Courses 2DCx, C3: Animation, Code Comments

In `CChildView.h` in front of `class CChildView : public CWnd`

`#define nMax 100` //nMax is the upper limit of the no. of vertices that can be stored. You can set nMax to 1000 or even higher. It is just a vaste of memory to reserve space for too much vertices. nMax does not influence the velocity of the animation as long as you do not fill its space by drawing.

`#include < math.h >` //needed for sin() and cos()

In `CChildView.h` inside `class CChildView : public CWnd`

`CPoint old_vertex;` // to store one vertex

`CPoint p[nMax];` //to store nMax vertices

`typedef struct { float x; float y; } FPoint;` //type for 2D vector graphics coordinates

`FPoint f[nMax];` //to store nMax vertices in float

`int n, t;` //n will be the counter of vertices and t the counter of timer events

`BOOL done;` //This flag indicates that the mouse drawing is over. It is initialized to `false` in `CChildView::OnCreate()` and `CChildView::OnLButtonDown()` and changed to `true` in `CChildView::OnLButtonUp()`

`CRect minmax;` //Will contain the surrounding rectangle of the drawing

`float zoom, sinus, cosinus;` //some help variables

In `int CChildView::OnCreate(LPCREATESTRUCT lpCreateStruct)`
This function is called by the operating system once at the first appearance of the client window.

`done = false;` //At first start nothing has been drawn.

`zoom = 0.995f;` //The animation starts with a down zoom of 0.5 percent.

`double arcus = 3.14159 / 180.;` //The animation rotates in steps of one degree.

`sinus = float( sin( arcus ) );` //This is the sinus of one degree.

`cosinus = float( cos( arcus ) );` //This is the cosinus of one degree.

`SetTimer( 1, 1, NULL );` //Start a timer no. 1 with the fastest possible velocity. Parmaeter no. 1 is the user no of this timer (64 parallel timers are possible). Parameter 2 requests an event every 1 msec. The operating system cannot react with 1000 events in a second but it will now produce such events as often as possible. Parameter 3 is not used.

`return 0;` //Exit with the normal return value of 0.

In `void CChildView::OnPaint()`

This function is called once at the first start of the program and later at any event that requires the redrawing of the window (f.i. by `Invalidate()`).

```
CPaintDC dc(this); //Device context of the current client area
```

```
dc.TextOut( 0, 0, "Press the left mouse button and draw something !"); //Text for users who do not know what to do.
```

In `void CChildView::OnLButtonDown(UINT nFlags, CPoint point`

The users begins to draw something.

```
done = false; //Ignore the timer events during drawing.
```

```
old_vertex = p[0] = point; //Store the first vertex.
```

```
n = t = 1; //Reset the no. of vertices and the no. of timer events.
```

```
Invalidate(); //If there was a former drawing it is swept now.
```

In `void CChildView::OnMouseMove(UINT nFlags, CPoint point`

The user moves the mouse.

```
if ( !nFlags ) return; //Ignore the mouse movement if no mouse buttons are pressed.
```

```
int dx = point.x - old_vertex.x; //horizontal distance to the former vertex
```

```
int dy = point.y - old_vertex.y; //vertical distance to the former vertex
```

```
if ( dx*dx + dy*dy < 100 ) return; //If the distance is less the 10 forget the point.
```

```
if ( n > nMax - 2 ) return; //If there are already more than 98 vertices forget the rest.
```

```
CClientDC dc( this ); //Device context of the current client area
```

```
dc.MoveTo( old_vertex ); dc.LineTo( point ); //Draw a straight line.
```

```
old_vertex = p[n++] = point; //Remember the current point as the latest one.
```

In `void CChildView::OnLButtonUp(UINT nFlags, CPoint point)`
The user has drawn something and releases the left mouse button.

`minmax.left = minmax.right = p[0].x;` //The surrounding rectangle is intitalized around the first vertex.

`minmax.top = minmax.bottom = p[0].y;` //The surrounding rectangle is intitalized around the first vertex.

`for ( int i = 1; i < n; i++ )` //Take one by one any of the following vertices.

`int x = p[i].x;` //x is a help variable to shorten the writing of `p[i].x`.

`int y = p[i].y;` //y is a help variable to shorten the writing of `p[i].y`.

`if ( x < minmax.left ) minmax.left = x;` //New vertex is left of the old surrounding rectangle.

`if ( x > minmax.right ) minmax.right = x;` //New vertex is right of the old surrounding rectangle.

`if ( y < minmax.top ) minmax.top = y;` //New vertex is above the old surrounding rectangle.

`if ( y > minmax.bottom ) minmax.bottom = y;` //New vertex is below the old surrounding rectangle.

`m.x = ( minmax.left + minmax.right )/2;` //mid of the rectangle

`m.y = ( minmax.top + minmax.bottom )/2;` //mid of the rectangle

`for ( i = 0; i < n; i++ )` //Take any vertex

`f[i].x = float(p[i].x - m.x);` //scroll it to the left upper corner of the client area and change its data type from integer to float.

`f[i].y = float(p[i].y - m.y);` //scroll it to the left upper corner of the client area and change its data type from integer to float.

`done = true;` //The animation can start now.

In `void CChildView::OnTimer(UINT nIDEvent)`

This function is started by the operating system as often as possible (under time control of parameter 2 of `SetTimer( 1, 1, NULL )` called in `CChildView::OnCreate()`

`if ( !done ) return;` //do nothing during the time between `OnLButtonDown` and `OnLButtonUp.`

`int i, ix, iy, ixmax=m.x, ixmin=m.x, width;` //local help variables

`CClientDC dc (this );` //Device context of the current client area

`for ( i = 0; i < n; i++ )` //Take any vertex

`float x = f[i].x * zoom;` //Zoom it.

`float y = f[i].y * zoom;` //Zoom it.

`f[i].x = cosinus * x - sinus * y;` //Rotate it.

`f[i].y = sinus * x + cosinus * y;` //Rotate it.

`ix = int(f[i].x) + m.x;` //Scroll it back from the left upper corner of the ClientArea to its original position.

`iy = int(f[i].y) + m.y;` //Scroll it back from the left upper corner of the ClientArea to its original position.

`if ( !i ) dc.MoveTo( ix, iy );` //If this is the firt vertex then start the polgone here.

`else dc.LineTo( ix, iy );` //else draw a straight line from the former vertex to the current one.

`if ( ix < ixmin ) ixmin = ix;` //x-position of the leftmost vertex.

`if ( ix > ixmax ) ixmax = ix;` //x-position of the rightmost vertex.

`width = ixmax - ixmin;` //current horizontal extend of the polygone

`CRect r; GetClientRect( r );` //current space on the ClientArea

`if ( width > r.right - r.left ) { MessageBeep(-1); Invalidate(); zoom = 0.95f; };` //The anmation is too big. Stop the zooming up and zoom down now in steps of 5 percent.

`if ( width < 20 ) { MessageBeep(-1); Invalidate(); zoom = 1.05f; };` //The anmation is too small. Stop the zooming down and zoom up now in steps of 5 percent.

`CString blabla;` //Instance of the class for text strings.

`blabla.Format("Timer=%d, Width=%d, Zoom=%f ", t++, width, zoom );` //Write some formatted text into the string.

`dc.TextOut( 0,20, blabla );` //During the animation this text informes the user what is going on.