CS 475/575 -- Spring Quarter 2015

Project #1

OpenMP: Numeric Integration with OpenMP

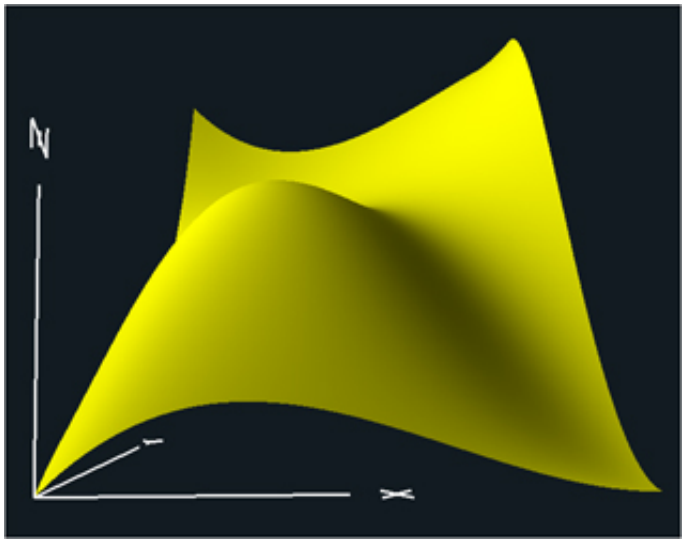
100 Points

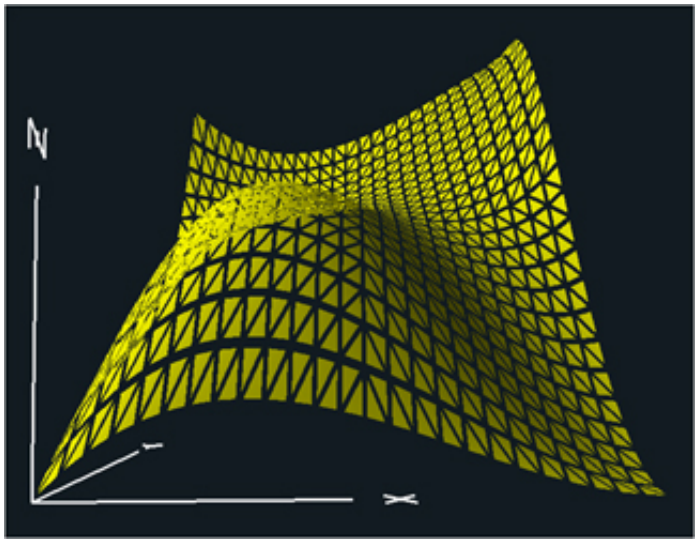
Due: April 21

This page was last updated: March 25, 2015

Introduction

Bézier surfaces are a way of dynamically sculpting shapes. Because they are analytically defined, they can be computed (and rendered) to any precision you want. In this project, we are going to take this Bézier surface:





and use numerical techniques to find the volume between this surface and the horizontal X-Y (Z=0) plane.

The surface's "floor" is the X-Y plane (i.e., Z=0), Using some number of subdivisions in both X and Y, `NUMSxNUMS`, take NUMS^2 height samples.

We will think of each height sample as sitting on a 2D tile on the floor. That really makes that height sample act as a volume where the tile is extruded vertically from the floor to the height.

The tiles in the middle of the floor are full-sized tiles. Tiles along the edges are half-sized. Tiles in the corners are quarter-sized. The volume contribution of each extruded height tile needs to be weighted accordingly. The logic of this is for you to figure out.

Requirements

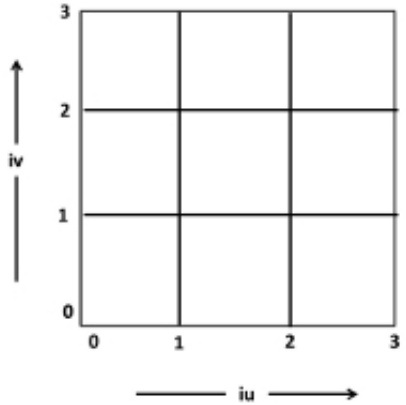
- Using OpenMP, compute the total volume under the surface.
- Use a variety of number of subdivisions (NUMS).
- Use a variety of number of threads (NUMT).
- Record the data in units of something that gets larger as speed increases. Joe Parallel used "MegaHeights Computed Per Second", but you can use anything that makes sense.
- From the speed-up that you are seeing, use the "Inverse Amdajl's law" to determine the Parallel Fraction for this application.

- From the Parallel Fraction, determine what maximum speed-up you could *ever* get, even with a million cores.
- Your commentary write-up (turned in as a PDF file) should include:

1. Tell what machine you ran this on
2. What do you think the actual volume is?
3. Show the performances you achieved in tables and graphs as a function of NUMS and NUMT
4. What patterns are you seeing in the speeds?
5. Why do you think it is behaving this way?
6. What is the Parallel Fraction for this application?
7. What is the maximum speed-up you could *ever* get?

The Height-Evaluation Code

In this code sample, `NUMS` is the number of lines subdividing the floor area. So, for example, `NUMS=4` means that there is one line at each edge and two lines in the interior like this.



I recommend a single for-loop over all the nodes that looks like this:

```
for( int i = 0; i < NUMS*NUMS; i++ )
{
        int iu = i % NUMS;
        int iv = i / NUMS;

        . . .
}
```

The code to evatuate the height at a given *iu* and *iv* is:

```
#define XMIN     0.
#define XMAX     3.
#define YMIN     0.
#define YMAX     3.

#define Z00     0.
#define Z10     1.
#define Z20     0.
#define Z30     0.

#define Z01     1.
#define Z11     6.
#define Z21     1.
#define Z31     0.

#define Z02     0.
#define Z12     1.
#define Z22     0.
#define Z32     4.

#define Z03     3.
#define Z13     2.
#define Z23     3.
#define Z33     3.

float
Height( int iu, int iv )        // iu,iv = 0 .. NUMS-1
{
        float u = (float)iu / (float)(NUMS-1);
        float v = (float)iv / (float)(NUMS-1);

        // the basis functions:

        float bu0 = (1.-u) * (1.-u) * (1.-u);
        float bu1 = 3. * u * (1.-u) * (1.-u);
        float bu2 = 3. * u * u * (1.-u);
        float bu3 = u * u * u;

        float bv0 = (1.-v) * (1.-v) * (1.-v);
        float bv1 = 3. * v * (1.-v) * (1.-v);
        float bv2 = 3. * v * v * (1.-v);
        float bv3 = v * v * v;
```

```
        // finally, we get to compute something:

        float height =    bu0 * ( bv0*Z00 + bv1*Z01 + bv2*Z02 + bv3*Z03 )
                        + bu1 * ( bv0*Z10 + bv1*Z11 + bv2*Z12 + bv3*Z13 )
                        + bu2 * ( bv0*Z20 + bv1*Z21 + bv2*Z22 + bv3*Z23 )
                        + bu3 * ( bv0*Z30 + bv1*Z31 + bv2*Z32 + bv3*Z33 );

        return height;
}
```

The main Program

Your main program would then look something like this:

```
float Height( int, int );

int main( int argc, char *argv[ ] )
{
        . . .

        // the area of a single full-sized tile:

        float fullTileArea = (  ( (XMAX-XMIN)/(float)(NUMS-1) )  *  ( ( YMAX - YMIN )/(float)(NUMS-1) )  );

        // sum up the weighted heights into the variable "volume"
        // using an OpenMP for loop and an addition reduction:

        ?????

        for( int i = 0; i < NUMS*NUMS; i++ )
        {
                int iu = i % NUMS;
                int iv = i / NUMS;

                ?????
        }

        . . .
}
```

Grading:

Feature	Points
Results for a variety of NUMS values	30
Results for a variety of NUMT values	30
Commentary, including tables, graphs, correct volume, Parallel Fraction, and ultimate speedup	40
Potential Total	**100**