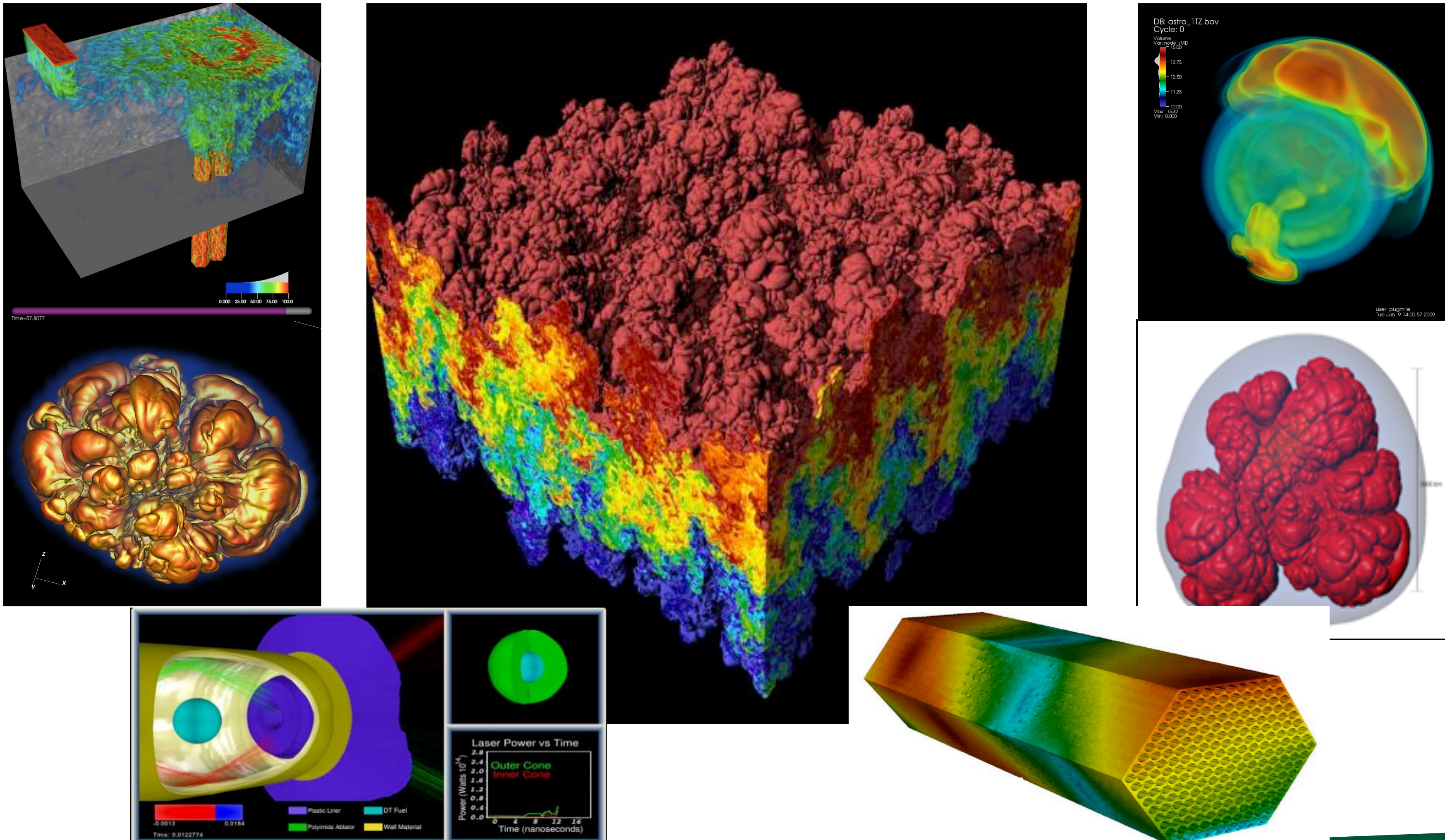
CIS 441/541: Intro to Computer Graphics
Lecture 7: Math Basics, Lighting Introduction & Phong Lighting



April 27, 2021

Hank Childs, University of Oregon

Midway Experience: thank you to the 2 people who responded

Midway Student Experience Survey opens next week

otp@uoregon.edu
Mon 4/12/2021 8:10 AM
To: Hank Childs

Dear Hank,

The Midway Student Experience Survey for your courses will open at 08:00 AM on Mon, Apr 19, 2021 PDT and will close at 06:00 PM on Fri, Apr 23, 2021 PDT. You can view the feedback from your students beginning April 26th at noon.

Students will receive an email from the Office of the Registrar directing them to Duckweb to complete the survey when it opens next week.

Other ways to increase response rates and quality feedback include:

1. Make it an assignment (you don't have to give points or extra credit or even keep track).
2. Tell your students that their feedback is valuable to you.
3. Provide students with examples of useful and actionable comments, in contrast to non-actionable comments.

Resources:
Office of the Provost: Revising UO's Teaching Evaluations
Teaching Engagement Program: Student Feedback
Office of the Registrar: Student Experience Survey FAQ

For questions, email the Office of the Provost at otp@uoregon.edu.

Thank you!
Office of the Provost

Proposed Change to Syllabus/Quiz Structure: YES

Proposed change to syllabus / quiz structure

Hank Childs

Apr 17 at 3:12pm

All Sections

Hello Everyone,

I would like to modify the course grading structure. Currently, we are planning on 5 quizzes, at 5 points each.

I would like to add a "quiz redo" at the end of the term. Consider three students, S1, S2, and S3, who have completed all 5 quizzes. S1 forgot about Quiz 3, missed class, and got a zero, S2 performed poorly on Quiz 2, and S3 got a perfect score on every quiz.

During Week 10, we would use one lecture as a "quiz redo." S1 would choose to "redo" Quiz 3. S2 would choose to retake Quiz 2. S3 could skip class (and may feel that the redo policy worked against them). If S1 or S2 scored higher on the redo, then they would get the higher score. (Lower scores would be ignored.)

I note the "redo" quizzes would be harder than the originals -- students would need to really understand the material to improve.

I plan to discuss this proposal on Tuesday's lecture. That said, if you object and you don't want your peers to know, then you are welcome to email me private comments offline.

Best,
Hank

Class Plan

- Abhishek and I are working hard on preparing Project 2 (OpenGL)
  - Quite frankly not going as well as we thought
- Projects will start coming faster
  - Want there to time to do great final projects
- 1E, 1F: simpler coding, harder concepts
- Cannot finish shading today
  - Also have another activity

Class Plan (New Slide)

- Option A:
  - Slow class down, more time on OpenGL, less time for final project
- Option B:
  - Keep up the pace, less time on OpenGL, more time for final project

Current Plan (1/2)

Week	Sun	Mon	Tues	Weds	Thurs	Fri	Sat
5			Lec 7 (shading), 1F assigned, 1E due		Lec 8 (finish shading, GL), 2A assigned		
6		1F due	Lec 9 (GL), 2B assigned		Discussion of final projects / Quiz 3		2A due
7			Lec 11 – ray tracing		More discussion of final projects (?)	2B due	

Current Plan (2/2)

- Weeks 8-10 → you work on final projects
- Lectures will be on misc. topics in graphics, esp. in support of final projects
- Quiz 3 (Week 6): likely on matrices
- Quiz 4 (Week 8): likely on GL
- Quiz 5 (Week 10): likely on topics in final weeks

Office Hours

Published | Edit

How to access Office Hours

Hank Childs

Apr 4 at 2:02pm

All Sections

Hi Everyone,

We currently have an asymmetry for accessing Hank and Abhishek's Office Hours.

As of now, Abhishek's are always at: COVERED UP (THIS IS POSTED ONLINE)

And Hank's are accessible via the Zoom Meetings area in Canvas.

Let's chat on Tuesday about the most standard way to do this.

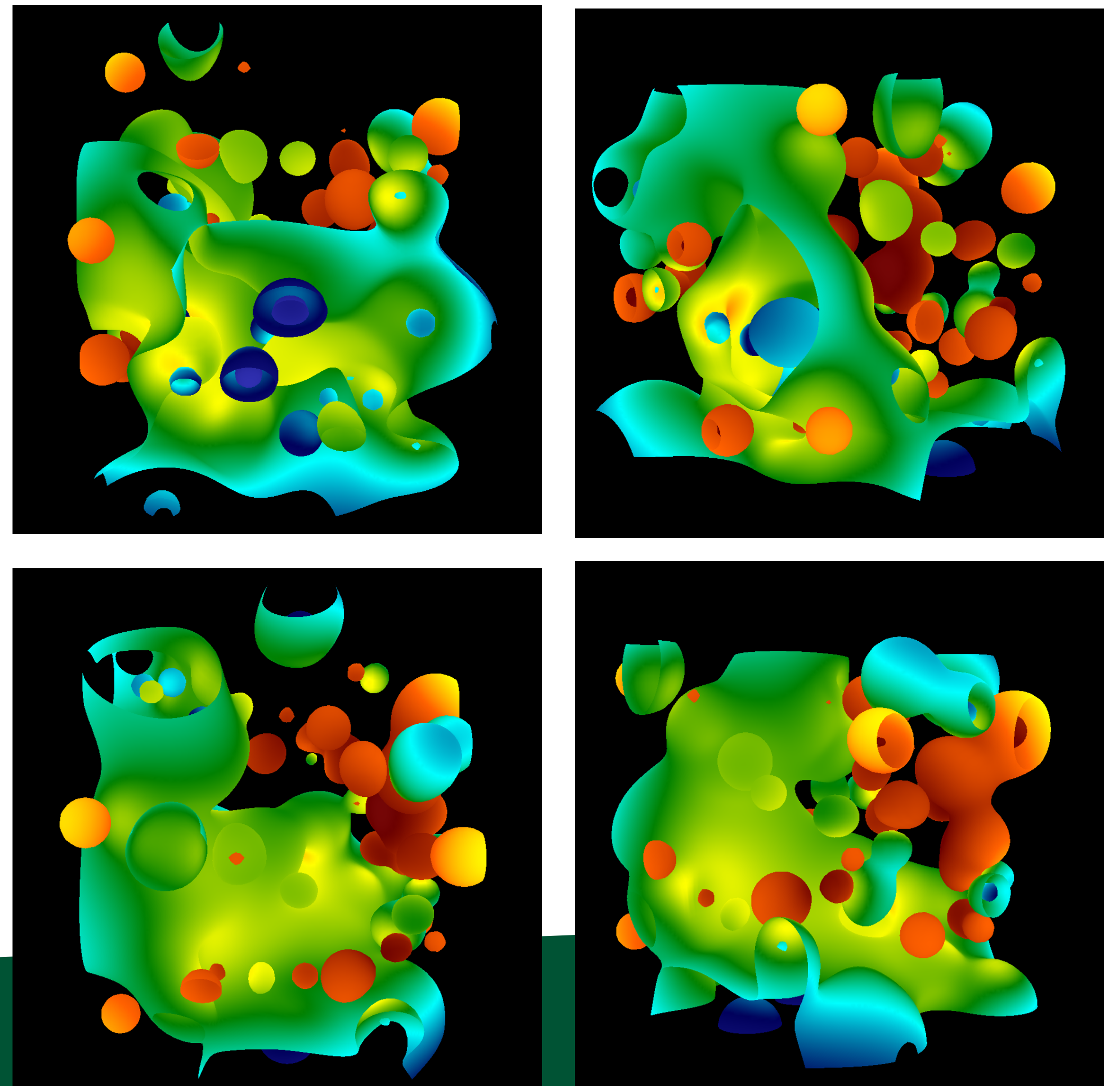
Finally, here is the OH schedule again:

Monday (Abhishek): 10am-11am
Tuesday (Abhishek): 945am-1045am
Wednesday (Hank): 230pm-330pm
Thursday (Abhishek): 945am-1045am

Best,
Hank

Project #1E (6%), Due Tues April 27th

- Goal: add arbitrary camera positions
- Extend your project1D code
- New: proj1e_geometry.vtk available on web (9MB), “reader1e.cxx”.
- New: Matrix.cxx, Camera.cxx
- No Cmake, project1E.cxx



Outline

- Math Basics
- Lighting Basics
- The Phong Model

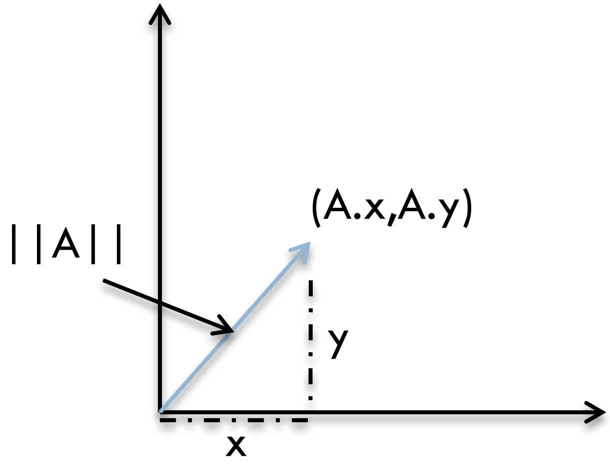
Outline

- Math Basics
- Lighting Basics
- The Phong Model

What is the norm of a vector?

- The norm of a vector is its length
  - Denoted with || · ||
- For a vector A = (A.x, A.y),

  ||A|| = sqrt(A.x*A.x+A.y*A.y)
- Physical interpretation:



- For 3D, ||A|| = sqrt(A.x*A.x+A.y*A.y+A.z*A.z)

What does it means for a vector to be normalized?

- The vector A is normalized if $||A|| = 1$.
  - This is also called a unit vector.
- To obtain a normalized vector, take $A/||A||$
- Many of the operations we will discuss today will only work correctly with normalized vectors.
- Example: $A=(3,4,0)$. Then:
  - $||A|| = 5$
  - $A/||A|| = (0.6, 0.8, 0)$

What is the normal of a triangle?

- A triangle coincides with a flat plane.
- A triangle's normal is the vector perpendicular to that plane.
- If a triangle is on plane = Ax+By+Cz = D,

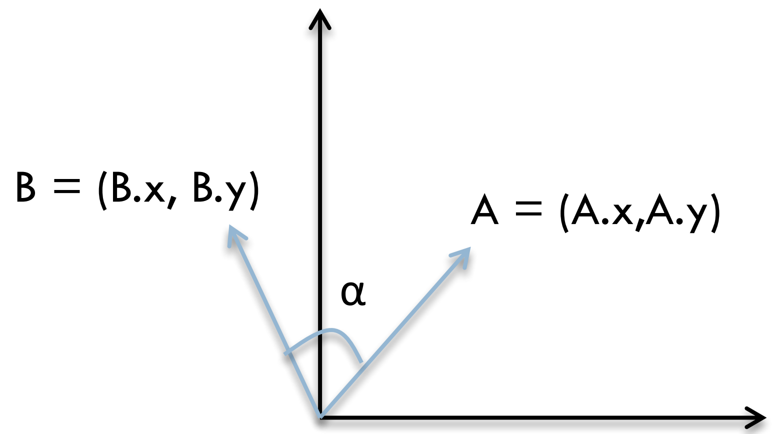
  then the triangle's normal is (A, B, C)

Norm, Normal, Normalize, Oh My!

- Norm: the length of a vector (||A||)
- Normal: a perpendicular vector to a plane coincident with geometry
- Normalize: the operation to create a vector with length 1 (A/||A||)
- All 3 are important for today's lecture

What is a dot product?

- $A \cdot B = A.x*B.x + A.y*B.y$
  - (or $A.x*B.x + A.y*B.y + A.z*B.z$)
- Physical interpretation:
  - $A \cdot B = \cos(\alpha)*(||A||*||B||)$



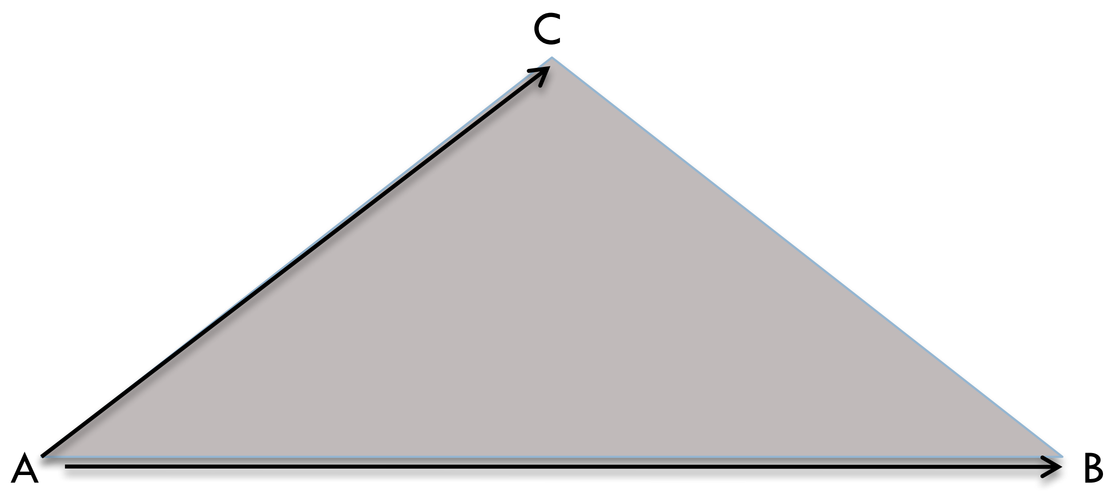
What is the cross product?

- AxB = (A.y*B.z - A.z*B.y,
  B.x*A.z - A.x*B.z,
  A.x*B.y - A.y*B.x)

- What is the physical interpretation of a cross product?
  - Finds a vector perpendicular to both A and B.

Easy Way to Calculate Normal For a Triangle

- Normal = (C-A)x(B-A)



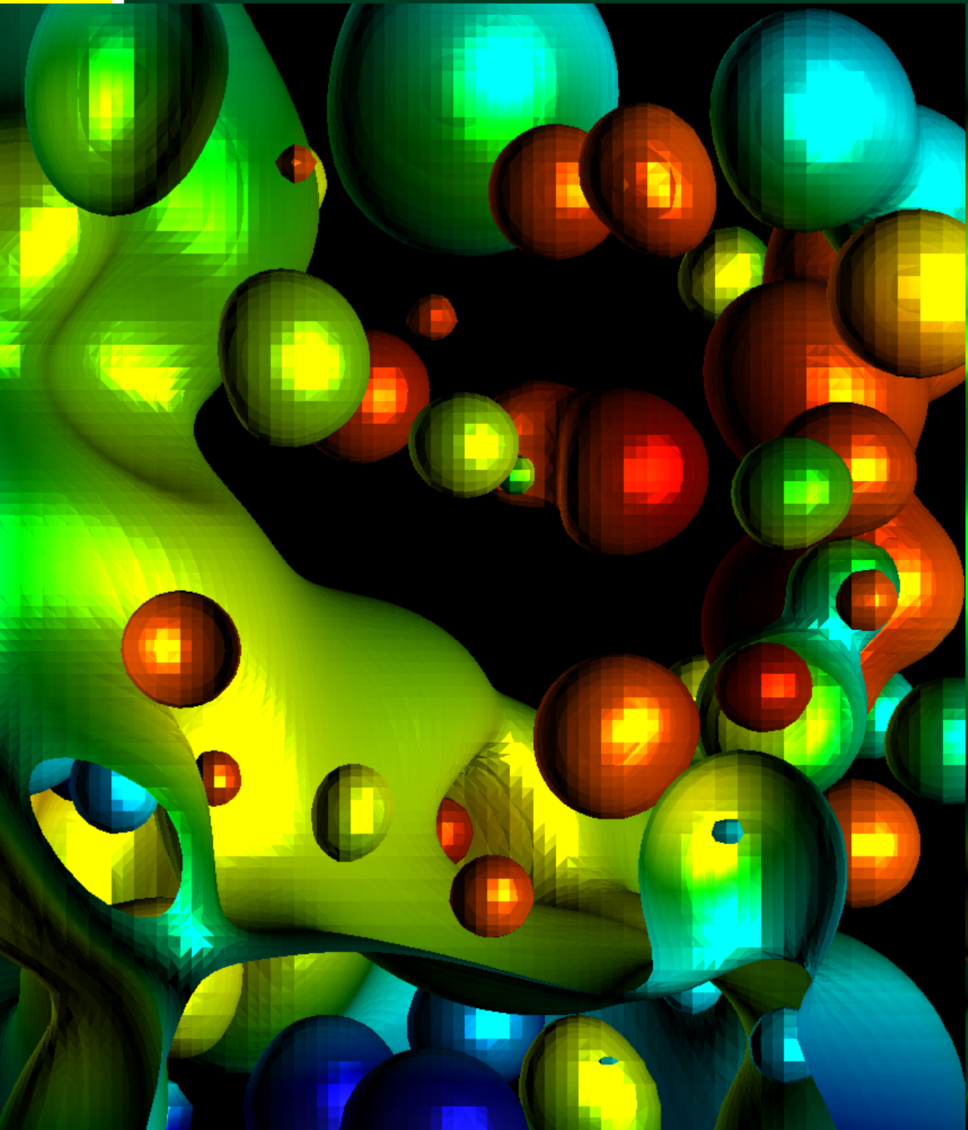
Important:
(C-A)x(B-A) != (B-A)x(C-A)
… we'll worry about this later

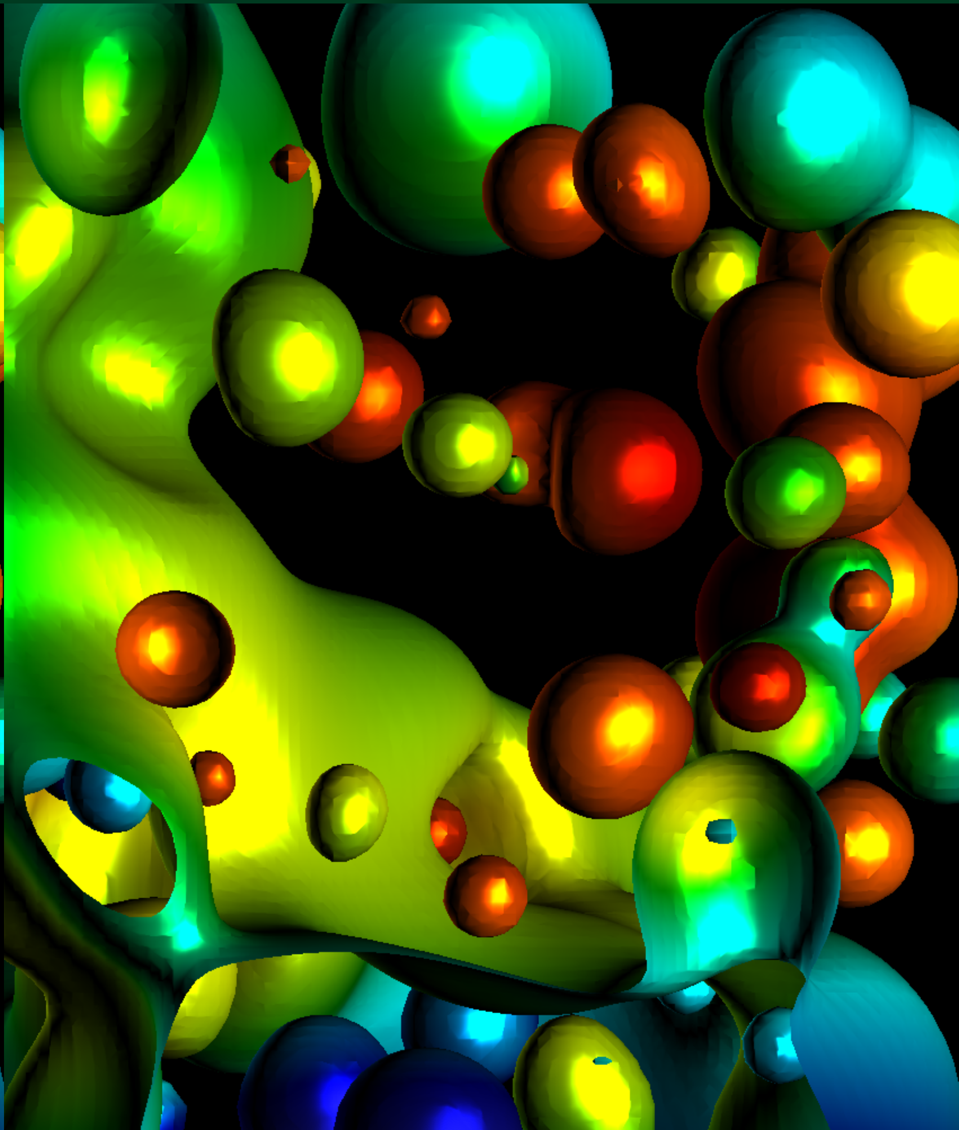
Lighting and Normals

- Two ways to treat normals:
  - Constant over a triangle
  - Varying over a triangle

- Constant over a triangle ←→ flat shading
- Varying over a triangle ←→ smooth shading

Flat vs Smooth Shading





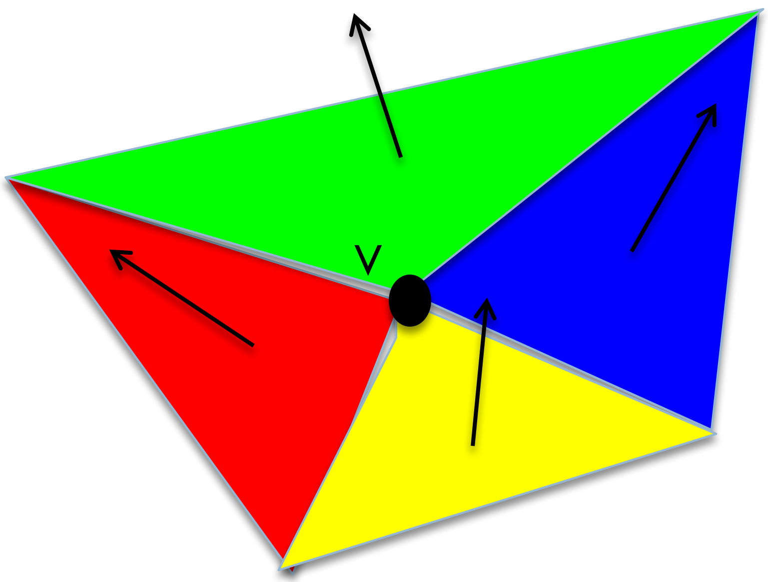
Lighting and Normals

- Two ways to treat normals:
  - Constant over a triangle
  - Varying over a triangle

- Constant over a triangle ←→ flat shading
  - Take (C-A)x(B-A) as normal over whole triangle
- Varying over a triangle ←→ smooth shading
  - Calculate normal at vertex, then calculate shading at vertex, then LERP shading
    - How do you calculate normal at a vertex?

Vertex Normals

- Algorithm:
  - For vertex V,
    - Find all triangles T_i incident to V
    - Normal(V) = {0,0,0}
    - NumIncident = 0
    - For each T_i,
      - calculate Normal(T_i)
      - Normal(V) += Normal(T_i)
      - NumIncident++
    - Normal(V) /= NumIncident



N(V) = (N(T1)+N(T2)+N(T3)+N(T4)) / 4

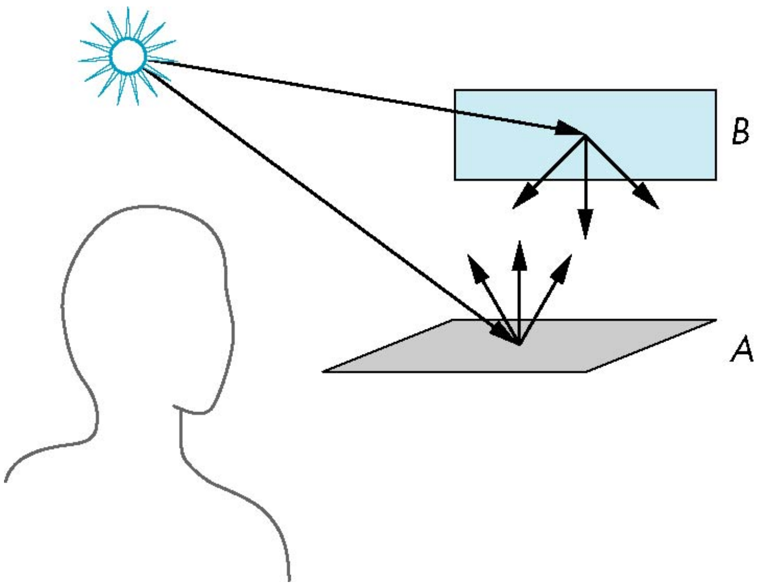
- Note: our data structures don't allow for "Find all triangles T_i incident to V" very easily.
  - Vertex normals are precalculated for 1F

Outline

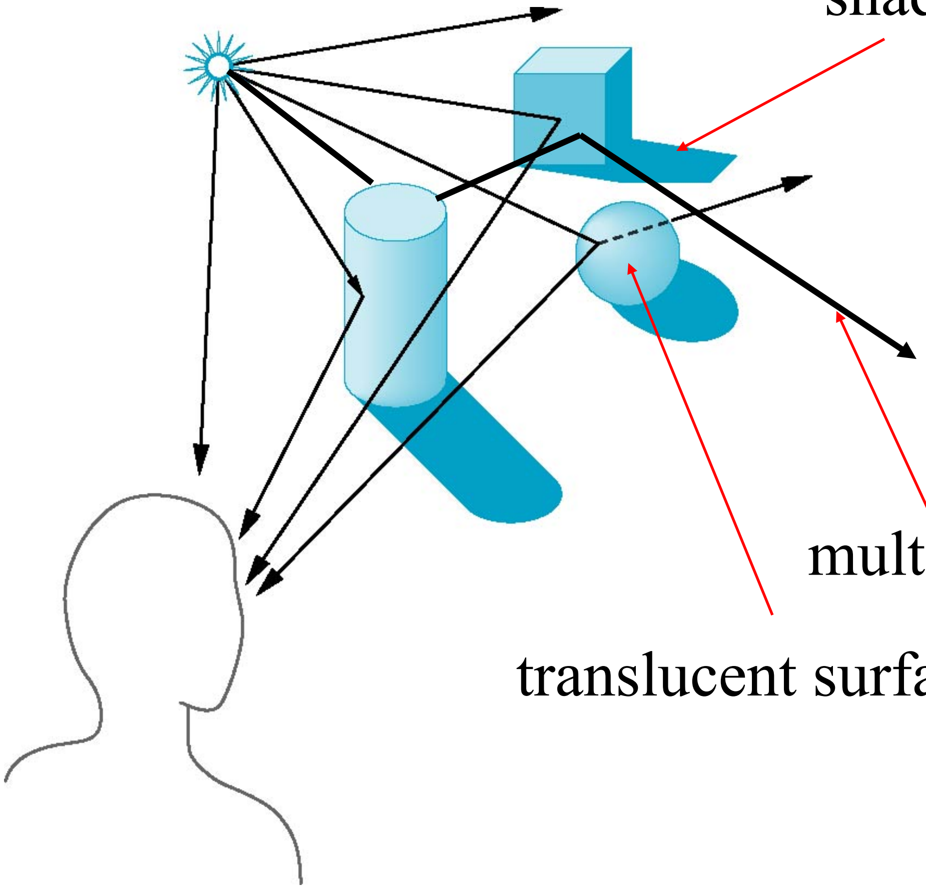
- Math Basics
- Lighting Basics
- The Phong Model

Scattering

- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A and so on



Global Effects



Local vs Global Rendering (1/2)

- Local rendering: when rendering one triangle, ignore the effects of other triangles
- Global rendering: when rendering one triangle, consider the effects of other triangles

Local vs Global Rendering (2/2)

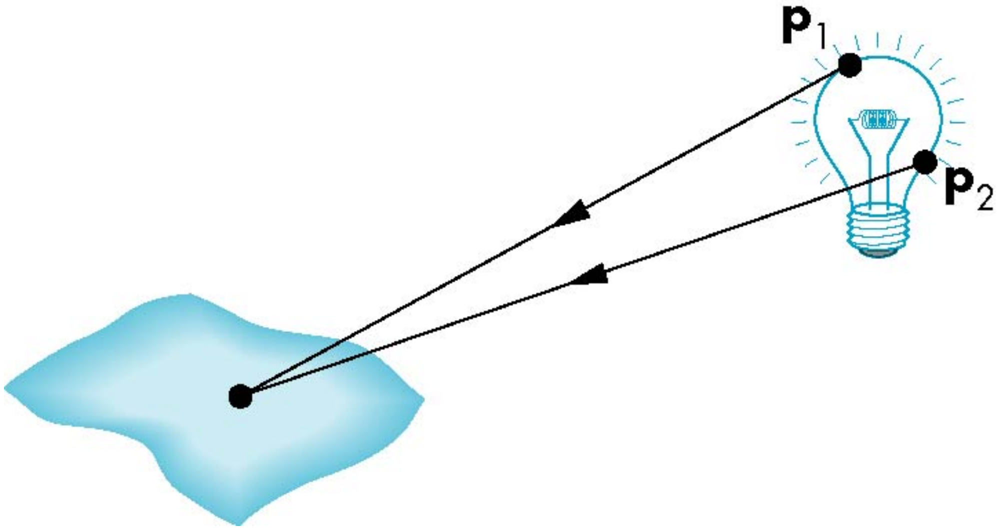
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things "look right"
  - Many techniques exist for approximating global effects
    - I.e., do local rendering, but bring in other knowledge to make it look like global rendering

Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Light Sources

General light sources are difficult to work with because we must integrate light coming from all points on the source

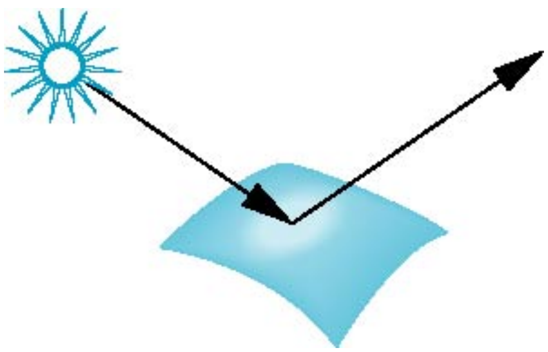


Simple Light Sources

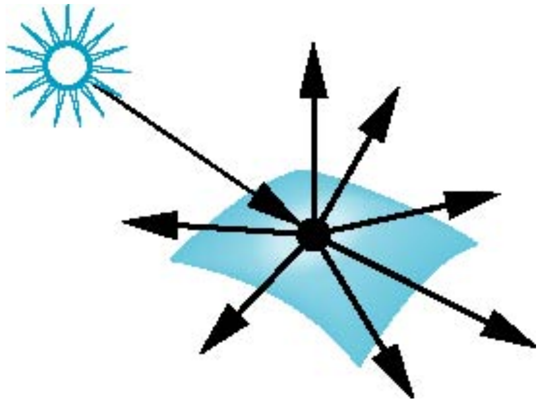
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source
- (We will do point sources for 1F … and this class)

Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction that a perfect mirror would reflect the light
- A very rough surface scatters light in all directions



smooth surface

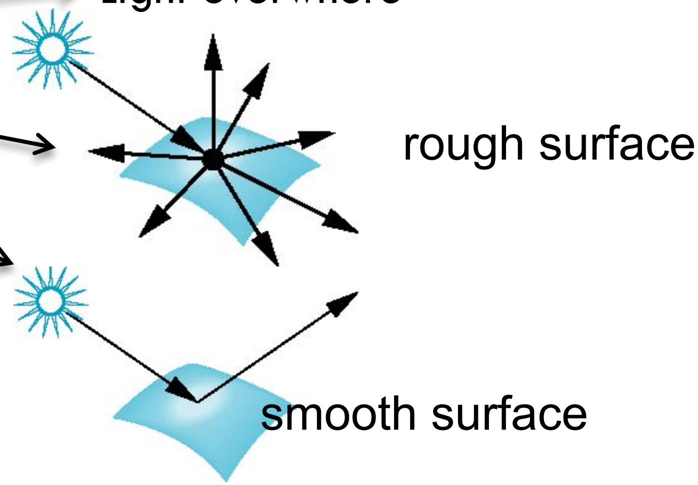


rough surface

Shading

- Our goal:
  - For each pixel, calculate a shading factor
  - Shading factor typically between 0 and 1, but sometimes >1
    - Shading >1 makes a surface more white
- 3 types of lighting to consider:
  - Ambient
  - Diffuse
  - Specular

Light everwhere



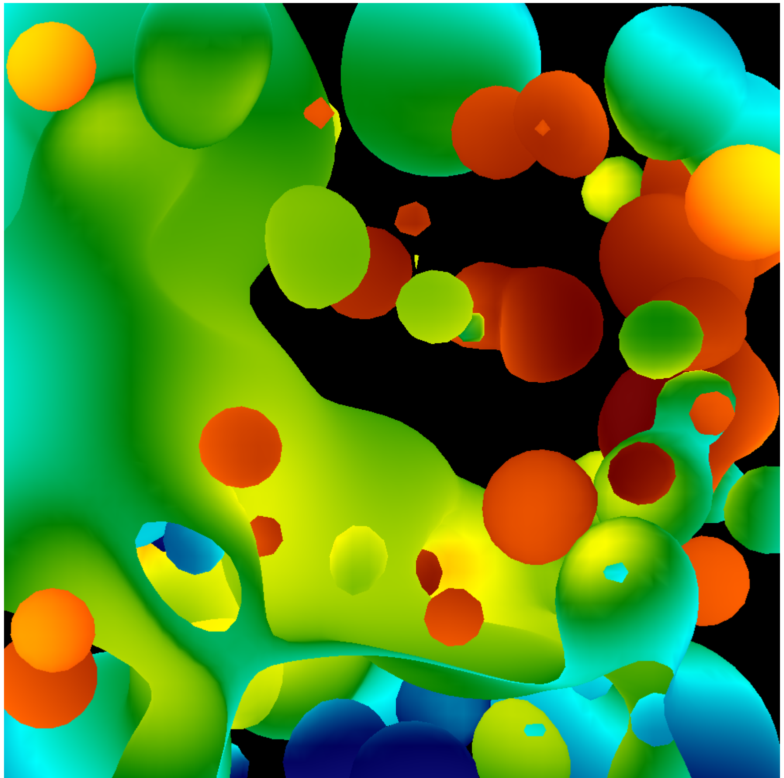
<u>Our game plan:</u>
Calculate all 3 and combine them.

How to handle shading values greater than 1?

- Color at pixel = (1.0, 0.4, 0.8)
- Shading value = 0.5
  - Easy!
  - Color = (0.5, 0.2, 0.4)→ (128, 52, 103)
- Shading value = 2.0
  - Color = (1.0, 0.8, 1.0) → (255, 204, 255)
- Color_R = 255*min(1, R*shading_value)
- This is how bright lights makes things whiter and whiter.
  - But it won't put in colors that aren't there.

Ambient Lighting

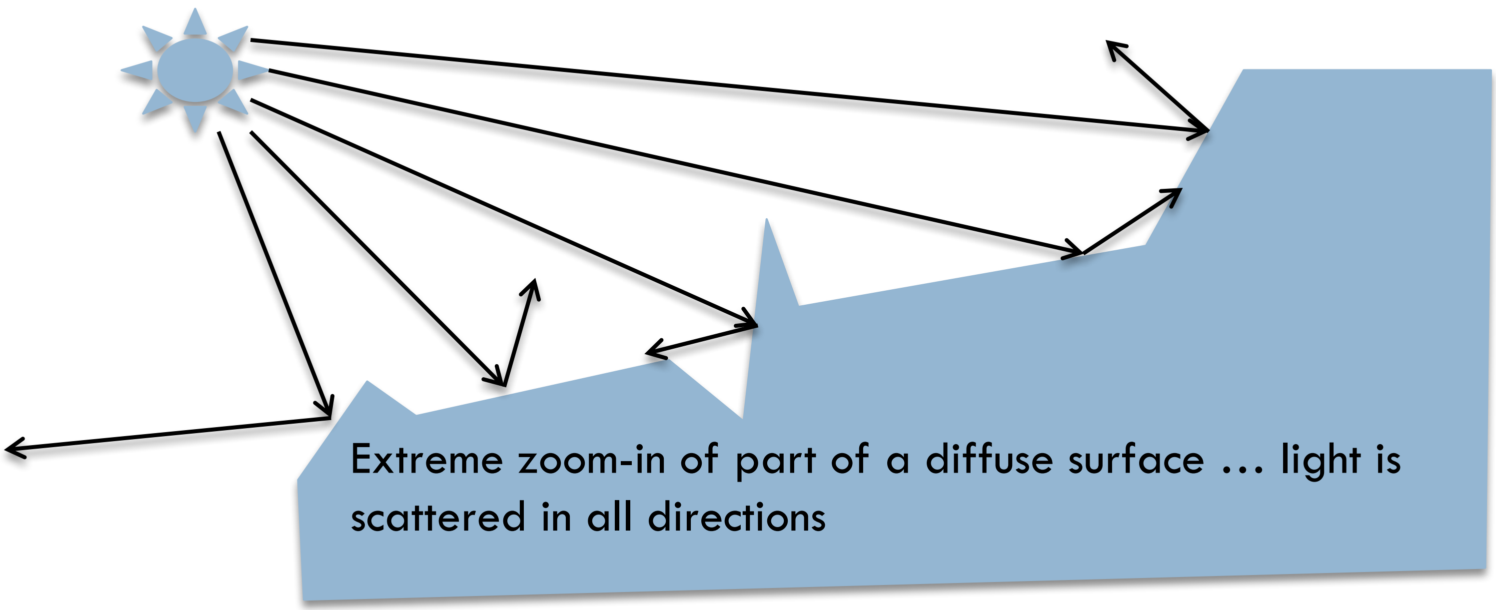
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces



Surface lit with ambient lighting only

Lambertian Surface

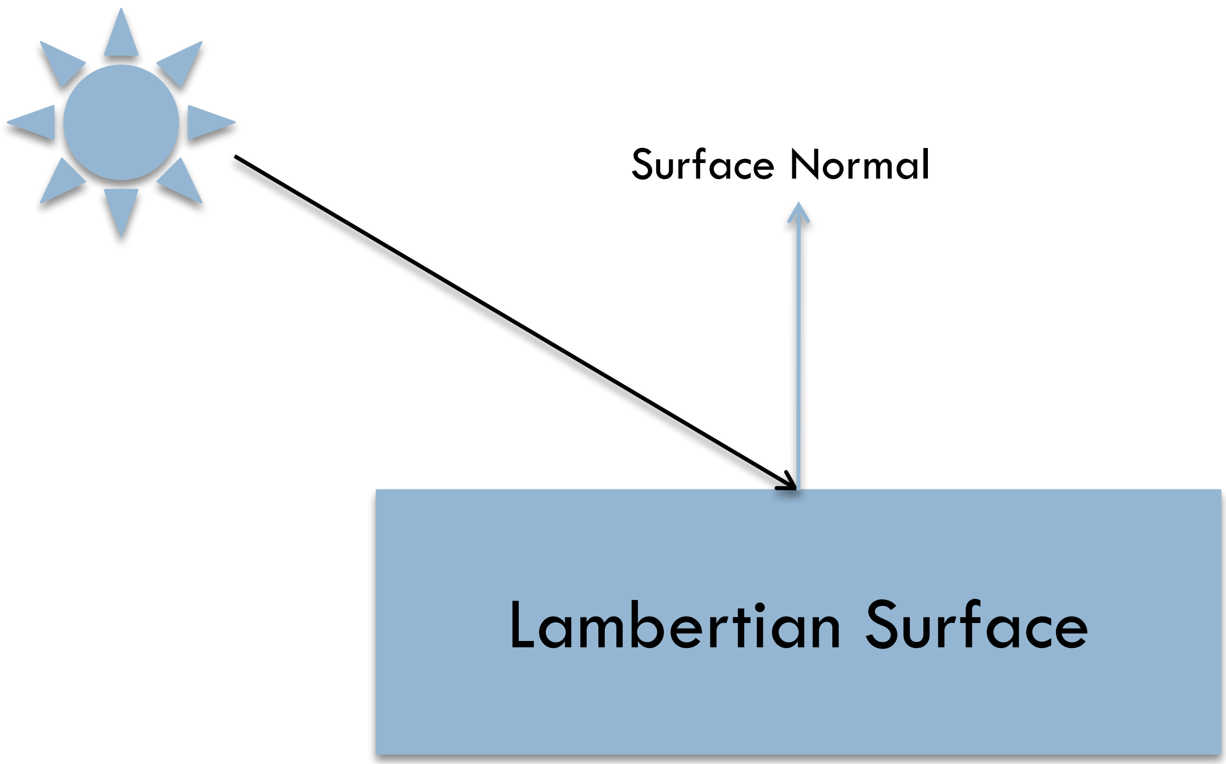
- Perfectly diffuse reflector
- Light scattered <u>equally</u> in all directions



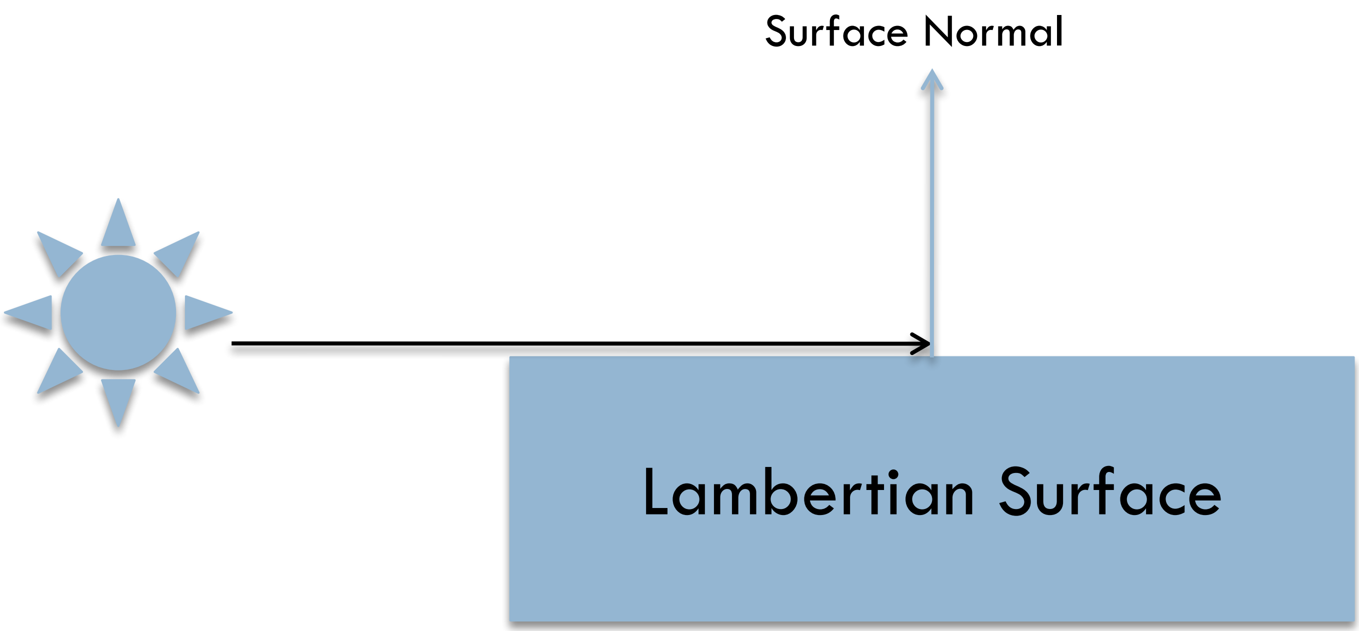
(this image shows 5 of the directions)

Slide inspired by Ed Angel Computer Graphics Book

Diffuse Lighting

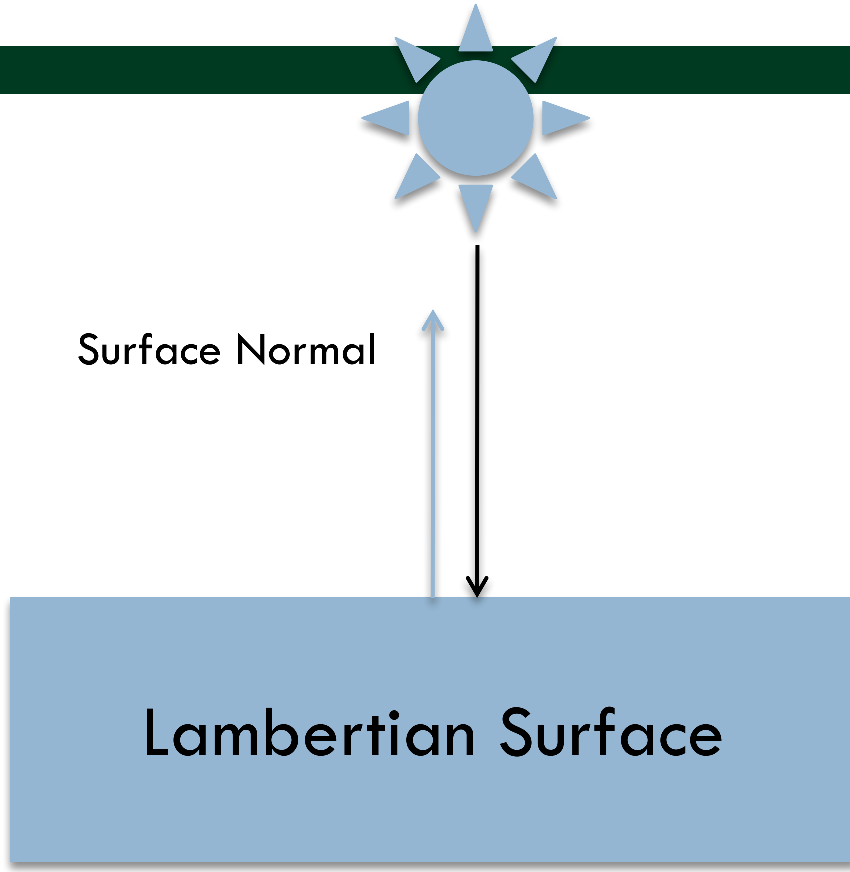


Diffuse Lighting



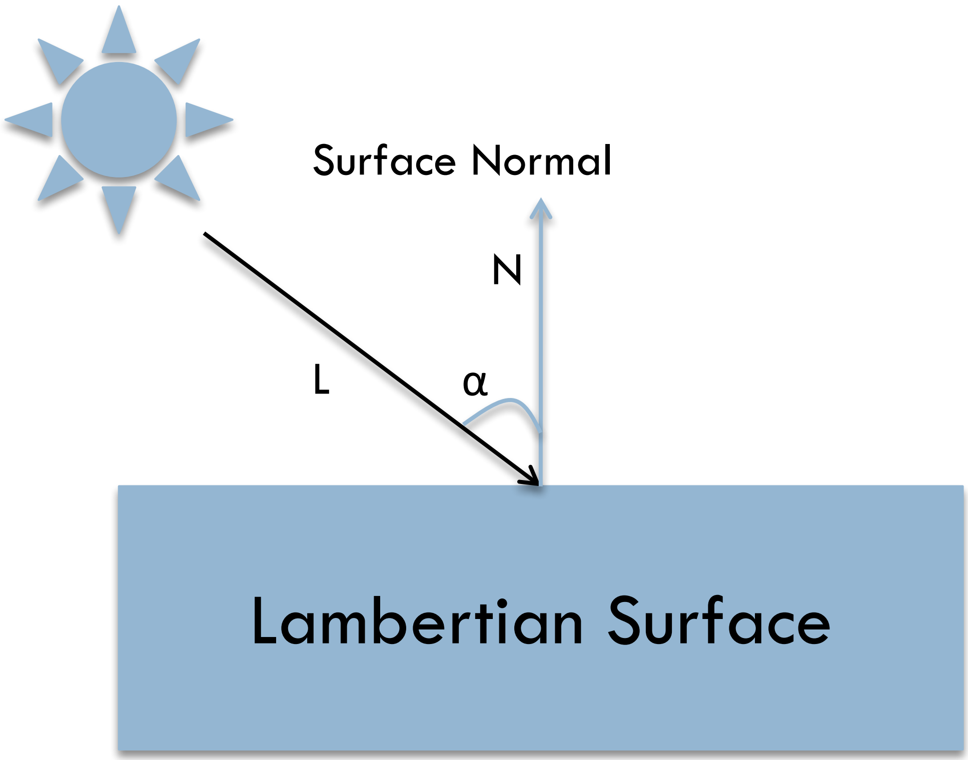
No light reflects off the (top) surface
(Light direction and surface normal are perpendicular)

Diffuse Lighting



When the light squarely hits the surface, then that's when the most light is reflected

Diffuse Lighting



How much light should be reflected in this case?

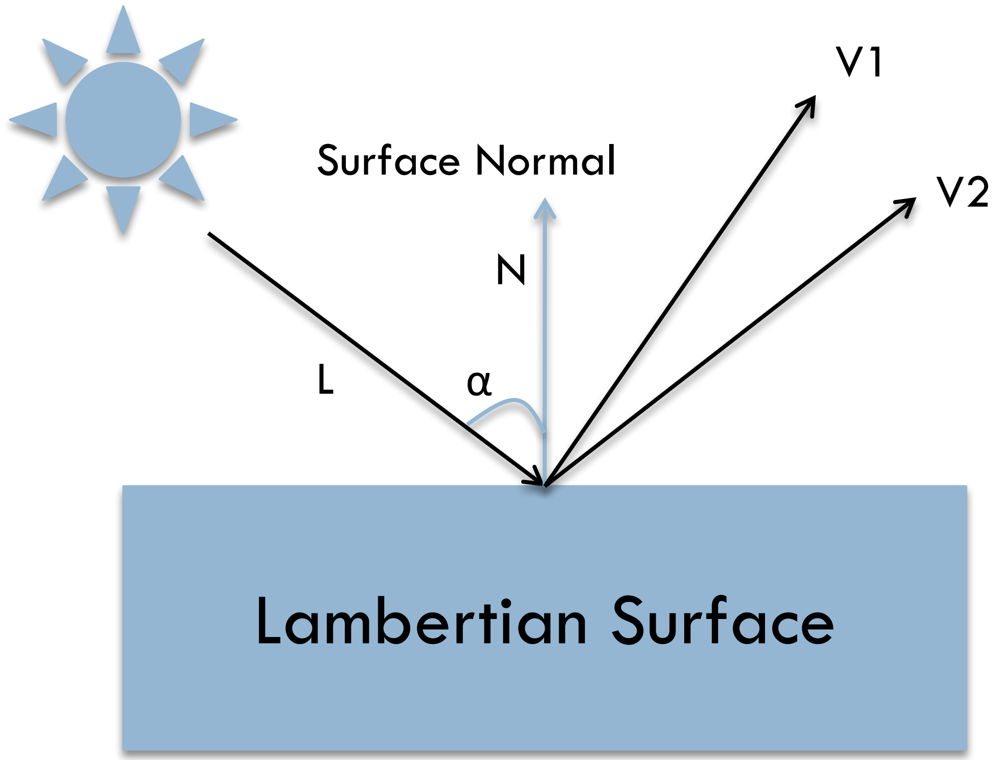
<u>A: cos(α)</u>

And note that:

$\cos(0) = 1$

$\cos(90) = 0$

Diffuse Lighting



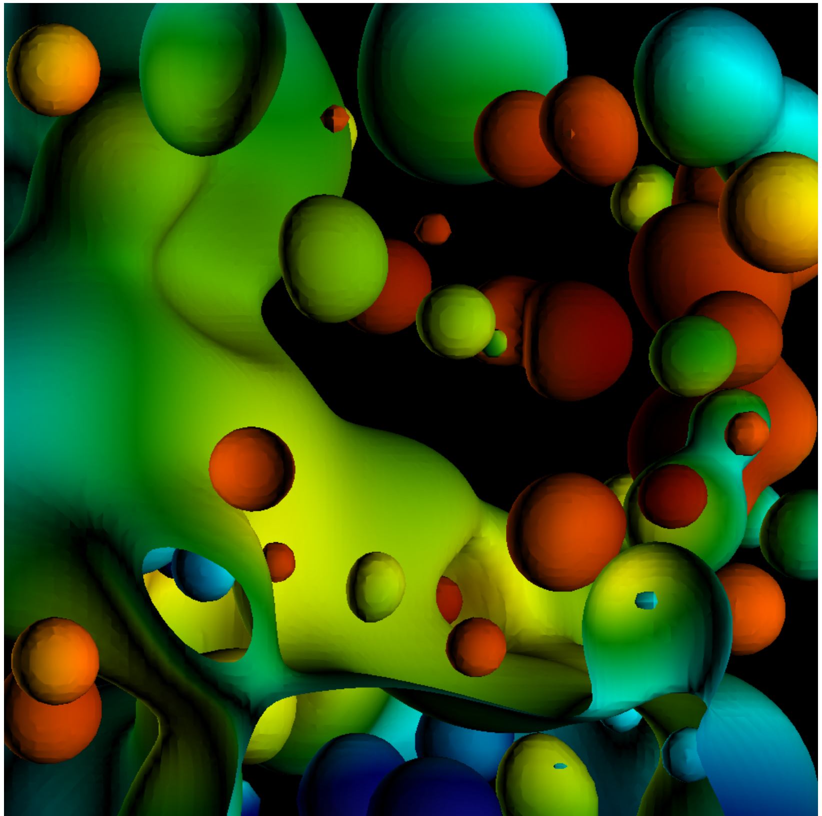
How much light makes it to viewer V1? Viewer V2?

A: cos(α) for both

Lambertian surfaces reflect light equally in all directions

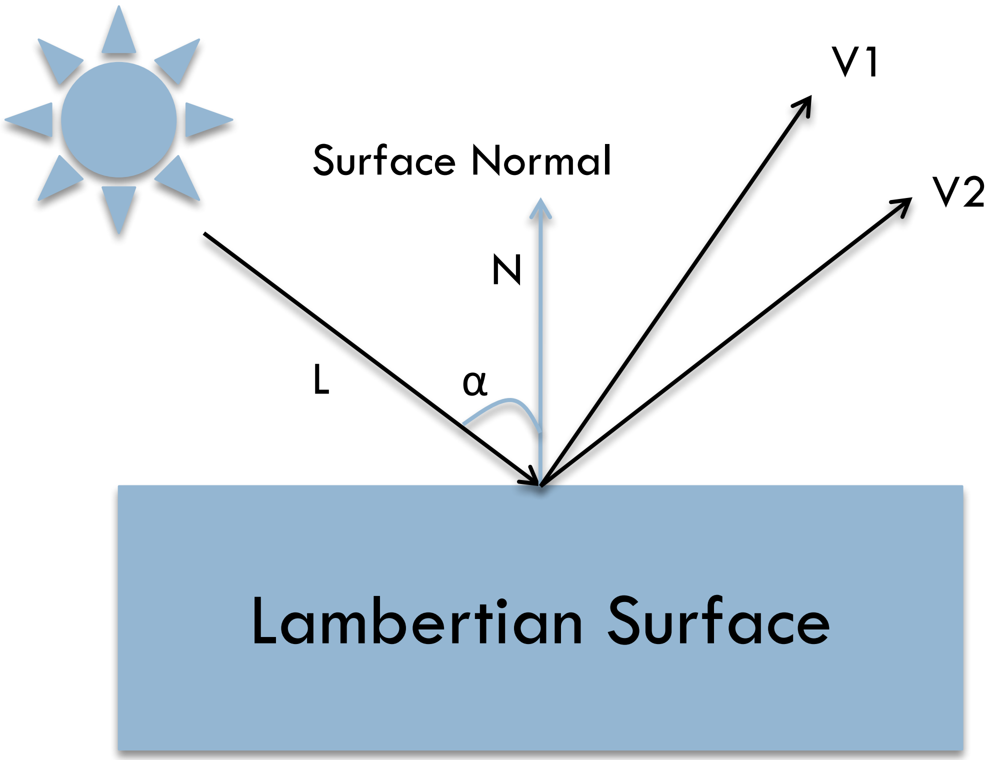
Diffuse Lighting

- Diffuse light
  - Light distributed evenly in all directions, but amount of light depends on orientation of triangles with respect to light source.
  - Different for each triangle



Surface lit with diffuse lighting only

SLIDE REPEAT: Diffuse Lighting



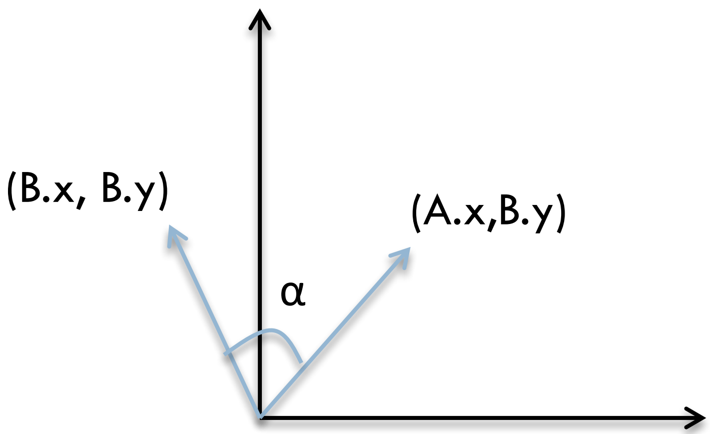
How much light makes it to viewer V1? Viewer V2?

A: cos(α) for both

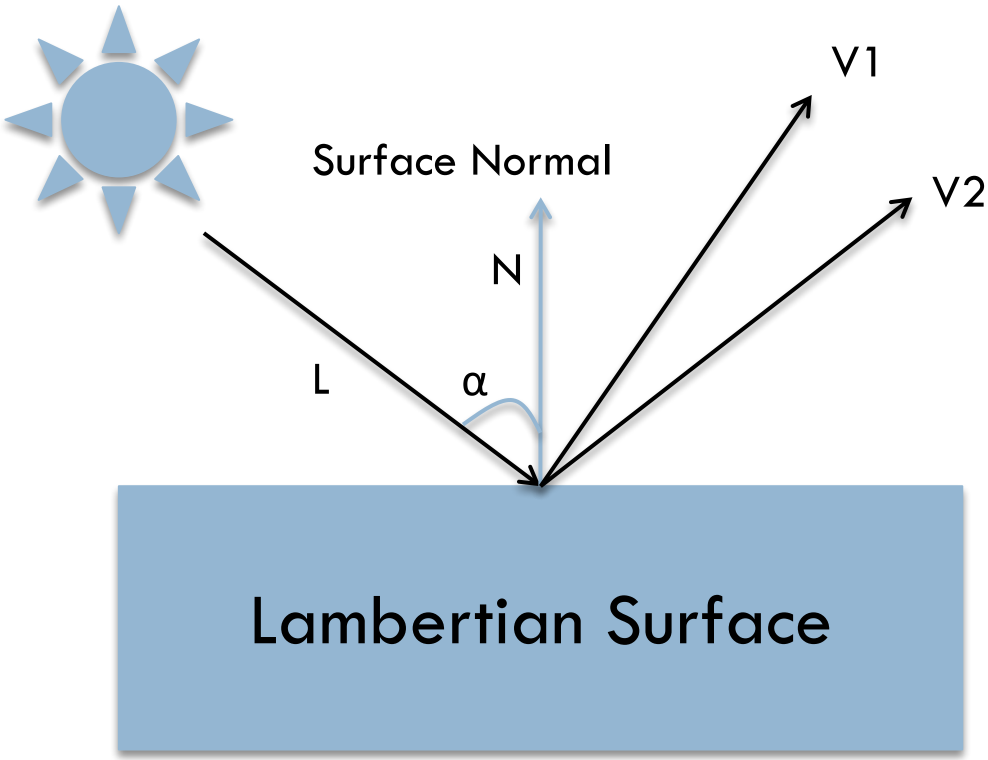
Lambertian surfaces reflect light equally in all directions

What is a dot product?

- $A \cdot B = A.x * B.x + A.y * B.y$
- Physical interpretation:
  - $A \cdot B = \cos(\alpha)/(||A||*||B||)$



Diffuse Lighting

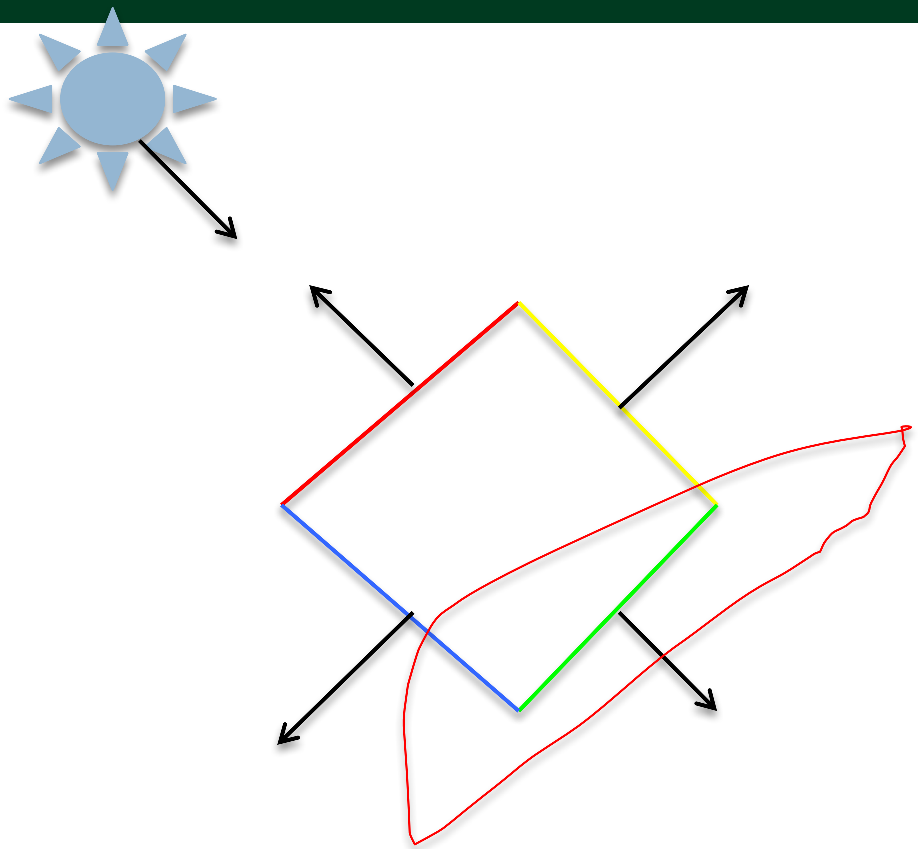


You can calculate the diffuse contribution by taking the dot product of L and N,

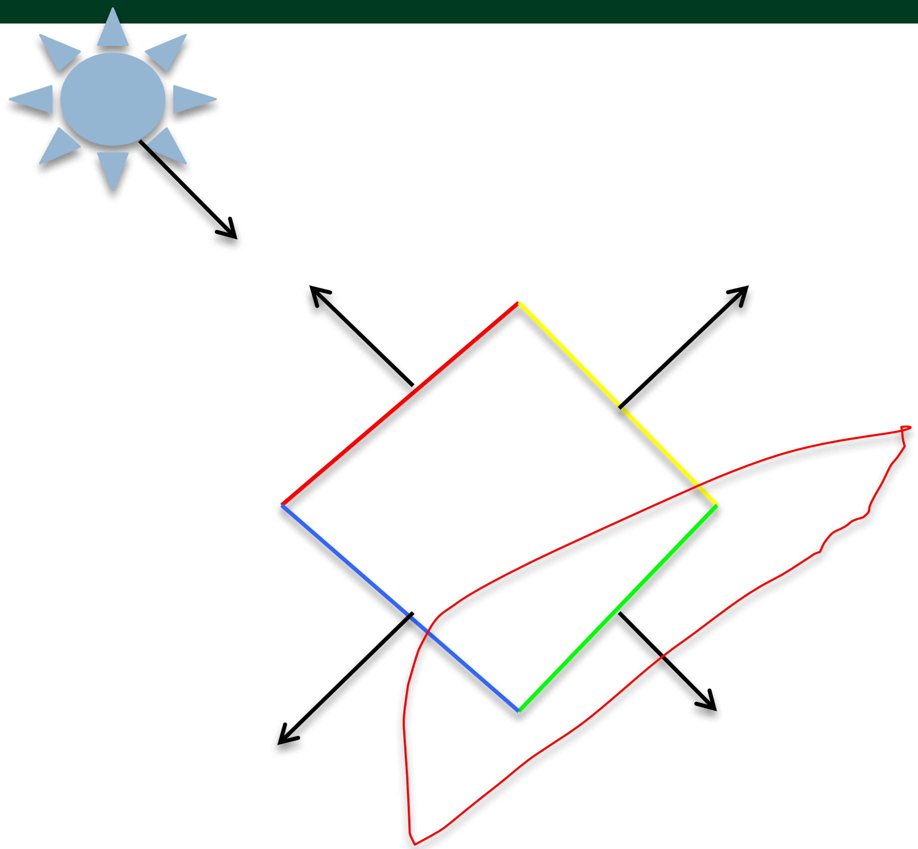
Since $L \cdot N = \cos(\alpha)$

(assuming L and N are normalized)

What about cases where L·N < 0?



What about cases where L·N < 0?



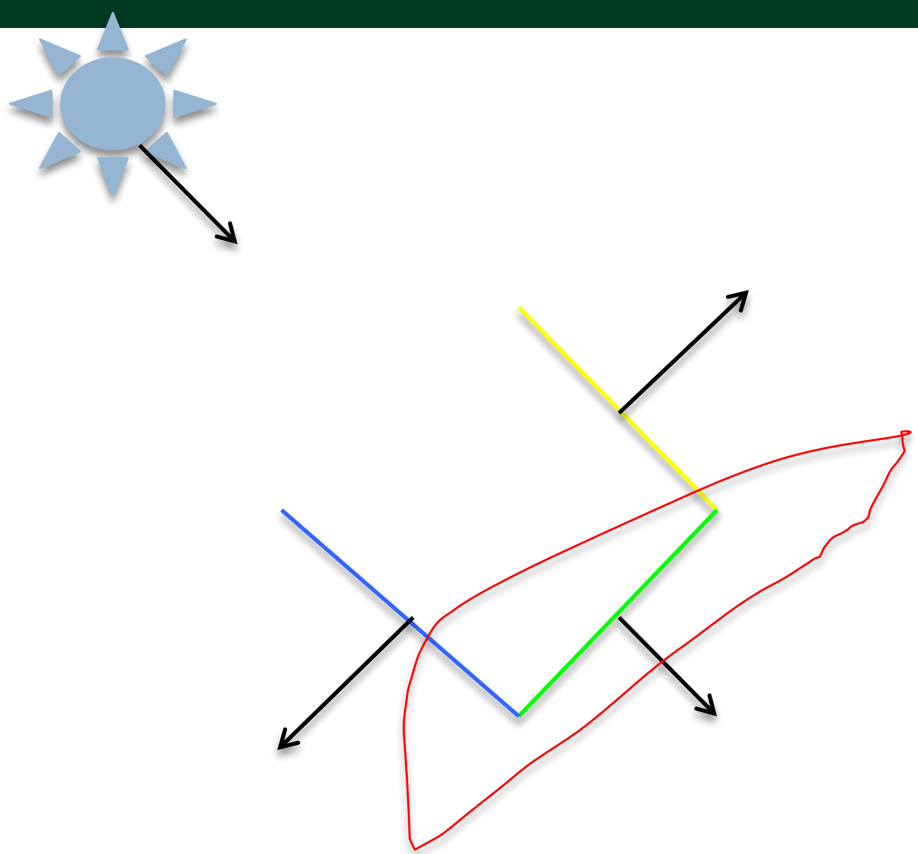
L·N = -1

Non-sensical … takes away light?

Common solution:

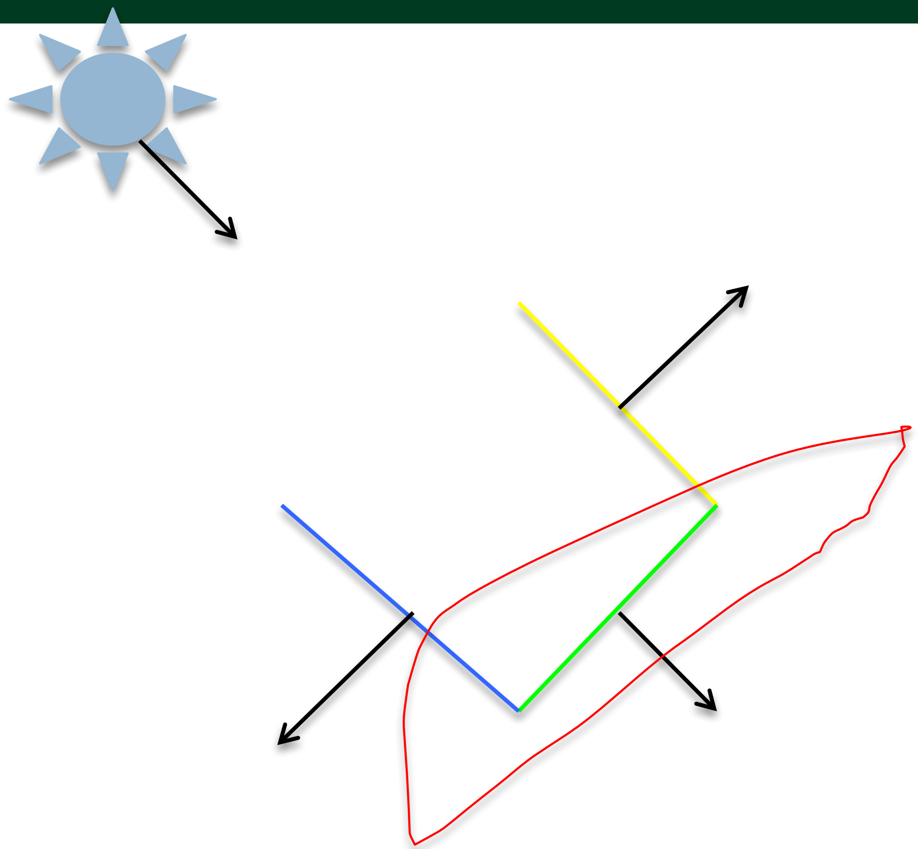
Diffuse light = max(0, L·N)

But wait…



If you have an open surface, then there is a "back face".
The back face has the opposite normal.

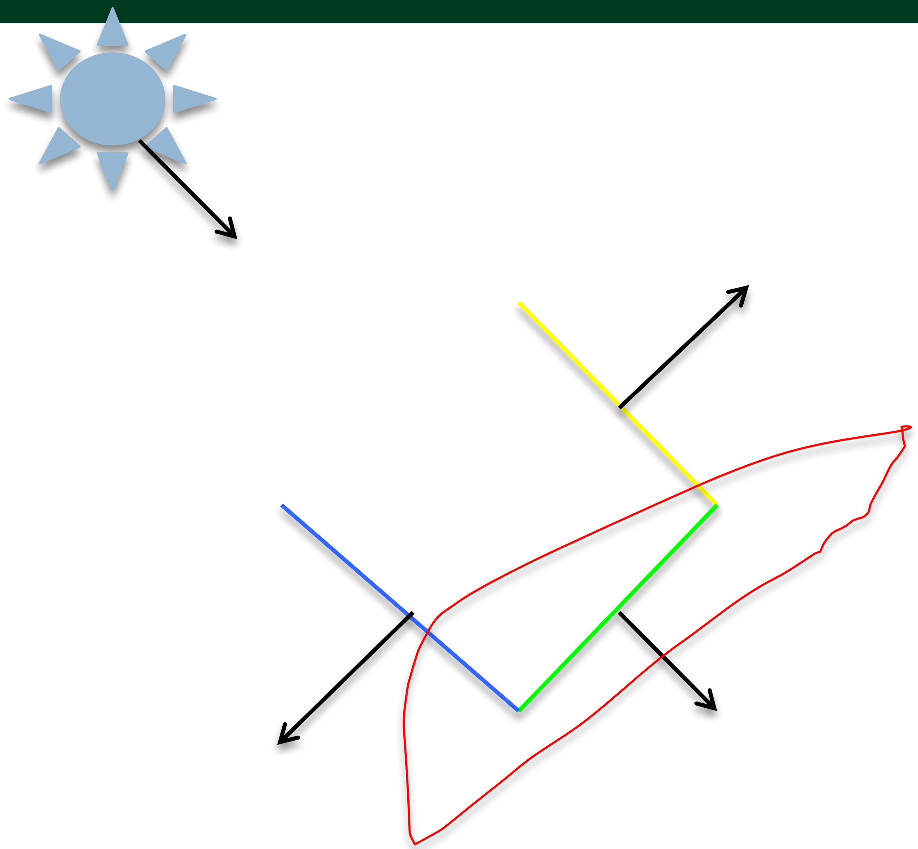
But wait…



If you have an open surface, then there is a “back face”.
The back face has the opposite normal.

How can we deal with this case?

But wait…

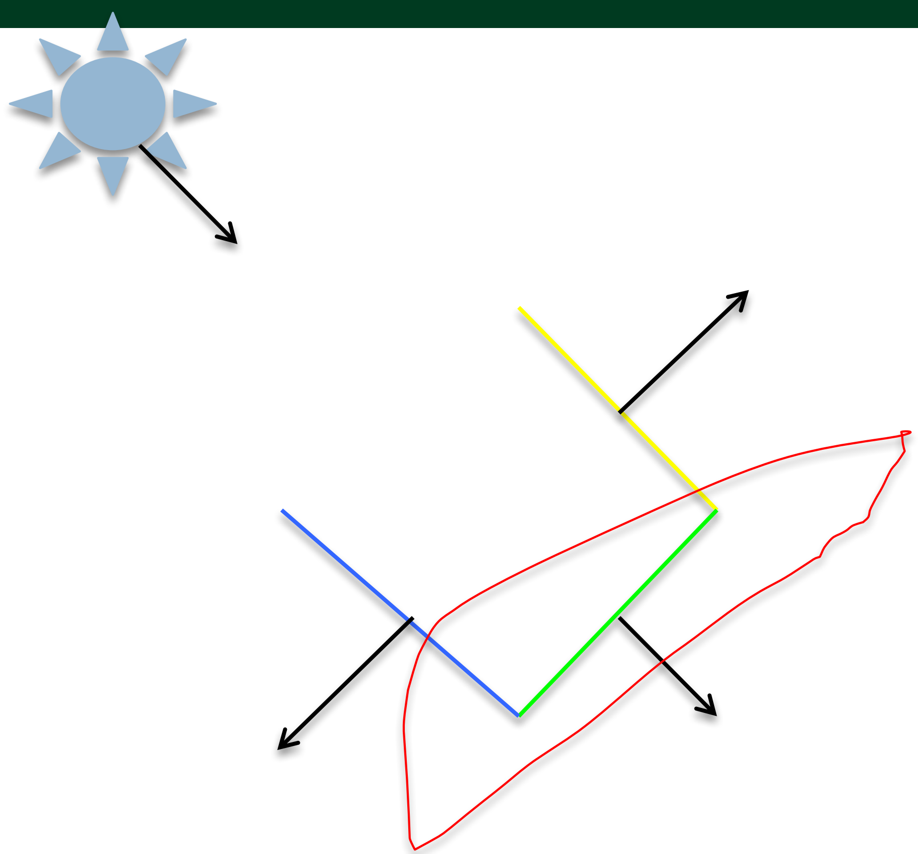


If you have an open surface, then there is a "back face".
The back face has the opposite normal.

How can we deal with this case?

Idea #1: encode all triangles twice, with different normals
Idea #2: modify diffuse lighting model

But wait…



If you have an open surface, then there is a "back face".
The back face has the opposite normal.

How can we deal with this case?

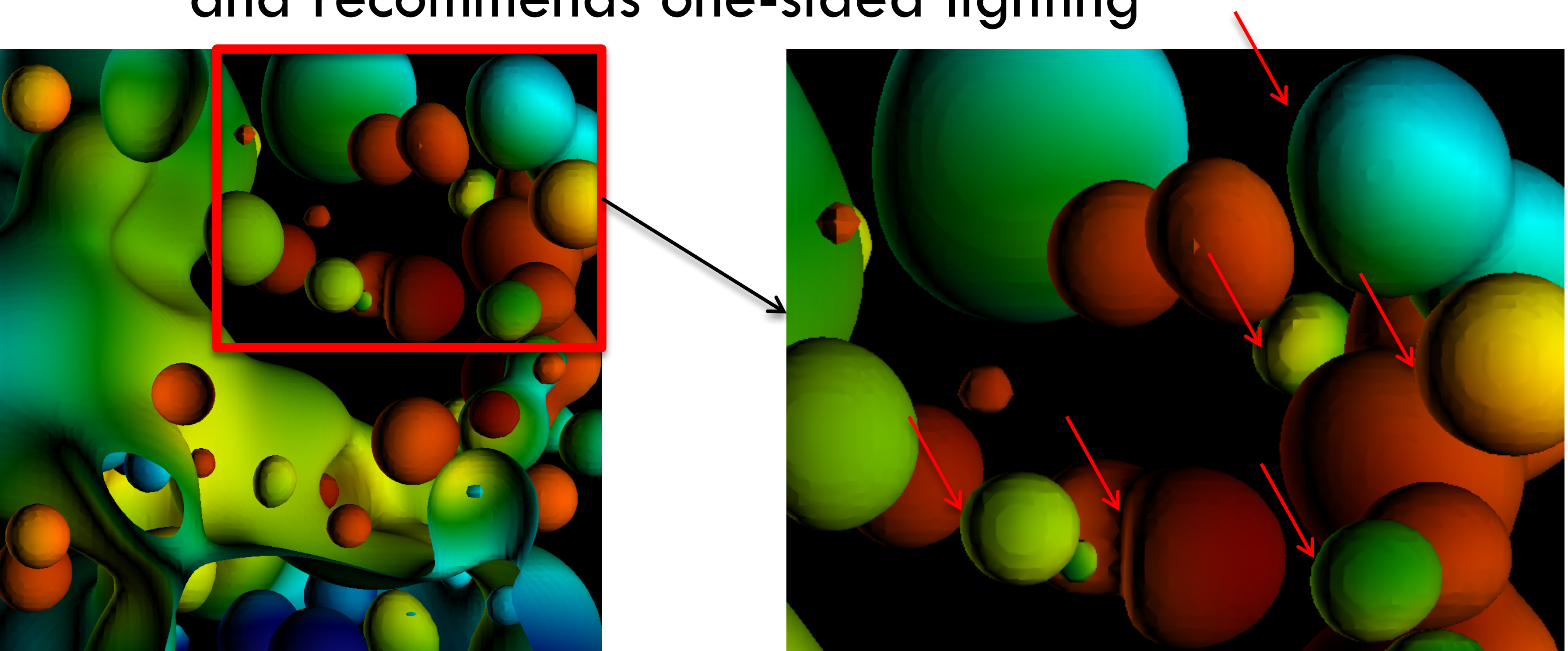
Idea #1: encode all triangles twice, with different normals
Idea #2: modify diffuse lighting model

Diffuse light = abs(L˙N)

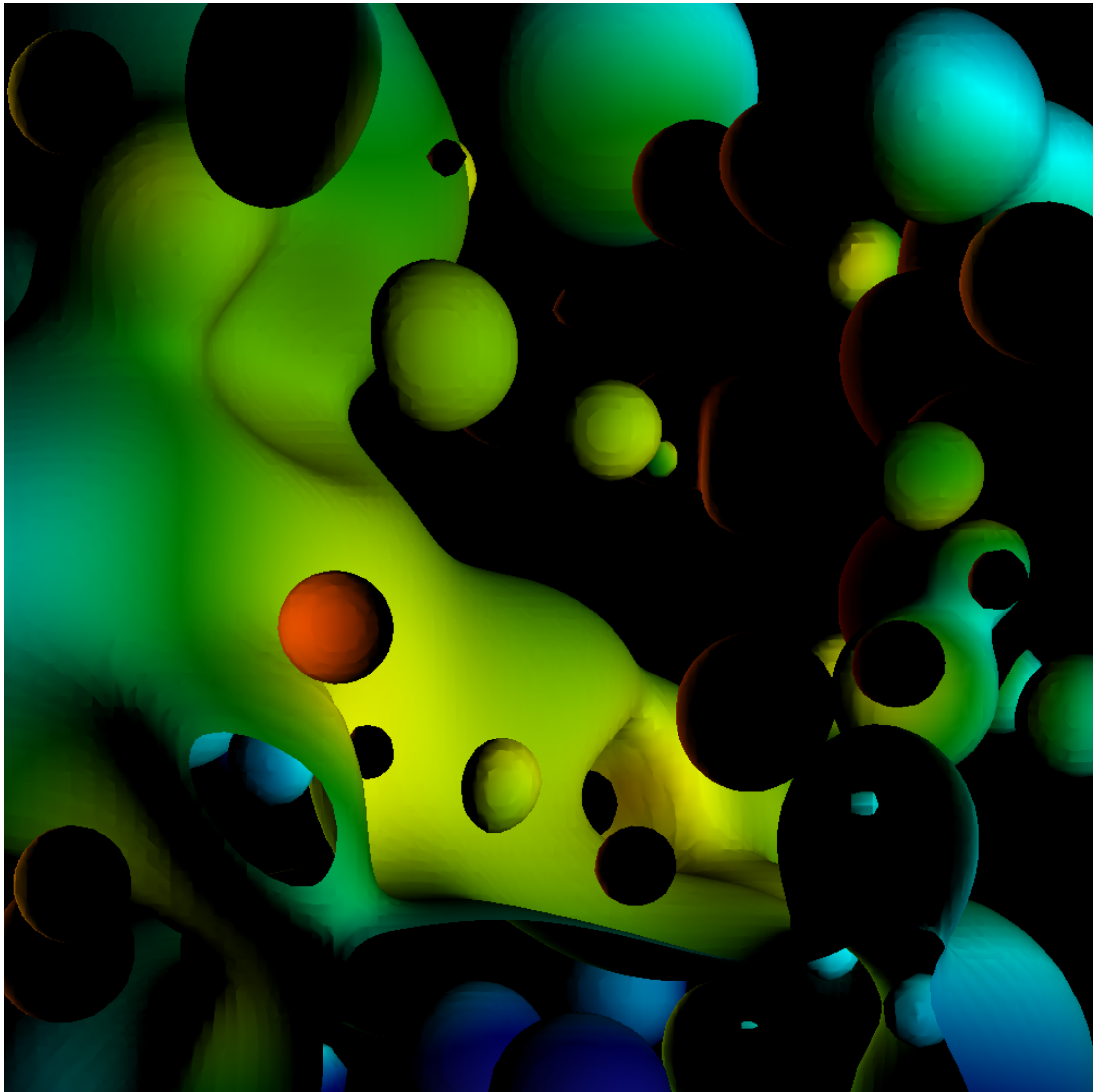
This is called two-sided lighting

Two-sided lighting

- We will use two-sided lighting for project 1F, since we have open surfaces
- Note that Ed Angel book assumes closed surfaces and recommends one-sided lighting

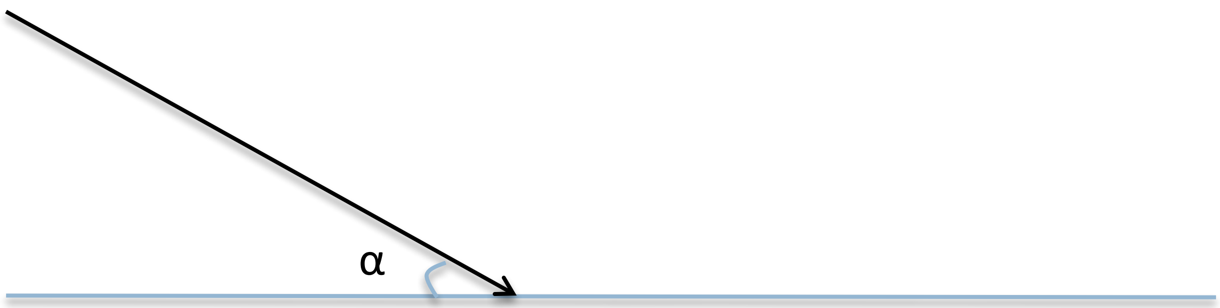


One-sided lighting with open surfaces is disappointing



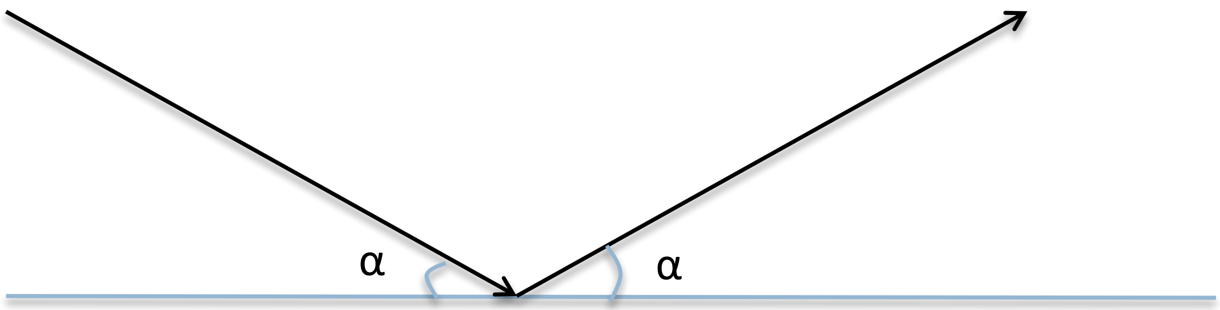
The most valuable thing I learned in Freshman Physics

- "angle in = angle out"

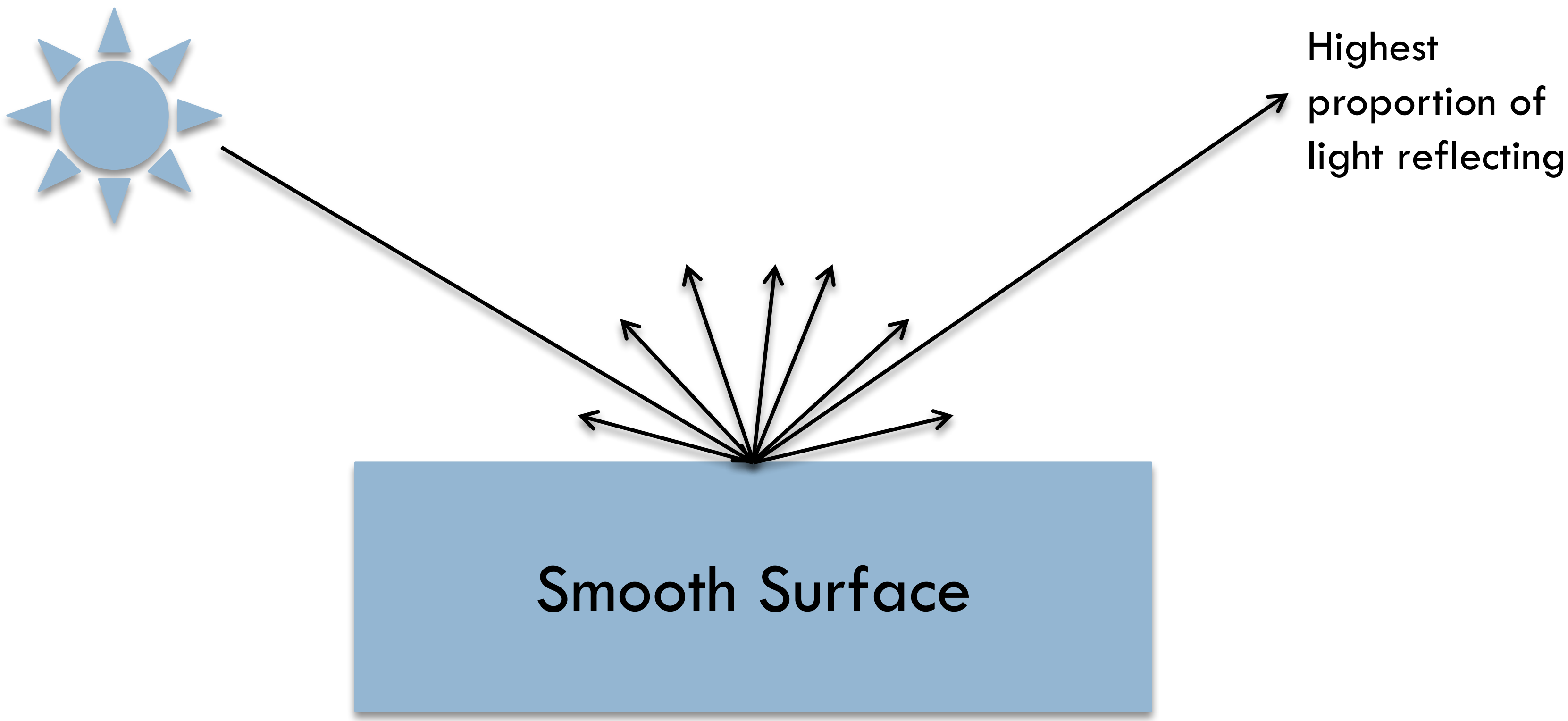


The most valuable thing I learned in Freshman Physics

- "angle in = angle out"

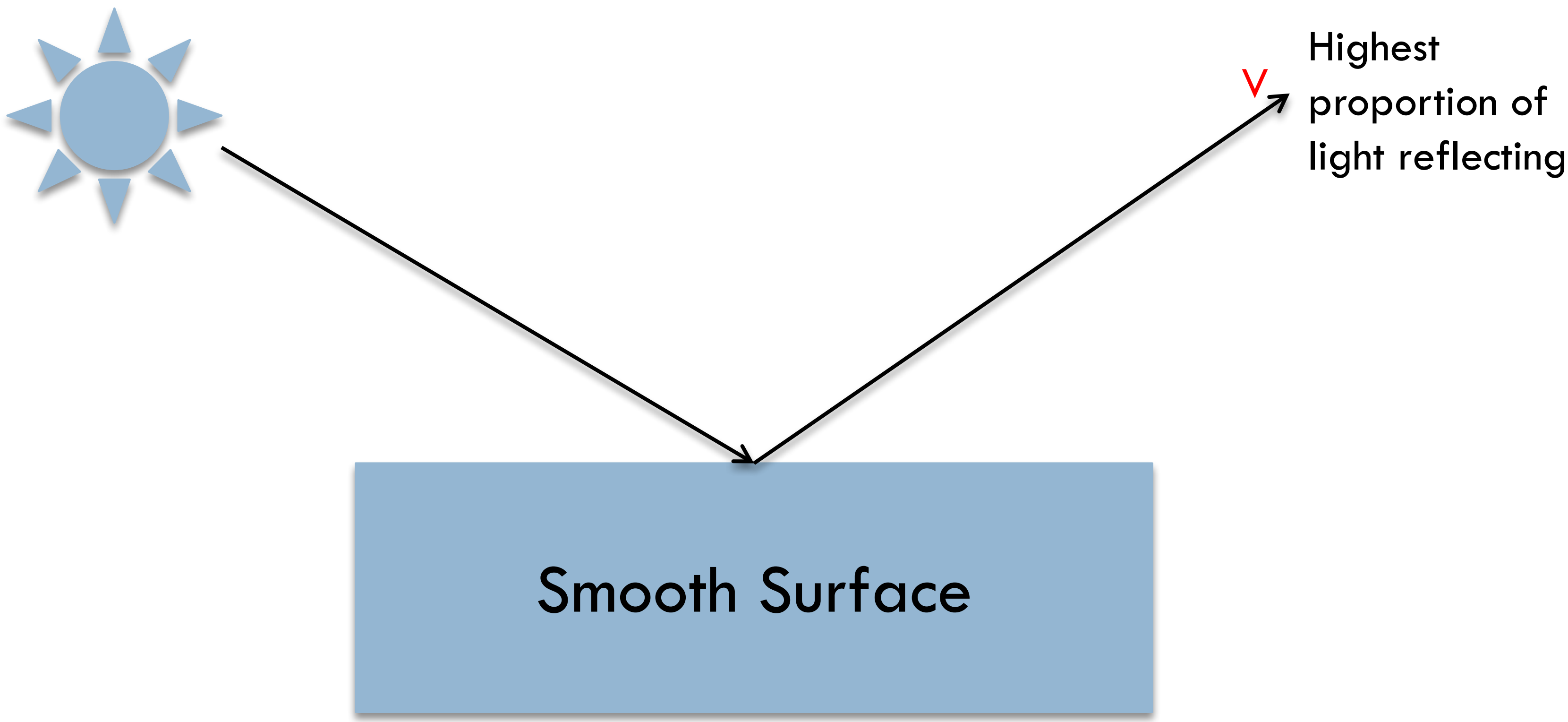


Specular Lighting



Light reflects in all directions.
But the surface is smooth, not Lambertian, so amount of reflected light varies.
So how much light??

How much light reflects with specular lighting?

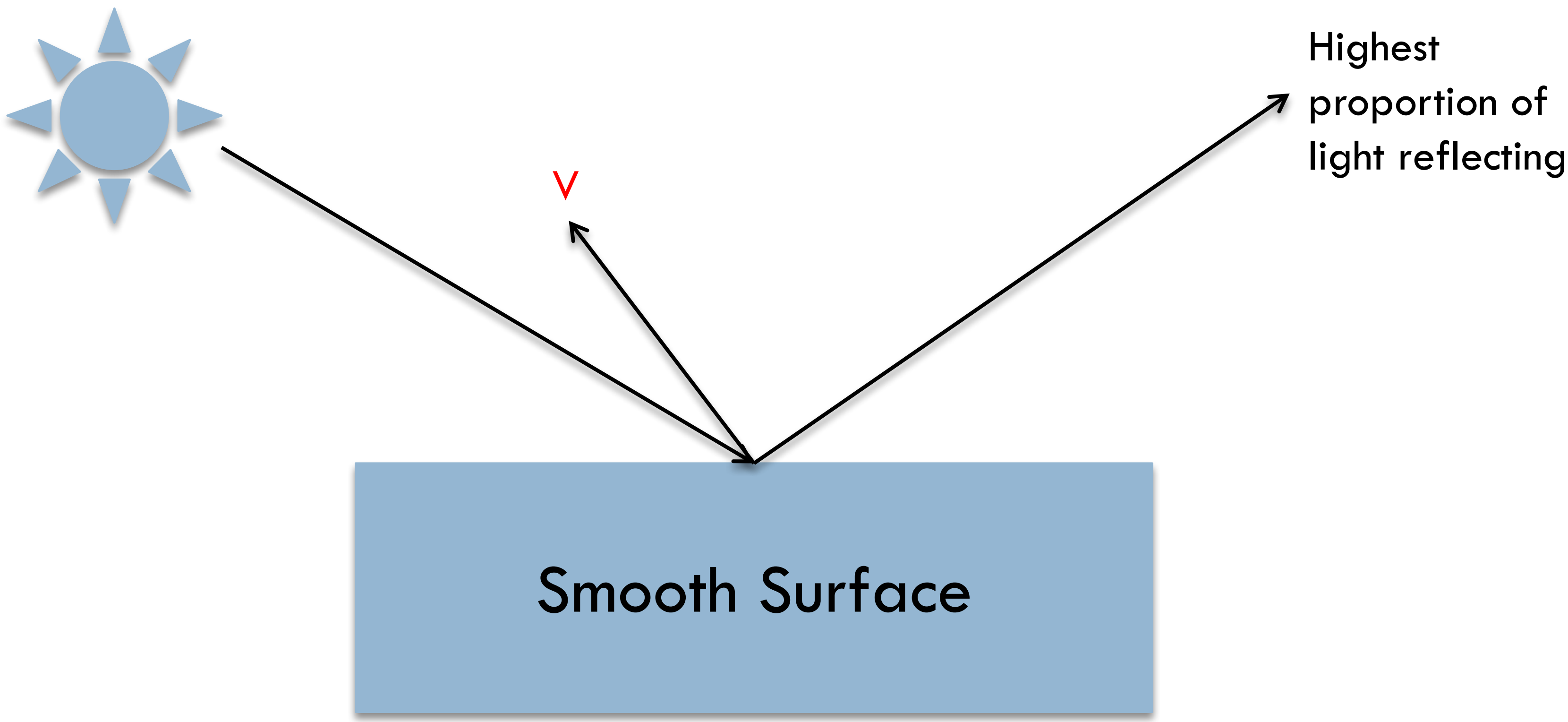


Consider V located along reflection ray.

Answer: most possible

Call this “1”

How much light reflects with specular lighting?

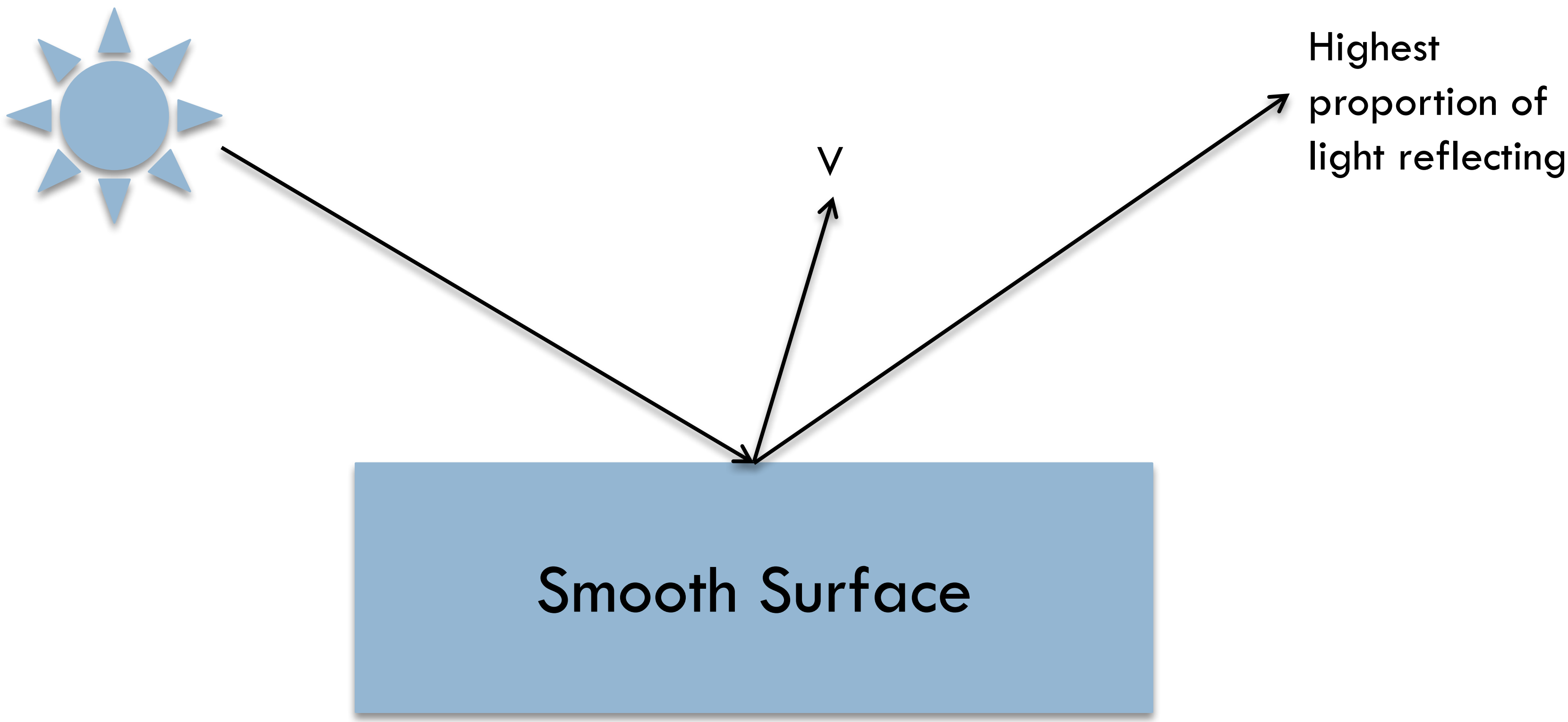


Consider V located along perpendicular ray.

<u>Answer</u>: none of it

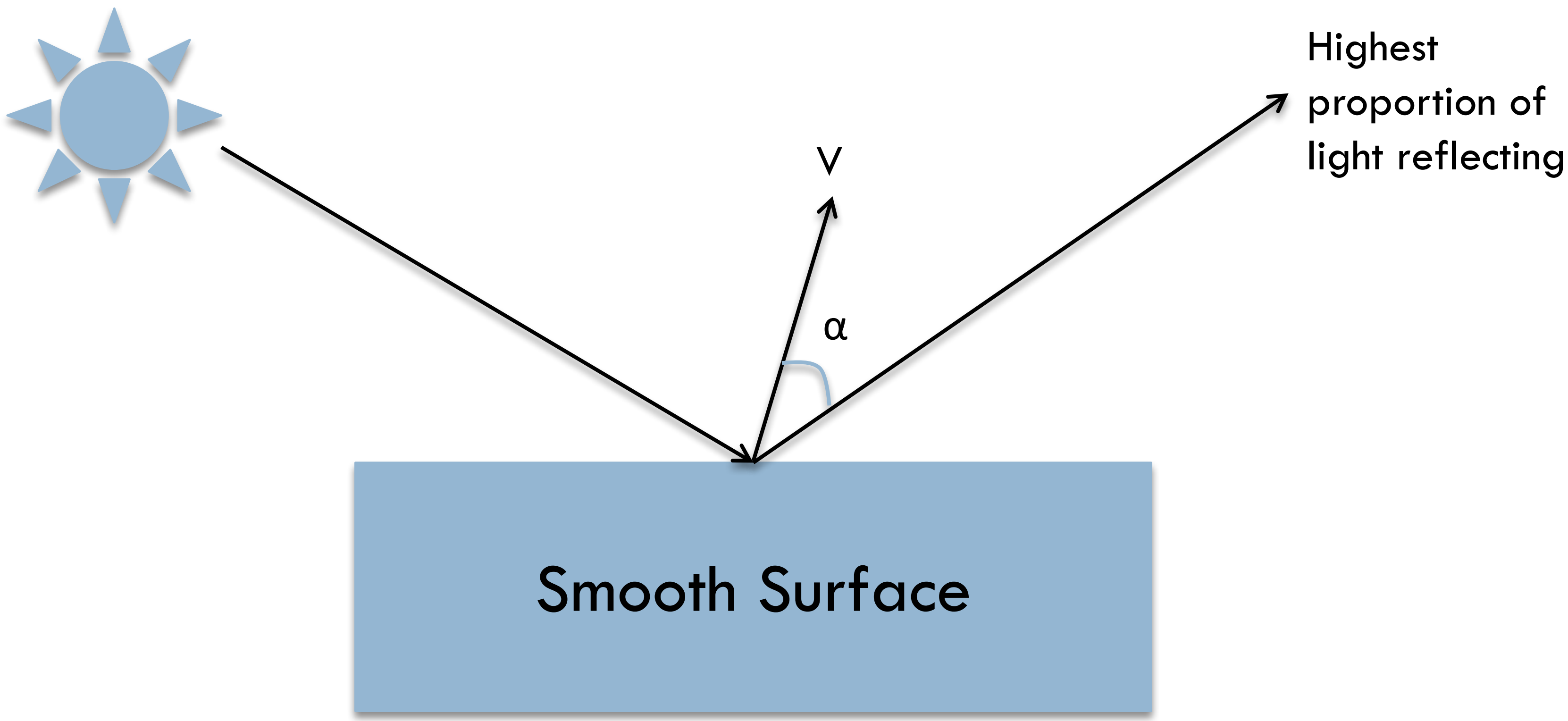
Call this “0”

How much light reflects with specular lighting?



How much light gets to point V?

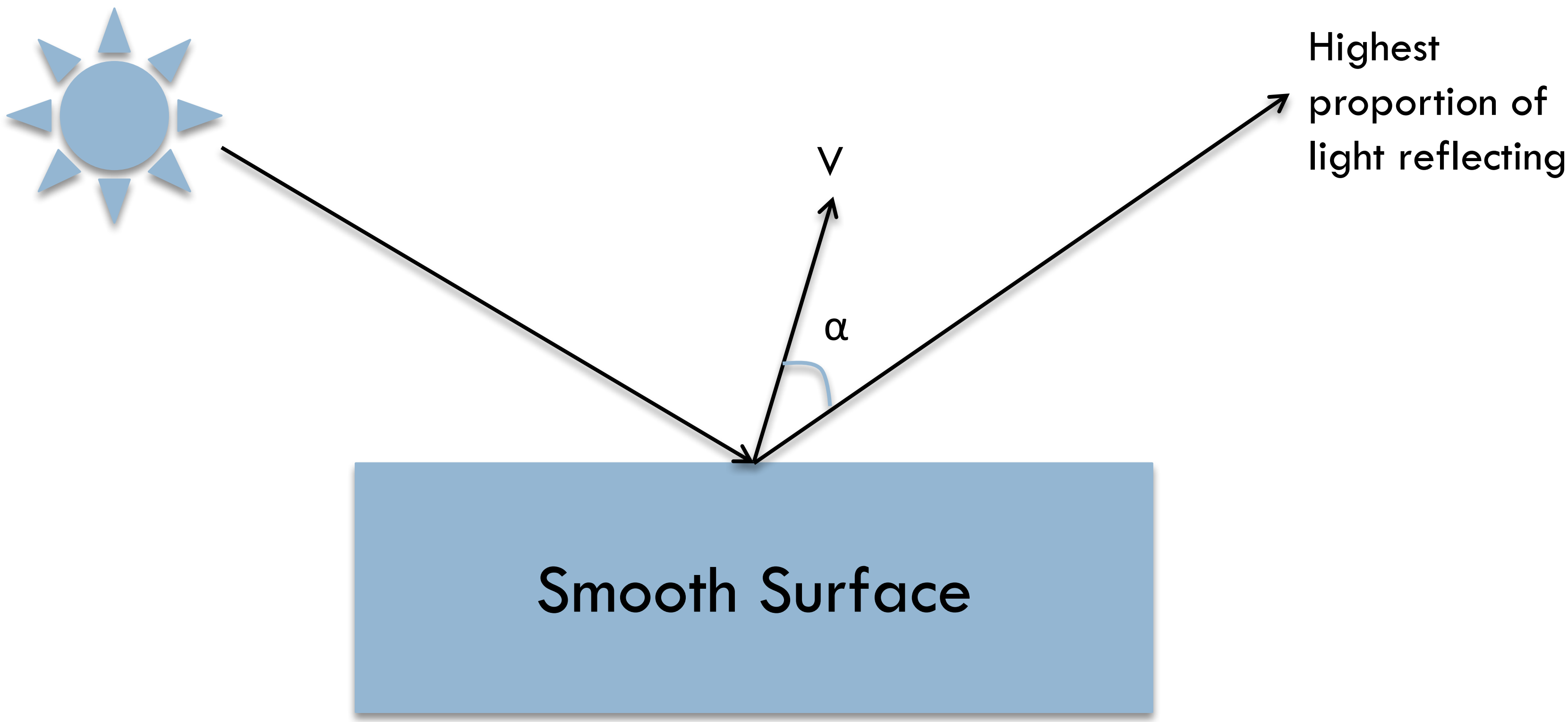
How much light reflects with specular lighting?



How much light gets to point V?

A: proportional to cos(α)

How much light reflects with specular lighting?



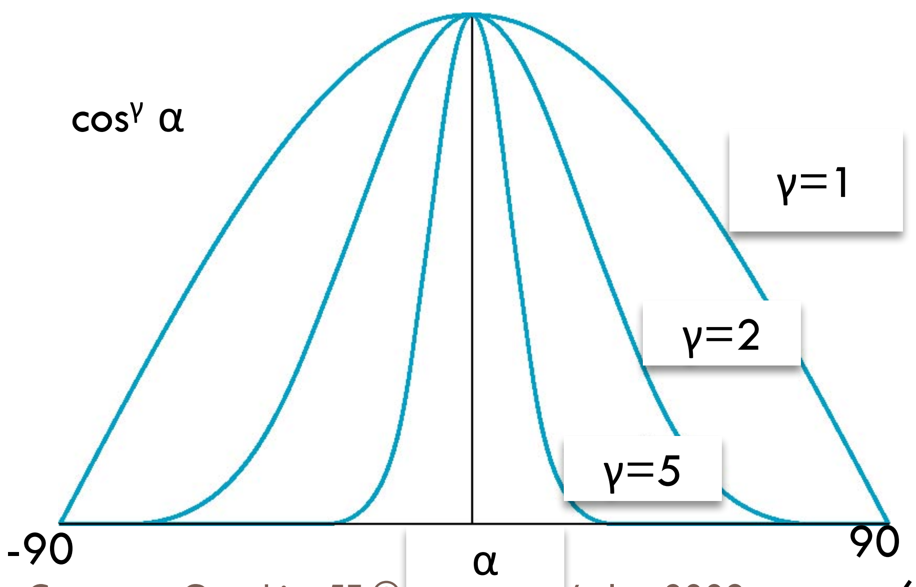
How much light gets to point V?

A: proportional to cos(α)

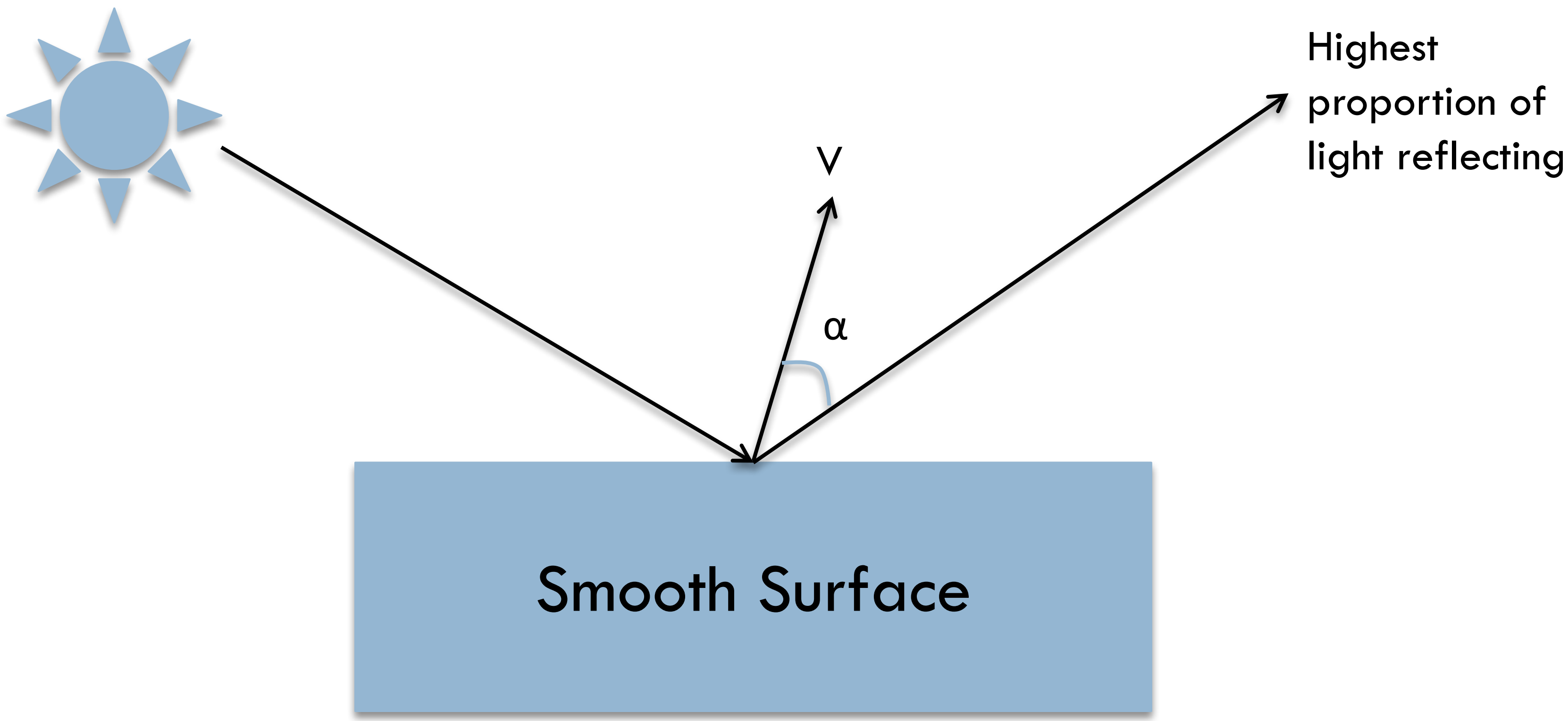
(Shininess strength) * cos(α) ^ (shininess coefficient)

γ: The Shininess Coefficient

- Values of γ between 100 and 200 correspond to metals
- Values between 5 and 10 give surface that look like plastic



How much light reflects with specular lighting?

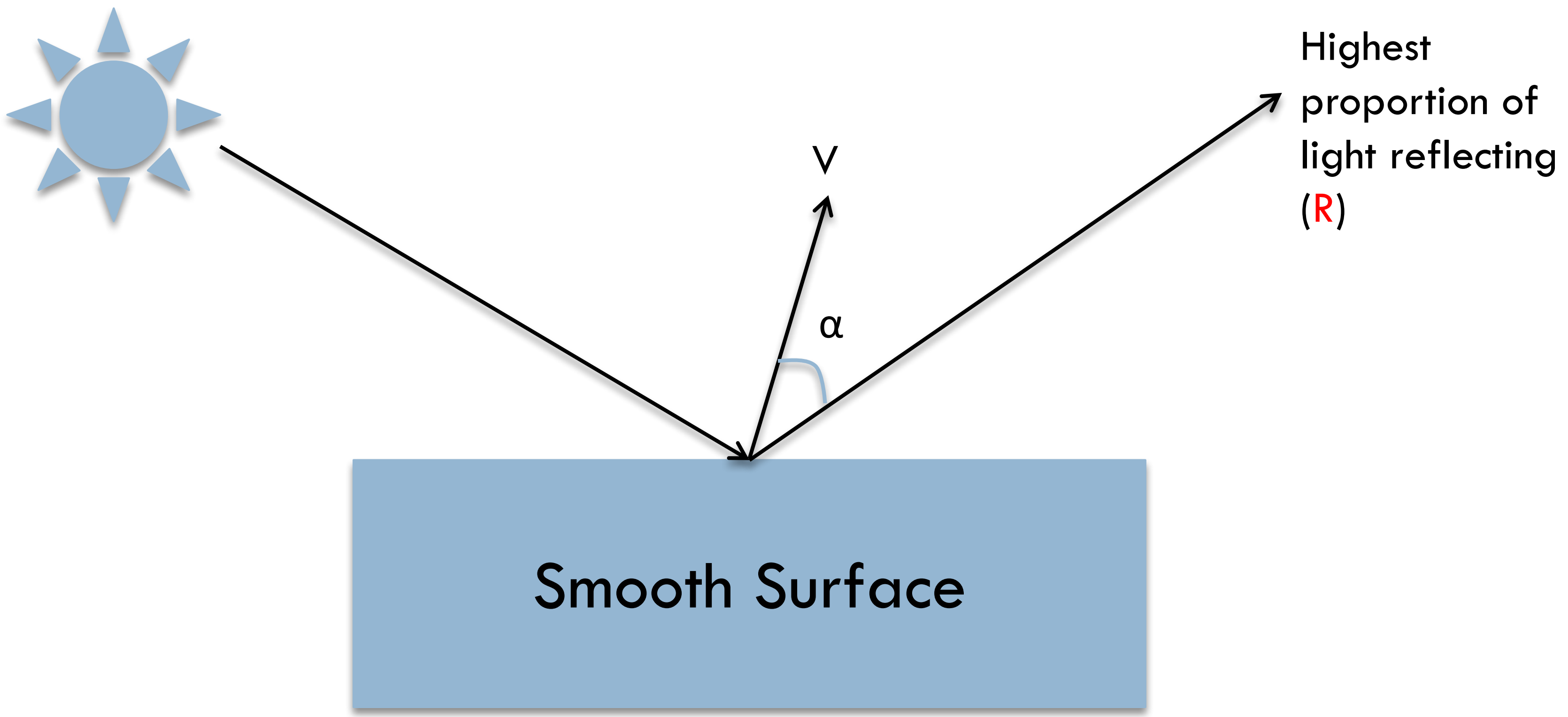


How much light gets to point V?

A: proportional to cos(α)

(Shininess strength) * cos(α) ^ (shininess coefficient)

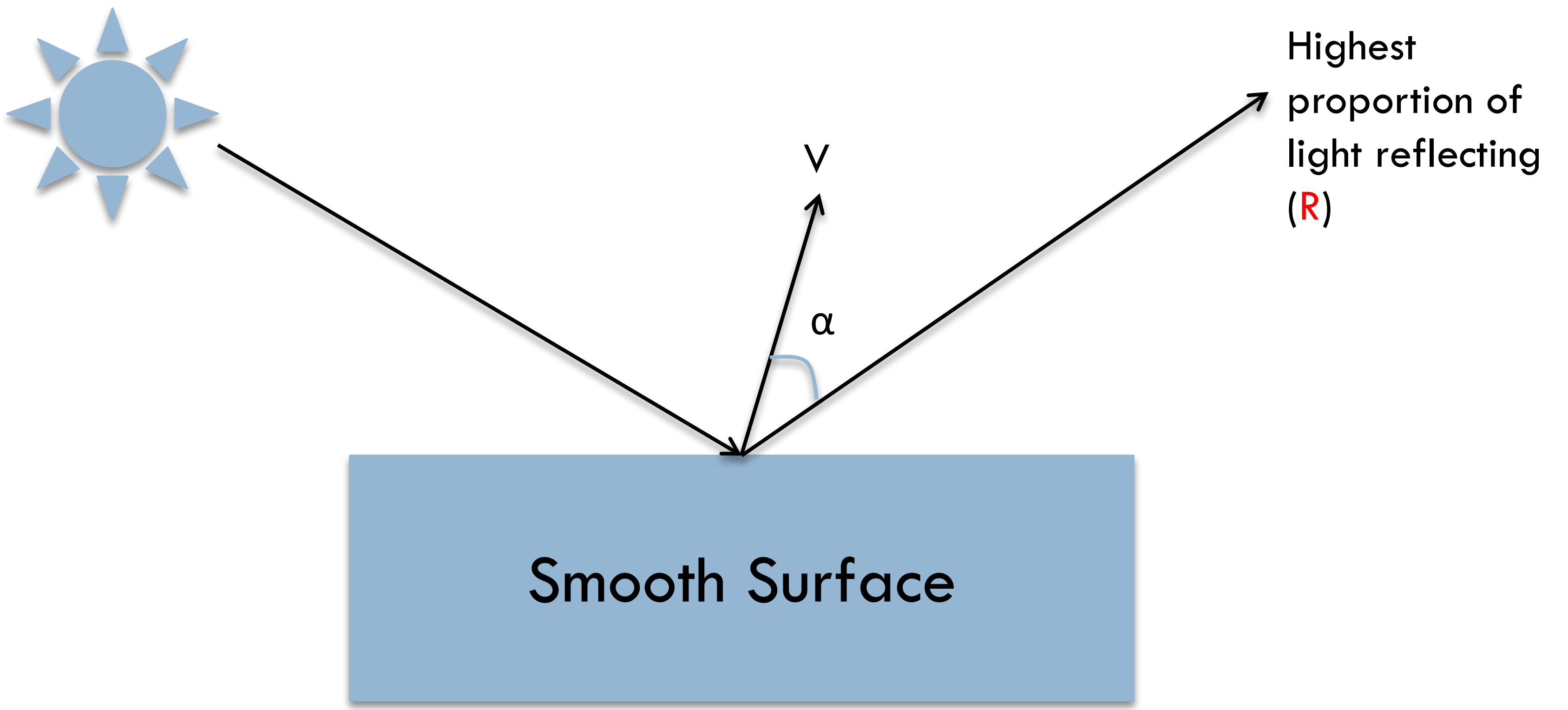
How much light reflects with specular lighting?



Great!

We know that cos(α) is V·R (provided V & R are normalized).

How much light reflects with specular lighting?



Great!

We know that cos(α) is V·R (provided V & R are normalized).

But what is R?

It is a formula: R = 2*(L·N)*N - L

Two-sided lighting

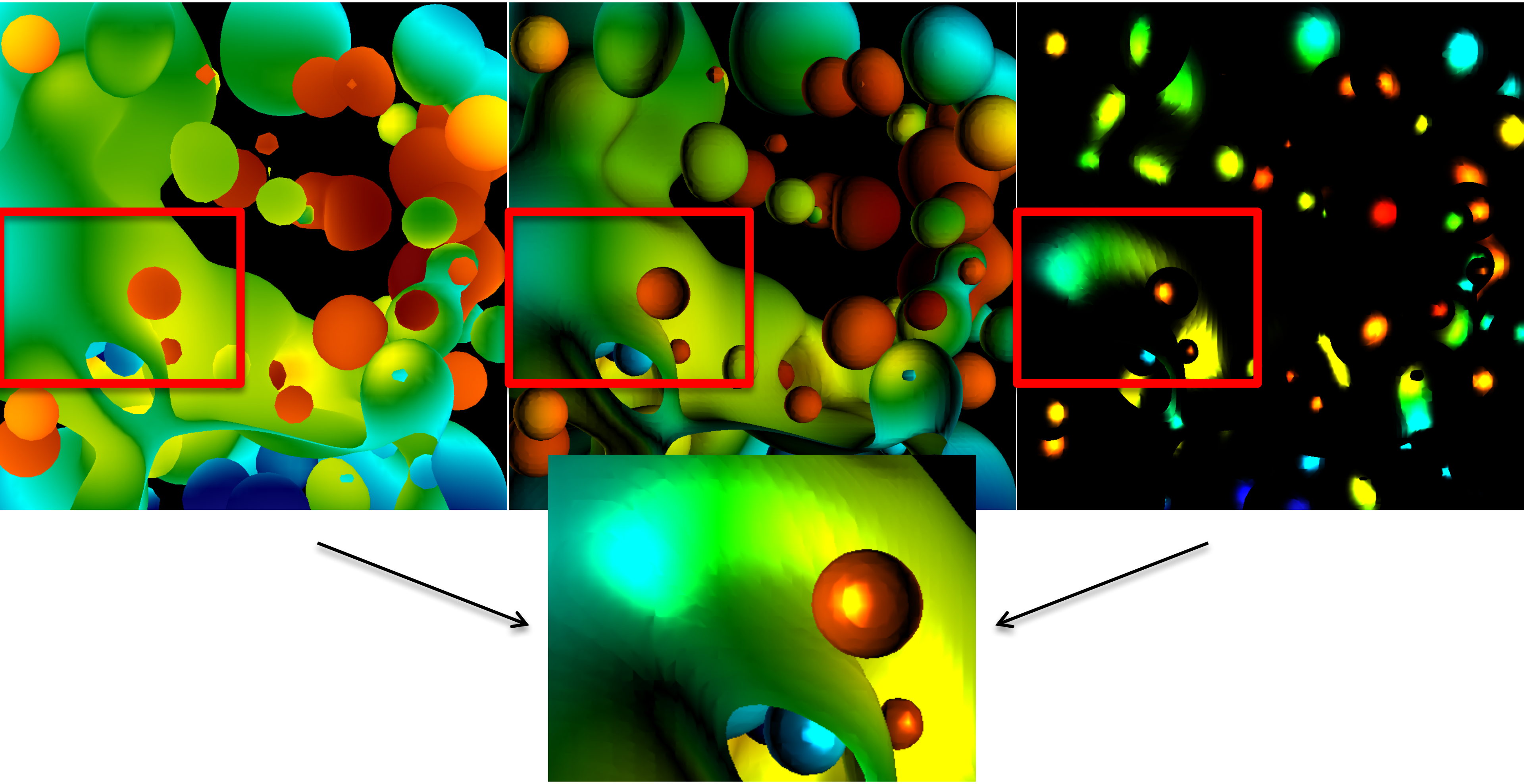
- For specular lighting, we will use two-sided lighting for project 1F
  - Diffuse: abs(L·N)
  - Specular: abs($S*(R \cdot V)^{\gamma}$)

Outline

- Math Basics
- Lighting Basics
- The Phong Model

Phong Model

- Combine three lighting effects: ambient, diffuse, specular



Phong Model

- Simple version: 1 light, with “full intensity” (i.e., don’t add an intensity term)
- Phong model
  - Shading_Amount = K_a + K_d*Diffuse + K_s*Specular
- Signature:
  - double CalculatePhongShading(LightingParameters &, double *viewDirection, double *normal)
  - Will have to calculate viewDirection for each pixel!

Specular Term of Phong Model

- Specular part of Phong: K_s*Specular
- and Specular is: (Shininess strength) * cos(α) ^ (shininess coefficient)
- Putting it all together would be:
  - K_s * (Shininess strength) * cos(α) ^ (shininess coefficient)
- But now we have two multipliers, K_s and (Shininess Strength). Not needed.
- So: just use one. Drop Shininess Strength and only use K_s
  - K_s * cos(α) ^ (shininess coefficient)

Lighting parameters

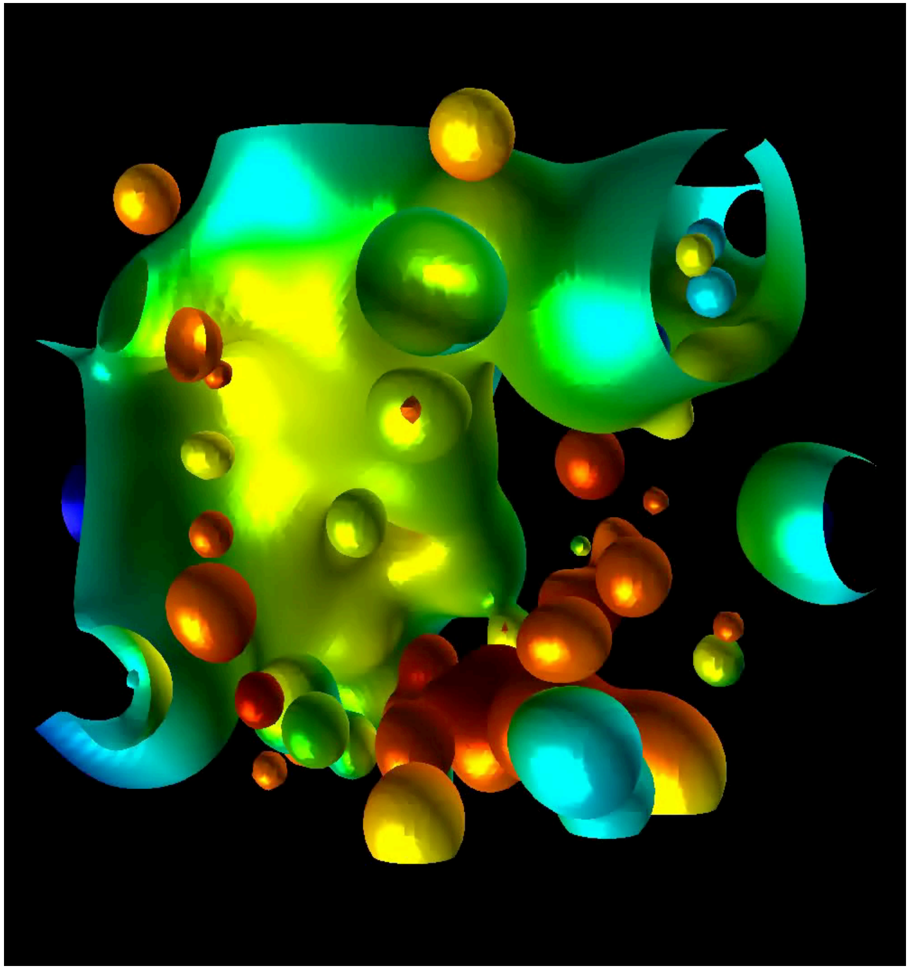
```
struct LightingParameters
{
    LightingParameters(void)
    {
         lightDir[0] = -0.6;
         lightDir[1] = 0;
         lightDir[2] = -0.8;
         Ka = 0.3;
         Kd = 0.7;
         Ks = 2.3;
         alpha = 2.5;
    };


    double lightDir[3]; // The direction of the light source
    double Ka;           // The coefficient for ambient lighting.
    double Kd;           // The coefficient for diffuse lighting.
    double Ks;           // The coefficient for specular lighting.
    double alpha;        // The exponent term for specular lighting.
};

LightingParameters lp;
```

Project #1F (8%), Monday May 3rd

- Goal: add shading, movie
- Extend your project1E code
- Important:
- add #define NORMALS



Changes to data structures

```
class Triangle
{
  public:
      double X[3], Y[3], Z[3];
      double colors[3][3];
      double normals[3][3];
};
```

→reader1e.cxx will not compile (with #define NORMALS) until you make these changes

→reader1e.cxx will initialize normals at each vertex

More comments (1/3)

- This project in a nutshell:
  - Add method called “CalculateShading”.
    - My version of CalculateShading is about ten lines of code.
  - Call CalculateShading for each vertex
  - This is a new field, which you will LERP.
  - Modify RGB calculation to use shading.

More comments (2/3)

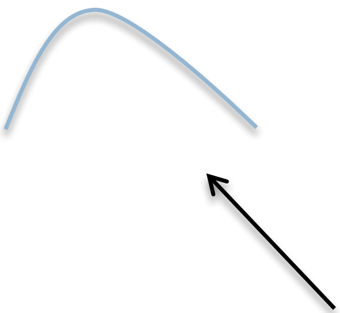
- New: more data to help debug
  - I will make the shading value for each pixel available.
  - I will also make it available for ambient, diffuse, specular.
- Don't forget to do two-sided lighting

More comments (3/3)

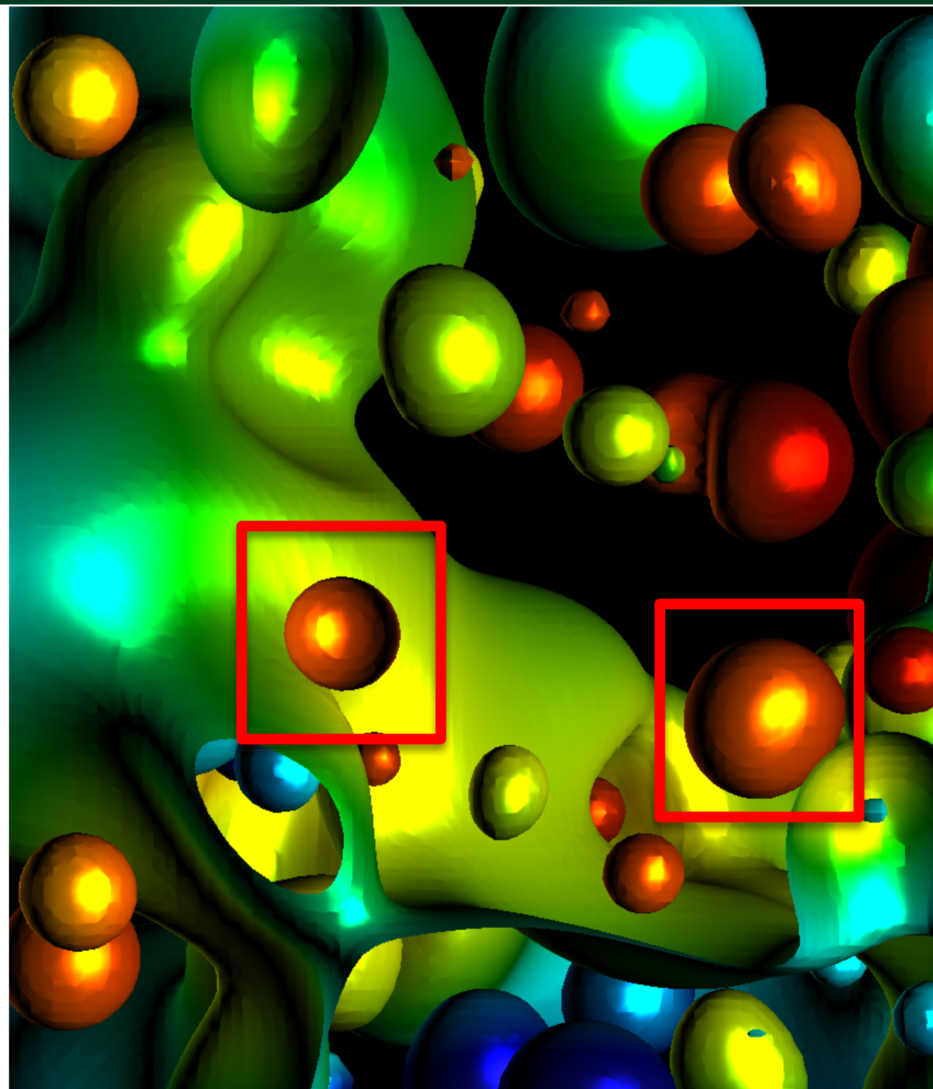
- I haven't said anything about movie encoders

Where Hank spent his debugging time…

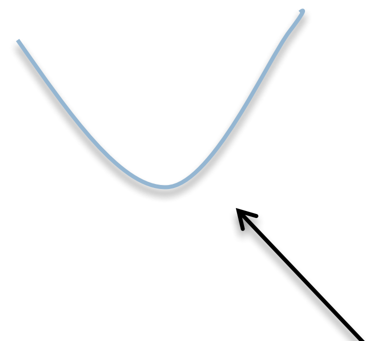
Concave surface



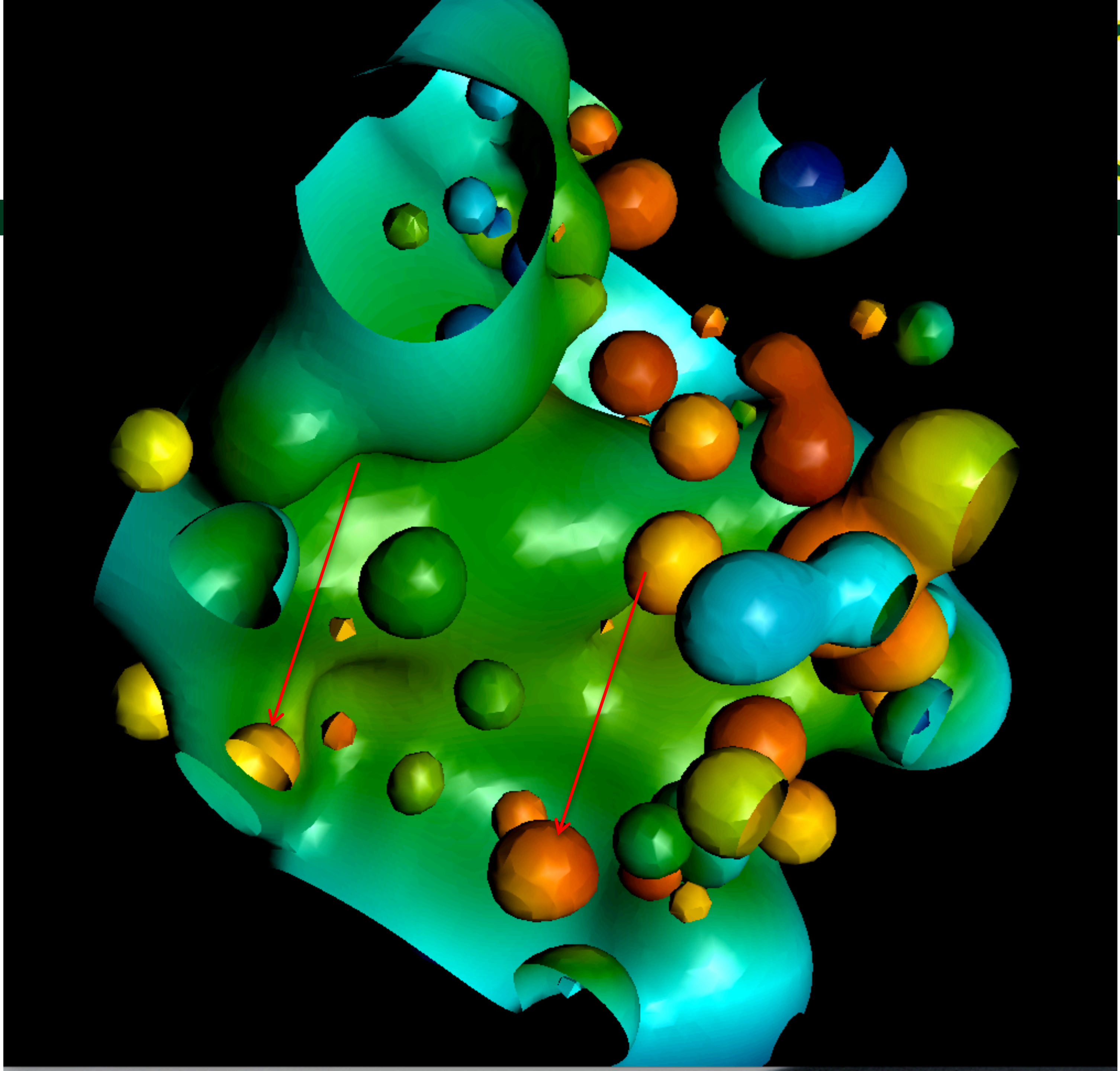
Lighting direction



Convex surface

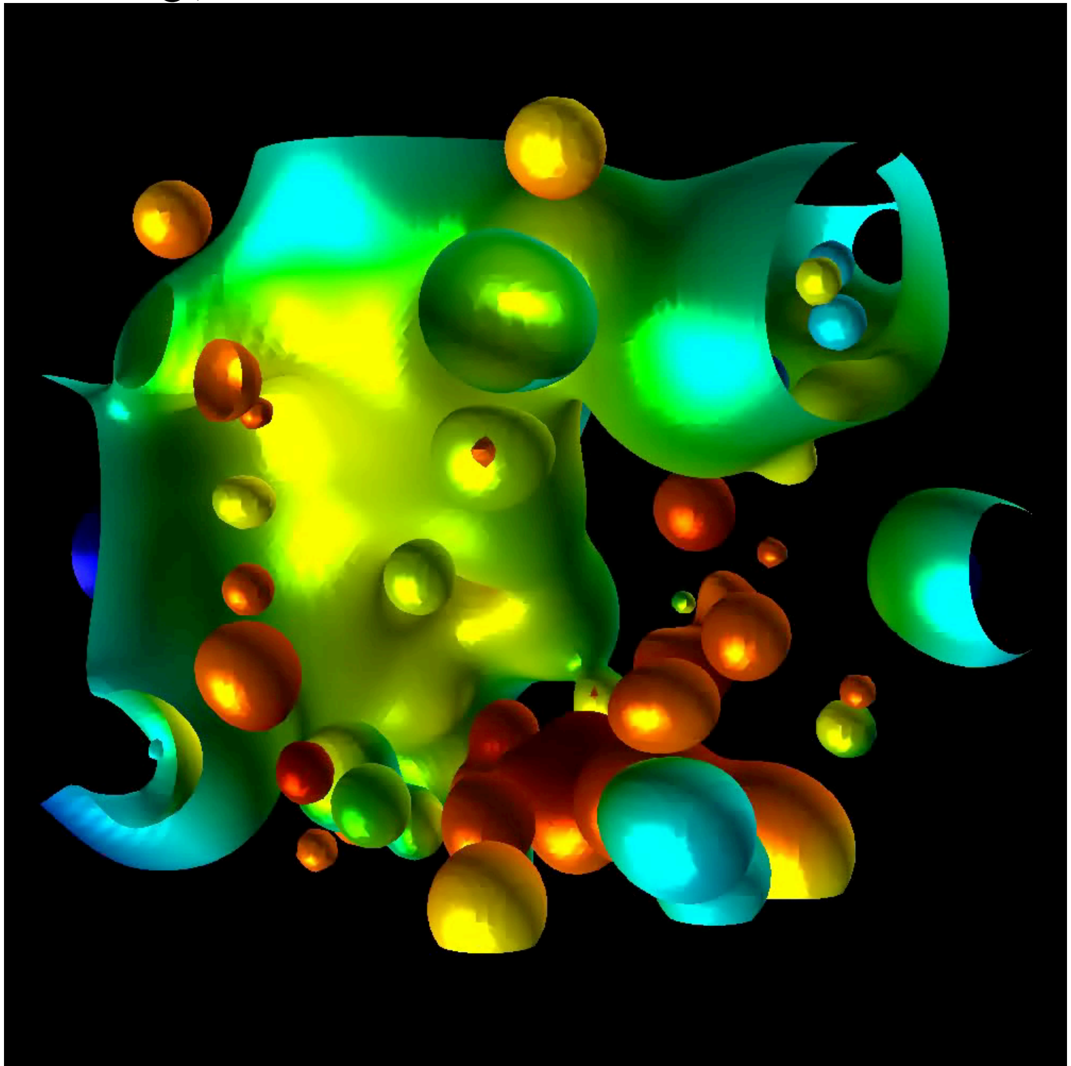


Lighting direction



Project #1F (8%), Due Monday May 3rd

- Goal: add shading, movie



static

- static memory: third kind of memory allocation
  - reserved at compile time
- contrasts with dynamic (heap) and automatic (stack) memory allocations
- accomplished via keyword that modifies variables

There are three distinct usages of statics

static usage #1: persistency within a function

```
fawcett:330 childs$ cat static1.C
#include <stdio.h>

int fibonacci()
{
   static int last2 = 0;
   static int last1 = 1;
   int rv = last1+last2;
   last2 = last1;
   last1 = rv;
   return rv;
}

int main()
{
   int i;
   for (int i = 0 ; i < 10 ; i++)
       printf("%d\n", fibonacci());
}
```

```
fawcett:330 childs$ g++ static1.C
fawcett:330 childs$ ./a.out
1
2
3
5
8
13
21
34
55
89
```

static usage #2: making global variables be local to a file

I have no idea why the static keyword is used in this way.

```
faw

#include <stdio.h>

static int count = 0;

int doubler(int);

int main()
{
   count++;
   doubler(3);
   printf("count is %d\n", count);
}
```

```
fawcett:330 childs$ cat file2.C
static int count = 0;

int doubler(int Y)
{
    count++;
    return 2*Y;
}
```

```
fawcett:330 childs$ g++ -c file2.C
fawcett:330 childs$ g++ file1.o file2.o
fawcett:330 childs$ ./a.out
count is 1
```

static usage #3: making a singleton for a class

```
fawcett:Downloads childs$ cat static3.C
#include <iostream>

using std::cout;
using std::endl;

class MyClass
{
  public:
             MyClass()   { numInstances++; };
   virtual  ~MyClass()   { numInstances--; };

   int       GetNumInstances(void) { return numInstances; };

  private:
   int       numInstances;
};

int main()
{
    MyClass *p = new MyClass[10];
    cout << "Num instances = " << p[0].GetNumInstances() << endl;
    delete [] p;
    cout << "Num instances = " << p[0].GetNumInstances() << endl;
}
fawcett:Downloads childs$ g++ static3.C
fawcett:Downloads childs$ ./a.out
Num instances = 1
Num instances = 0
fawcett:Downloads childs$
```

static usage #3: making a singleton for a class

```
fawcett:Downloads childs$ cat static3.C
#include <iostream>

using std::cout;
using std::endl;

class MyClass
{
  public:
                MyClass()
   virtual     ~MyClass()

   int          GetNumInstar

  private:
   static int        numInstances;
};

i
{
```

```
fawcett:Downloads childs$ g++ static3.C
Undefined symbols:
  "MyClass::numInstances", referenced from:
      MyClass::MyClass()in ccoao8Hf.o
      MyClass::MyClass()in ccoao8Hf.o
      MyClass::GetNumInstances()      in ccoao8Hf.o
      MyClass::~MyClass()in ccoao8Hf.o
      MyClass::~MyClass()in ccoao8Hf.o
      MyClass::~MyClass()in ccoao8Hf.o
      MyClass::~MyClass()in ccoao8Hf.o
ld: symbol(s) not found
collect2: ld returned 1 exit status
```

We have to tell the compiler where to store this static.

```
    delete [] p;
    cout << "Num instances = " << p[0].GetNumInstances() << endl;
}
```

What do we get?

static usage #3: making a singleton for a class

```
fawcett:Downloads childs$ cat static3.C
#include <iostream>

using std::cout;
using std::endl;

class MyClass
{
  public:
             MyClass()   { numInstances++; };
   virtual  ~MyClass()   { numInstances--; };

   int       GetNumInstances(void) { return numInstances; };

  private:
   static int       numInstances;
};

int MyClass::numInstances = 0;

int main()
{
    MyClass *p = new MyClass[10];
    cout << "Num instances = " << p[0].GetNumInstances() << endl;
    delete [] p;
    cout << "Num instances = " << p[0].GetNumInstances() << endl;
}
```

static methods

```
fawcett:Downloads childs$ cat static3.C
#include <iostream>

using std::cout;
using std::endl;

class MyClass
{
  public:
                 MyClass()   { numInstances++; };
   virtual  ~MyClass()   { numInstances--; };

   static int        GetNumInstances(void) { return numInstances; };

  private:
   static int        numInstances;
};

int MyClass::numInstances = 0;

int main()
{
    MyClass *p = new MyClass[10];
    cout << "Num instances = " << MyClass::GetNumInstances() << endl;
    delete [] p;
    cout << "Num instances = " << MyClass::GetNumInstances() << endl;
}
fawcett:Downloads childs$ g++ static3.C
fawcett:Downloads childs$ ./a.out
Num instances = 10
Num instances = 0
```

Static data members and static methods are useful and they are definitely used in practice