


# Balloons by B.Arnold

Balloons by B.Arnold

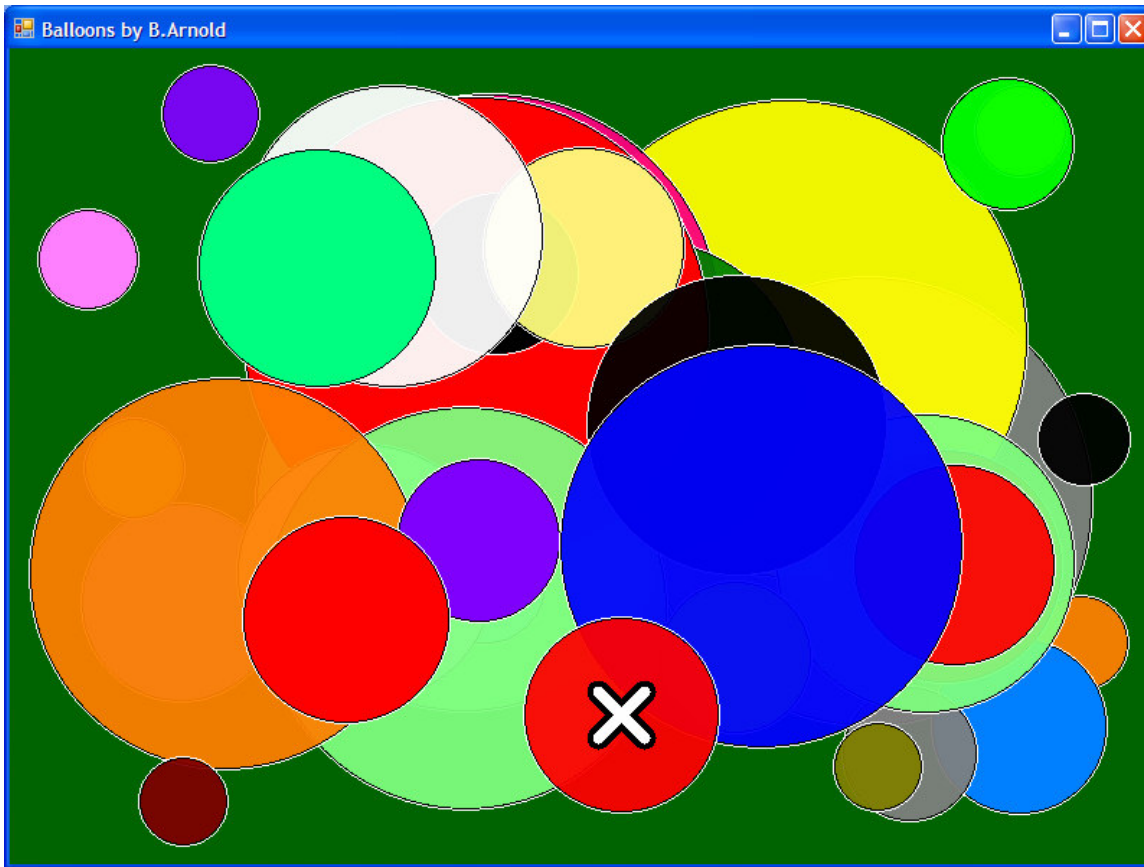
Balloons version 2.00 7/08 6/09 by B.Arnold

EASY - Single Click Smiley  
HARDER - Double Click Smiley  
HARDEST - Rubber Band Skills

Burst balloons using Space-Bar or Mouse.  
To start, click one of the icons below.



999



## Create several arrays to hold the location information.

```
#define MAXBALLOONS 35 // use 5 for debug, 35 for real.  
  
static array<int ^ > ^ pBalloonRadius = gnew array<int ^> (MAXBALLOONS);  
static array<Point ^ > ^ pBalloonCenter = gnew array<Point ^> (MAXBALLOONS);  
static array<Color ^ > ^ pBalloonColor = gnew array<Color ^> (MAXBALLOONS);
```

## Draw the balloons on the screen.

```
System::Void Form1_OnPaint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)
{
    topBalloon = -1;
    Rectangle windowRect = ClientRectangle;
    windowRect.Inflate(-10, -10); // No mans land. Ver 2.00

    if (firsttime)
    {
        Form1_InitBalloonArray(windowRect);
        ZoomStart = Point((*pBalloonCenter[MAXBALLOONS-1]).X, (*pBalloonCenter[MAXBALLOONS-1]).Y);
        firsttime = false;
    }
    Graphics ^g = e->Graphics;

    for (int i = 0; i < MAXBALLOONS; i++)
    {
        if (pBalloonCenter[i]->X < 0) continue;

        SolidBrush^ balloonBrush = gcnew SolidBrush(*pBalloonColor[i]);
        g->FillEllipse(balloonBrush,
            (*pBalloonCenter[i]).X - *pBalloonRadius[i],
            (*pBalloonCenter[i]).Y - *pBalloonRadius[i],
            2 * (*pBalloonRadius[i]),
            2 * (*pBalloonRadius[i]));
        g->DrawEllipse(Pens::Black,
            (*pBalloonCenter[i]).X - *pBalloonRadius[i],
            (*pBalloonCenter[i]).Y - *pBalloonRadius[i],
            2 * (*pBalloonRadius[i]),
            2 * (*pBalloonRadius[i]));
        g->DrawEllipse(Pens::White,
            (*pBalloonCenter[i]).X - *pBalloonRadius[i] - 1,
            (*pBalloonCenter[i]).Y - *pBalloonRadius[i] - 1,
            2 * (*pBalloonRadius[i]) + 2,
            2 * (*pBalloonRadius[i]) + 2);
        // ~balloonBrush(); or balloonBrush->Dispose();
        topBalloon = i;
    }
}
```

## Initialize the colors and locations

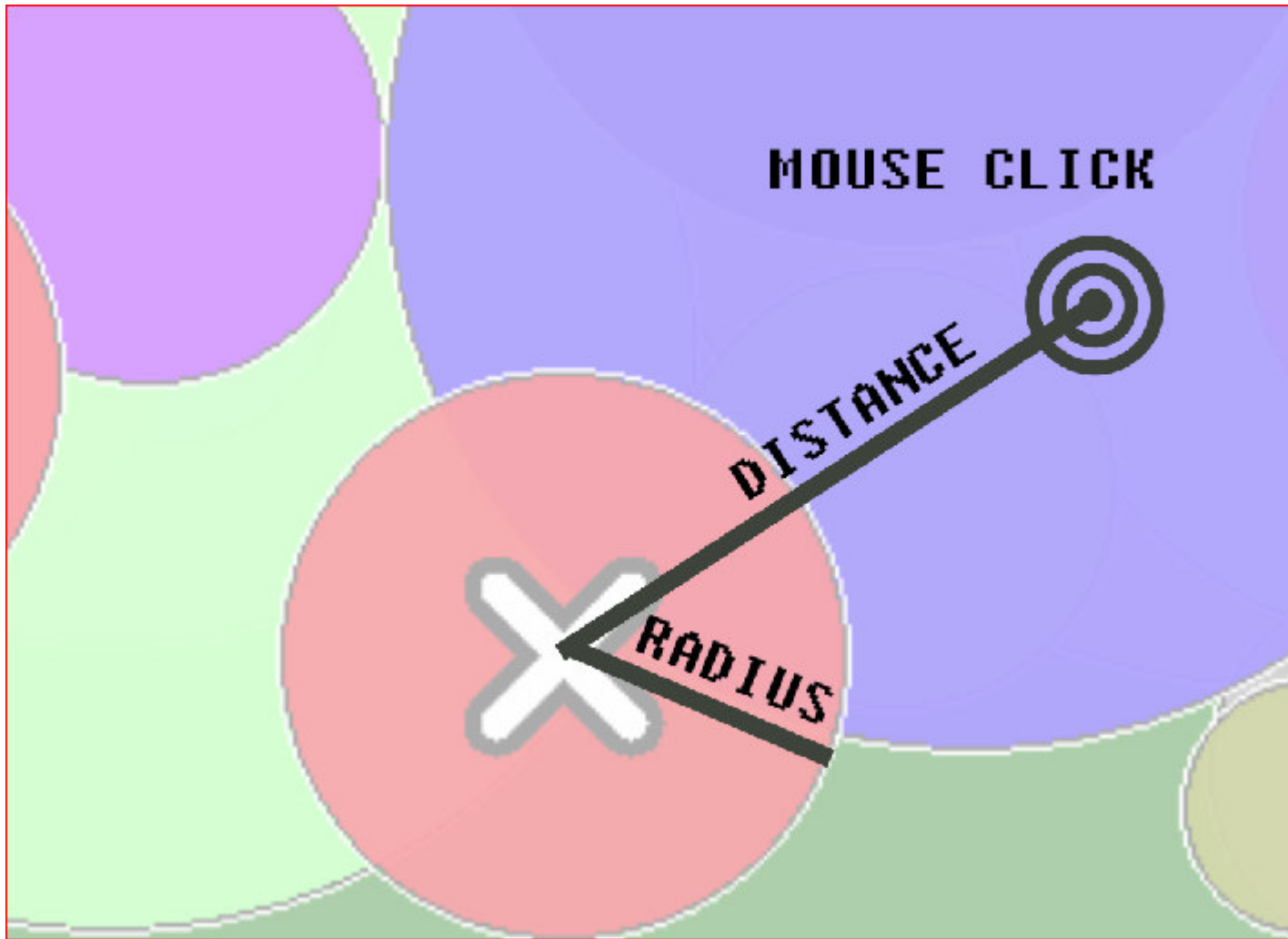
```
void Form1_InitBalloonArray(Rectangle rr)
{
    int i;
    int arg[3]; arg[0] = 0;    arg[1] = 128; arg[2] = 255;
    int alpha[3]; alpha[0] = 0xEF; alpha[1] = 0xF0; alpha[2] = 0xFF;
    for (i = 0; i < MAXBALLOONS; i++)
    {
        pBalloonCenter[i] = Form1_GetRandPoint(rr);
        pBalloonRadius[i] = Form1_GetRandRadius(rr, pBalloonCenter[i]->X, pBalloonCenter[i]->Y);
        pBalloonColor[i] = Color::FromArgb(
            alpha[pRand->Next(3)],
            arg[pRand->Next(3)],
            arg[pRand->Next(3)],
            arg[pRand->Next(3)]
        );
    }
    m_Smiley->Hide();
    m_Smiley2->Hide();
    RubberBand->Hide();
    PressEnter->Hide();
    Time->Hide();
}
```

## Randomize the center of the balloon

```
Point Form1_GetRandPoint(Rectangle rr)
{
    Point pp; Rectangle zone = rr;
    zone.Inflate(-30, -30);
    if ((rr.Right - rr.Left) < 1 || (rr.Bottom - rr.Top) < 1)
    {
        pp.X;
        pp.Y = 0; return pp;    // trap potential divide by zero
    }
    do
    {
        pp.X = rr.Left + pRand->Next(rr.Right - rr.Left);
        pp.Y = rr.Top + pRand->Next(rr.Bottom - rr.Top);
    } while (!zone.Contains(pp.X, pp.Y));    // loop if outside zone
    return pp;
}
```

## Randomize the radius of the balloon

```
int Form1_GetRandRadius(Rectangle rr, int x, int y)
{
    int limit, rad;
    if ((rr.Right - rr.Left) < 1 || (rr.Bottom - rr.Top) < 1)
    {
        rad = 0; return rad;    // trap potential divide by zero
    }
    limit = ((rr.Bottom - rr.Top) + (rr.Right - rr.Left)) / 8;
    do
    {
        rad = 30 + pRand->Next(limit);
    } while (
        !rr.Contains(x+rad,y) ||
        !rr.Contains(x-rad,y) ||
        !rr.Contains(x,y+rad) ||
        !rr.Contains(x,y-rad)    ); // loop if outside rectangle
    return rad;
}
```



## When the mouse is clicked, check the location.

```
Void Form1_ProcessClick(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e)
{
    int i, sqDistance, sqRadius;

    for (i = MAXBALLOONS - 1; i >= 0; i--)          // loop downward through array.
    {
        if ((*pBalloonCenter[i]).X < 0) continue; // skip balloons already popped.

        // Calculate squared distance from mouse to center of circle
        sqDistance =
            (e->Location.X - (*pBalloonCenter[i]).X) * // Note: we're at target balloon.
            (e->Location.X - (*pBalloonCenter[i]).X) +
            (e->Location.Y - (*pBalloonCenter[i]).Y) *
            (e->Location.Y - (*pBalloonCenter[i]).Y);

        // Calculate squared circle radius
        sqRadius =
            *pBalloonRadius[i] *
            *pBalloonRadius[i] ;

        // If the mouse click is within the balloon, burst it.
        if (sqDistance < sqRadius)
        {
            mouseflag = true;
            Form1_BurstBalloon(i);
            if (!firstballoonpopped)
            {
                pTimeStart = System::DateTime::Now; // Start timer. 1.35
                firstballoonpopped = true;
            }
            break;
        }
        else
            break; // Added for version 1.35
    }
}
```