

Re: Mandelbrot set with bitmap texture applied

Source: <http://sci.tech-archive.net/Archive/sci.fractals/2006-01/msg00013.html>

- *From:* "dave" <dave@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 9 Jan 2006 10:33:41 -0000
-

Hi Chris,

Hey – I like it.

I could never work out how to get the rotation repeat value correctly !!

bye
Dave

"eNZedBlue" <chris@xxxxxxxxxxxx> wrote in message
<news:1136769254.827062.155460@xx>

>
> adam majewski wrote:
>
>> Interesting.
>>
>> Could you give:
>> – more detailed decription of algorithm
>> – program
>> – sorce code.
>>
>> Adam Majewski
>> republika.pl/fraktal
>
> Hi Adam,
>
> Apologies for not replying sooner. The algorithm used to produce the
> texture co-ordinates is identical to normal iteration of the Mandelbrot
> fractal, except that I store the last value of z before it escapes the
> boundary circle, and then convert it to a polar co-ordinate to use as a
> texture u/v (with some clamping and scaling applied for aesthetic
> reasons). I have made a few changes since my last post in order to
> calculate the number of times that the texture has repeated moving
> clockwise around the set before you get to the point. This allows more
> complex texturing, such as adding text or textures that repeat over a
> larger periodicity to the image:
>
> <http://www.enzedblue.com/Fractals/Images/plato1.jpg>

Re: Mandelbrot set with bitmap texture applied

> <http://www.enedblue.com/Fractals/Images/plato2.jpg>

> <http://www.enedblue.com/Fractals/Images/plato3.jpg>

>

> To work out the nTextureRepeat value, I use a geometric interpretation
> of complex-number multiplication – convert to polar co-ordinates and
> perform the multiplication by adding the angles and multiplying the
> distance from the origin. This means that if I have for example a
> complex number with a polar co-ordinate angle of 190, then after
> squaring it I record the angle as 380, instead of 20, (which is what
> you would get if you used normal complex number arithmetic for the
> iteration and then converted to polar co-ordinates at the end.)

>

> Here is the source code to my point-testing function:

>

> #define IN

> #define OUT

> #define INOUT

>

>

//-----

> // Name: TestMandelbrotPoint

> // Description: Tests a point in the Mandelbrot Set and returns the

> number

> // of iterations before the point escapes, and set of a texture

> // co-ordinates for the point

> // Parameters:

> //

> // IN: dA, dB – point on the complex plane being tested

> // nMaxIterations – maximum number of times to iterate the function

> // bUsePolarCoords – iterate the function using polar coordinates

> // for the complex numbers

> // bJulia – set to true to test a julia set point

> // dJuliaA, dJuliaB – the value of C for the Julia set (ignored

> // if bJulia is false

> //

> // OUT: dTextureU, dTextureV – texture co-ordinates that can be used

> // to texture the set with a tiling texture, in the range of

> 0.0–1.0.

> // nTextureRepeat – the number of times that the texture-mapping

> // repeats moving anti-clockwise around the set before getting

> // to this point (can be thought of as the whole number component

> // of TextureU)

> // bInSet – set to true if the point is in the set (in which case

> // the texture coordinates are invalid and the point should be

> // rendered black)

> // Return: The number of iterations before the point escapes

>

//-----

> int TestMandelbrotPoint(double dA IN, double dB IN,

Re: Mandelbrot set with bitmap texture applied

Re: Mandelbrot set with bitmap texture applied

```
> int nMaxIterations IN,  
> bool bUsePolarCoords IN,  
> bool bJulia IN,  
> double dJuliaA IN, double dJuliaB IN,  
> double& dLastValidA OUT, double& dLastValidB OUT,  
> double& dTextureU OUT, double& dTextureV OUT,  
> int& nTextureRepeat OUT,  
> bool& bInSet OUT)  
> {  
> double dX = dA;  
> double dY = dB;  
> int nIterations = 0;  
> double dLength = 0;  
> if(bJulia)  
> {  
> dA = dJuliaA;  
> dB = dJuliaB;  
> }  
>  
> dLastValidA = dX;  
> dLastValidB = dY;  
> nTextureRepeat = 0;  
>  
> if(bUsePolarCoords)  
> {  
> double dLastAngle = atan2(dY, dX);  
>  
> for(int nK = 0; nK < nMaxIterations; nK++)  
> {  
> double dU = dX * dX;  
> double dV = dY * dY;  
> double dW = 2 * dX * dY;  
> dX = dU - dV + dA;  
> dY = dW + dB;  
>  
> // ApplyDistortions(&dX, &dY);  
>  
> dLength = dU + dV;  
> if(dLength > 16)  
> break;  
>  
> dLastValidA = dX;  
> dLastValidB = dY;  
>  
> // to multiply two complex numbers together  
> // in polar co-ordinate form you add the angles  
> // and multiply the distance from the origin.  
>  
> // this bit of code keeps track of the number  
> // of revolutions that the point under iteration  
> // has travelled around the origin by repeated
```

Re: Mandelbrot set with bitmap texture applied

```
> // squaring.
>
> // The algorithm separates the whole number component
> // of the number of revolutions and stores it
> // as a separate integer value (nTextureRepeat)
>
> dLastAngle *= 2;
> double dAngle = atan2(dY, dX);
> double dDiff = dAngle - dLastAngle;
> int nRevs = (int)(dDiff / TWO_PI);
> dDiff -= (double)nRevs * TWO_PI;
> if(dDiff < -PI)
> dDiff += TWO_PI;
> if(dDiff > PI)
> dDiff -= TWO_PI;
> dLastAngle += dDiff;
>
> nTextureRepeat <<= 1;
> while(dLastAngle > PI)
> {
> dLastAngle -= TWO_PI;
> nTextureRepeat += 1;
> }
> while(dLastAngle < -PI)
> {
> dLastAngle += TWO_PI;
> nTextureRepeat -= 1;
> }
>
> nIterations++;
> }
> }
> else
> {
> // Simple Mandelbrot iteration:
> for(int nK = 0; nK < nMaxIterations; nK++)
> {
> double dU = dX * dX;
> double dV = dY * dY;
> double dW = 2 * dX * dY;
> dX = dU - dV + dA;
> dY = dW + dB;
>
> // ApplyDistortions(&dX, &dY);
>
> dLength = dU + dV;
> if(dLength > 16)
> break;
>
> dLastValidA = dX;
> dLastValidB = dY;
```

Re: Mandelbrot set with bitmap texture applied

```
>
> nIterations++;
> }
> }
>
> // The length is distorted to make the texture V
> // co-ordinate match up between iterations bands of
> // the rendered fractal. There is no real
> // mathematical meaning to this, it just looks nicer
> dLength = (dLength - 16) / 240;
> dLastValidA -= (dLength * dA);
> dLastValidB -= (dLength * dB);
>
> // compute the texture co-ordinates:
> dTextureV = ((sqrt((dLastValidA * dLastValidA) + (dLastValidB *
> dLastValidB))) / 16.0);
> dTextureV = ((dTextureV * 4) - 1) / 3;
> if(dTextureV < 0)
> dTextureV = 0;
> dTextureV = sqrt(dTextureV);
>
> dTextureU = 0.0;
> dTextureU = atan2(dLastValidA, dLastValidB) / PI;
> dTextureU += 0.5;
> if(dTextureU < -1.0)
> dTextureU += 2.0;
> if(dTextureU >= 1.0)
> dTextureU -= 2.0;
>
> // indicate whether the point was in the set
> if(nIterations == nMaxIterations)
> bInSet = true;
> else
> bInSet = false;
>
> return nIterations;
> }
>
> Hope this helps.
>
> Cheers,
> Chris
>
```

• *References:*

- ◆ *Re: Mandelbrot set with bitmap texture applied*

Re: Mandelbrot set with bitmap texture applied

◇ *From:* eNZedBlue

- Prev by Date: *Call for Papers: CGVR'06 (part of WORLDCOMP'06)*
- Next by Date: *Re: abrasives modelled by fractal technique*
- Previous by thread: *Re: Mandelbrot set with bitmap texture applied*
- Next by thread: *draw next iteration by copying last iteration*
- Index(es):
 - ◆ *Date*
 - ◆ *Thread*