CAGD Kit Reference

3D point

```
typedef struct {
    GLdouble x, y, z;
} CAGD_POINT;
```

Supported types of segment

```
enum {
    CAGD_SEGMENT_UNUSED = 0,
    CAGD_SEGMENT_POINT,
    CAGD_SEGMENT_TEXT,
    CAGD_SEGMENT_POLYLINE
};
```

Events to register callback functions with

```
enum {
    CAGD_LBUTTONDOWN = 0,
    CAGD_LBUTTONUP,
    CAGD_MBUTTONDOWN,
    CAGD_MBUTTONUP,
    CAGD_RBUTTONDOWN,
    CAGD_RBUTTONUP,
    CAGD_MOUSEMOVE,
    CAGD_TIMER,
    CAGD_CLOSE,
    CAGD_LOSEFOCUS,
    CAGD_SHUTDOWN,
    CAGD_LAST
};
```

Callbacks used in event handling

```
typedef void (*CAGD_CALLBACK)(int, int, PVOID);
```

Id's of standard user interface controls

```
enum {
    CAGD_LOAD = WM_USER,
    CAGD_SAVE,
    CAGD_EXIT,
    CAGD_DRIVE,
    CAGD_DIR,
    CAGD_FILE,
    CAGD_FILENAME,
    CAGD_SENDKEYS,
    CAGD_FILEOPEN,
    CAGD_FILESAVE,
    CAGD_RESET,
    CAGD_HELP,
    CAGD_USER           /* use this one to create your own control */
};
```

General purpose functions

```
void cagdNewScr(PCSTR title, int width, int height);
```

- DESCRIPTION:
  Creates drawable CAGD window and initializes all internal data structures. It should be invoked PRIOR to any cagdcall.
- PARAMETERS:
  title text for window's caption.
  width width of window in pixels (CW_DEFAULT can be used).
  height height of window in pixels (CW_DEFAULT can be used).
- RETURN VALUE:
  None.

```
void cagdMainLoop();
```

- DESCRIPTION:
  Runs application event loop. Function exits only when user issues exit command from user interface or quit message is sent from OS or application code. You can install event callbacks that process application events.
- RETURN VALUE:
  None.

```
void cagdRedraw();
```

- DESCRIPTION:
  Forces application to redraw its main window. Usually used when CAGD segments where modified from the application to obtain immediate update.
- RETURN VALUE:
  None.

```
HWND cagdGetWinHwd();
```

- DESCRIPTION:
  Use this function to retrive valid handle of CAGD window.
- PARAMETERS:
  None.
- RETURN VALUE:
  Handle of window.

```
HMODULE cagdGetModule();
```

- DESCRIPTION:
  Use this function to retrive valid handle of application instance.
- RETURN VALUE:
  Handle of instance.

Menu and dialog functions

```
BOOL cagdAppendMenu(HMENU hMenu, PCSTR name);
```

- DESCRIPTION:
  Append user defined menu to menu bar of window (before 'Help').
- PARAMETERS:
  hMenu handle of menu to append;
  name menu's caption.
- RETURN VALUE:
  True if success;

```
BOOL cagdRemoveMenu(HMENU hMenu);
```

- DESCRIPTION:
  Remove user defined menu from menu bar of window.
- PARAMETERS:
  hMenu handle of menu to remove;
- RETURN VALUE:
  True if success;

```
WORD cagdPointMenu(HMENU hMenu, int x, int y);
```

- DESCRIPTION:
  Post user defined menu at some point of window.
- PARAMETERS:
  hMenu handle of menu to post;
  x,y where to post the menu (window's coordinates).
- RETURN VALUE:
  ID of control that belongs to pop-up menu posted and selected.

```
void cagdHelpHelpText(PCSTR text);
```

- DESCRIPTION:
  Specifies text shown in dialog window.
- PARAMETERS:
  text text shown in help dialog.
- RETURN VALUE:
  None.

```
BOOL cagdShowHelp();
```

- DESCRIPTION:
  Popups help dialog box.
- RETURN VALUE:
  True, when dialog was successfully shown.

Geometric transformations and view functions

```
void cagdRotatexy(GLdouble angle, GLdouble x, GLdouble y, GLdouble z);
```

- DESCRIPTION:
  Performs a counterclockwise rotation of the coordinate system by *angle* degrees about the vector from the origin through the point (x, y, z).
- PARAMETERS:
  angle the angle of rotation, in degrees.
  x, y, z the x, y, and z coordinates of a vector, respectively.
- RETURN VALUE:
  None.

```
void cagdTranslate(GLdouble, GLdouble, GLdouble);
```

- DESCRIPTION:
  Moves the origin of the coordinate system that moves origin to the point (x, y, z).
- PARAMETERS:
  x, y, z the x, y, and z coordinates of a vector, respectively.
- RETURN VALUE:
  None.

```
void cagdScale(GLdouble, GLdouble, GLdouble);
```

- DESCRIPTION:
  Performs a scaling of the coordinate system.
- PARAMETERS:
  x, y, z the x, y, and z scale factors respectively.
- RETURN VALUE:
  None.

```
void cagdReset();
```

- DESCRIPTION:
  Restores default coordinate system.
- RETURN VALUE:
  None.

```
WORD cagdGetView();
```

- DESCRIPTION:
  Current projection view type.
- RETURN VALUE:
  CAGD_ORTHO orthogonal projection
  CAGD_PERSP perspective projection

```
void cagdSetViewWORD which);
```

- DESCRIPTION:
  Changes projection view to which.
- PARAMETERS:
  which is one of projection types
  CAGD_ORTHO orthogonal projection
  CAGD_PERSP perspective projection
- RETURN VALUE:
  None.

```
BOOL cagdGetDepthTest();
```

- DESCRIPTION:
  Current depth cue mode.
- RETURN VALUE:
  True when enabled, false when disabled.

```
void cagdSetDepthTest(BOOL enable);
```

- DESCRIPTION:
  Changes depth cue mode to *enable*.
- PARAMETERS:
  enable the depth cue mode enable when true.
- RETURN VALUE:
  None.

```
BOOL cagdToObject(int x, int y, CAGD_POINT *where2[]);
```

- DESCRIPTION:
  Translates window coordinates at depth 0.0 and 1.0 into object coordinates.
- PARAMETERS:
  x,y the x, y window coordinates respectively.
  where the pair of object coordinates.
- RETURN VALUE:
  True when transformation can be done.

```
BOOL cagdToWindow(CAGD_POINT *where, int *x, int *y);
```

- DESCRIPTION:
  Translates object coordinates into window coordinates.
- PARAMETERS:
  where the pointer to object coordinates.
  x,y the pointers to x, y, window coordinates respectively.
- RETURN VALUE:
  True when transformation can be done.

General segment functions

```
void cagdSetColor(BYTE r, BYTE g, BYTE b);
```

- DESCRIPTION:
  Sets color context for segments created after this call.
- PARAMETERS:
  r, g, b the RGB color.
- RETURN VALUE:
  None.

```
void cagdGetColor(BYTE *r, BYTE *g, BYTE *b);
```

- DESCRIPTION:
  Gets current color context for newly created segments.
- PARAMETERS:
  r, g, b the RGB color.
- RETURN VALUE:
  None.

```
BOOL cagdSetSegmentColor(UINT id, BYTE r, BYTE g, BYTE b);
```

- DESCRIPTION:
  Changes color of the id segment.
- PARAMETERS:
  id the segment identificator.
  r, g, b the RGB color.
- RETURN VALUE:
  True when segment exists and color can be changed.

```
BOOL cagdGetSegmentColor(UINT id, BYTE *r, BYTE *g, BYTE *b);
```

- DESCRIPTION:
  Obtains color of the id segment.
- PARAMETERS:
  id the segment identificator.
  r, g, b the pointers to RGB color components.
- RETURN VALUE:
  True when segment exists.

```
BOOL cagdSegmentExist(UINT id);
```

- DESCRIPTION:
  Is a state of the id segment is visible. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id of the segment identificator.
- RETURN VALUE:
  True when segment exists.

```
BOOL cagdSetSegment(UINT id);
```

- DESCRIPTION:
  Sets a state of the id segment to not visible. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id segment identificator.
- RETURN VALUE:
  True when segment exists.

```
BOOL cagdClearSegment(UINT id);
```

- DESCRIPTION:
  Is a state of the id segment visible?
- PARAMETERS:
  id segment identificator.
- RETURN VALUE:
  True when segment exists and is visible.

```
BOOL cagdFreeSegment(UINT id);
```

- DESCRIPTION:
  Destroys segment and releases resources associated with it. The segment identifier is not valid after that and may be reused. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id the segment identificator.
- RETURN VALUE:
  True when segment exists.

```
void cagdFreeAllSegments();
```

- DESCRIPTION:
  Destroys ALL segments and releases resources associated with them. All segment identifiers is not valid after that and may be reused. You need to call **cagdRedraw** to effect the view.
- RETURN VALUE:
  None.

```
UINT cagdGetSegmentType(UINT id)
```

- DESCRIPTION:
  Obtains segment type or **CAGD_SEGMENT_UNUSED**.
- PARAMETERS:
  id the segment identificator.
- RETURN VALUE:
  CAGD_SEGMENT_POINT
  CAGD_SEGMENT_TEXT
  CAGD_SEGMENT_POLYLINE

```
UINT cagdGetSegmentLength(UINT id);
```

- DESCRIPTION:
  Obtains length of segment or zero.
- PARAMETERS:
  id the segment identificator.
- RETURN VALUE:
  Segment's length. Meaning depends on type:
  CAGD_SEGMENT_POINT 1 point
  CAGD_SEGMENT_TEXT number of characters
  CAGD_SEGMENT_POLYLINE number of vertexes

```
BOOL cagdGetSegmentLocation(UINT id, CAGD_POINT *where);
```

- DESCRIPTION:
  Location of existing segment.
  NOTE: make sure to provide enough storage.
- PARAMETERS:
  id the segment identificator.
  where the segment location.
  Meaning of the *where* depends on type of the segment.
  CAGD_SEGMENT_POINT location in object space
  CAGD_SEGMENT_TEXT location in object space
  CAGD_SEGMENT_POLYLINE locations of the vertexes
- RETURN VALUE:
  True when existing segment location.

```
void cagdPick(int x, int y);
```

- DESCRIPTION:
  Finds identification of the segments intersecting with a window location. Accuracy depends on *fuzziness* controlled from the application menu.
- PARAMETERS:
  x, y the window coordinates.
- RETURN VALUE:
  None.

```
UINT cagdFirstHit();
```

- DESCRIPTION:
  Next segment in enumeration initialized by **cagdPick**.
- RETURN VALUE:
  A segment identifier or zero when there are no more segments left to enumerate.

Point segment functions

```
UINT cagdNewPoint(CAGD_POINT *where);
```

- DESCRIPTION:
  Creates new CAGD_SEGMENT_POINT segment at *where* location and sets its state as visible. Uses current color context to set segment's color. See **cagdSetColor**. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  where the point location in object space.
- RETURN VALUE:
  A segment identifier.

```
BOOL cagdMovePoint(UINT id, const CAGD_POINT *where);
```

- DESCRIPTION:
  Modifies location of the segment. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id the segment identificator.
  where the point location in object space;
- RETURN VALUE:
  True when existing **point** segment is moved.

Text segment functions

```
UINT cagdAddText(const CAGD_POINT *where, PCSTR str);
```

- DESCRIPTION:
  Creates new CAGD_SEGMENT_TEXT segment at *where* location and sets its state as visible. Uses current color context to set segment's color. See **cagdSetColor**. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  where the text location in object space;
  str the text string;
- RETURN VALUE:
  A segment identifier or zero when invalid parameters.

```
BOOL cagdMoveText(UINT id, const CAGD_POINT *where, PCSTR str);
```

- DESCRIPTION:
  Modifies location and text string of the segment. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id the segment identificator;
  where the point location in object space;
  str the text string.
- RETURN VALUE:
  True when existing **text** segment is moved.

```
BOOL cagdGetText(UINT id, PSTR str);
```

- DESCRIPTION:
  Copies text string of the segment into buffer *str*.
- PARAMETERS:
  id the segment identificator;
  str the text string storage.
- RETURN VALUE:
  True when existing **text** segment identifier.

Polyline segment functions

```
UINT cagdAddPolyline(const CAGD_POINT *where, UINT length);
```

- DESCRIPTION:
  Creates new CAGD_SEGMENT_POLYLINE segment at *where* locations and sets its state as visible. Uses current color context to set segment's color. See **cagdSetColor**. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  where locations of vertexes locations in *iges* space;
  length the number of the vertexes.
- RETURN VALUE:
  A segment identifier or zero when invalid parameters.

```
BOOL cagdMovePolyline(UINT id, const CAGD_POINT *where, UINT length);
```

- DESCRIPTION:
  Modifies the vertexes of the segment. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id the segment identificator;
  where locations of vertexes locations in *iges* space;
  length the number of the vertexes.
- RETURN VALUE:
  True when existing **polyline** segment is moved.

```
BOOL cagdGetVertex(UINT id, UINT index, CAGD_POINT *where);
```

- DESCRIPTION:
  Copies location in object space of the vertex of the polyline segment into *where* point.
- PARAMETERS:
  id the segment identificator;
  index the vertex index;
  where the vertex location storage.
- RETURN VALUE:
  False when not existing **polyline** segment identifier or invalid parameters.

```
BOOL cagdSetVertex(UINT id, UINT index, const CAGD_POINT *where);
```

- DESCRIPTION:
  Modifies location in object space of the vertex of the segment. You need to call **cagdRedraw** to effect the view.
- PARAMETERS:
  id the segment identificator;
  index the vertex index;
  where the vertex location in object space;
- RETURN VALUE:
  False when not existing **polyline** segment identifier or invalid parameters.

```
UINT cagdGetNearVertex(UINT id, int x, int y);
```

- DESCRIPTION:
  Index of the vertex of the polyline segment nearest to the window location by Euclidian norm.
- PARAMETERS:
  id the segment identificator;
  x, y the window coordinates;
- RETURN VALUE:
  Index of the vertex.

Callback management functions

```
BOOL cagdRegisterCallback(UINT message, CAGD_CALLBACK CBF, PVOID cbData);
```

- DESCRIPTION:
  Installs callback function to handle event message. If message handler has already been installed, it is replaced.
- PARAMETERS:
  message message identificator;
  cbf the callback function address;
  cbdata the argument passed to the callback function;
- RETURN VALUE:
  False when invalid parameters.