

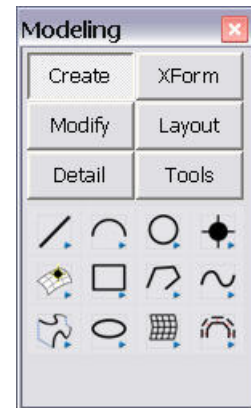
# 4 Creating Geometry

## 4.0 Overview

---

Good planning is the first step in creating a high quality CAD drawing. In the previous chapter, some basic strategies for preparing a new KeyCreator file were discussed. As you saw, file setup involves using a combination of *part-specific* and *session-specific* settings. Some of these settings simply control the part's appearance, while others have an effect on the "intelligence" of the part. And since KeyCreator has such a wide range of uses, it's hard to prescribe a specific setup that will suit all needs. What will determine the most suitable setting environment is your particular design application.

Having completed a discussion of basic file setup, we can now introduce the various kinds of geometry KeyCreator can create, and apply them to an actual production drawing. These functions are all conveniently located under the **Modeling** menu as shown. In KeyCreator, geometric shapes are treated as objects and are referred to as **entities**. Entities created under this menu can be added directly to your file without the help of existing geometry. There are many other functions that will indirectly "create" or add entities to the file. However, those functions are geared more to the editing process, and rely on existing entities. For example, later you'll learn about the Modify functions. The **Transform** functions, introduced in the next chapter, will add new geometry to the part by *copying* existing entities. Detailing, discussed in Chapter 7, will also introduce new geometry to your part in the form of notes (text), dimensions, cross hatching, and labels. However, detailing is considered to be an entirely separate phase of the drawing process.



## What's Covered in This Chapter

Earlier in Chapter 1, our brief tour of the KeyCreator Interface revealed the numerous functions contained within the Modeling menu. These included tools for creating lines, arcs, points, circles, etc. A typical function such as “Create a Line” will also offer a number of methods for creating that entity. Often, these options are so closely related, that a detailed explanation of each would be quite time consuming. For this reason, we recommend that you consult either the **On-line help** for basic command usage. For example, if you're not sure exactly how to use (or where to find) the **Create>Polyline** function, simply click “**Contents Help**” as shown at right. Within the Help screen, you'll find all of the menus conveniently indexed.



This chapter begins by explaining KeyCreator's unique style of command workflow and user interaction. Next, you'll find a discussion concerning the different *types* of KeyCreator entities, followed by a detailed look at the **Universal Position Menu**. A brief overview of these options was given when the conversation bar was first introduced back in Chapter 1. Creating and modifying geometry often go hand in hand. Therefore, the tools for **trimming**, **extending**, and **breaking** entities are also covered in this chapter, even though these functions are actually part of the **Modeling** menu.

## Chapter Projects

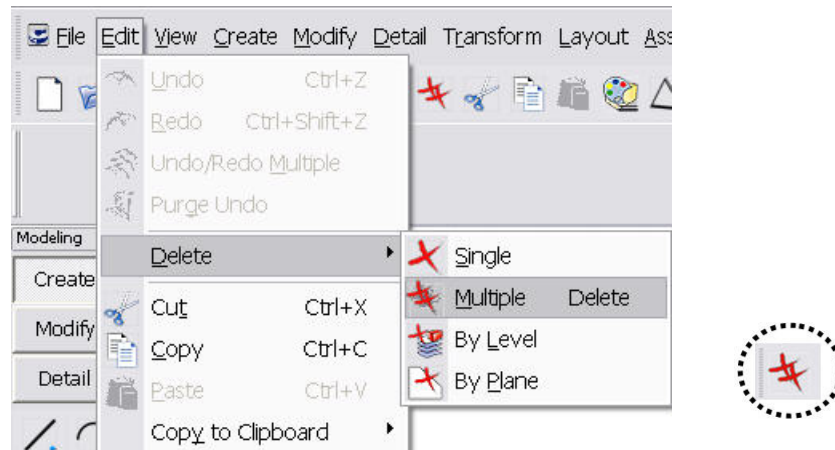
To apply the concepts that you've learned, exercises are included at the end of each section. An **exercise** is a procedure that suggests one possible way to construct the part. There are usually a number of strategies that can be used to attack the same construction. Apart from the exercises, you'll also find a wealth of **projects** that you can try on your own. Projects are more challenging versions of a sample exercise, and can be developed using similar techniques. But unlike an exercise, a project is not outlined using a step-by-step presentation. But rather provides an excellent opportunity to plan and develop the part using your own particular approach.

The first part you'll create is a simple one entitled “Digital Doodle”, an exercise intended to give the user a “feel” for the KeyCreator workspace. This is followed by a succession of sketching exercises designed to familiarize the user with the concept of randomly placing entities in the file. A final application exercise that incorporates the use of the positional options is also included. Here, you will gain the expertise needed to complete the more challenging projects that require a great deal of trimming and extending. Before we proceed with any of these topics, the first order of business will be to introduce the two most frequently used functions when creating geometry - **Delete** and **Undo/Redo**.

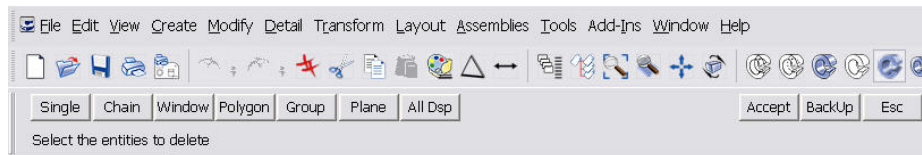
## 4.1 Deleting and Recalling Entities

### Delete Entities

We all make mistakes! To erase entities, select **Edit>Delete>Multiple** from the **Edit** pull down menu as shown below. For convenience, you can also use the corresponding icon shown below.



The **Universal Selection Menu (USM)**, shown below, will allow you to specify the entity or entities you want to delete. Selection sets are discussed in more detail in Chapter 5. For now, you can use the “**Single**” select option as shown below. The Selection Menu options appear on the Conversation Bar whenever you are required to select entities on which an action is to be performed.

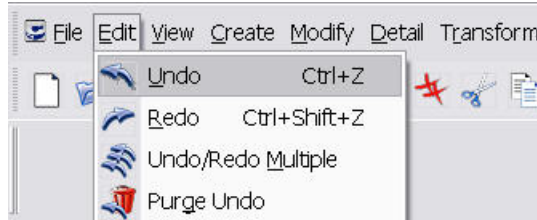


The Selection Menu defaults to a toggle selection method that allows the selection of single entities, or using a drag and window choose to select multiple entities. The selected entities are either added or removed from the current selection. Using the cursor, you select the entities you intend to modify. Or, if you wish to use another selection option available from the Conversation bar, select the appropriate button, then select the entity, or entities, and proceed with your changes.

## Undo/Redo

The **Undo/Redo** functions will restore the previous part state. You can undo an unlimited number of operations. Conversely, you can go forward (Redo) to reverse the Undo chain.

Once you have made a final **File>Save**, and have closed the file, the Undo/Redo chain will begin anew when the file is next loaded. The Undo “buffer” is not saved with the file.



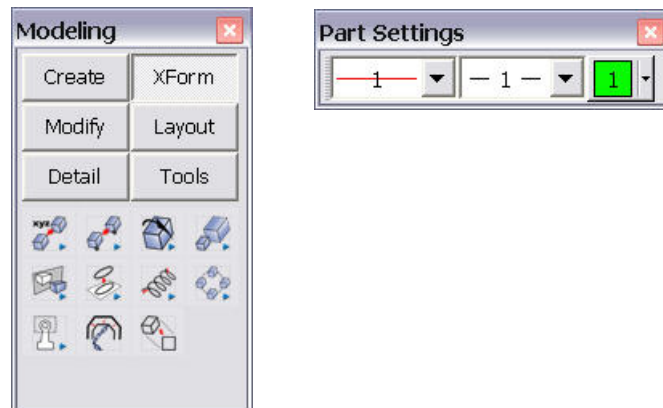
## 4.2 Command Workflow

---

A basic overview of the entire KeyCreator interface was provided earlier in Chapter One. Recall that a key component of the interface was the **conversation bar**. When a command is active, this is the area where the various prompts are displayed. We also learned that depending upon the command, different options will appear within the conversation bar area. Before proceeding with any drawing exercises, let's take a closer look at how commands are organized and the different ways that you can interact with them via the conversation bar.

### Command Modes

KeyCreator functions are designed to execute in either of two modes: **root** or **immediate**. With the exception of **macro** functions, each function has a pre-defined mode that can not be changed. In general, all functions contained in the **Modeling menu** are considered root mode commands, whereas functions built into the **Settings Window** are considered to be immediate mode. Other menus such as Pull-downs and toolbars, will contain a mix of both types. The differences between root and immediate are explained below and on the following page.



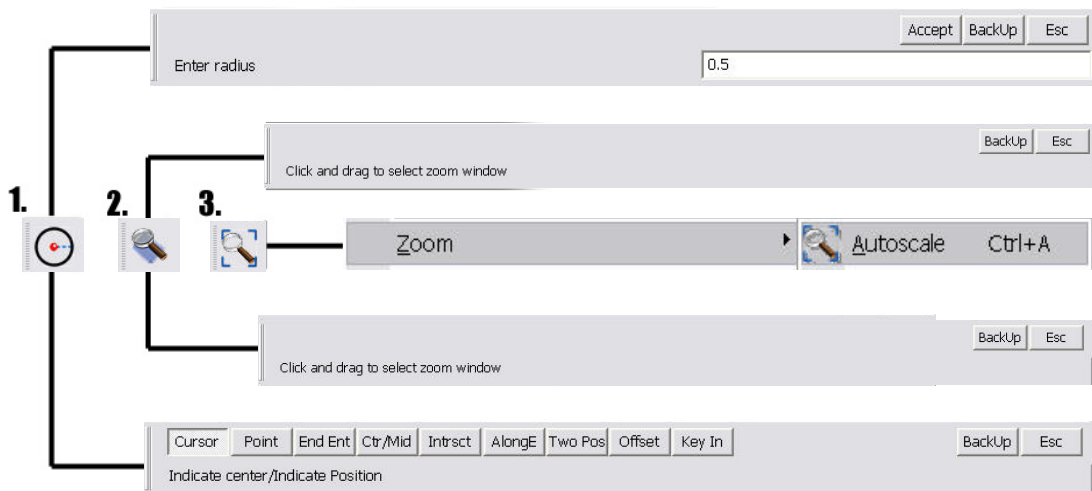
### Root Functions

When invoked, a **root function** forces any conversation currently in progress to automatically terminate, in favor of itself. For example, if you decide to create a circle while the line function is still active, the line command is automatically ended, and you're placed at the **root** of the circle command. In this case, both the line and the circle command are root mode commands.

## Immediate Mode Functions

Unlike root mode, **Immediate mode** functions *may* be invoked, or “nested” within other functions without causing them to automatically terminate. In doing so, it temporarily suspends the execution of the other function. There are numerous immediate mode commands. Among the more popular ones are view related functions such as **Pan**, **Zoom**, and **Autoscale** introduced earlier in Chapter 2.

Immediate mode functions can even be nested within *other* immediate mode commands. For example, while creating a **circle**, the **View>Zoom>Window** function may be safely executed. But before completing the zoom function, you decide to execute **View>Zoom>Autoscale** in order to further enhance the current view. The autoscale function completes, and returns control to the Window function still in progress. When that function is complete, control is then passed back to the line function. Diagrammatically, this immediate mode nesting appears as shown below.



## Dialog Boxes and the Conversation Bar

Most of the KeyCreator commands utilize, or converse with, dialog boxes whenever possible. Dialog boxes provide a variety of components such as edit fields, drop-down lists, radio buttons, and image thumbnails, all designed to make command entry decisions much more convenient. However, a great majority of basic functions found in the Modeling palette are still implemented directly within the **conversation bar** exclusively. In fact, even dialog-driven commands must rely on the conversation bar for the **positional** and **selection** menus. For example, the popular “**Change Attribute of Entity**” function will first activate a dialog box, and then pass the remainder of the command back to the conversation bar as shown.

**ACTIVATE COMMAND**



**DIALOG BOX**



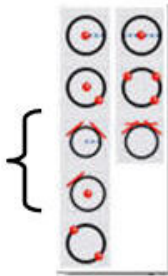
**CONVERSATION BAR  
(SELECTION MENU SHOWN)**

## Direct Functions

KeyCreator allows you to *directly* specify the type of construct to be used. The **Create>Circle** menu contains eight unique tools for creating circles.

Of course, KeyCreator also provides options that offer control over the placement of geometry. But options like tangency and perpendicularity are “built” directly into the functions. The line and arc functions will have similar variations.

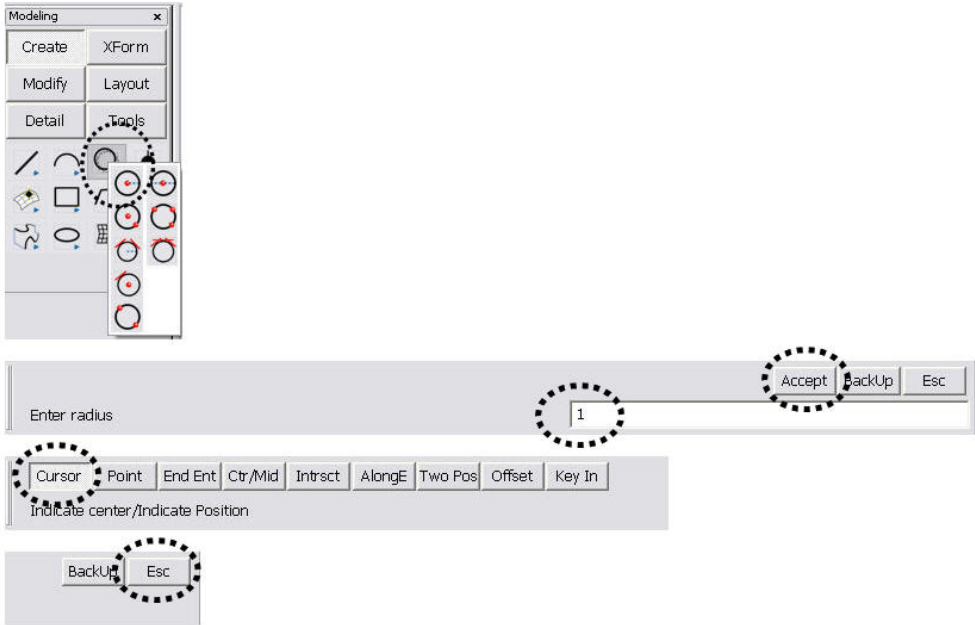
The direct function approach has a few advantages. First of all, the icons themselves clearly depict the various functions. There is no question that the particular function shown below will produce a *circle that is tangent to two neighboring entities*. This, when compared to the rather ambiguous nature of a text-based command prompt, is already a vast improvement, especially for new users. Secondly, when used in connection with user-defined hotkeys, you also gain “one-touch” command activation. Hotkeys and related productivity aids are discussed in detail in Chapter 1.



# Conversation Prompts

Another unique characteristic of a KeyCreator function is the orderly manner in which the prompts are presented. When executed, a function will sequentially interrogate the user within the conversation bar. This step-by-step interaction makes the entire process less prone to errors. Invalid entries are immediately rejected, and incomplete ones stop the function from going on to the next prompt. In either case, the command can not proceed until the entry is corrected.

Some commands may be as simple as a three step conversation, while others may contain as many as ten steps. For example, the conversation sequence for creating a **Circle by center and radius** is shown below.



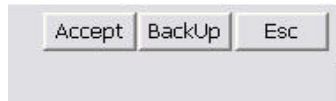
## Accept, Backup, and Escape

A typical KeyCreator function will not self-stop, instead it remains active until you choose to end the routine. This allows you to complete a number of similar tasks by preserving the same command inputs. For example, you can create **25** circles that are *all* of the same radius by staying within the loop of the same function.

There are two ways to terminate a KeyCreator command. The first is by clicking on the “**Esc**” (escape) activity button shown below. For convenience, this button is also mapped directly to the **esc** key located on your keyboard. Either of these will instantly force any command in progress to end, *including* any immediate mode commands that may be nested within the command. This is perhaps the most surefire way to stop a command. The second way to end a command is to invoke a **root** command. These types of commands were discussed in the previous section.

Another significant component of the conversation bar is the “**Accept**” activity button. Some prompts, particularly the data entry fields, will not proceed until this button is clicked.

The “**Back Up**” button can be used to reverse the sequence of a conversation. This lets you correct or adjust previously entered values without having to restart from the beginning. During the selection phase of a function, the “**BackUp**” activity has the same effect of un-selecting.



### 4.3 The Create Functions

---

An overview of the entire KeyCreator Modeling menu was given back in Chapter 1. At that time, the objective was to reveal the various functions and sub-menus residing within this very compact window. It also helped us to appreciate the organization of the menu, which divides into five basic categories, each heading a particular family of tools needed in the workflow of a standard technical drawing. Among them is the **[Create]** menu as shown at right.



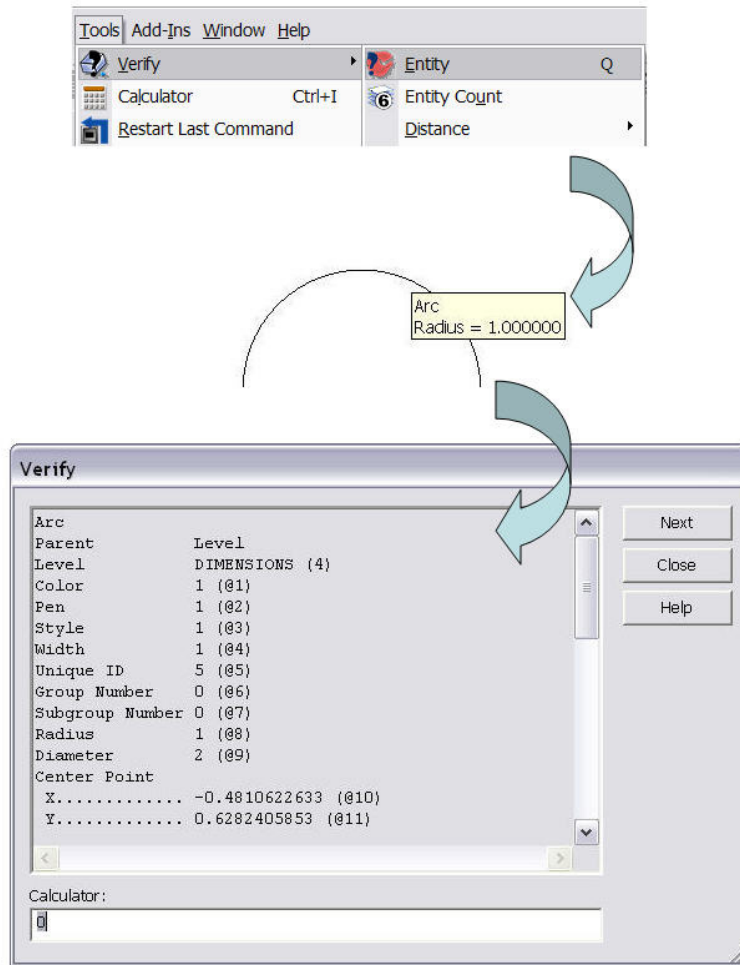
#### Help and Other KeyCreator Resources

In this section, we probe deeper into the [Create] menu, which contains over eighty functions. However, an explanation of each would be quite time consuming, not to mention, downright boring. It is highly recommended that you complement this guide with either the **On-line Help**, since the resources can assist with command usage and syntax. Otherwise, the best way to explore the various Create functions is to apply them in an actual exercise. At the conclusion of each section, you'll find a number of classical parts that apply various geometric constructs.



## Entities and the Geometric Database

When a geometric shape such as an **ARC** is added to the file's *database*, it becomes known as an **entity** as indicated by the **Tools>Verify>Entity** report card shown below. Incidentally, this handy utility, along with a number of other measurement tools, is explained in Chapter 6.



However, not all entities are geometric shapes. A single entity can also be a collection of many entities, all working together in what is known as a **collective**. Some entities are not geometry at all, but will behave much like an ordinary geometric entity.

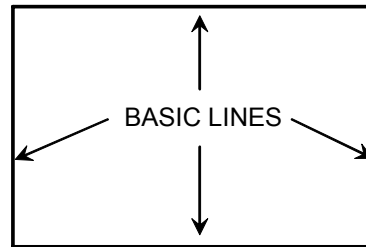
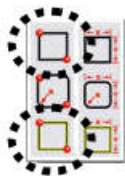
The term “database” was emphasized earlier, because that’s exactly what a KeyCreator file is; It’s a collection of records that are sequentially stored in a file, similar to an ordinary mailing list. But instead of containing names, addresses and zip codes, a KeyCreator file contains the records of various entities like lines, arcs, and instances.

## Basic Entities

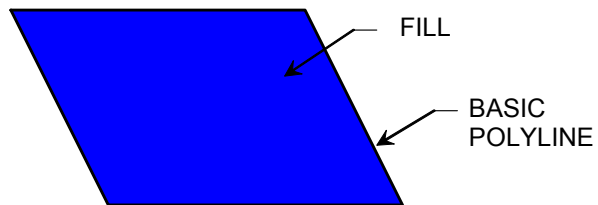
Most of the entities within the [Create] menu are considered *basic* entities. *Basic* entities have their own identity. A line for example, is considered a basic entity. They are also often used internally by KeyCreator to implement more complex entities. Later, you may choose to explore the benefits of solids modeling.

Some of the [Create] functions are actually “routines” that combine basic entities into compound shapes. For example, the **rectangle** function produces a four-sided figure that is assembled using basic **lines** as shown below. Note that the option at the bottom will produce a rectangle using a single basic **polyline** instead.

In other cases, KeyCreator will use a basic entity to redefine a more complex entity type. For example, the **polygon** entity is created by a specific function that “links” a basic polyline together into a loop. As a loop, the polygon entity has the ability to be optionally **filled** using a solid color instead of just an outlined object as shown.



**RECTANGLE  
FUNCTION**



**POLYGON ENTITY**

## Summary of the Create Functions

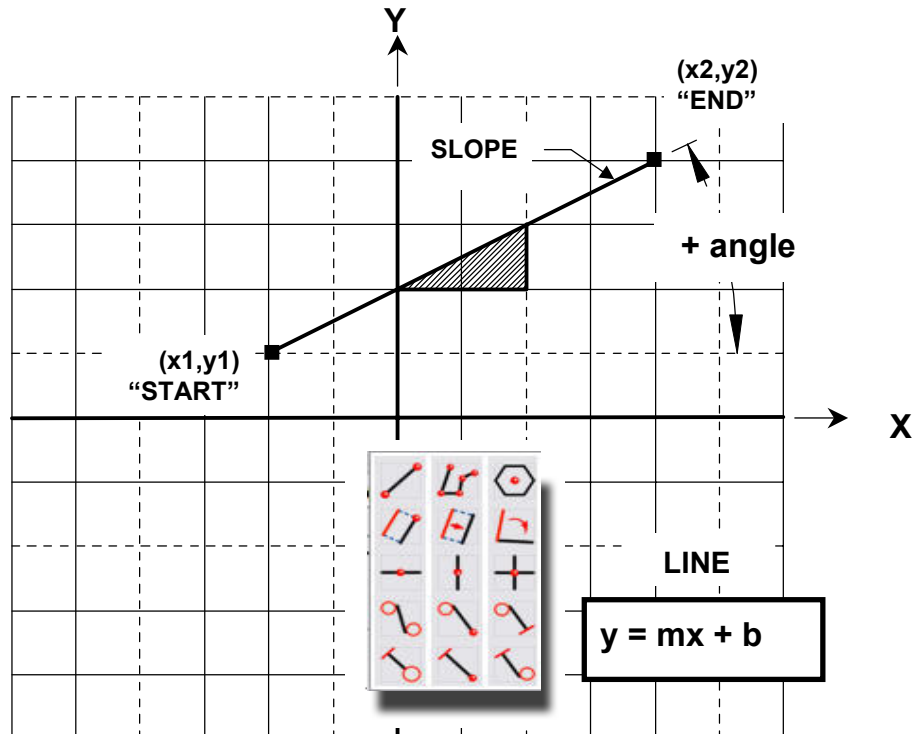
The Create functions are reasonably intuitive and for the most part, self-explanatory in nature. Although every one of these can be used in 2D, some are best suited for use in 3D construction applications. Not all of them are discussed in this section, such as the Mesh function. Esoteric shapes such as hyperbolas and parabolas are also not covered here for, today, these particular shapes are best derived directly from solid models. Conic, another “advanced” entity type, will be explained in this section although only the ellipse form.

The following table is a summary of the various standard KeyCreator entities. Those checked are covered in this section.

|   |                    |            |
|---|--------------------|------------|
| ✓ | LINE               | (basic)    |
| ✓ | ARC                | (basic)    |
| ✓ | CIRCLE             | (basic)    |
| ✓ | POINT              | (basic)    |
| ✓ | POLYLINE           | (basic)    |
| ✓ | FILLET AND CHAMFER | (compound) |
|   | GENERAL CONIC      | (basic)    |
|   | CONIC PARABOLA     | (basic)    |
| ✓ | CONIC ELLIPSE      | (basic)    |
| ✓ | POLYLINE           | (complex)  |
| ✓ | SPLINE             | (basic)    |
|   | MESH               | (compound) |
| ✓ | RECTANGLE          | (compound) |
|   | AUTOSEGMENT        | (compound) |

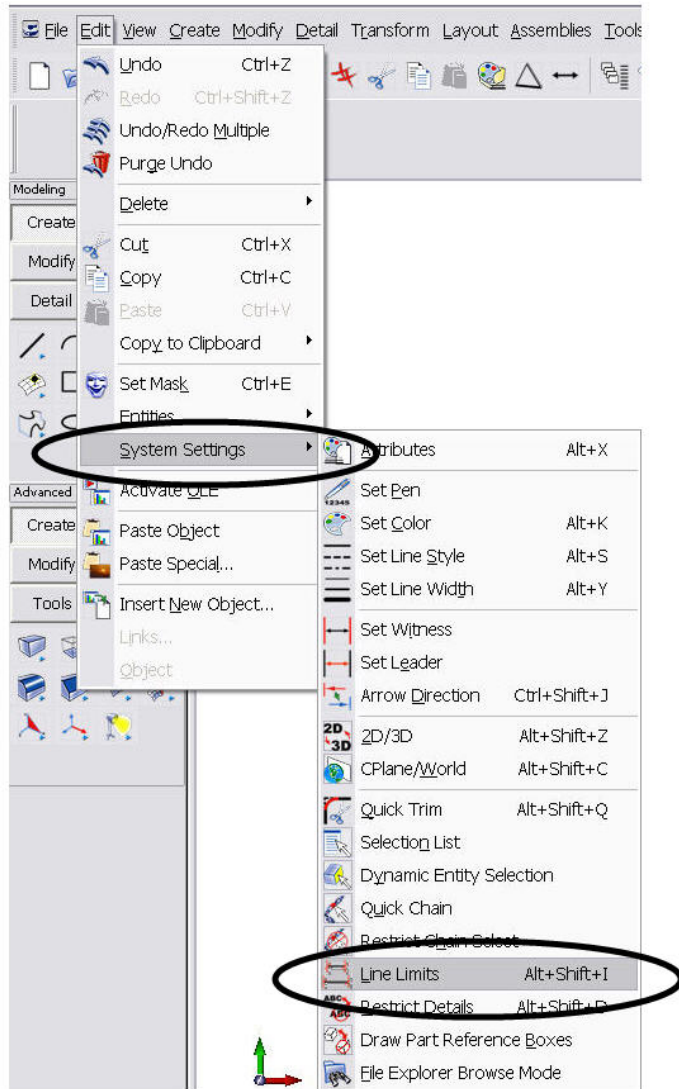
## Lines

This basic entity possesses a length that is determined by the distance between the coordinates of its endpoints. The same two points also act to define its **angle** as well as **slope**. In KeyCreator, positive angles are measured counterclockwise with respect to the positive X-axis as shown. Here, the semantics of direction are determined by which of the two endpoints is created first. The first point, shown as  $x_1, y_1$ , becomes the “start point” of the line. Bear in mind there is a “z” coordinate associated with this and all entities which is not shown here for clarity.



## Construction Lines

As you saw earlier, there are a variety of ways to create line entities. The **Line Limits** toggle, shown at bottom, will add even more flexibility to the existing fifteen line tools. **Line Limits** provides a setting that allows a line to over extend its natural endpoints, thus producing a semi-infinite construction line. Only line functions that produce a *single line segment* such as “**Create a line by indicating endpoints**” or “**Create a Line at a specified angle**” are affected by the Line Limits function. The line will extend out to the boundaries of the Viewport.



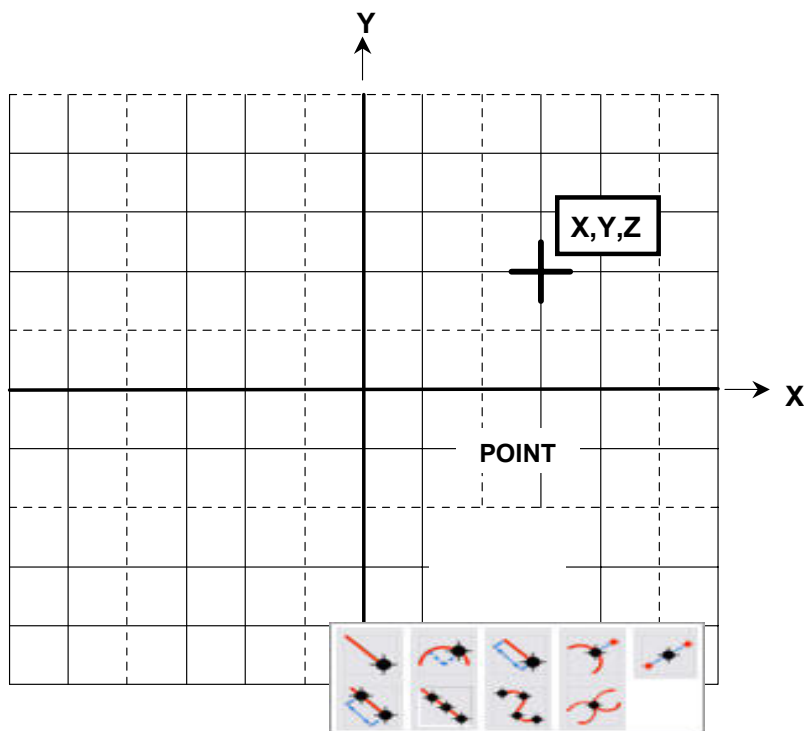
## Arcs and Circles

KeyCreator allows you to create both **Arcs** and **Circles** although there is no such entity as the “circle” listed within KeyCreator’s database. Instead, KeyCreator treats circles as full **360°** arcs. This means that a “circle” possesses an unique start and end point, which are coincident. The circle functions are generally easier to work with, since they don’t require values of angle that are otherwise needed to define an ARC. For this reason, many users often find it easier to construct entire circles, and later remove the unwanted segments using construction lines as references.

## Points

The **Point** is the most simple of entities, and represents the graph of a coordinate in 3D space. A point entity has no size, even though it may appear as a pair of intersecting lines. See also Chapter 3, Section 3.6, **User Preferences {display} "Graphics Marker Size"**.

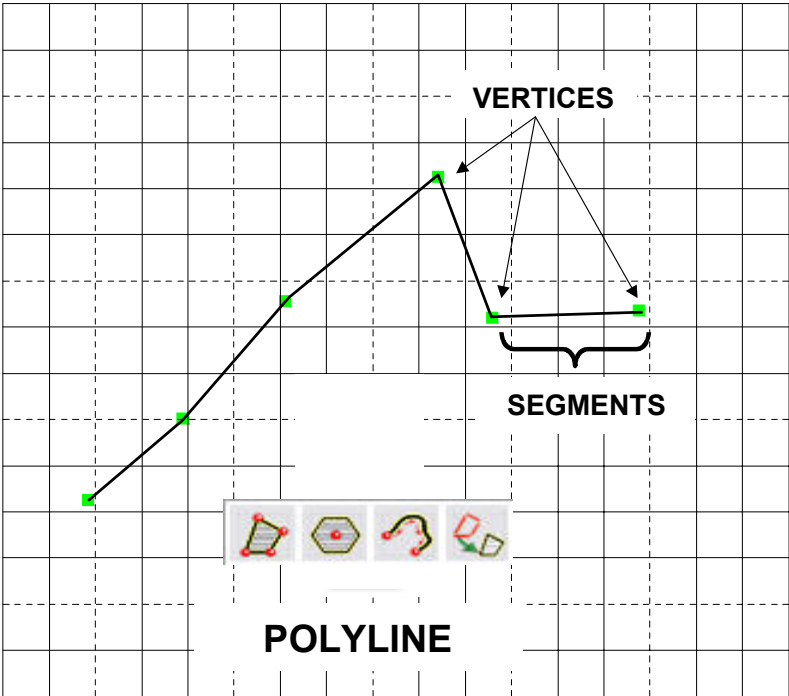
Point entities are used in a variety of applications. They are particularly useful as references when constructing other geometry. Points are often used to indicate measured distances along entities, or to denote a location in space where two entities intersect. Since points are basically a Cartesian coordinate, their data is often extracted to an x,y,z table file for use in certain CAM operations.



# Polylines

Unlike a Line, a **polyline** is not expressed as an equation. Instead, this basic entity is defined and saved as a series of points or **vertices**. This means that certain information such as *angle* and *slope* can not be directly extracted from this entity. You'll find that out quick enough if you mistakenly use a polyline instead of a line, and then later try to dimension the entity. However, between the vertices, polylines consist of linear segments which makes it possible for *some* applications to interpret the angle between consecutive segments.

When selected, a polyline will respond as a single entity. This makes it an attractive entity for use in many applications such as digitizing of 3D models or CNC machining. Since a Polyline entity is defined using the same inputs as a spline, it can be later directly converted into a spline if needed.



## Fillets and Chamfers

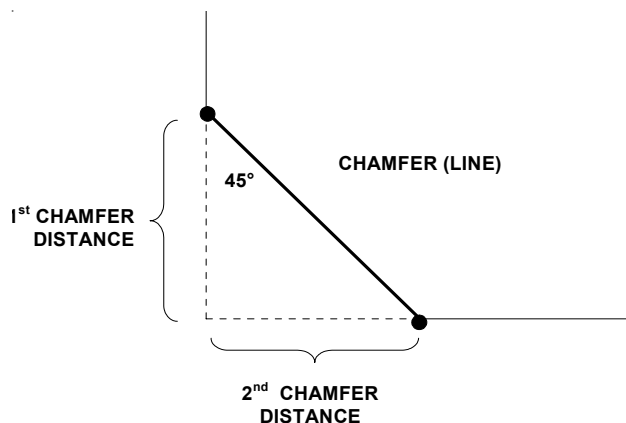
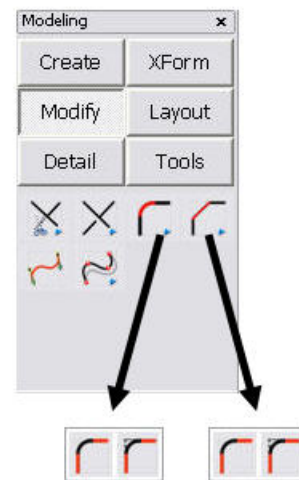
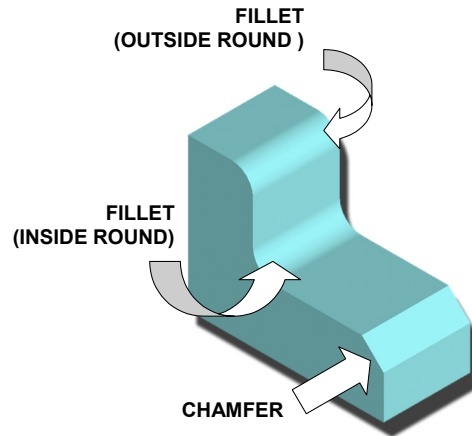
**Fillet** and **Chamfer** are terms given to two popular shop operations. Typically, these “features” are applied to either smooth or eliminate a sharp edge on a machined part as shown at right. The

**Fillet** function can be applied to form either an inside or an outside round feature. Similarly, the **Chamfer** function can be used to apply a face between a combination of entities. Both are considered compound functions and will introduce a new basic entity to the file. Both functions modify a sharp corner, and appear under the Modify menu as shown.

The Fillet and Chamfer functions only support wireframe construction.

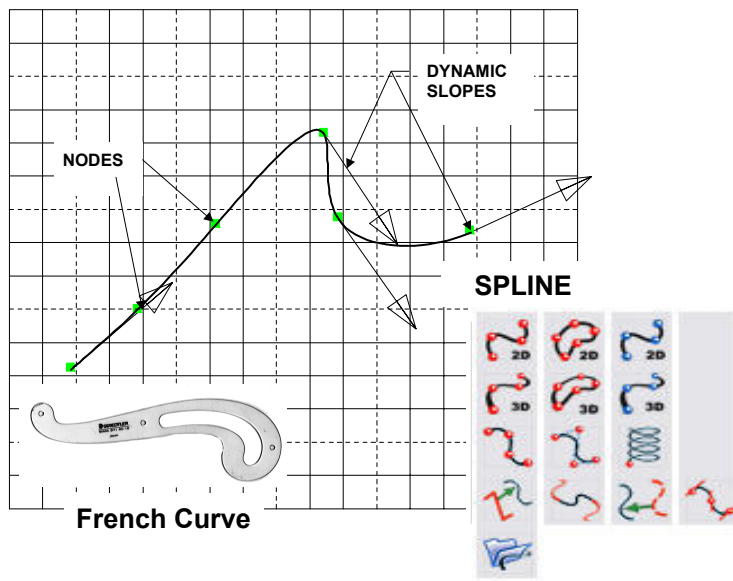
KeyCreator actually “constructs” a fillet using a basic ARC entity that is tangent to the two selected entities. Depending on the selected option, the function will also automatically trim away the excess segments beyond the point of tangency.

A Chamfer is constructed using a basic LINE entity to produce the feature’s characteristic “flat” area as shown below. For added flexibility, the chamfer function uses a distance-distance method to determine the angle of the chamfer line. For example, if both distances are of equal length, this will produce a typical  $45^\circ$  x (distance) chamfer.



## Splines

A spline entity is best described as an “electronic french curve”. Anyone who has used a french curve recognizes the instrument for its ability to create smooth shapes that can change directions many times. For example, a spline may be relatively constant for a moment. Farther along the same curve, it may then abruptly change its direction as shown below. This makes expressing spline behavior in equation form quite difficult, if not impossible. So like a Polyline, a spline is not defined in terms of curvature, but rather by a combination of control “**nodes**” and **slopes**. The slope at a given node is largely responsible for the curvature of the spline at that location. KeyCreator provides fourteen functions for creating **bi-cubic** and NURB spline curves. Both 2D (planar) and 3D splines are supported.



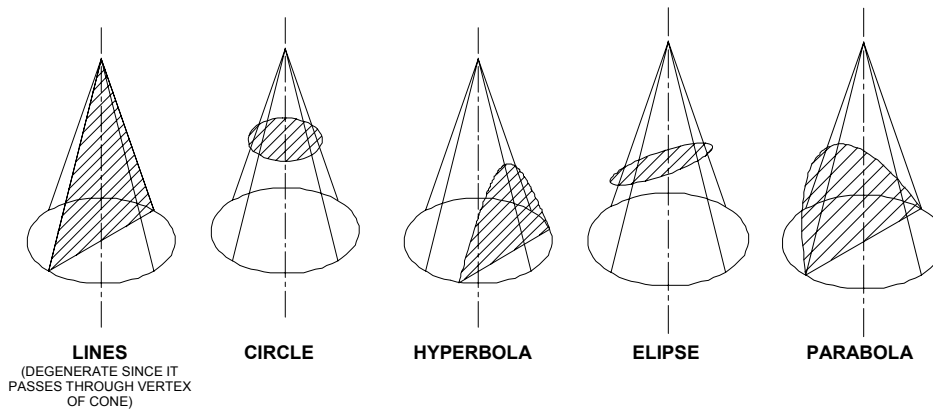
## Introduction to Conic Entities

Interestingly enough, every standard KeyCreator entity, with the exception of splines and polylines, can be expressed by the general quadratic equation:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

This general equation can be rearranged or even reduced to represent even the simplest of entities such as a line. If you're handy with algebra, you might even spot the equation of a line ( $x + y = 0$ ) nested in there somewhere. Who would ever have guessed that mathematics would play such a behind the scene role in the future of tech drafting?

The Greeks had a more practical way of looking at this equation - pictorially. They defined all of the possible shapes that could be derived from the equation in terms of the intersection of a plane and a cone as shown in Figure 4.18.



Hence, the term “conic” emerged. Take a moment and examine some of these shapes. A **degenerate conic** is formed if the plane passes directly through the vertex of the cone. A pair of intersecting lines, for example is considered a degenerate conic, since the equation “degenerates” down to nothing.

Although all of these entities are conics, KeyCreator labels only the **ellipse**, **hyperbola**, and **parabola** as conic entities in its database. This is perhaps to distinguish them in terms of their functionality. Later in Chapter 6, you'll verify these types of entities using certain tools like the **Tools>Verify>Entity** command.

## Conic Ellipse

In the previous module, we saw that a variety of shapes are theoretically derived from the intersection between a plane and a cone. One of these shapes in particular is the **ellipse**. Referring to the graphic on the previous page, we see that an ellipse begins to emerge as the plane begins tilting off-perpendicular. This begins to skew the otherwise perfect “projection” of the circle. This effect is much like shining a flashlight onto a wall from different positions in the room.

In algebraic terms, an ellipse can be expressed by rewriting the standard quadratic to:

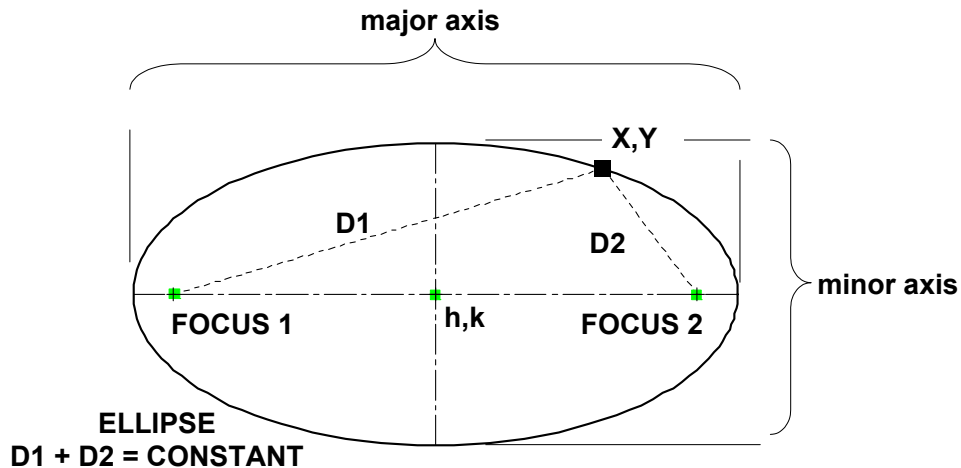
$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

## Focus Points

An ellipse is defined as the set of all points  $(x,y)$ , the sum of whose distances from two distinct points, known as the **foci**, is constant. The **foci** for a KeyCreator ellipse, may be obtained using the **Tools>Verify>Entity** utility.

In the example below, we clearly see the relationship between the focus points, and any point  $(x,y)$  that lies on the ellipse. An ellipse *approaches* the shape of a circle as the distance between the focus points closes and becomes the center. In such case,  $d_1$  and  $d_2$  combine to form the circle’s radius.

From the equation above, we can also see that there is no distance measured from the center  $(h,k)$  that is constant. This makes it impossible to resolve a fixed radius for the ellipse. However, the true center of a KeyCreator ellipse can be located using the **Center** positional option. The position options are introduced in the next section.



## Defining a KeyCreator Ellipse

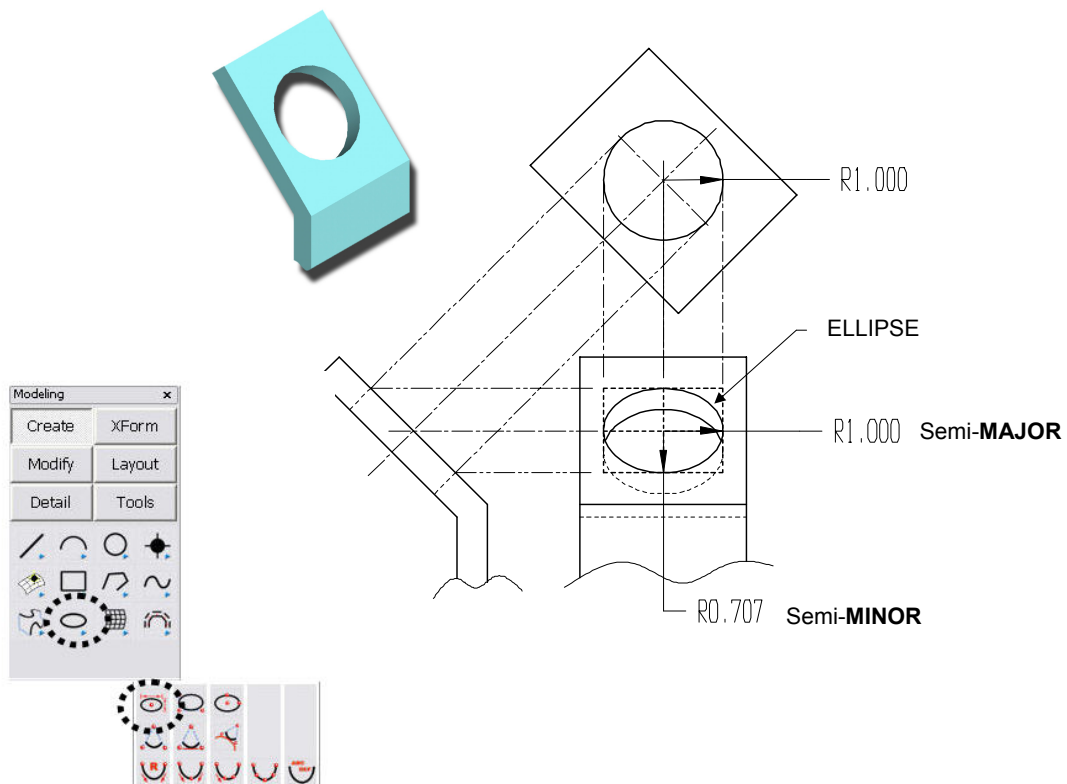
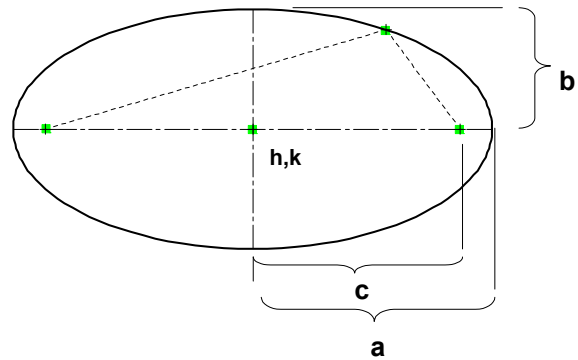
In KeyCreator, the convention used to construct an **ellipse** is to specify both the semi-major and semi-minor axis distances, shown below as distances **a** and **b** respectively.

Partial ellipses can also be created by specifying the desired *start* and *end* angle. The ellipse is positioned into the part based on its center (h,k).

The **eccentricity** of an ellipse is simply the ratio of its semi-major axis length to its semi-major focus length or:

$$e = c/a$$

The ellipse is a vital entity in 2D drawing applications, particularly when developing a multi-view drawing. An ellipse is often the result of a circular entity, such as a through-hole, that doesn't appear in true projection. For example, the tapered 45° surface contains the  $\varnothing 2.0$  hole. In the front view, this hole appears as an ellipse. Its true width is preserved as the semi-major axis dimension while the semi-minor axis becomes the projected or "skewed" axis. From right triangle geometry, the 45° surface causes the semi-minor axis to be 0.707 as shown.



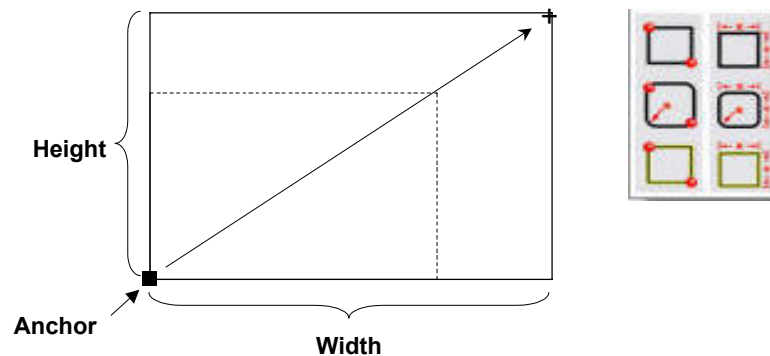
## Rectangle

A function is included that allows you to quickly create a rectangular shape. The shape can be constructed using either basic lines, polylines, or polygon entities. When the polygon option is used, the rectangle can also be filled using a solid color.

Traditionally, KeyCreator has offered two options for creating the rectangle:

**Corners** - Creates a rectangle by indicating its diagonal positions (opposite corners).

**Width/Height** - Creates a rectangle using specified values for the width and height, and an option for anchor position of the rectangle.



**Rounded Rectangle by Corners** - This options is similar to “Corners” except that filleted corners are added to the figure. Eight total entities result when complete.

**Rounded Rectangle by Width/Height** - Creates a rectangle with filleted corners with specified values for the width and height and an option for anchor position of the rectangle.

*The Rounded Corners options do not apply to polyline and polygon rectangles, only line rectangles.*

**Anchor** - When defining by diagonal, the first point becomes the anchor point. You have the option of 9 placement points to anchor the rectangle. This option creates a rectangle with specified values for the width and height and an indicated position for the anchor point of the rectangle.

|  |         |         |          |         |         |          |         |         |        |        |       |  |
|--|---------|---------|----------|---------|---------|----------|---------|---------|--------|--------|-------|--|
| Top Left                                       | Top Ctr | Top Rgt | Mid Left | Mid Ctr | Mid Rgt | Bot Left | Bot Ctr | Bot Rgt | Accept | BackUp | Esc   |  |
| Select anchor type then enter width and height |         |         |          |         |         |          |         |         | dXC    | 2      | dYC = |  |

## Polygons

The KeyCreator polygon entity is an “n” sided figure and is constructed using a polyline entity. When completed, the entire shape can be managed as a single entity. A polygon can have an unlimited number of sides.



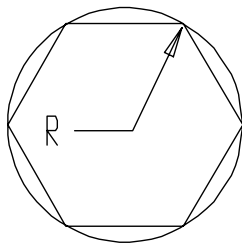
Two of the more popular options for creating a polygon include “**By Center and Radius**”, and Polygon “**Center and Length of one Side**”.

### ***Polygon Using Center and Radius Method***

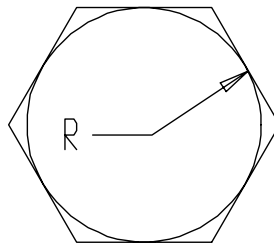
This function creates a multiple sided entity, with a rotation angle, **radius**, fill type, and **center position**. The function also provides two methods for defining the polygon: corner and flat.

The **Corner** Rad option measures the polygon’s radius from the center to the end point of the segment of the polygon. This is also known as an “circumscribed” polygon since the entire polygon fits within the specified radius. By contrast, the **Flat** Rad option measures the radius from the center perpendicularly to any one side of the polygon. This is also commonly referred to as an “inscribed” polygon since it fits entire inside of the specified radius as shown.

The Fill option may be chosen to automatically apply a color to the interior of the



**CORNER**



**FLAT**

polygon. If you choose to fill the polygon, select a color from the “Color Selection” dialog box. If you choose to outline the polygon, it is outlined using the current **color attribute**.

### ***Polygon Using Center and Length of Side Method***

This function creates a multiple sided entity with a specified number of sides, rotation angle, **side** length, fill type, and **center position**.