Појмови тачка, линија, полигон. Цртање круга.

Тачке, линије, полигони

Појмови тачке, линије и полигона у ***OpenGL***-у имају слично значење као и у математици, али не исто. Пре свега због прецизности њиховог приказа (ограничења растерске графике). На таквим екранима, најмања јединица која се приказује јесте један пиксел. Колико год мали он био, далеко је већи од математичког концепта бесконачно мало (за тачке) или бесконачно танко (за линије).

Тачке се у ***OpenGL***-у представљају чворовима са одређеним координатама. Уколико корисник дефинише дводимензионалну тачку (x и y координате) ***OpenGL*** аутоматски поставља трећу координату на 0 ($z = 0$).

Линија (линијски сегмент) се дефинише помоћу два чвора (почетни и крајњи). Линија у ***OpenGL***-у се може поистоветити са математичким појмом дужи. Постоје једоставни начини за дефинисање серије линијских сегмената (отворених или затворених) о којима ће бити речи.

Полигон је област ограничена серијом линијских сегмената. У OpenGL-у ивице полигона се не смеју сећи. Такође, полигони морају да буду конвексни. Тачније за сваке две тачке које припадају полигону, линијски сегмент који је одређен са те две тачке такође мора припадати полигону. Неконвексни полигони се креирају као уније конвексних полигона.

Слика 1: Конвексни и неконвексни полигони

Како се цртање **правоугаоника** често користи у графичким апликацијама, ***OpenGL*** садржи методе

```
void glRect{sifd}(TYPE x1, TYPE y1, TYPE x2, TYPE y2)
void glRect{sifd}v(const TYPE *v1, const TYPE *v2)
```

којима се црта правоугаоник са свим пикселима унутар његових ивица. У првој функцији се прослеђују два наспрамна темена правоугаоника (x_1, y_1) и (x_2, y_2). Креира се полигон са задатим наспрамним теменима који лежи у равни $z = 0$ и странице су му паралелне осама x и y. У другој функцији обе тачке правоугаоника се прослеђују као низ са два елемента који представљају координате (x, y) тих тачака.

Глатке **криве** и **закривљене** површи се цртају приближно. Оне се апроксимирају кратким линијским сегментима или малим полигонима. На следећим сликама се може видети како се један кружни лук може апроксимирати кратким линијским сегментима.

Слика 2:.Апроксимирање кружног лука линијским сегментима

Да би се могло почети са цртањем тачака, линија и полигона прво је потребно научити како се дефинишу чворови. Дефинисање чворова се врши искључиво између наредби `glBegin(GLenum mode)` и `glEnd()`. Ове две функције се заједно са чворовима дефинисаним између њих користе за цртање геометријске примитиве одређене аргументом `mode`. Чворови се дефинишу функцијама

`void glVertex[234]{sifd}(`*`TYPE coords`*`)` и

`void glVertex[234]{sifd}v(const `*`TYPE* coords`*`).`

У првом случају прослеђују се координате, док се у другом прослеђује вектор са координатама чвора који се дефинише.

Тачка се црта наредбама:

```
glBegin(GL_POINTS);
  glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Линија или линијски сегмент се може нацртати наребама:

```
glBegin(GL_LINES);
  glVertex3f(1.0, 1.0, 0.0);
  glVertex3f(3.0, 1.0, 0.0);
glEnd();
```

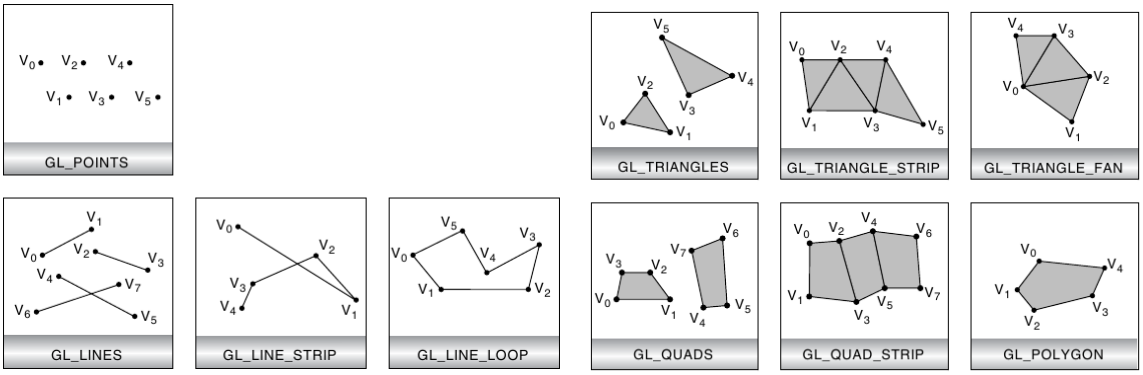
док се полигон црта следећим скупом наредби:

```
glBegin(GL_POLYGON);
  glVertex3f(1.0, 1.0, 0.0);
  glVertex3f(3.0, 1.0, 0.0);
  glVertex3f(3.0, 4.0, 0.0);
  glVertex3f(1.0, 4.0, 0.0);
glEnd();
```

Константа `mode` може имати следеће вредности (примери су на сликама испод):

- `GL_POINTS` – Цртају се појединачне тачке.
- `GL_LINES` – Парови узастопних чворова се интерпретирају као индивидуални линијски сегменти.
- `GL_LINE_STRIP` – Црта се серија повезаних линијских сегмената.

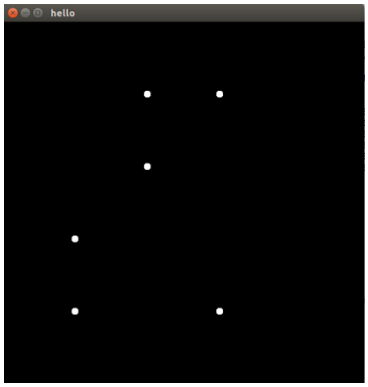
- `GL_LINE_LOOP` – Црта се серија повезаних линијских сегмената где су спојени први и последњи чвор.
- `GL_TRIANGLES` – Тројке чворова се интерпретирају као троуглови.
- `GL_TRIANGLE_STRIP` – Црта се повезана трака троуглова тако да сваке три узастопне тачке представљају један троугао.
- `GL_TRIANGLE_FAN` – Црта се повезана лепеза троуглова. Први чвор припада сваком троуглу, а заједно са свим паровима узастопних тачака дефинише по један троугао.
- `GL_QUADS` – Четворке чворова се интерпретирају као четвороструки полигони.
- `GL_QUAD_STRIP` – Црта се повезана трака четвороуглова. Прве четири тачке чине један четвороугао, а потом друге две од ових тачака са две следеће тачке креирају још по један четвороструки полигон све до краја листе тачака.
- `GL_POLYGON` – Црта се конвексни полигон чије су ивице дефинисане наведеним тачкама.



Слика 3: Различити модови за цртање

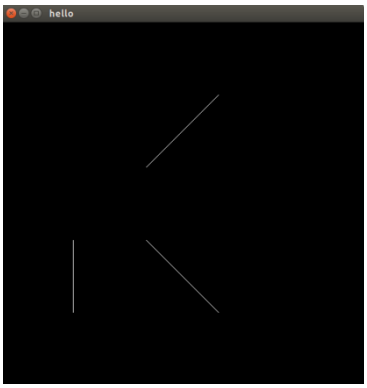
Пример 1.1 `GL_POINTS` - цртање тачака: Да би се боље виделе тачке на слици поред повећана је њихова величина на 10 пиксела наредбом `glPointSize(10.0)`, а да не би тачке биле цртане као квадрат величине 10×10 пиксела укључена је опција ублажавања оштрих ивица наредбом `glEnable(GL_POINT_SMOOTH)`. Мод цртања `GL_POINT_SMOOTH` говори о томе да ће наведени чворови бити представљени као тачке на екрану, па за шест наведених чворова добијамо шест тачака као на слици испод.

```
glEnable(GL_POINT_SMOOTH);
glPointSize(10.0);
glBegin(GL_POINTS);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(1.0, 2.0, 0.0);
    glVertex3f(2.0, 3.0, 0.0);
    glVertex3f(3.0, 4.0, 0.0);
    glVertex3f(2.0, 4.0, 0.0);
    glVertex3f(3.0, 1.0, 0.0);
glEnd();
```



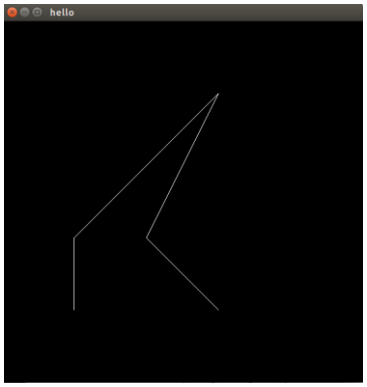
Пример 1.2 GL_LINES - цртање линија: Шест наведених чворова одређује три линије, и то прва два чвора једну линију, друга два чвора другу линију, а пети и шести чвор трећу линију.

```
glBegin(GL_LINES);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(1.0, 2.0, 0.0);
   glVertex3f(2.0, 3.0, 0.0);
   glVertex3f(3.0, 4.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
glEnd();
```



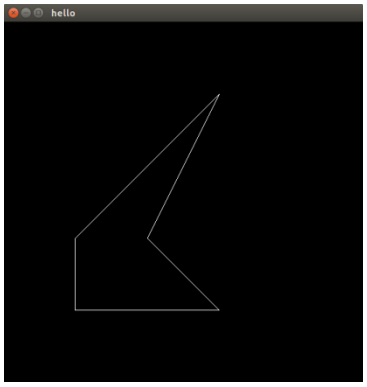
Пример 1.3 GL_LINE_STRIP - цртање низа линија: Сваки пар узастопних чворова одређује једну линију. Редом, први и други чвор прву линију, други и трећи другу линију, трећи и четврти трећу линију итд. У примеру је наведено шест чворова којим је дефинисано пет линија.

```
glBegin(GL_LINE_STRIP);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(1.0, 2.0, 0.0);
   glVertex3f(2.0, 3.0, 0.0);
   glVertex3f(3.0, 4.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
glEnd();
```



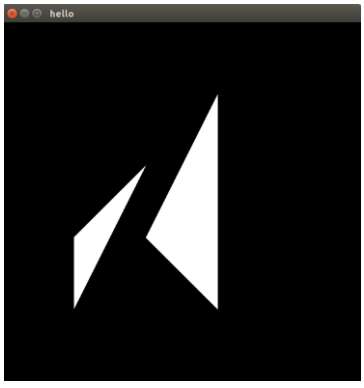
Пример 1.4 GL_LINE_LOOP - цртање затвореног низа линија: Исто као и у претходном примеру, али ће још бити нацртана линија између последњег и првог наведеног чвора.

```
glBegin(GL_LINE_LOOP);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(1.0, 2.0, 0.0);
   glVertex3f(2.0, 3.0, 0.0);
   glVertex3f(3.0, 4.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
glEnd();
```



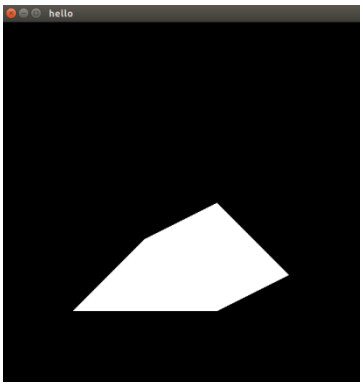
Пример 1.5 GL_TRIANGLES - цртање троуглова: Наводе се тројке чворова које чине један троугао, па прве три тачке у примеру одређују један троугао, а друге три тачке други троугао.

```
glBegin(GL_TRIANGLES);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(1.0, 2.0, 0.0);
   glVertex3f(2.0, 3.0, 0.0);
   glVertex3f(3.0, 4.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
glEnd();
```



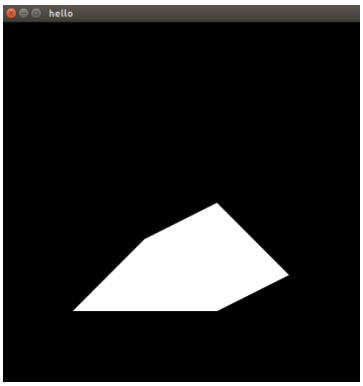
Пример 1.6 GL_TRIANGLES_STRIP - цртање спојених троуглова: Троуглови су дефинисани тројкама узастопних чворова (прва три чвора чине један троугао; други, трећи и четврти чвор чине други троугао; трећи, четврти и пети чине трећи троугао; четврти, пети и шести чвор четврти троугао).

```
glBegin(GL_TRIANGLE_STRIP);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(2.0, 1.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
   glVertex3f(3.0, 2.5, 0.0);
   glVertex3f(4.0, 1.5, 0.0);
glEnd();
```



Пример 1.7 GL_TRIANGLES_FAN - цртање узастопних троуглова са заједничким чвором (прва тачка припада свим троугловима; први чвор са другим и трећим чини први троугао; први чвор са трећим и четвртим чини други троугао; први чвор са четвртим и петим чини трећи троугао; први чвор са петим и шестим чини четврти троугао).

```
glBegin(GL_TRIANGLE_FAN);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(3.0, 1.0, 0.0);
   glVertex3f(4.0, 1.5, 0.0);
   glVertex3f(3.0, 2.5, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
glEnd();
```



Детаљније о тачкама

Подразумевана ширина рендероване тачке јесте један пиксел. Та вредност се може променити функцијом

```
void glPointSize(GLfloat size).
```

Вредност *size* мора бити већа од 1.0. Ако је ширина 2.0 тада је величина тачке 2×2 пиксела. Ефектом ублажавања оштрих ивица се избегава да тачка буде квадрат што се постиже наредбом

```
glEnable(GL_POINT_SMOOTH).
```

Ова наредба важи за цртање свих тачака које се креирају након њеног позива. Да би се искључило ублажавање оштрих ивица позива се функција

```
glDisable(GL_POINT_SMOOTH).
```

Детаљније о линијама

Подразумевана ширина линије која се црта је 1.0, али се и то може променити функцијом

`void glLineWidth(GLfloatsize).`

Параметар `size` мора бити већи од 0. Могуће је ублажавање ивица функцијом

`glEnable(GL_LINE_SMOOTH).`

Као и за тачке, ово важи са све линије које се цртају до позива функције

`glDisable(GL_LINE_SMOOTH).`

Уколико желимо да нацртамо испрекидану линију, прво морамо да омогућимо такво исцртавање позивом функције

`glEnable(GL_LINE_STIPPLE)`

пре почетка цртања линије (пре наредбе `glBegin`).

Да би се линија нацртала испрекидано, пре позива креирања линије потребно је дефинисати изглед испрекидане линије. То се постиже наредбом

`glLineStipple(GLint factor, GLushort pattern)`

где је $pattern$ 16-битни број састављен од нула и јединица који представља начин исцртавања линије, и то тако да 1 означава да се црта тај пиксел, а 0 да се не црта, и то почевши од цифре најмање тежине. Шаблон се може развући параметром `factor` који умножава број узастопних нула и јединица. Ако имамо да је $pattern = 1111110000000011$ и параметар $factor = 1$ прво се цртају два пиксела, па прескачу осам, па цртају шест и све тако док се не исцрта цела линија. Да је вредност параметра $factor = 2$, тада би се цртало четири пиксела, па прескакало шеснаест, па цртало дванаест пиксела, итд.

Детаљи о полигонима

Полигони садрже две стране и на различите начине могу бити рендеровани у зависности која је страна окренута ка посматрачу и како је одређено при самом цртању. То омогућава различит поглед на објекте где је очигледна разлика између делова који се налазе унутар или ван тог објекта. Ако није другачије наведено и предња и задња страна се цртају на исти начин, односно испуњени пикселима. Могуће је цртати само контуре или темена полигона. Начин исцртавања полигона бира се функцијом

`glPolygonMode(GLenum face, GLenum mode).`

Ова функција контролише интерпретацију полигона за растеризацију. Параметар `face` одређује на коју страну полигона се односи мод који се наводи и може бити `GL_FRONT`, `GL_BACK` или `GL_FRONT_AND_BACK`, односно предња, задња или обе стране полигона, редом. Начин исцртавања изабране стране полигона одређује се параметром `mode`. Полигон може да се црта испуњен пикселима (`GL_FILL`), да се цртају само ивице полигона (`GL_LINE`) или само крајње тачке ивица полигона (`GL_POINT`). У примеру испод црта се полигон коме се предња страна исцртава испуњена пикселима, док се са задње стране полигона цртају само ивице.

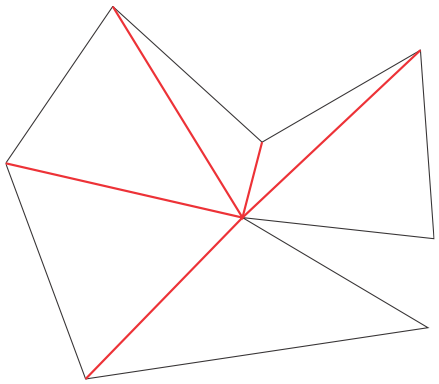
```
glPolygonMode(GL_FRONT, GL_FILL);
glPolygonMode(GL_BACK, GL_LINE);

glBegin(GL_POLYGON);
   glVertex3f(1.0, 1.0, 0.0);
   glVertex3f(4.0, 1.0, 0.0);
   glVertex3f(2.0, 2.0, 0.0);
   glVertex3f(3.0, 3.0, 0.0);
   glVertex3f(2.0, 4.0, 0.0);
glEnd();
```

Уколико се темена полигона наводе у смеру супротном од кретања казаљке на часовнику гледано из угла камере, страна ка камери је предња. Оно шта ***OpenGL*** сматра предњом страном могуће је променити употребом функције `glFrontFace(GLenum mode)`. Тачније, задаје се смер којим се добија да је предња страна ка камери. Параметар `mode` може бити или `GL_CCW` или `GL_CW`. Ако је параметар `mode`= `GL_CCW` тачке је потребно наводити супротно кретању казаљке на часовнику да би предња страна полигона била ка камери, а ако је `mode`= `GL_CW` тачке треба наводити у смеру кретања казаљке часовника.

У потпуно затвореним површима конструисаним од непрозирних полигона константне оријентације, уколико се камера налази изван површи, задње стране тих полигона никада неће бити видљиве, па задње стране тих полигона нема потребе рендеровати. Супротно, уколико је камера унутар површи предње стране тих полигона неће бити видљиве и онда нема потребе рендеровати предње стране тих полигона. У ***OpenGL***-у је могуће избећи рендеровање стране полигона која није видљива. Прво се мора омогућити та опција наредбом `glEnable(GL_CULL_FACE)`, а тек потом позвати функција `glCullFace(GLenum mode)`, где `mode` може бити `GL_BACK`, `GL_FRONT` или `GL_FRONT_AND_BACK`, што указује на страну полигона коју не треба цртати.

Као што је речено, неконвексне полигоне је потребно поделити на конвексне полигоне (углавном троуглове) пре исцртавања. Поступак у коме се неконвексни полигони деле на троуглове назива се триангулација. Ако се цртају делови неконвексног полигона може се десити да се неке линије исцртавају два пута. Уколико би неконвексан полигон са слике испод био нацртан са шест троуглова (подела на слици), тада би црвене линије са слике биле цртане по два пута, а чак не морају ниједном. Навођењем функције `glEdgeFlag(GLboolean flag)` пре прве тачке неке ивице полигона можемо изабрати да ли се та ивица црта или не. Уколико је параметар `flag` једнак `GL_TRUE` тада се та ивица црта, а уколико је једнака `GL_FALSE` онда се не исцртава. Постављањем на један мод, он важи све до тренутка док се не промени поновним позивом функције.



Слика 4: Цртање неконвексних полигона

У примеру испод се може видети како се црта контура неконвексног полигонапомоћу три конвексна (троугла). Избегнуто је цртање линија које припадају тим троугловима, а не припадају контури неконвексног полигона.

```
glBegin(GL_POLYGON);
    glEdgeFlag(GL_TRUE);
    glVertex3f(1.0, 1.0, 0.0);
    glEdgeFlag(GL_TRUE);
    glVertex3f(4.0, 1.0, 0.0);
    glEdgeFlag(GL_FALSE);
    glVertex3f(2.0, 2.0, 0.0);
 glEnd();

 glBegin(GL_POLYGON);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(2.0, 2.0, 0.0);
    glEdgeFlag(GL_TRUE);
    glVertex3f(2.0, 4.0, 0.0);
 glEnd();

 glBegin(GL_POLYGON);
    glVertex3f(2.0, 2.0, 0.0);
    glVertex3f(3.0, 3.0, 0.0);
    glEdgeFlag(GL_FALSE);
    glVertex3f(2.0, 4.0, 0.0);
 glEnd();
```

Круг

Задатак

Цртање круга са задатим центром, полупречником и бројем тачака којим се дефинише кружница (подела кружнице).

Решење

Прво кружницу којом је дефинисана унутрашњост круга који цртамо поделимо на коначан број ($circle\_points$) подједнако удаљених тачака тако да прва тачка те поделе легне на праву која пролази кроз центар кружнице и паралелна је x оси. Један од начина да се креира круг јесте да се креира полигон чија су темена овако описане тачке.

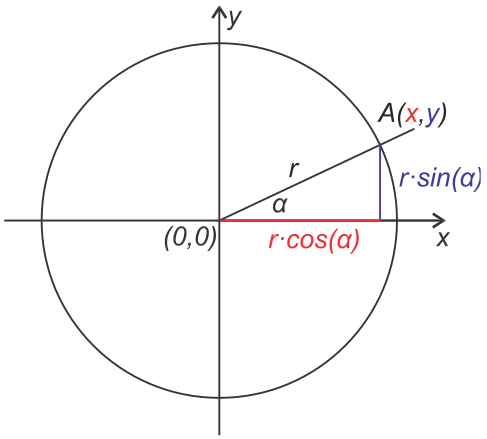
Подсетник:

Нека је A произвољна тачка на кружници са центром у координатном почетку и полупречника r. Нека је α угао који полуправа са почетком у тачки (0,0) и која пролази кроз тачку $A(x,y)$ заклапа са x осом. Важе следеће једнакости:

$$cos(\alpha) = \frac{x}{r},\ sin(\alpha) = \frac{y}{r}$$

Тада се координате тачке A могу добити на следећи начин:

$$(x,y) = \big(r \cdot cos(\alpha), r \cdot sin(\alpha)\big)$$



Центар круга треба да буде у задатој тачки $C(x_0, y_0)$, па је произвољна тачка $A(x,y)$ тог полигона

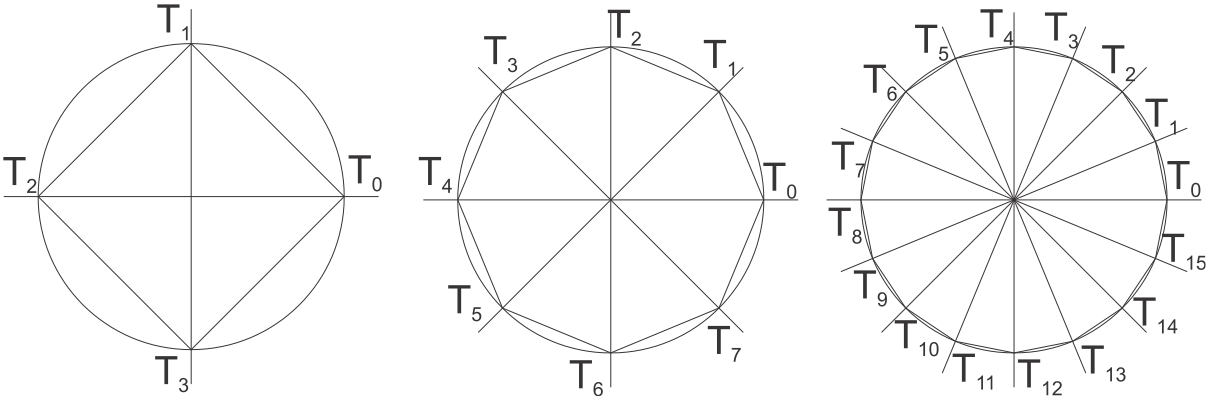
$$\big(x_0 + r \cdot cos(\alpha), y_0 + r \cdot sin(\alpha)\big)$$

где је α угао који полуправа са почетком у тачки $C(x_0, y_0)$ која пролази кроз тачку $A(x,y)$ заклапа са x осом. Тај угао је једнак 0 за прву тачку на кружници, а потом је угао сваке следеће тачке већи за угао $\frac{2\pi}{circle\_points}$ од угла претходне тачке. Тада се тачке полигона могу дефинисати формулом:

$$T_i(x_i, y_i) = \left(x_0 + r \cdot cos\left(i \cdot \frac{2\pi}{circle\_points}\right), y_0 + r \cdot sin\left(i \cdot \frac{2\pi}{circle\_points}\right)\right),$$

$$\text{за } i = \overline{0, circle\_points - 1}$$

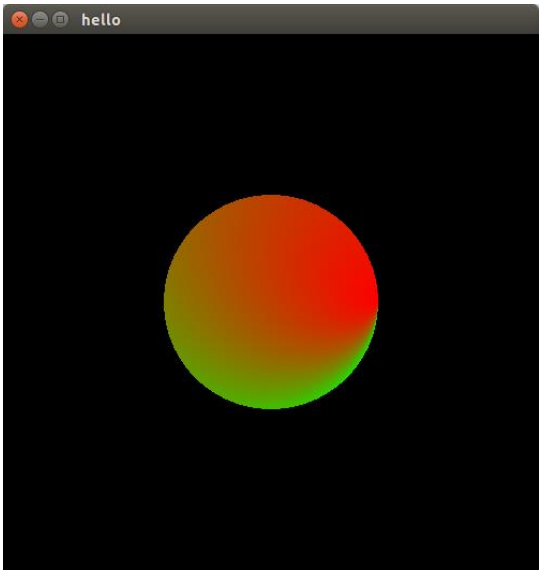
Пример цртања круга полигоном са 4, 8 и 16 темена дат је на слици испод.



$circle\_points = 4$ $circle\_points = 8$ $circle\_points = 16$

Слика 5: Представљање круга полигоном са 4, 8 и 16 темена

У примеру 2 приказано је како се црта задати круг помоћу полигона. Приликом навођења чворова полигона у петљи сваки пут се бира другачија боја за цртања (од црвене ка зеленој) командом `glColor3f (red, green, blue)`, где се параметри `red` и `green` мењају са `red -= 1.0 / circle_points` и `green += 1.0 / circle_points`, док вредност параметра `blue = 0` остаје иста. Као резултат добија се круг (Слика 6: Рендерован круг коришћењем OpenGL-а*Слика 6*).



Слика 6: Рендерован круг коришћењем OpenGL-а

Програмски код

```
void drawCircle(GLdouble x, GLdouble y, GLdouble r, GLint circle_points)
{
	double red = 1.0;
	double green = 0.0;
	double blue = 0.0;
	double alpha = 2 * PI // circle_points;

	glBegin(GL_POLYGON);
	//  glBegin(GL_LINE_LOOP);
	for (int i = 0; i < circle_points; i++)
	{
		glColor3f (red, green, blue);
		red -= 1.0 / circle_points;
		green += 1.0 / circle_points;
		double angle = i * alpha;
		glVertex2f(x + r * cos(angle), y + r * sin(angle));
	}
	glEnd();
}
```

Домаћи задатак:

Нацртати цртаног лика или животињу по избору.

Радове слати на е-маил: dragutin.ostojic@pmf.kg.ac.rs.