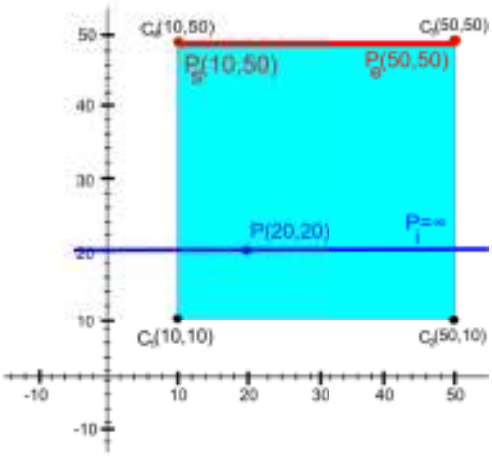
شکل ۹-۱: محاسبه PIP نقطه (۲۰،۲۰)

پاره خط دوم



شکل ۱۰-۱: محاسبه PIP نقطه (۲۰،۲۰)

پاره خط سوم



شکل ۱۱-۱: محاسبه PIP نقطه (۲۰،۲۰)

پاره خط چهارم

۱,۳,۱۲. متد drawPixel

این متد برای سادگی رسم نقطه در صفحه نوشته شده است یک نقطه توسط تابع glVertex2f بر روی صفحه رسم می کند.

```
void Polygon::drawPixel(float x,float y){
	glBegin(GL_POINTS);
	glVertex2f(x,y);
	glEnd();
}
```

۱,۳,۱۳. متد draw

این متد بدنه اصلی چند ضلعی را با استفاده از خطوط می کشد.

```
void Polygon::draw(){
	for(int i=0;i<_sides;i++)
	{
```

در اینجا به دو پارامتر اول مقدار اندیس i ام corners داده می شود.

```
		drawLine(corners[i]->X,
			corners[i]->Y,
```

به عنوان پارامتر دوم، ابتدا بررسی می‌شود که اگر حلقه به آخرین ضلع نرسیده باشد، نقطه بعدی موجود در corners را به متد drawLine می دهد، در غیر این صورت، همان نقطه انتخاب شده برای دو پارامتر اول برای دو پارامتر آخر نیز در نظر گرفته می شود.

```
			(i<_sides-1)?corners[i+1]->X:corners[i]->X,
			(i<_sides-1)?corners[i+1]->Y:corners[i]->Y);
	}
```

```
}
```

۱,۳,۱۴. متد fill

این متد در محدوده مشخص شده توسط نقاط minPoint و maxPoint به دنبال نقاطی که درون چند ضلعی هستند می گردد، و در صورت وجود در آن نقطه، یک نقطه رسم می کند. به این ترتیب چند ضلعی پر می شود.

```
void Polygon::fill(){
	for(int x= minPoint.X ;x <= maxPoint.X;x++)
		for(int y=minPoint.Y;y<=maxPoint.Y;y++)
			if(pointInPolygon(new Point(x,y)))
				drawPixel(x,y);
}
```

۱,۳,۱۵. متد Render

این متد، چند ضلعی را رسم می کند، و در صورت بسته شدن آن را پر می کند.

```
void Polygon::Render(){
	glColor3ub(color.red,color.green,color.blue);
	draw();
	if(_finished)
		fill();
}
```

۲. رابط کاربری برنامه

فایل ها: main.cpp

در این قسمت پیاده‌سازی رابط کاربری برنامه شرح داده می شود.

۲,۱. متغیر های سراسری

نام متغیر	نوع داده ای	شرح
currentPolygon	Polygon*	چندضلعی در حال رسم
Polygons	vector<Polygon*>	تمام چند ضلعی های موجود در صفحه

جدول ۱-۲: متغیر های سراسری

۲,۲. توابع

نام متد	پارامترهای ورودی	نوع برگشتی	شرح
main	int,char	int	تابع اصلی و آغازین برنامه
menu	int	void	تابع پردازش منو
InitializeScene	_	void	آماده کردن صفحه برای ترسیم
WindowsSizeChangedEvent	int,int	void	رویداد تغییر اندازه پنجره
MouseEvent	int,int,int,int	void	رویداد ماوس
RenderGrid	int	void	ترسیم صفحه مشبک
RenderScene	_	void	ترسیم اشکال

جدول ۲-۲: توابع رابط کاربری

۲,۳. پیاده سازی

۲,۳,۱. main

تابع main برنامه که اول از همه اجرا می شود. ورودی های آن برای استفاده از خط فرمان است که در این برنامه کاربردی ندارد.

```
int main(int argc, char* argv[])
{


	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

اندازه صفحه و ایجاد صفحه

```
	glutInitWindowSize(800,600);
	glutCreateWindow("Abbas Allahyari's Polygon");
	glutReshapeFunc(WindowsSizeChangedEvent);
	glutMouseFunc(MouseEvent);
	InitializeScene();
```

ایجاد منوی رنگ ها

```
	int colorMenu = glutCreateMenu(menu);
```

اضافه کردن گزینه های مختلف به منو

```
	glutAddMenuEntry("Black", 1);
	glutAddMenuEntry("Gray", 2);
	glutAddMenuEntry("Red", 3);
```

```
	glutAddMenuEntry("Green", 4);
	glutAddMenuEntry("Blue", 5);
	glutAddMenuEntry("Orange", 6);
	glutAddMenuEntry("Yellow", 7);
```

نسبت داده کلیک راست ماوس به منو، پس هنگامی که کلیک راست شود منو ظاهر می شود.

```
	glutAttachMenu(GLUT_RIGHT_BUTTON);
```

ساختن اولین چند ضلعی که به طور پیش‌فرض چند ضلعی جاری در نظر گرفته می شود.

```
	currentPolygon=new Shapes::Polygon();


	glutDisplayFunc(RenderScene);
	glutMainLoop();
	return 0;
}
```

menu ۲,۳,۲.

در این قسمت با توجه به ورودی که گزینه انتخاب شده در منو است رنگ چند ضلعی را تعیین می کنیم.

```
void menu(int value){
	switch(value){
	case 1: //black
		currentPolygon->setColor(0,0,0);
		break;
	case 2://gray
		currentPolygon->setColor(128,128,128);
		break;
	case 3:		//red
```

```
            currentPolygon->setColor(255,0,0);
            break;
      case 4: //green
            currentPolygon->setColor(0,255,0);
            break;
      case 5: //blue
            currentPolygon->setColor(0,0,255);
            break;
      case 6: //orange
            currentPolygon->setColor(252, 188,103);
            break;
      case 7: //yellow
            currentPolygon->setColor(255, 255,0);
            break;
      }
```

در آخر منو را از کلیک راست ماوس جدا می‌کنیم پس دیگر در هنگام کلیک راست منویی ظاهر نمی شود.

```
      glutDetachMenu(GLUT_RIGHT_BUTTON);
}
```

InitializeScene ۲,۳,۳.

در اینجا رنگ زمینه را سفید قرار می دهیم.

```
void InitializeScene(){
      glClearColor(1.0,1.0,1.0,1.0);
      glMatrixMode (GL_PROJECTION);
```

```
}
```

۲,۳,۴. WindowsSizeChangedEvent

این تابع میدان دید را با توجه به اندازه صفحه تغییر می دهد.

```
void WindowsSizeChangedEvent(int w, int h){

{

		GLfloat nRange = 100.0f;
```

از تقسیم بر صفر جلو گیری می کنیم.

```
		if(h == 0)

			h = 1;
```

محدوده میدان دید را مشخص می کنیم.

```
		glViewport(0, 0, w, h);


		glMatrixMode(GL_PROJECTION);

		glLoadIdentity();
```

در اینجا جهت میدان دید را تعیین می کنیم.

```
		if (w <= h)

			glOrtho (-nRange, nRange, -nRange*h/w, nRange*h/w, -nRange, nRange);

		else

			glOrtho (-nRange*w/h, nRange*w/h, -nRange, nRange, -nRange, nRange);


		glMatrixMode(GL_MODELVIEW);

		glLoadIdentity();

	}

}
```

۲,۳,۵. MouseEvent

این تابع برای پاسخ‌گویی به رویداد های مربوط به ماوس است که از آن برای درج نقطه در صفحه استفاده می شود.

```
void MouseEvent(int button,int state,int x,int y){
```

محور مختصات داخلی OpenGL

```
	double worldX=0, worldY=0, worldZ=0;
```

تعریف سه آرایه برای نگهداری ماتریس های projection، view و model

```
	GLint view[4];

	GLdouble  projMatrix[16];//projection matrix

	GLdouble  modelMatrix[16];
```

پر کردن آرایه های گفته شده، با ماتریس های موجود در صفحه.

```
	glGetDoublev (GL_MODELVIEW_MATRIX, modelMatrix);

	glGetDoublev (GL_PROJECTION_MATRIX,projMatrix);

	glGetIntegerv( GL_VIEWPORT, view );
```

با استفاده از تابع gluUnProject مختصات تقطه صفحه[1] را به نقطه جهان[2] در OpenGL تبدیل می کنیم. این کار این امکان را می‌دهید تا در آن محل نقطه مورد نظر را ترسیم کنیم، و با استفاده از ماوس نقاط را دریافت کنیم.

```
	gluUnProject(x,view[3]-y,0.5,modelMatrix,projMatrix,view, &worldX, &worldY,
&worldZ);
```

بعد از اجرای این دستور مختصات نقطه جهان به سه متغیر worldX، wolrdY و worldZ نسبت داده می شوند. از آنجایی که این برنامه سه بعدی نیست، با محور Z ها کاری نداریم.

1. Screen point
2. World point

به محض اینکه کلیک چپ زده شد، منو را از روی کلیک راست ماوس برداشته، و نقطه جاری را به عنوان اولین ضلع چند ضلعی به آن اضافه می کنیم.

```
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN){
        glutDetachMenu(GLUT_RIGHT_BUTTON);
        currentPolygon->addVertex(worldX,worldY);
    }
    if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
    {
```

در زمانی که کلیک راست زده شود، چند ضلعی را با فراخوانی متد finishPolygon بسته. منو را دوباره به کلیک راست نسبت می دهیم.

```
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        currentPolygon->finishPolygon();
```

چند ضلعی جدید را به درون لیست چند ضلعی ها قرار می دهیم. و سپس یک چند ضلعی جدید می سازیم.

```
        Polygons.push_back(currentPolygon);
        currentPolygon=new Shapes::Polygon();
    }
```

با استفاده از این تابع باعث می‌شویم تا صفحه دوباره رندر شود.

```
    glutPostRedisplay();
}
```

۲,۳,۶. RenderGrid

برای آسان سازی کار با برنامه با این تابع یک صفحه مشبک ترسیم می کنیم. که به عنوان ورودی فاصله بین هر خانه را می دهیم.

```
void RenderGrid(int distance){
```

GL_POLYGON_STIPPLE را فعال می‌کنیم تا در پس زمینه این صفحه کشیده شود.

```
    glEnable(GL_POLYGON_STIPPLE);
```

تعیین رنگ خطوط

```
	glColor3ub(64,64,64);
```

رسم خطوط افقی

```
	for(int i=-400;i<=400;i+=distance){

		glBegin(GL_LINES);
```

عدد ۳۰۰ از تقسیم عرض صفحه بر ۲ بدست آمده است

```
		glVertex2i(i,-300);

		glVertex2i(i,300);

		glEnd();

	}
```

رسم خطوط عمودی

```
	for(int i=-300;i<=300;i+=distance){

		glBegin(GL_LINES);
```

عدد ۴۰۰ از تقسیم طول صفحه بر ۲ بدست آمده است

```
		glVertex2i(-400, i);

		glVertex2i(400,i);

		glEnd();

	}
```

در آخر GL_POLYGON_STIPPLE را غیر فعال می کنیم.

```
	glDisable(GL_POLYGON_STIPPLE);

}
```

۲,۳,۷. RenderScene

```
void RenderScene()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPointSize(6);
	using namespace Shapes;
```

ترسیم صفحه مشبک

```
	RenderGrid(5);
```

نمایش چند ضلعی فعلی، این کار باعث می‌شود تا به هنگام رسم خطوط نمایش داده شوند.

```
	if(currentPolygon){
		currentPolygon->Render();
	}
```

نمایش چند ضلعی های رسم شده، این تکه کد چند ضلعی هایی که قبلاً رسم شده‌اند را نمایش می دهد.

```
	for(int i=0;i<Polygons.size();i++){
		Polygons[i]->Render();
	}
	glFlush();
}
```

۳. کار با برنامه

ابتدا می‌توان با «کلیک راست» و مشاهده منو، یک رنگ را انتخاب کرد.

سپس با «کلیک چپ» بر روی مختصاتی از صفحه آن نقطه رسم می شود.

نقطه های دیگر فقط با «کلیک چپ» بر روی صفحه ترسیم شده، و به ازای هر ۲ نقطه یک خط بین آن‌ها کشیده می شود.

در نهایت برای بستن چند ضلعی در مختصاتی از صفحه باید «کلیک راست» دوباره زده شود. این بار بجای باز شدن یک منو، چند ضلعی بسته و پر می شود.