Fractal X plans

1/16/18

See Direct3D Win32 Game1 in code\directx code

Made simple vertex and indices and loaded

Next try to set matrices and try rotating -ok

Then try making a texture

1/20/18

It rotates but it seems like some sections are transparent – try to figure out

1. Add a menu with something to stop the timer
2. Add something to set a fixed phi and theta
3. Figure out if the vertexes are drawing correctly
4. Add lights and menu to set
5. Add texture

In looks like the current project is Win32 see if we can transfer the code to a MFC project. Call the new project FractalX

Made project FractalX in git\_working

Find out what property view and others are about

need to import direct x libraries

Seem to have libraries but still a conflict with *D3D11CreateDevice and precompiled headers*

After that:

1. Connect renderer to view and get it to display
2. Clean up exceptions, etc.

d3d11.lib

dxguid.lib

1/27/18

Got it to display

Commit -ok

Next clean up. Examine and see how it can be refactored. Maybe replace exceptions or throw specific one and catch.

2/3/18

Make exceptions with descriptions

Cleaned up.

Moved vertex creation to a factory and set before initialization.

Then

It rotates but it seems like some sections are transparent – try to figure out

1. Add a menu with something to stop the timer
2. Add something to set a fixed phi and theta
3. Figure out if the vertexes are drawing correctly
4. Add lights and menu to set
5. Add texture

2/10/18

Change namespace to DXF

Make new folder DFX and put all DXF files there –ok

Change implementation of Renderer to use new settings – problem is I can’t find MathHelper::ToRadians() may have to implement manually –ok

2/13/18

Implemented different rotation modes –ok

commit -ok

Fix dialog to do input validation -ok

Add z angle -ok

Add texture

2/17/18

Look at how to implement texture

we need to make the colors first in the form of

*std*::*vector*<*uint32\_t*> GradientColors;

*uint32\_t* ColorrefToXmcolor(*COLORREF* cr, *uint32\_t* alpha)

{

// XMCOLOR is structured 0xaarrggbb

// COLORREF is structured 0x00bbggrr

return (alpha << 24) | (*GetRValue*(cr) << 16) | (*GetGValue*(cr) << 8) | *GetBValue*(cr);

// An alternative method

// return XMCOLOR(GetRValue(cr) / 255.0f, GetGValue(cr) / 255.0f, GetBValue(cr) / 255.0f, alpha / 255.0f);

}

violet -> blue -> Cyan -> Green -> Yellow -> Red -> White

0 -> 0.167 -> 0.333 -> 0.5 -> 0.666 -> 0.833 -> 1

1. Make function that spreads colors taking RGB and index ratio (0 to 1) and spreads colors between those values to make palette
2. Check Bulb and see if we can use texture

2/20/18

1. Put Dx code in DLL
2. link to it
3. Create a unit test project

2/24/18

Need to used an MFC DLL because windows is included in DirectX (kind of)

So use project Dxf. Delete project DxHelper when done

Main problem remaining is how to use shared\_ptr in export.

Seems like it’s not solvable. Try making the vertex calculation internal in the DLL so it doesn’t need to be passed back and forth.

2/27/18

Updated FractalX3 from work code

Commit

Changed format to VertexPositionNormalTexture and added texture and DxfColorFactory

Next try making a static library and see if we can export a vector

3/3/18

Make new project DxSupport as static link library

Commit

Next make a test project

Commit

Added DxTests

Remove changes to old FractalX sandbox -ok

Add more tests

3/6/18

1. Move menu up for rotation dialog -ok
2. Have it start on rotation x? -ok
3. Make test for enums -ok
4. Move texture to external method – ok
5. Add buffer creation to external methods –ok
6. Commit
7. Make vertex buffer a cube –ok
8. commit
9. Allow selection of different textures – need to finish dialog
10. Research fractal code
11. Consider error log

3/17/18

Added texture dialog

Cleaned up menus

Next think about how to implement fractal vertexes

Look at solution Fracture, project Bulb

Algorithm Ray Tracing and Stretch Ray Tracing

Plain ray tracing is RayTracingBulbCalculator

Stretch ray tracing is – DistanceRayTraceBulbCalculator

Study plain first

Also BulbDistanceCalculator

RayTracingBulbCalculator

double distance = RayMarch(start, direction, rayMarchParams, distanceEstimateParams);

public double RayMarch(Vertex start, Vec3D direction, RayMarchParameters rayMarchParams, DistanceEstimatorParameters distanceEstimateParams)

{

double totalDistance = 0.0;

int steps;

double lastDistance = *Double*.*MaxValue*;

for (steps = 0; steps < distanceEstimateParams.MaxRaySteps; steps++)

{

\_p = VectorMethods.Add(totalDistance \* direction, start);

double distance = EstimateDistance(\_p);

totalDistance += distance / distanceEstimateParams.StepDivisor; // note change

if (distance < distanceEstimateParams.MinRayDistance || distance > lastDistance)

break;

lastDistance = distance;

}

return 1.0 - ((double)steps) / distanceEstimateParams.MaxRaySteps;

}

public double EstimateDistance(Vec3D pos)

{

Vec3D z = new Vec3D(pos);

double dr = 1.0;

double r = 0.0;

for (int i = 0; i < Iterations; ++i)

{

r = z.Length();

if (r > Bailout)

break;

if (*Double*.*IsNaN*(r))

break;

BulbCalculator.CalculateNextCycle(ref z, ref r, ref dr);

z += pos;

}

return 0.5 \* *Math*.*Log*(r) \* r / dr;

}

public void CalculateNextCycle(ref Vec3D z, ref double r, ref double dr)

{

// convert to polar coordinates

double theta = *Math*.*Acos*(z.Z / r);

double phi = *Math*.*Atan2*(z.Y, z.X);

dr = *Math*.*Pow*(r, \_power - 1.0) \* \_power \* dr + \_C;

// scale and rotate the point

double zr = *Math*.*Pow*(r, \_power);

theta = theta \* \_power;

phi = phi \* \_power;

// convert back to Cartesian coordinates

z = zr \* new Vec3D(*Math*.*Sin*(theta) \* *Math*.*Cos*(phi), *Math*.*Sin*(phi) \* *Math*.*Sin*(theta), *Math*.*Cos*(theta));

}

// calculate normal

// vec3 n = normalize(vec3(DE(pos + xDir) - DE(pos - xDir),

// DE(pos + yDir) - DE(pos - yDir),

// DE(pos + zDir) - DE(pos - zDir)));

Vec3D CalculateNormal(Vec3D pos, double normalDelta)

{

double plusX = EstimateDistance(VectorMethods.AddX(pos, normalDelta));

double minusX = EstimateDistance(VectorMethods.AddX(pos, -1.0 \* normalDelta));

double plusY = EstimateDistance(VectorMethods.AddY(pos, normalDelta));

double minusY = EstimateDistance(VectorMethods.AddY(pos, -1.0 \* normalDelta));

double plusZ = EstimateDistance(VectorMethods.AddZ(pos, normalDelta));

double minusZ = EstimateDistance(VectorMethods.AddZ(pos, -1.0 \* normalDelta));

var norm = new Vec3D(plusX - minusX, plusY - minusY, plusZ - minusZ);

return norm.Normalize();

}

Problem – how to scan a sphere and convert to vertices and indices

Vertices to approximate a sphere

Start with 6 vertices

0 (0,1,0),

1 (1,0,0),

2 (0,0,-1),

3 (-1,0,0),

4 (0,0,1),

5 (0,-1,0)

Make counter clockwise

Make 8 triangles with indices: 0,1,2, 0,2,3, 0,3,4, 0,4,1, 5,2,1, 5,3,2, 5,4,3, 5,1,4

Then for n cycles divide each triangle into 4. Divide each line in half and add a new vertex between the two points. Create a new triangle for each old vertex with 2 new points and a new triangle with all 3 new vertices. Need to search if a new vertex was recently created.

Total number of triangles T = 2 x 4^(n+1)

Total number of vertices V = T/2 + 2 or 2 + 4^(n+1)

3/27/18

Next

1. Make unit tests for TriangleData GenerateCrudeTriangles(int depth);-ok
2. Normalize vertices -ok
3. Make conversion function for models -ok
4. Test models

3/31/18

n 0 1 2 3 4 5

V 6 18 66 258 1026 4098

T 8 32 128 512 2048 8192

It works!

Commit

4/3/18

Committed changes including SphereApproximator

Next plan fractal calculation

Started on Ray Tracing method with distance calculation.

4/14/18

3 problems

1. Have to scale world. Currently using 0.1 for each dimension- temp fix
2. The object is still a sphere – the return vector is always the same distance away?
3. The color is always white

EstimateDistance is returning a greater value after the initial and it always seems to be the same? Fracture returns smaller values.

VS not working. Try repair. Then see

<https://social.msdn.microsoft.com/Forums/en-US/aaaa3e11-7835-47b8-9c4c-15171dce05b1/visual-studio-not-starting-error-no-inprocserver32-registered-for-package-visual-studio-logging?forum=vsx>

4/21/18

Have the program building again use FractalX2 sandbox

Don’t know what the problem was, seems like solution was broke.

Not sure why we’re not getting a fractal. The distance does not decrease but it does in fracture. Math looks the same but something is not.

Compare distance calculation carefully to fracture. Maybe a function behaves differently.

4/24/18

Got it working in FractalX2 sandbox. The problem was the direction was the same as the starting point.

Committed

Next add a dialog to set the parameters, such as view distance, both perspective and distance from the camera to start the calculation. Then other tracing parameters. Also the number of points or n depth.

4/28/18

Remove Properties, Output and File view –ok

Consider adding a ribbon bar

<https://docs.microsoft.com/en-us/cpp/mfc/walkthrough-updating-the-mfc-scribble-application-part-1>

c:\users\steve\source\repos

Working on ribbon bar.

m\_wndRibbonBar commented out in mainframe because resources not loading right. Problem with IDR\_RIBBON copied from c:\users\steve\source\repos\TempRibbonBar

I think the problem is with the IDB\_WRITESMALL IDB\_WRITELARGE which weren’t copied correctly. Give up for now.

5/1/18

Tried to set perspective but it doesn’t work

5/5/18

Fixed perspective and added View dialog

Works good n= 7 is slow > 10 min

Next save to hard disk. OK

Next add tracing params and consider not making whole sphere

5/8/18

Created CModelSheet and ModelData

Need to set a DXF Render SetVertexData to take a ModelData and also need it to return one. -ok

Then add handler in view to call sheet to fill in data and a method to call renderer to recalculate

Need to work out how to determine if we can call the method to recalculate. Some kind of progress dialog

5/12/18

Hook up property sheet in view

Add method to recalculate model -ok

Progress

Display number of vertexes in property page

Try calculating single octant of sphere

5/15/18

Added progress. The last stage of GenerateCrudeTriangles()

is a little slow. When n is big the last cycle is long. Leave for now and hook up octants.

Next display in Param dialog the number of vertexes for n

Total number of triangles T = 2 x 4^(n+1)

Total number of vertices V = T/2 + 2 or 2 + 4^(n+1)

This can’t be right, check

1. Display number of vertexes and triangles
2. Add cancel button
3. Add partial sphere calculation

IDC\_VERTEX\_NUMBER\_EDIT

IDC\_TRIANGLE\_NUMBER\_EDIT

5/19/18

Got the display of the vertices and triangles.

Cancel is more complicated. Will need to do the calculations in a worker thread and check an event or modern equivalent set from the progress dialog.

Need to research. Don’t do same thing as before use more modern approach.

Old method had the progress dialog send a message to the parent that set an event. The worker thread method checked the event.

5/22/18

Examples for threading

bool failed = false;

CDialogTaskMonitor ProgressDialog(*AfxGetMainWnd*());

std::*thread* exportThread( [&](){ExportToCsvThreadMethod( SaveFileDlg.*GetPathName*(), pMatrix, &ProgressDialog, failed, isGCxGC, isSaturn);} );

// Show the progress dialog (until the task has completed)

ProgressDialog.SetTitle( *\_T*("Exporting Mass Calibration Matrix") );

ProgressDialog.*DoModal*();

exportThread.*join*();

std::mutex m\_mutex;

std::lock\_guard<std::mutex> guard(m\_mutex);

#include <condition\_variable>

m\_condVar.notify\_one();

    std::unique\_lock<std::mutex> mlock(m\_mutex);

    // Start waiting for the Condition Variable to get signaled

    // Wait() will internally release the lock and make the thread to block

    // As soon as condition variable get signaled, resume the thread and

    // again acquire the lock. Then check if condition is met or not

    // If condition is met then continue else again go in wait.

    m\_condVar.wait(mlock, std::bind(&Application::isDataLoaded, this));

[std::condition\_variable](http://en.cppreference.com/w/cpp/thread/condition_variable) cv;

[std::mutex](http://en.cppreference.com/w/cpp/thread/mutex) cv\_m;

int i;

void waits(int idx)

{

[std::unique\_lock](http://en.cppreference.com/w/cpp/thread/unique_lock)<[std::mutex](http://en.cppreference.com/w/cpp/thread/mutex)> lk(cv\_m);

    if(cv.wait\_for(lk, idx\*100ms, []{return i == 1;}))

[std::cerr](http://en.cppreference.com/w/cpp/io/cerr) << "Thread " << idx << " finished waiting. i == " << i << '**\n**';

    else

[std::cerr](http://en.cppreference.com/w/cpp/io/cerr) << "Thread " << idx << " timed out. i == " << i << '**\n**';

Plans for progress

1. Create progress dialog as member of view -ok
2. Make messages for progress and closing progress -ok
3. Add handlers to view for those -ok
4. Call methods on progress dialog or close from the handlers -ok
5. Close handler should update the renderer
6. Add method to view that calculates the model in worker thread
7. Thread method should take a method for updating progress – it should call postmessage and a condition variable to wait on for cancel
8. Thread method should also take a method for done
9. How to pass model back from worker? Used to return data as a pointer in message but there must be a better way.
10. Use a shared variable that is passed to the thread and mutex
11. promise <https://cpppatterns.com/patterns/pass-values-between-threads.html>
12. task <https://cpppatterns.com/patterns/execute-task-asynchronously.html>

I think you can put a shared or unique ptr in a future. Store the future in the view while the thread is calculating then when the complete message is done retrieve from the future

5/25/18

Have it set up to do calculation with future but OnInitialUpdate() tries to initialize renderer without any vertices which fails.

Have it start without vertexes and then add cancel.

5/27/18

Got it working with registered messages and std::thread

Commit -ok

Pimple and commit -ok

Add cancel – couldn’t get it to work with condition\_variable or atomic??????

6/1/18

Got it to work with an atomic bool.

Commit

Start on Seed portions – work out math.

6/4/18

Vertices and Triangle counts for number of starting triangle

Starting triangles == 8

T = 2 x 4^(n+1)

V = T/2 + 2 or 2 + 4^(n+1)

Starting triangle = 1

T= 4^n

levels =

l = 2;

for x = 1 to n: l =+ l-1

V = Sum(0 to levels)

n l V

0 2 3

1 3 6

2 5 15

3 9 45

4 17 153

Starting triangle = 2

T = 2\* T= 4^n

V = 2\*V(1) – l

Starting triangle = 4

T= 4^n

V = 4\*V(1) – 3l

6/9/18

1. Add seed triangles combo -ok
2. Calculate triangles and vertices -ok
3. Implement in calculation –ok
4. Commit –ok
5. Update back up drive-ok

Implemented SeedTriangles so partial sphere of vertices could be calculated for model.

Put hard drive in ceiling

6/23/18

1. Allow setting of TraceParams in SetModel
2. Compare BulbDistanceEstimator and FractionalDistanceEstimator in Fracture and figure out what we’re missing
3. Add C
4. Allow setting colors for texture. Consider how to enter and store textures

struct *TraceParams*

{

float *Distance* = 10.0f;

*DirectX*::*SimpleMath*::*Vector3* *Origin* = *DirectX*::*SimpleMath*::*Vector3*(0.0f, 0.0f, 0.0f);

// Distance estimate params

int *MaxRaySteps* = 1000;

double *MinRayDistance* = 0.0001;

double *StepDivisor* = 10.0;

double *Bailout* = 2.0;

int *Iterations* = 256;

float *NormalDelta* = 0.01f;

};

For 1:

Add a TraceParams to the view -ok

Pass to CreateBulb() in VertexFactory.cpp -ok

Add property page to CModelSheet -ok

Make new page IDD\_TRACE\_PARAMS\_PAGE CTraceParamsPage -ok

Add fields

Just added max ray steps so far – commit

Added CTraceParamsPage and TraceParams were added to view.

6/26/18

Commit

Added the rest of the trace params to the trace params page.

Next see 2 and 3 above

Then experiment

Consider saving flat image to bmp

Consider saving image to file with vertex data

Consider speeding up. Maybe storing seed triangles

Consider adding progress for calculating seed triangles

6/30/18

There are two calculations we can add 1)fractional and 2) Stretch.

1. Add a bool for fractional to params. This adds distances/minDistance to the number of steps when calculating color index. See fracture, FractionalDistanceEstimator::RayMarch()
   1. Add bool to params
   2. Implement calculation
   3. Connect
2. Implement stretch method as an option. In fracture this uses RayMarchDistance() it adds an extra step when the distances is stretched before calculating the color index
   1. Study
   2. Implement method to use min max
   3. Add params
   4. Connect
   5. Figure out how to calculate min/max
   6. Implement
3. Add constant C to input params and connect
4. Study how to implement colors/palettes
5. Save image
6. Save to file
7. Save triangles to file to speed up
8. Add progress for calculating seed triangles
9. Maybe make a common method for creating progress methods or think about a better approach

7/3/18

Did 1 and committed

7/5/18

Hooked up Stretch Parameters

Next implement

Two big differences

1. distance is calculated directly and not from the number of steps
2. There can be an earlier estimation calculation which changes progress too

Now we have to return the traceParams with the vertex data because the min/max could change. This is possible because the data is returned in FractalXView::OnCalcFinished()

So when returning to this:

1. put StretchDistanceParams in DxVertexData, set it in the RayTrace() and RayTraceStretch(). Fill in all intermediate steps as needed. -ok
2. Add a method to RayTraceStretch() to calculate min max as needed-ok
3. Commit -ok
4. Update progress in RayTraceStretch().-ok
5. Commit –ok

7/10/18

1. Added constant C param -ok
2. Added Power param -ok
3. Make new page for Fractal params –ok
4. Commit – Added FractalParamsPage -ok
5. Update view when calculation finished in case rotation is fixed

7/14/18

Made it redraw when rotation is fixed and a new model is calculated. Commit

Looking at how colors are saved in other programs

Fractals stores all colors and the pins

Chaos – can’t really tell. Looks like it serializes pins. The pins can include 3 pairs of colors. It include an enum for the spread type and double for the curve

Fracture has alpha. Has pins and each pin has a color and an index (double between 0 and 1)

Bulb has alpha

Plans for colors

1. Make a ColorPin with 3 colors for up to 3 stripes, an enum for strips, and an index. Color should be compatible with texture and include alpha. -ok
2. Make a palette class that stores a vector of pins. Add a serialization method to this. Think about how to serialize in and external to doc. -ok
3. Make method to convert to texture format -ok
4. Make a modeless dialog to display the palette. First just colors then pins. For now just add child dialog to set pin position and color. Later allow sliding pin with mouse.
5. Add method to save and load pin files
6. Connect so texture of renderer can be set
7. Work on serialization of whole doc.

Texture uses uint32\_t and just seems to spread whatever colors are given it. So we will store the palette as a bunch of rgbs (4 bytes one for each argb) and a double for the index or position between 0 and 1. The doc will have a resolution for the total number of colors in the palette, maybe 10,000. When converting from palette to texture, the app will generate the resolution number of colors (10,000). The colors will be calculated at each position by stretching.

Don’t worry about different types of stretching or the other 2 colors stored for each pin. Just serialize something for those values for now.

There is a gdiplus class called color which is close to what we want but I don’t want to have to use a strange header that might not always be supported. The documentation isn’t clear (<https://docs.microsoft.com/en-us/windows/desktop/api/gdipluscolor/nl-gdipluscolor-color>).

Make our own struct

RawColor

{

unsigned char red,

unsigned char green,

unsigned char blue,

unsigned char alpha

}

And make namespace with conversion methods

enum ColorStretchType {Linear = 0};

ColorPin

{

RawColor rawColor1;

RawColor rawColor2;

RawColor rawColor3;

double index;

ColorStretchType colorStretchType;

double curve;

}

7/17/18

For exporting colors see Chaos void cImageDoc::OnFileExportcolors()

Implemented serialization

1. Think about how to use with texture -ok
2. Make hard coded pins collections -ok
3. Make method to convert to texture
   1. Include method to make sure pin indices are from 0 to 1 and in order –put in DxColor -ok
4. Add one to view class and hook up to texture
5. Make a modeless dialog to display, load and save to file.
6. Hook up dialog
7. Test
8. Work on serializing doc with image.
9. Think about serializing raw triangles to speed up calculations

texture colors are in *std*::*vector*<*uint32\_t*>

we use this to make the color

static *uint32\_t* ColorrefToXmcolor(*COLORREF* cr, *uint32\_t* alpha)

{

// XMCOLOR is structured 0xaarrggbb

// COLORREF is structured 0x00bbggrr

return (alpha << 24) | (*GetRValue*(cr) << 16) | (*GetGValue*(cr) << 8) | *GetBValue*(cr);

}

We need a fixed number of colors like 10,000 and spread the colors using pins. This means making a color for each one.

The renderer doesn’t have to know about the palette, only the view. The view can call SetTextureColors with the color array

Before continuing we need to include linkage to DxSupport and DxColor.

7/24/18

Corrected ConvertPalette, etc

Correct library linking for both debug and release, may want to change later

commit

Added initial palette implementation with serialization and conversion. Fixed linking to DxColors.lib

Up to 4

Got exception trying to create texture

D3D11 ERROR: ID3D11Device::CreateTexture2D: The Dimensions are invalid. For feature level D3D\_FEATURE\_LEVEL\_10\_0, the Width (value = 9981) must be between 1 and 8192, inclusively. The Height (value = 1) must be between 1 and 8192, inclusively. And, the ArraySize (value = 1) must be between 1 and 512, inclusively. [ STATE\_CREATION ERROR #101: CREATETEXTURE2D\_INVALIDDIMENSIONS]

D3D11: \*\*BREAK\*\* enabled for the previous message, which was: [ ERROR STATE\_CREATION #101: CREATETEXTURE2D\_INVALIDDIMENSIONS ]

7/28/18

Commit

Reduced number of texture colors to 5000 because of Dx limits

We need a 2D texture because the vertex coordinates specify it. Maybe if we didn’t use an Effect we could code a 1D texture but it’s not worth it.

So revert and discard changes.

inclusively. And, the ArraySize (value = 1) must be between 1 and 512, inclusively. [ STATE\_CREATION ERROR #101: CREATETEXTURE2D\_INVALIDDIMENSIONS]

7/31/18

Commit

Got it to build with *D3D\_FEATURE\_LEVEL\_11\_0 and 10000 colors.*

Next

Have the initial colors created using this method. –ok

Commit

Work on dialog to load, save and display palettes

After pin palette dialog, move fields to doc

Plans for colors

Try a modeless dialog to display the colors then add a dialog that displays all the palettes in a directory. Look at chaos.

See ColorView.cpp

For palette dialog see ColorTreeDlg.h. But this serializes in a bitmap from the file. This draws the bitmaps in a treeview. I don’t know what we can draw out image in.

Maybe a picture control or just draw on the surface. For the palette selection dialog we can use the treeview approach.

8/4/18

Make a modeless dialog

Created a modeless dialog CPaletteViewDlg

Next add call back method to set palette back -ok

Then make double buffered bitmap and draw in dialog

[https://docs.microsoft.com/en-us/cpp/mfc/reference/cdc-class#stretchblt](https://docs.microsoft.com/en-us/cpp/mfc/reference/cdc-class" \l "stretchblt)

[https://docs.microsoft.com/en-us/cpp/mfc/reference/cdc-class#bitblt](https://docs.microsoft.com/en-us/cpp/mfc/reference/cdc-class" \l "bitblt)

Plan for double buffering

See void cColorView::OnDraw(*CDC*\* pDC) for example in Fractals

Make background image of fixed size with width = number of colors and height ~10

Fill background (solid now, stripes later for transparency)

Draw vertical lines, one pixel wide for each color

StretchBlt onto main bitmap/dc

draw pins (later)

Draw to screen

Need 2 CBitmap and CDC s

8/11/18

Removed dxgi.lib from DxSupport but still need in FractalX

Commits

Implemented DoubleBuffer and part of PaletteViewDlg.

Drew bmp on dialog

Next: fix position relative to controls. The draw colors.

8/14/18

1. Fixed bitmap position on dialog – commit
2. Populate palette name -commit
3. Draw colors in bitmap

Two problems

1. The number of colors does not always come out to the requested number – maybe fixed
2. The colors are not smooth, there are some lines – first bad color is about 100

Make sure StretchPaletteIndices(PinPalette& palette, double oldMin, double oldMax) in ColorPin.cpp is correct

8/17/18

Added a pin.

Working on CPinTracker

Made it display all the pins and committed

Next:

1. Add mouse handlers to move the pins
2. Make it update the palette and colors
3. Add methods to save and load the palette to a file
4. Serialize everything

8/21/18

Tried to implement pin movement but failed. Need to figure out

commit bad pins

8/25/18

Fixed at work.

Added NextUp.txt file to directory

This is at the start of it:

1) Fix so left most pin can go to end. Currently it's block about 1/4 th inch to right. -ok

2) Make pins stay in the plane. -ok

3) When pin is let go and it's exactly on top of another pin index, make it move 1 position rt or left -ok

4) Check pin movement

5) Add grid below

First: Palette is sometimes coming out with more or less than 1000 colors? It might be moving the pins to the ends

Commit

Fixed number of colors generated for palette so it should be the requested number.

Fixed problem with space on left of palette dialog

Made pins stay on the x plane

Made sure pins have different indices

8/28/18

Updated at work to import / export palette

Next allow editing of name -ok

Fixed palette color calculation –ok

Add context menu with empty handler to delete pin

IDR\_PALETTE\_VIEW\_CONTEXT\_MENU

ID\_PALETTE\_DELETE\_PIN

9/1/18

Commit

Converted PinEditDlg (not hooked up or tested)

I D\_PALETTE\_EDIT\_PIN

hook up dialog -ok

get working

make max index 999

Commit – Corrected pin index

remove junk from dialog and resources

Remove connect checks IDC\_CONNECT\_CHECK1 remove split checks

IDC\_SPLIT\_CHECK1 remove 3rd color bands

IDC\_BANDC\_EDIT2

Commit - Removed unused resources

Move controls to shrink (maybe add more pins later)

test remaining controls

9/4/18

Move controls

Commit

Hook up to parent so palette is passed back - Transferred pins back from pin edit dlg if OnOk

Test each function

Next - ok

Previous -ok

Delete 1 -ok

Delete 2 -ok

Delete 3 -ok

Insert 1 – inserts correct index but before first pin

Insert 2 – inserts correct index before first pin

Commit - Fixed insert function

Color Index 1 -ok

Color Index 2 -ok

Color Index 3 – bad mixes up other indices

Commit - Fixed pin index 3 bug

Color top 1 -ok

Color top 2 -ok

Color top 3 -ok

Color bottom 1-ok

Color bottom 2 -ok

Color bottom 3 -ok

Color 1 band 1 –isn’t enabled until you reopen the dialog

Color 1 band 2

Color 2 band 1

Color 2 band 2

Curve 1

Curve 2

9/8/18

Commit – Fixed all controls on pin edit dlg

Implement curve palette – it works – no changes needed

Implement banded palette

Commit - Implemented double banded pins

Implement Update button

Commit Made pin edit dlg modeless and implemented Update button

9/11/18

Working on serialization of doc

make extension frc

Chaos\nChaos\nChaos\nChaos Files (\*.cao)\n.cao\nChaos.Document\nChaos.Document

\nFractalX\nFractalX\n\n\nFractalX.Document\nFractalX.Document

FractalX\\nFractalX\nFractalX\ nFractalX Files (\*.frc)\n.frc\nFractalX.Document\nFractalX.Document

9/15/18

Try to remove shared\_ptr’s in doc members

Fix serialization

Commit Fixed serialization

Need SetModified in handlers

Commit: Added Save As button and SetModified() to setters

Need to add mutex to set/get in doc because it can be accessed from another thread. Maybe just vertex data

Commit: Added locks to doc data called from worker thread

Rethink color conversion from palette so we don't need to pass non-const palette all the time

When you change the palette it doesn’t update if it’s not rotating. Thought I fixed this before.

9/18/18

Add function to save file first 6 iterations 8 segments

save as triangles6\_8.triangles

31 sec for triangles only

55 sec for whole thing

24 for the whole thing with the file

0x00000083bdafe9e0 L"C:\\code\\github\\FractalXgit\\FractalX\\x64\\Debug\\triangles6\_8.triangles was not found."

C:\code\github\FractalXgit\FractalX\x64\Debug\Triangles

triangles6\_8.triangles

Next

triangles7\_8.triangles

Commit

Implemented loading triangle files

9/22/18

Commit

Added Triangle folder and ReadMe.txt describing them. – undid because files are too big

Note: Triangles10\_8 and Triangles10\_4 were too big to commit

Working on ConvertPalette

This should be 2 functions

ValidatePalette() – which exists and CalculatePaletteColors

When called:

View:: InitializeColors() – no need for validate

View::OnColors() – no need but should call it

View::SetNewPalette() – should call validate\*

PaletteViewDlg – OnInitDialog() no need to validate

PaletteViewDlg – PaletteChanged() should call or method that sets the palette should validate\*

// this can throw std::exception if there are less than 2 colors

DxColor::ValidatePalette(palette);

Next:

1. Rethink color conversion from palette so we don't need to pass non-const palette all the time-ok
2. Add ReadMe file to explain Triangle files
3. When you change the palette it doesn’t update if it’s not rotating. Thought I fixed this before.
4. Contrast and other color modes HVS?
5. Transparency

Commit

Spit ConvertPalette into ValidatePalette and CalculatePaletteColors –ok

Commit

Made it render if palette, colors, view or perspective changed.

Commit

Added Readme.txt explaining Triangle files

To speed up rendering files containing coordinates of triangles used to build the model can be placed in a folder named "Triangles" within the same path as the executable. The files have the form of triangles10\_2.triangles where the first digit is the number of iterations and the second the number of Seed Triangles (see the Model dialog vertex page). These can be built using the Export Triangles menu item under the Help menu. But the source has to be changed for each different number of iterations and seed triangles. See CFractalViewImpl:: void OnExportTriangles().

The seed triangles are used to calculate the model and if they need to be calculated they take almost the same amount of time as the model which can be over an hour. Using triangle files speeds up the calculation almost 2x.

9/29/18

Contrast params consist of two byte[3] for min and max

HSL params consist of 3 pairs of double for min/max hue, saturation and lightness

Also need something to determine which mode to use none, contrast or HSL

Apply when calculating colors from palette. Can just scan palette pins since these include min max colors

Dialog should set the values (min/max) and calculating the colors should apply

1. Scan code for locations of where to put dialog, where colors are generated, and where to store params
2. Make param class, ColorContrast-ok
3. Make dialog
4. Make calculation for contrast
5. Hook up
6. Test
7. Make calculation for HSL

1 – First put dialog button on palette dlg then on main menu/toolbar

ColorUtilities:: CalculatePaletteColors() and internally at AddColor()

Store Contrast Params in Doc

2 – put params and calculations in DxColors

Commit

Added ColorContrast Params

Made IDD\_CONTRAST\_DLG with IDC\_MODE\_COMBO

CContrastDlg

Made skeleton of dlg. Need to add to palette Dlg and connect params

10/2/18

Commit

Partial implantation of contrast dialog

10/6/18

1. Implement OnHScroll -ok
2. Add buttons for Palette dialog and handler -ok
3. Display and test -ok
4. Hook up edit boxes
   1. Implement DDX for HSL edit -ok
   2. Fix so it doesn’t say enter a number and the initial values are correct -ok
   3. OnLoseFocus for Edits-ok
   4. Make sure min < max-ok
5. Add implementation to use contrast
6. Add implementation to use HSL
7. Maybe make modeless
8. Add to main window

Commit- Implemented more contrast dlg

Commit – Fixed OnModeChanged()

Commit – Connected Edits in contrast dlg

10/9/18

Commit

Fixed contrast dlg validation

Commit Added contrast to color method

Next

HSL – look at Fracture RawImage.cs – not sure how to handle because it determine HSL min max from colors in image. Apply to palette colors instead of image colors

Modeless contrast update

Add to main window

10/13/18

Implemented converting colors with contrast but

1. Palette looks choppy -ok
2. The contrasted colors aren’t used in the model –ok

Commit - Implemented contrast

10/19/18

see <https://stackoverflow.com/questions/39118528/rgb-to-hsl-conversion>

<https://stackoverflow.com/questions/3018313/algorithm-to-convert-rgb-to-hsv-and-hsv-to-rgb-in-range-0-255-for-both>

Commit HSL implementation with bugs

Got bugs in RGB to HSL Hue calculation –

1) for magenta hue is -60 and should be 300. This might just be adding 360 when the hue is negative but I don’t see that in Fracture or other examples

2) I also think there is bug converting back in rounding since only primary colors come back

Commit Fixed to hsl

Try the calculation from

<https://gist.github.com/fairlight1337/4935ae72bcbcc1ba5c72>

First increase H to 360

Made ToHSL() function

need to replace ToRgb()

10/27/18

Commit: Fixed hsl/rgb conversion

11/3/18

Hook up CCOlorSelectorDlg to CPinEditDlg -ok

Add button to bring up windows dlg –ok

Add color square with DC -ok

Change to gdiplus

draw strips behind square

convert the rest of the colors in pin edit dlg

Add color square – draw stripes in background and color on top with transparency – will need gdiplus

Next work on transparency

1. Display a with RGB in pin dialog when you select a pin
2. When you click on a color allow entering an argb and add another button to use color dialog
3. Turn on gdiplus for app
4. Display stripes behind color square and draw color square with gdiplus.

Commit: Display ARGB in PinEdit dlg

Next Features

1. Transparency at model level and pin level
2. Add buttons to the main window/toolbar
3. Background image
4. Mouse handlers or buttons to move model or rotate

12/15/18

Commit

Got transparency working