Wrapping up Physically-Based Animation and PDEs

Computer Graphics

CMU 15-462/15-662

Model Equations

- **Fundamental behavior of many important PDEs is well-captured by three model linear equations:**

LAPLACE EQUATION ("ELLIPTIC") $\Delta u = 0$

"Laplacian" (more later!)

"what's the smoothest function interpolating the given boundary data"

HEAT EQUATION ("PARABOLIC") $\dot{u} = \Delta u$

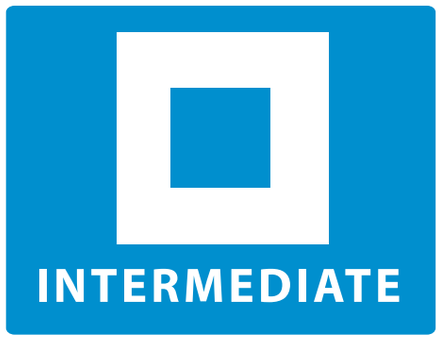
"how does an initial distribution of heat spread out over time?"

WAVE EQUATION ("HYPERBOLIC") $\ddot{u} = \Delta u$

"if you throw a rock into a pond, how does the wavefront evolve over time?"

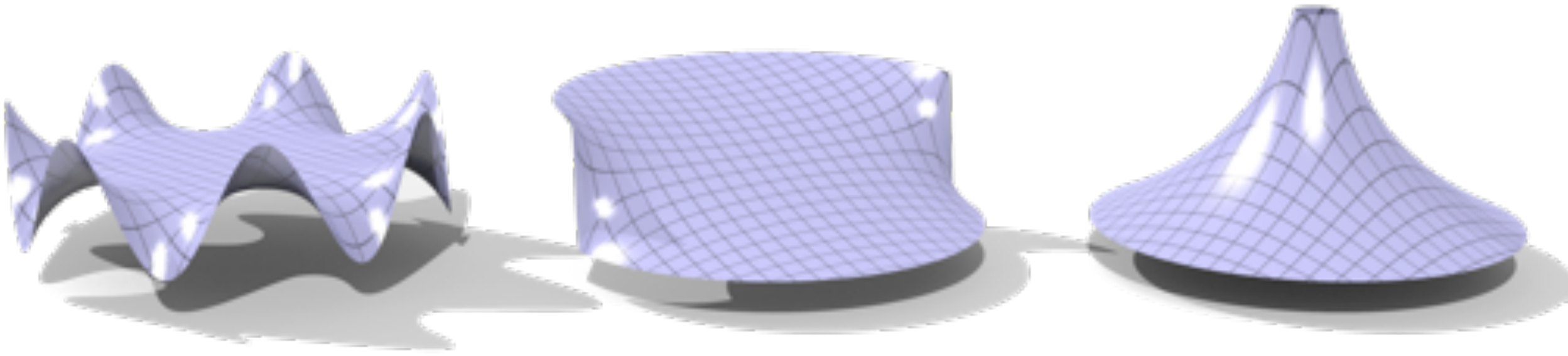
[NONLINEAR + HYPERBOLIC + HIGH-ORDER]

Solve numerically?



Elliptic PDEs / Laplace Equation

- "What's the smoothest function interpolating the given boundary data?"

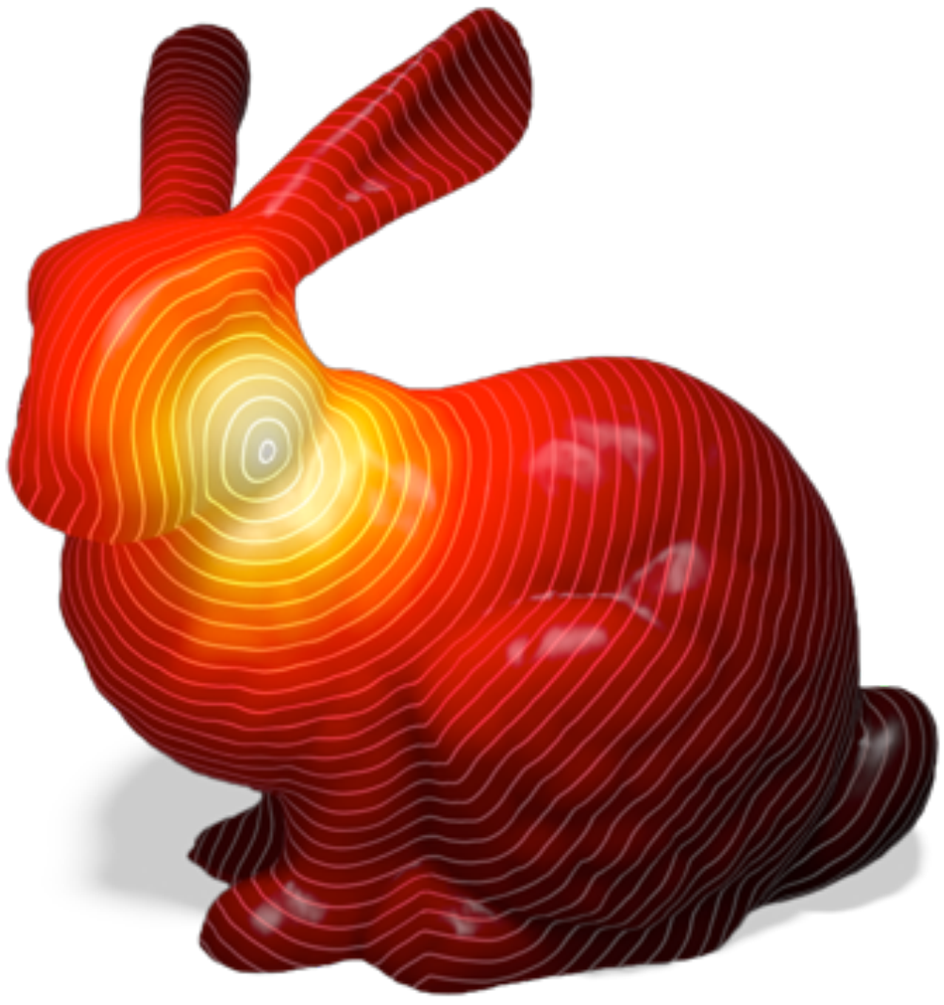


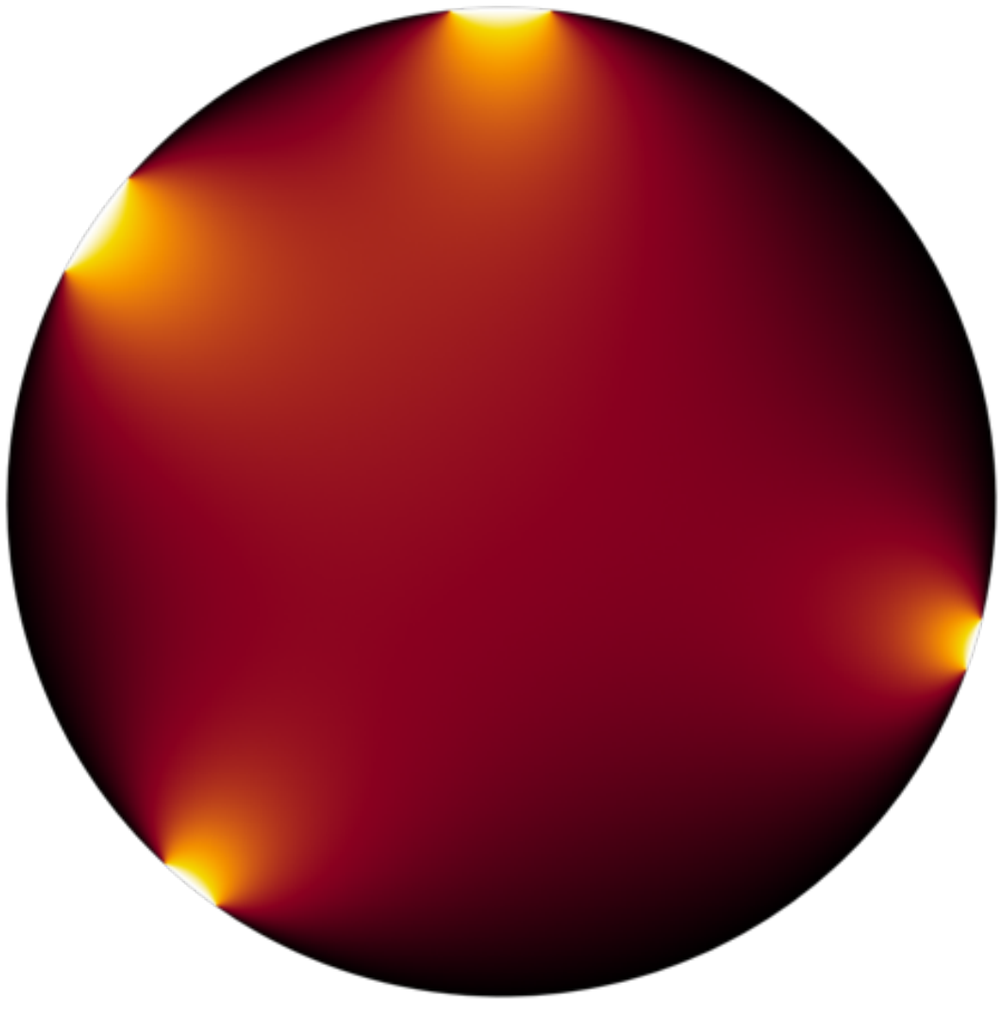
- Conceptually: each value is at the average of its "neighbors"
- Roughly speaking, why is it easier to solve?
- Very robust to errors: just keep averaging with neighbors!

Image from Solomon, Crane, Vouga, "Laplace-Beltrami: The Swiss Army Knife of Geometry Processing"

Parabolic PDEs / Heat Equation

- "How does an initial distribution of heat spread out over time?"





- After a long time, solution is same as Laplace equation!
- Models damping / viscosity in many physical systems

Hyperbolic PDEs / Wave Equation

- "If you throw a rock into a pond, how does the wavefront evolve over time?"

- Errors made at the beginning will persist for a long time! (hard)

Numerical PDEs—Basic Strategy

- Pick PDE formulation
  - Which quantity do we want to solve for?
  - E.g., velocity or vorticity?
- Pick spatial discretization
  - How do we approximate derivatives in space?
- Pick time discretization
  - How do we approximate derivatives in time?
  - When do we evaluate forces?
  - Forward Euler, backward Euler, symplectic Euler, ...
- Finally, we have an update rule
- Repeatedly solve to generate an animation

Richard Courant

The Laplace Operator

- All of our model equations used the Laplace operator
- Different conventions for symbol:

Δ ← same symbol used for "change"

∇^2 ← same symbol used for Hessian!

- Unbelievably important object showing up everywhere across physics, geometry, signal processing, …
- Ok, but what does it mean?
- Differential operator: eats a function, spits out its "2nd derivative"
- What does that mean for a function $u : \mathbb{R}^n \to \mathbb{R}$?

  –divergence of gradient

  –sum of second derivatives

  –deviation from local average

div, grad

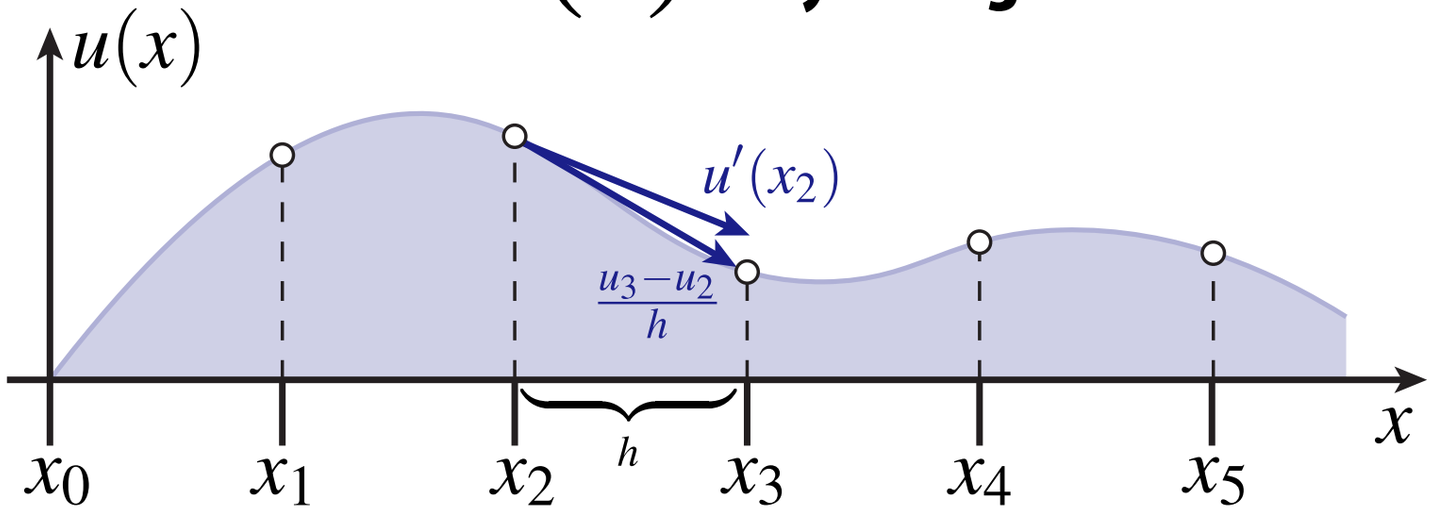
$$\Delta u = \nabla \cdot \nabla u$$

$$\Delta u = \frac{\partial u^2}{\partial x_1^2} + \cdots + \frac{\partial u^2}{\partial x_n^2}$$

For more intuition about the Laplacian: https://youtu.be/oEq9R0l9Umk

Discretizing the First Derivative

- To solve any PDE, need to approximate spatial derivatives (e.g., Laplacian)
- Suppose we know a function $u(x)$ only at regular intervals h



- Q: How can we approximate the first derivative of u?
- A: Recall definition of a derivative in terms of limits:

$$u'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Can hence get an approximation using known values:

$$u'(x_i) \approx \frac{u_{i+1} - u_i}{h}$$

- Approximation gets better for finer grid (smaller h)

Discretizing the Second Derivative

- **Q: How can we get an approximation of the second derivative?**
- **A: One idea*: approximate the first derivative of the approximate first derivative!**

$$u''(x_i) \approx \frac{u_i' - u_{i-1}'}{h} \approx \frac{\left(\frac{u_{i+1} - u_i}{h}\right) - \left(\frac{u_i - u_{i-1}}{h}\right)}{h} =$$

$$\boxed{\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}}$$

- **In general, this approach of approximating derivatives with differences is the "finite difference" approach to PDEs**
- **Not the only way! But works well on regular grids.**

*Can show this is also a reasonable thing to do, using Taylor series

Numerically Solving the Laplace Equation

- Want to solve $\Delta u = 0$

	$u_{i,j+1}$	
$u_{i-1,j}$	$u_{i,j}$	$u_{i+1,j}$
	$u_{i,j-1}$	

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = 0$$

$$\iff u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right)$$

- If u is a solution, then each value must be the average of the neighboring values (u is a "harmonic function")
- How do we solve this?
- One idea: keep averaging with neighbors! ("Jacobi method")

Aside: PDEs and Linear Equations

- How can we turn our Laplace equation into a linear solve?
- Have a bunch of equations of the form

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = 0$$

- On a 4x4 grid, assign each cell $u_{i,j}$ a unique index 1, …, 16

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

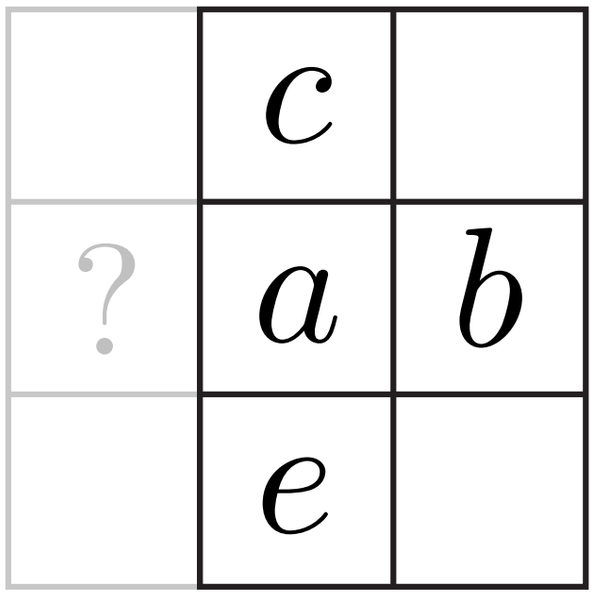
$$\begin{bmatrix}
-4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & -4
\end{bmatrix}
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \\ u_{16} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Compute solution by calling sparse linear solver (SuiteSparse, Eigen, …)
- Q: By the way, what's wrong with our problem setup here? :-)

*assuming neighbors wrap around left/right and top/bottom

Boundary Conditions for Discrete Laplace

- What values do we use to compute averages near the boundary?

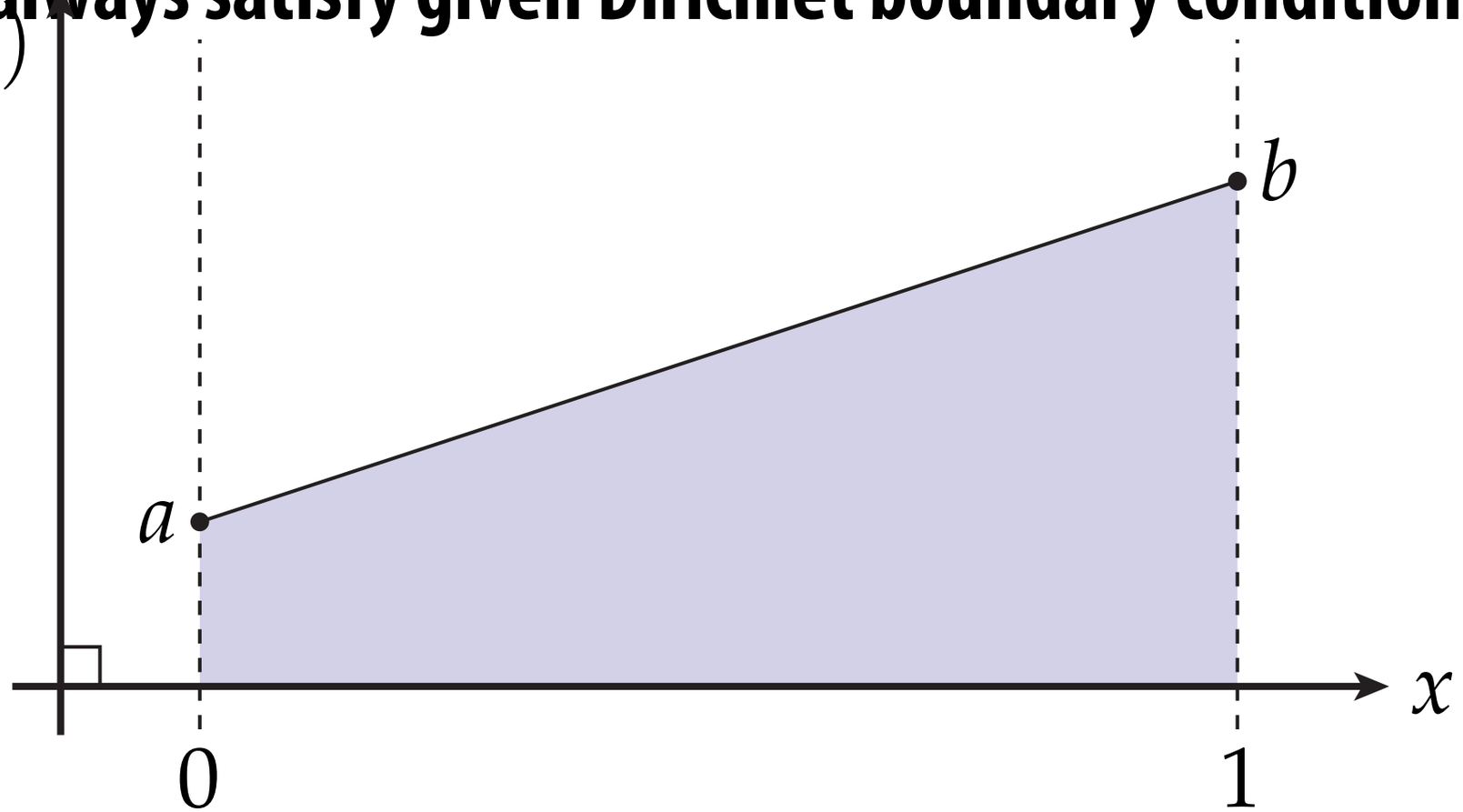


$$a = \frac{1}{4}(b + c + ? + e)$$

- A: We get to choose—this is the data we want to interpolate!
- Two basic boundary conditions:
  1. Dirichlet—boundary data always set to fixed values
  2. Neumann—specify derivative (difference) across boundary
- Also mixed (Robin) boundary conditions (and more, in general)

1D Laplace w/ Dirichlet BCs

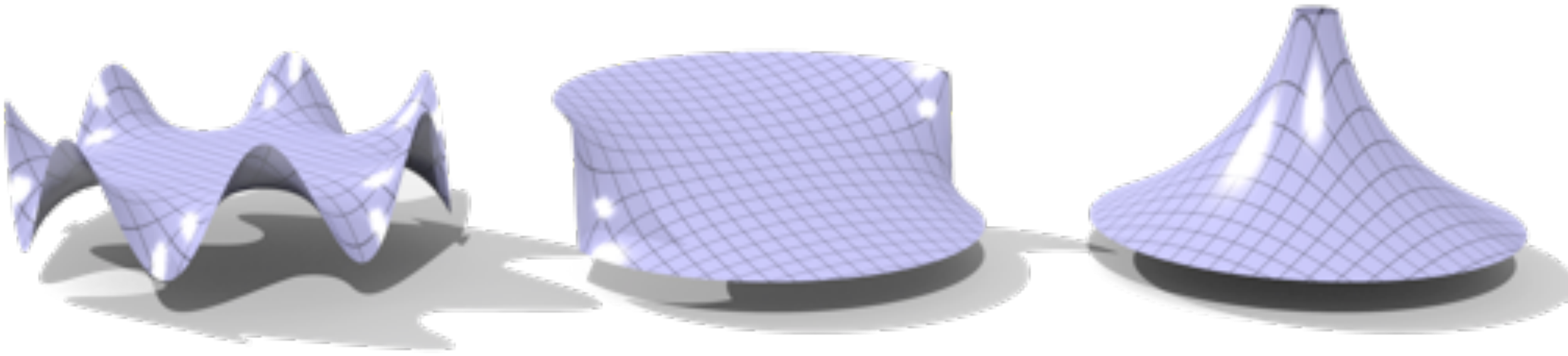
- **1D Laplace:** $\partial^2\phi/\partial x^2 = 0$
- **Solutions:** $\phi(x) = cx + d$
- **Q: Can we always satisfy given Dirichlet boundary conditions?**



- **Yes: a line can interpolate any two points.**

2D Laplace w/ Dirichlet BC

- The smoothest function interpolating the given boundary data



- Each value is still the average of its "neighbors"

Image from Solomon, Crane, Vouga, "Laplace-Beltrami: The Swiss Army Knife of Geometry Processing"

A variation on Laplace: The Poisson Equation

- **Laplace equation:** $\Delta u = 0$
- **Poisson equation:** $\Delta u = f$

A variation on Laplace: The Poisson Equation

Real-Time Gradient-Domain Painting

James McCann*
Carnegie Mellon University

Nancy S. Pollard†
Carnegie Mellon University

Finding the image u with gradients close, in the least-squares sense, to given – potentially non-conservative – edited gradient images G^x, G^y is equivalent to solving a Poisson equation:

$$\nabla^2 u = f \tag{5}$$

Where f is computed from the edited gradients:

$$f_{x,y} = G^x_{x,y} - G^x_{x-1,y} + G^y_{x,y} - G^y_{x,y-1} \tag{6}$$

SIGGRAPH 2008

A variation on Laplace: The Poisson Equation

SIGGRAPH 2008

Let's move on to Solving the Heat Equation

- Back to our three model equations, want to solve heat eqn.

$$\dot{u} = \Delta u$$

- Just saw how to discretize Laplacian
- Also know how to do time (forward Euler, backward Euler, ...)
- E.g., forward Euler:

$$u^{k+1} = u^k + \tau \Delta u^k$$

- Q: On a grid, what's our overall update now at $u_{i,j}$?

$$u_{i,j}^{k+1} = u^k + \frac{\tau}{h^2}(4u_{i,j}^k - u_{i+1,j}^k - u_{i-1,j}^k - u_{i,j+1}^k - u_{i,j-1}^k)$$

- Not hard to implement! Loop over grid, add up some neighbors.

Solving the Wave Equation

- Finally, wave equation:

$$\ddot{u} = \Delta u$$

- Not much different; now have 2nd derivative in time
- By now we've learned two different techniques:
  - Convert to two 1st order (in time) equations:

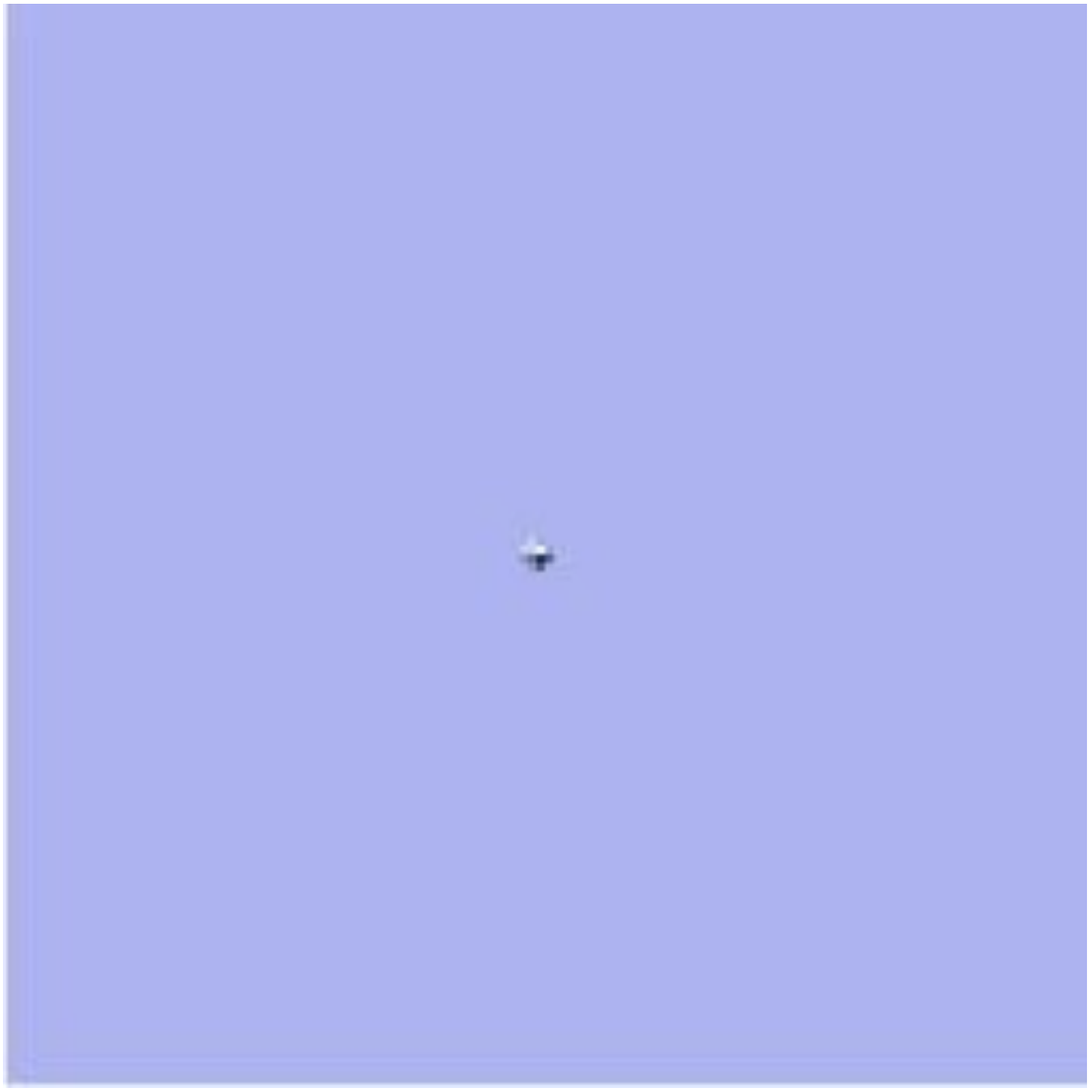
$$\dot{u} = v, \quad \dot{v} = \Delta u$$

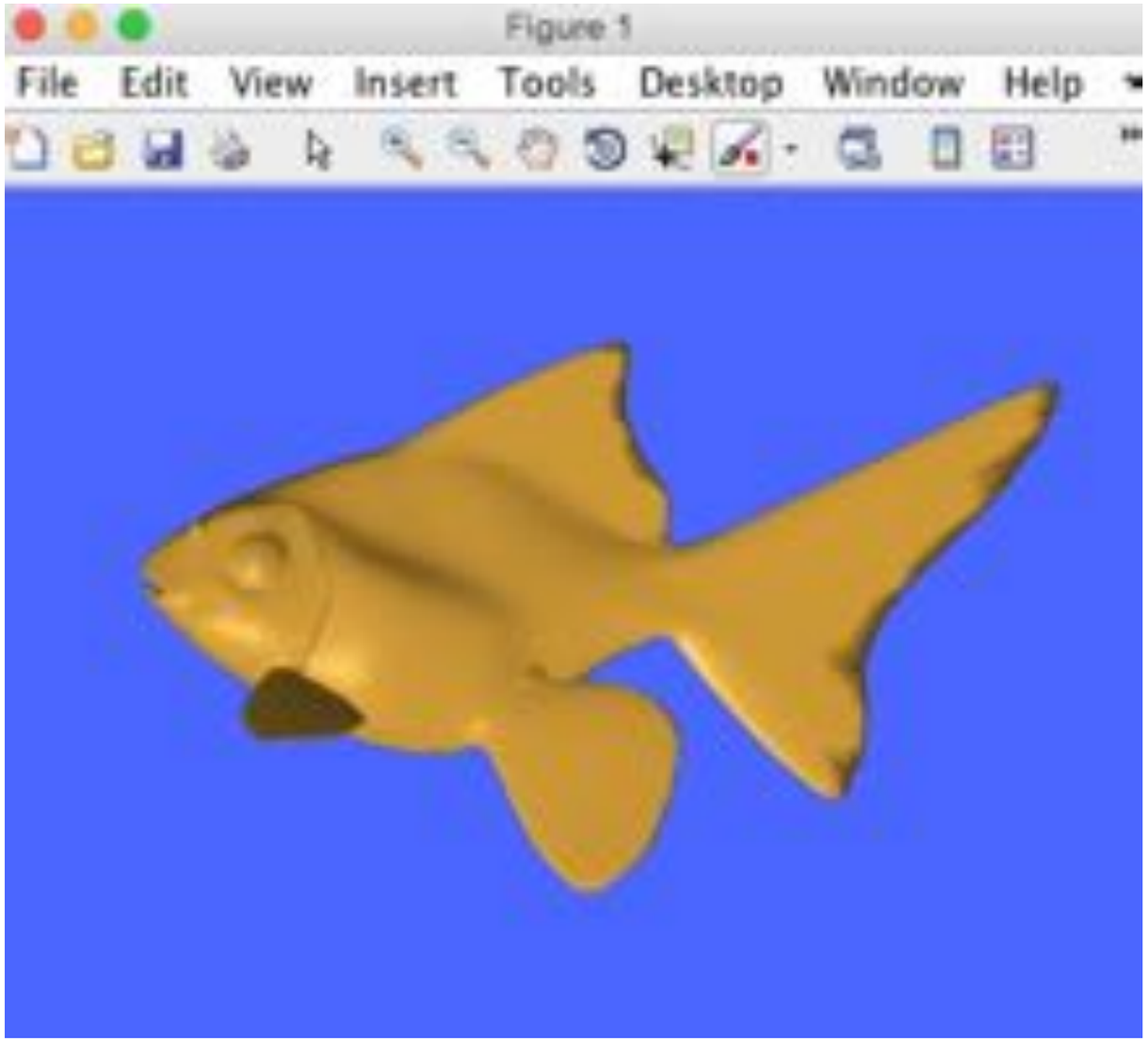
  - Or, use centered difference (like Laplace) in time:

$$\frac{u^{k+1} - 2u^k + u^{k-1}}{\tau^2} = \Delta u^k$$

- Plus all our choices about how to discretize Laplacian.
- So many choices! And many, many (many) more we didn't discuss.

Wave Equation on a Grid, Triangle Mesh





Fish credit: Alec Jacobson (http://www.alecjacobson.com/weblog/?p=4363)

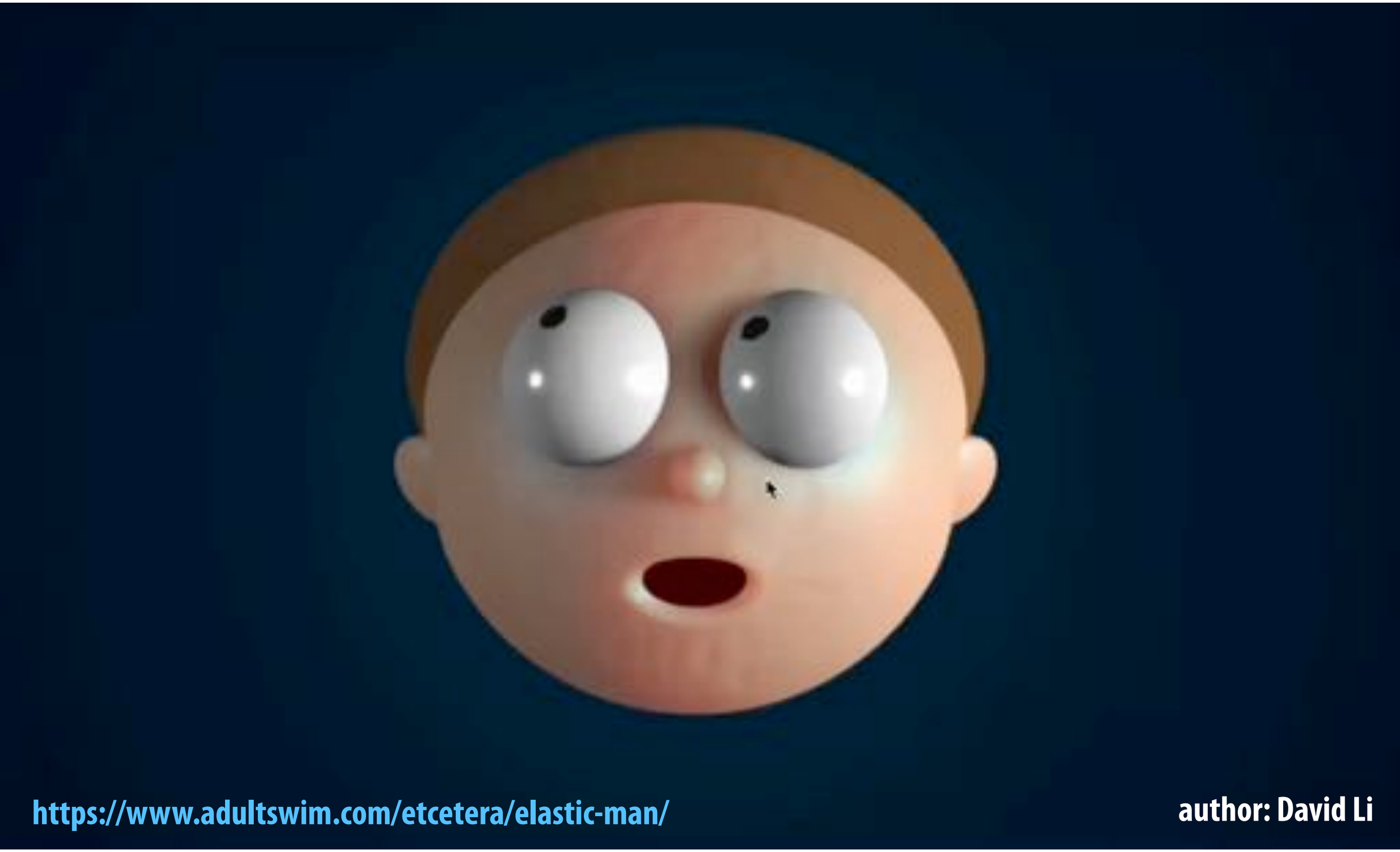
Solving a PDE in Code

Don't be intimidated—very simple code can give rise to beautiful behavior!

```
void simulateWaves2D() {
   const int N = 128; // grid size
   double u[N][N]; // height
   double v[N][N]; // velocity (time derivative of height)
   const double tau = 0.2; // time step size
   const double alpha = 0.985; // damping factor

   for( int frame = 0; true; frame++ ) {  // loop forever
      // drop random "stones"
      if( frame % 100 == 0 ) u[rand()%N][rand()%N] = -1;
      // update velocity
      for( int i = 0; i < N; i++ )
      for( int j = 0; j < N; j++ ) {
         int i0 = (i + N-1) % N; // left
         int i1 = (i + N+1) % N; // right
         int j0 = (j + N-1) % N; // down
         int j1 = (j + N+1) % N; // up
         v[i][j] += tau * (u[i0][j] + u[i1][j] + u[i][j0] + u[i][j1] - 4*u[i][j])
         v[i][j] *= alpha; // damping
      }
      // update height
      for( int i = 0; i < N; i++ )
      for( int j = 0; j < N; j++ ) {
         u[i][j] += tau * v[i][j];
      }
      display( u );
   }
}
```

Fun with wave-like equations…



https://www.adultswim.com/etcetera/elastic-man/

author: David Li

Technique: low-res thin shell simulation (via “position-based dynamics”) + Loop subdivision

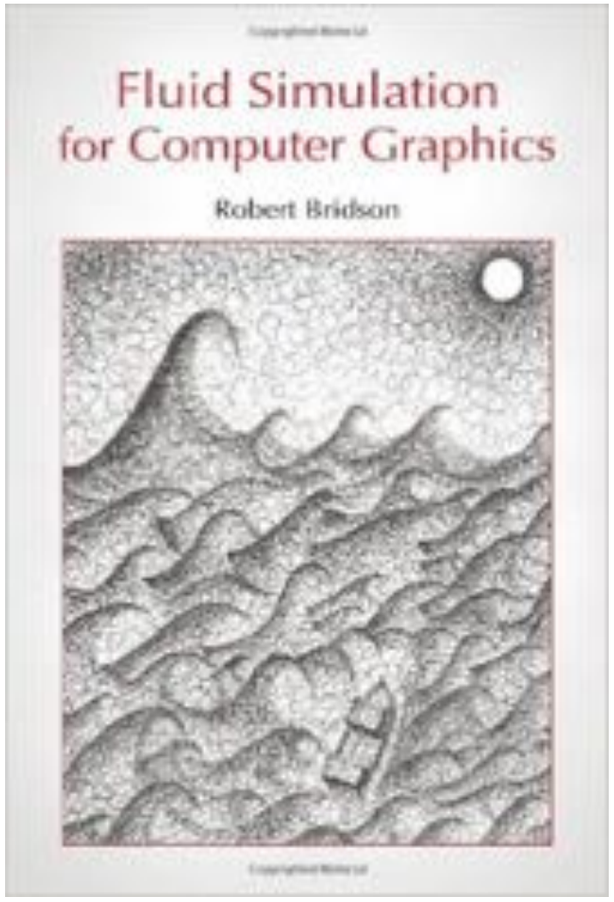
Wait, what about all that other cool stuff?

(Fluids, hair, cloth, . . .)

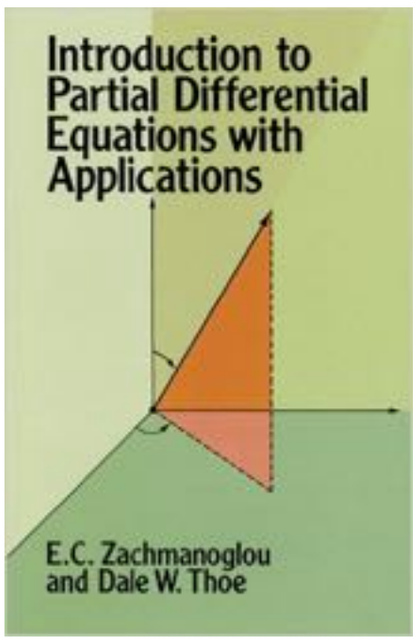
Want to Know More?

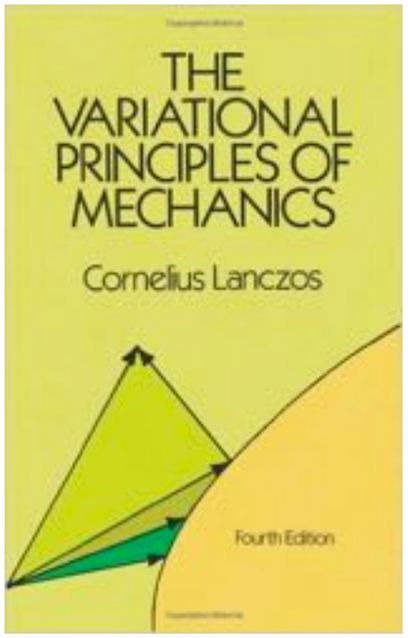
- There are some good books:
- And papers:

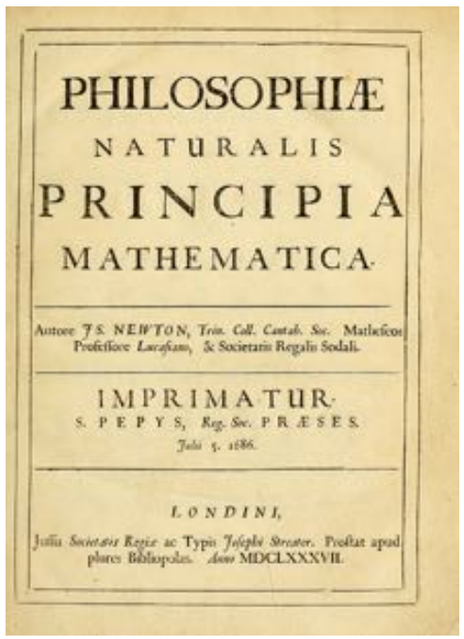
http://www.physicsbasedanimation.com/



- Also, what did the folks who wrote these books & papers read?







Final Exam Review

15-462 / 15-662 Computer Graphics

Lecture 1: Introduction

- For any given setup where we place a camera in the environment, pointing down any of the main coordinate axes (x, y, or z), compute a projection of points in the world onto an image plane.
- Write an algorithm for drawing lines that handles all edge cases (i.e., including edges that are exactly horizontal or vertical).

Lecture 2: Math Review 1 (part 1 of 2):

- How can we measure vectors?
- What is a Vector Space?
- Draw a geometric representation of each rule that vectors seem to obey.
- Can a function be a vector? Explain
- Add and scale vectors.
- Add and scale functions
- What is the norm of a vector?
- Associate each property of a norm with a geometric interpretation.
- Compute the Euclidean norm in Cartesian coordinates.
- Compute the L2 norm of a function.
- Associate each property of the inner product with a geometric interpretation.
- Compute the inner product in Cartesian coordinates.

Lecture 2: Math Review 1 (part 2 of 2):

- Use the inner product for operations such as projection.
- Compute the inner product of functions.
- Give properties and an example of a linear map.
- Define span and basis..
- Compute an orthonormal basis from a set of vectors.
- Be able to use Gram-Schmidt orthonormalization.
- Know that orthonormalization of functions can be done by decomposing them into sinusoids.
- Be able to solve a simple system of linear equations, depict it geometrically, and represent it in matrix form.
- Be able to represent a linear map in matrix form.

Lecture 3: Math Review 2 (Part 1 of 2):

- Euclidean norm is any notion of length preserved by rotations/translations/ reflections of space. Be able to calculate it for a vector of any dimension.
- Compute the inner product of two n-dimensional vectors. What is the geometric meaning if an orthonormal basis is used?
- Compute the cross product of two three-dimensional vectors. What is the geometric meaning of this cross product? What is the geometric meaning of its magnitude?
- Use the cross product to do a quarter rotation of a vector within a plane.
- Represent the dot product using matrix notation.
- Represent the cross product using matrix notation.
- Understand what the determinant measures. What does the determinant of a linear map tell us? Give an example.
- What is a directional derivative?
- Bonus: Compute the gradient of a function.

Lecture 3: Math Review 2 (Part 2 of 2):

- Understand gradient as the best linear approximation and the direction of steepest ascent.
- When is the gradient not defined?
- Bonus: Be able to express gradients of simple matrix expressions
- What is a vector field? Give an example.
- Be able to compute divergence, curl, and the Laplacian of a vector field. Also be able to express the meaning of these terms geometrically, for example by drawing a diagram.
- What is the Hessian? Be able to compute the Hessian of a function.

Lecture 4: Drawing a Triangle

1. How should we choose the correct color for a pixel? There is not an exact right answer. However, you should be able to discuss some of the issues involved.
2. What is aliasing, and what artifacts does it produce in our images and our animations?
3. One form of aliasing is where high frequencies masquerade as low frequencies. Give an example of this phenomenon.
4. Suppose we have a single red triangle displayed against a blue background. Does this scene contain high frequencies?
5. What does the Nyqvist-Shannon theorem tell us about how image frequencies relate to required sampling rate?
6. One practical solution on your graphics card for reducing aliasing (i.e., for antialiasing) is to take multiple samples per pixel and average to get pixel color. Try to use what we learned about sampling theory to explain as precisely as you can why taking multiple samples per pixel can reduce aliasing artifacts.
7. Bonus: Be able to write an implicit representation of an edge given two points.
8. Bonus: Be able to use the implicit edge representation to determine if a point is inside a triangle.

Lecture 5: Transforms (Part 1 of 2)

1. Which of the following operations are linear transforms: scale, rotation, shear, translation, reflection, rotation about a point that is not the origin?
2. Express scale as a linear transform
3. Express rotation as a linear transform
4. Express shear as a linear transform
5. Express reflection as a linear transform
6. Express translation as an affine transform
7. Know what makes a transform linear vs. affine
8. Know how to build transformation matrices from start and end configurations of your object

Lecture 5: Transforms (Part 2 of 2)

- Create 2D and 3D transformation matrices to perform specific scale, shear, rotation, reflection, and translation operations
- Compose transformations to achieve compound effects
- Rotate an object about a fixed point
- Rotate an object about a given axis
- Create an orthonormal basis given a single vector
- Understand the equivalence of [x y 1] and [wx wy w] vectors
- Explain/illustrate how translations in 2D (x, y) are a shear operation in the homogeneous coordinate space (x, y, w)

Lecture 6: 3D Rotations

- What is the problem of gimbal lock? Give an example where this problem occurs.
- How does using quaternions solve this problem?
- Know that every rotation can be expressed as rotation by some angle about some axis.
- Know how to go between quaternions and axis-angle format for rotations.
- Know that quaternions are expressed as higher dimensional complex numbers.
- Be able to work out quaternion multiplication from the complex number representation of a quaternion.

Lecture 7: Projection

- Review:
  - Form an orthonormal basis
  - Create a rotation matrix to rotate any coordinate frame to xyz
  - Create the rotation matrix to rotate the xyz coordinate frame to any other frame
  - Know basic facts about rotation matrices / how to recognize a rotation matrix
    - Rows (also columns) are unit vectors
    - Rows (also columns) are orthogonal to one another
    - If our rows (or columns) are u, v, and w, then uXv=w
    - The inverse of a rotation matrix is its transpose
- Create a projection matrix that projects all points onto an image plane at z=1
- Propose a projection matrix that maintains some depth information
- Understand the motivation behind the projection matrix that projects the view frustum to a unit cube
- Be able to draw / discuss the details of the view frustum
- Prove that a standard projection matrix preserves some information about depth

Lecture 7: Barycentric Coordinates

- Interpolate colors using barycentric coordinates
- Bonus: Compute barycentric coordinates of a point using implicit edge functions
- Compute barycentric coordinates of a point using triangle areas
- Estimate the location of a point inside a triangle given its barycentric coordinates
- Estimate the location of a point outside a triangle given its barycentric coordinates

- Estimate barycentric coordinates of a point from a drawing.
- Show that interpolation in 3D space followed by projection can give a different result from projection followed by interpolation in screen space. In other words, explain why interpolation using barycentric coordinates in screen space may give a result that is incorrect.
- How, then, can we obtain a correct result using interpolation in screen space?

Lecture 8: Textures

- Textures are used for many things, beyond pasting images onto object surfaces.
  - Normal maps (create appearance of bumpy object on smooth surface by giving false normal to the lighting equations)
  - Displacement maps (encode offsets in the geometry of a surface, which is difficult to handle in a standard graphics pipeline)
  - Environment maps (store light information in all directions in a scene)
  - Ambient occlusion map (store exposure of geometry to ambient light for better representation of surface appearance with simple lighting models)
  - Can you think of / discover others?
- Know how to interpolate texture coordinates
- Know how to index into a texture and compute a correct color using bilinear interpolation
- Be able to create a mipmap and store it in memory
- Be able to compute color from multiple levels of mipmaps using trilinear interpolation
- What is the logic behind selecting an appropriate level in a mipmap?
- What can happen if we select a level that is too high resolution? too low resolution?

Lecture 8: Depth and Transparency

- What is the depth buffer (Z-buffer) and how is it used for hidden surface removal?
- Where does the depth for each sample / fragment come from? Where is it computed in the graphics pipeline?
- Is the depth represented in the depth buffer the actual distance from the camera? If not, what is it?
- What is the meaning of the alpha parameter in the [R G B a] color representation?
- Be able to use alpha to do compositing with the "Over" operator.
- Is "Over" commutative? If not, create a counterexample.

$$C = \alpha_B B + (1 - \alpha_B)\alpha_A A$$

- What is premultiplied alpha, and how does it work?
- Be able to use premultiplied alpha for "Over" composition.
- Why is premultiplied alpha better?
- How do we properly render a scene with mixed opaque and semi-transparent triangles? What is the rendering order we should use? When is the depth buffer updated?
- Draw a rough sketch of the graphics pipeline. Think about transforming triangles into camera space, doing perspective projection, clipping, transforming to screen coordinates, computing colors for samples, computing colors for pixels, the depth test, updating color and depth buffers.

Lecture 9: Introduction to Geometry

- List some types of implicit surface representations
- What types of operations are easy with implicit surface representations?
- List some types of explicit surface representations
- What types of operations are easy with explicit surface representations?
- What is CSG (constructive solid geometry)? Give some examples of CSG operations.
- What type of representation is best for CSG operations?
- Describe how to do union, intersection, and subtraction of geometry using simple operators on a surface representation.
- What is a level set representation? When is it useful?
- What types of splines are common in computer graphics?
- Why are they popular? What properties make them most useful?
- BONUS: Derive the equation on the slide labeled "Bézier Curves — tangent continuity" from the definitions given on the previous slides. Draw a diagram to illustrate any terms you use.

Lecture 10: Meshes and Manifolds

- What is a manifold surface?
- Distinguish manifold from non-manifold surfaces
- Can a manifold surface have a boundary? Give an example.
- Describe how to store mesh information in vertex and face tables. Give an example. What are good and bad points of this data structure?
- How would you store a mesh using incidence matrices? What are good and bad points of this data structure?
- What do you need to store in a halfedge data structure? What are good and bad points of this data structure?
- How can you find all vertices in a face with this data structure?
- How can you find all faces that contain a vertex with this data structure?
- BONUS: Think of an algorithm to traverse every face in a manifold using this data structure.

Lecture 11: Geometry Processing Part 1

- List practical applications that you can relate to for good geometry processing algorithms.
- Give criteria for what makes a good quality mesh. Be sure to state your assumptions (e.g., good quality for what purpose?)
- Give pseudocode for one iteration of Catmull-Clark subdivision
- Give pseudocode for one iteration of Loop subdivision
- Explain the important properties of various subdivision algorithms (interpolation, continuity, behavior at the boundaries)
- Be prepared to calculate vertex updates in a simple example of Loop or other subdivision. (The vertex weighting masks will be given to you.)
- Understand how the K matrix of the Quadric Error error metric encodes squared distance to a plane. How can it encode the sum of squared distances to many planes? How is this idea used in generating a good error metric for mesh decimation using edge collapse?

Lecture 11: Geometry Processing Part 2

- Express distance from a plane, given a point on the plane and a normal vector
- Show how the K matrix (the quadric error matrix) represents squared distance from a plane
- Given K matrices encoding triangles in a mesh, how do we get the K matrix for each vertex?
- If we collapse an edge, what is the K matrix for the new vertex that is added in the edge collapse?
- Given a K matrix and a proposed point, what is the cost (the quadric error)?
- How does this cost represent distance to the original surface?
- Describe some techniques for improving the quality of a mesh to make it more uniform and regular.

Lecture 12: Geometric Queries

- Find the closest point to a point, line, or line segment.
- Compute ray-triangle intersection, including checking whether the ray passed through the inside of the triangle.
- Be prepared to compute ray-primitive intersection for other primitives (e.g., a sphere) and primitive-primitive intersections (e.g., triangle-triangle or line-line)
- In all (most?) of these queries, what is the basic strategy that you use to find the closest point or perform the intersection?

Lecture 13: Spatial Data Structures

- Compute ray - bounding box intersection
- Construct a bounding box hierarchy for a given collection of objects.
- Calculate traversal order of a bounding box hierarchy for a given ray.
- What is the Surface Area Heuristic (SAH) and what goals is it trying to achieve?
- Explain how to choose a bounding box partition using the SAH
- Be able to distinguish between object-centric (primitive partitioning) acceleration structures and space-centric (space-partitioning) acceleration structures
- Know the difference between these acceleration structures, how to build them, how to traverse them, and when to use each type:
  - bounding box and bounding sphere hierarchies
  - KD-trees
  - octrees
  - grids

Lecture 14: Color and Radiometry Part 1

- How would you describe the emission spectrum of a light source?
- What are the rods and cones? Which are present in greater number in the human eye? What can you say about how they are distributed in the retina?
- Since color is best described by a spectrum of emissions, why is it that we can get away with just three values for color (e.g., R-G-B)?
- What are additive and subtractive models for color and when are they used?
- Describe some of the different color spaces that are used to express color.
- An alternative to RGB color space is the CIE color space, with X, Y, and Z primaries. What is Y in this color space? What problem with RGB color space does the CIE color space solve?
- BONUS: Given a color space expressed by some three-dimensional basis it be converted into any other basis through a linear operation (True or False)
- BONUS: What is gamma correction? Give an example where gamma correction is useful.

Lecture 14: Color and Radiometry Part 2

- Visible light consists of a small range of wavelengths along the spectrum from gamma rays to radio waves.
- Energy of a photon depends on wavelength (and speed of light, and Planck's constant)
- What is radiant energy? ..flux? .. irradiance?
- Why (how) does irradiance depend on the angle between the light source and a patch of surface area?
- How does irradiance fall off with distance from the light source?
- What is a solid angle?
- What is radiance, and how many dimensions do you need to capture the radiance in a scene (i.e., to capture a light field)?
- What effect is an ambient occlusion map trying to capture?
- What is radiant intensity?
- (Bonus) Characterize the spectral signatures of different familiar light sources.
- (Bonus) Figure out how to read a Goniometric diagram

Lecture 15: The Rendering Equation (Part 1 of 2)

- Ray tracing algorithms make use of radiance estimates to build up an image of a scene. What is radiance? Why is radiance the fundamental quantity of interest? How are radiance estimates used to compute the color that would be observe by a camera at a point on a material surface?
- What is the difference between radiance and irradiance? Between incident and exitant radiance?
- Explain and illustrate with a sketch all of the terms in the Rendering Equation.
- Draw diagrams to illustrate reflection for (1) a perfectly specular (mirror) surface, (2) a glossy surface, (3) a diffuse surface, (4) a retroreflective surface
- What is a BRDF? What are the inputs and outputs of a BRDF function?
- How would you measure a BRDF? If you were to mount a camera and light source on two robot arms, how many degrees of freedom (joints) would you need to do these measurements?

Lecture 15: The Rendering Equation (Part 2 of 2)

- Given a ray and a surface normal, calculate the direction of perfect reflection
- Given a ray, a surface normal, and indices of refraction, calculate the direction of perfect transmission using Snell's Law.
- What is total internal reflection? Give a detailed example of how / when it can occur.
- What is Fresnel reflection? Sketch curves to illustrate the effect as we have seen in class. Label your axes. Informally, what does this effect show?
- What is subsurface scattering?
- How can we extend the idea of BRDF to subsurface scattering? What additional parameters must be sampled?

Lecture 16 was an overview of A3

- No material from lecture 16 is required for the final exam

Lecture 17: Numerical Integration

- In rendering (global illumination), what are we integrating, i.e., what integral do we want to evaluate?
- How do we use the trapezoid rule to integrate a function?
- How does work increase with dimensionality of our function?
  - This is why we typically use Monte Carlo integration in graphics!
- Give a high level overview of the process of Monte Carlo integration
- What is a probability density function (PDF)?
- What is a Cumulative Distribution Function (CDF)?
- The Inversion Method can be used to correctly draw a sample from a PDF.
  - Sketch the overall step by step process for using the Inversion Method
- What is rejection sampling? Show how to use rejection sampling to sample area of a circle, volume of a sphere, directions on a sphere, and solid angles from a hemisphere.

Lecture 18: Monte Carlo Rendering

- What is Expected Value? .. Variance? .. Bias?
- What is Importance Sampling? How is it used in cosine-weighted sampling of the hemisphere?
- What is the Monte Carlo method (in general)?
- Explain how a Monte Carlo method can be used to solve the Rendering Equation.
- What is Russian Roulette and why do we need it?
- How can we use Russian Roulette and still have an unbiased estimator?

Lecture 19: Variance Reduction

- Be familiar with the following expression for Monte Carlo integration. What is the role of each term?

$$I = \lim_{n \to \infty} V(\Omega) \frac{1}{n} \sum_{i=1}^{n} f(X_i)$$

- Give an example of how we can reduce variance in our rendered results in a path tracing algorithm without increasing the number of samples.
- What does it mean for an estimator to be consistent?
- What does it mean for an estimator to be unbiased?
- Give a concrete example of how a renderer could give a biased estimate of an image. Is the renderer in your example consistent? Explain your answer.
- Give five examples of how you can reweight samples in a pathtracing algorithm in order to do importance sampling.
- What are the main ideas behind bidirectional path tracing?
- How would you enumerate all possible paths in a scene?
- How does Metropolis-Hastings sampling work?
- Assume you have code to generate random paths and code to mutate existing paths. Write pseudocode for Metropolis-Hastings path tracing.

Lecture 20: More Variance Reduction

- What is Stratified Sampling?
- Why is it preferred to random sampling?
- Hammersley and Halton points are pseudo random sampling techniques to generate points with low discrepancy. What is discrepancy? Why do we want to generate low discrepancy samples?
- Give a concise one sentence description of each of the following rendering algorithms that makes it clear the differences between them. Use a diagram to illustrate your description:
  - Rasterization
  - Ray casting
  - Ray tracing
  - Path tracing
  - Bidirectional path tracing
  - Metropolis Light Transport
  - Photon Mapping
  - Radiosity
- Which of these algorithms are best for capturing reflective surfaces? caustics? color bleeding? subsurface scattering? refraction? . . .

Lecture 21: Intro to Animation (Part 1 of 3)

- How were the first animations created? How were the first films created? Give some examples.
- Describe some of the first computer generated animations, giving the developer / artist and timeframe.
- Animations are created from keyframes. How do we interpolate between those keyframes?
- Why do we avoid splines of degree higher than three in computer graphics?
- Write a cubic polynomial P(t) in parameter t, which may describe a cubic spline.
- What are the constraints for P(t) to interpolate endpoints p1 at time t=0 and p2 at time t=1?
- What are the constraints for P(t) to have tangent vector r1 at time t=0 and r2 at time t=1?

Lecture 21: Intro to Animation (Part 2 of 3)

- This pair of constraints describes a Hermite spline. Derive the polynomial coefficients for the Hermite spline and write the cubic polynomial in terms of p1, p2, r1, and r2.
- Put your result in matrix form.
- Give properties of the Hermite spline in terms of continuity (C1, C2, etc..), interpolation, and local control.
- What type of spline has C2 continuity and interpolation, but not local control?
- What type of spline has C2 continuity and local control, but does not interpolate its key points?
- What are Catmull-Rom splines? How are tangents computed for Catmull-Rom splines?
- What are blend shapes and where are they used? What exactly is interpolated when using blend shapes for animation?

Lecture 21: Intro to Animation (Part 3 of 3)

- Bezier, Hermite, and Catmull-Rom splines are really all the same thing. Any one representation can be converted to any of the others. Explain the differences between them.
- Be able to express a Hermite spline in different ways — as a cubic polynomial, in matrix form, or derive it from its control parameters.
- How do we ensure continuity between cubic spline segments? C0 continuity? .. C1 continuity? .. is C2 continuity possible in general?

Lecture 22: Dynamics and Time Integration

- What is the “animation equation”?
- What is the difference between an ODE and a PDE? Give some examples of systems we can simulate by integrating an ODE.
- Sketch an overall system for simulating an ODE using a block diagram. Be clear about what is the state, how you advance the state forward in time, and what integrator you are choosing.
- When is the Euler-Lagrange equation useful?
- Be able to work through a simple example of obtaining dynamic equations of motion using Lagrangian mechanics.
- Describe how to put together a mass-spring system to simulate cloth.
- What are the forces on each cloth “particle”?
- What is Forward Euler integration and what is its disadvantage? Can you show a simple example where it fails?
- What is Backward Euler integration and what are its pros and cons?
- What is Symplectic Euler integration?

Lecture 23: Optimization

- Describe some problems in Computer Graphics where optimization is important.
- Be able to describe an optimization problem in standard form and give a couple of simple examples.
- How do you know you have a minimum of an objective function in an optimization problem without constraints? What properties must be true?
- What is meant by convexity of the domain in an optimization problem? convexity of the objective? Give examples of each. Why do we care about convexity in optimization?
- What is the difference between forward and inverse kinematics?
- Write an expression for the forward kinematics of a simple character or robot.
- What is the Jacobian? Compute the Jacobian for a simple character or robot.
- What is the Jacobian Transpose technique for inverse kinematics? Is it guaranteed to converge? What does that mean in practice? Does it give a locally optimal solution or a globally optimal one? Why?

Lecture 24: Physically Based Animation and PDEs

- What is the difference between a PDE and an ODE? Give examples of when you might use each one and why.
- Interpret this sentence using an equation and a diagram: Solving a PDE looks like *"use neighbor information to get velocity (...and then add a little velocity each time)"*
- Burger's equation is first order in time and second order in space. What does that mean? What are the orders of the Laplace equation? The heat equation? The wave equation? Be able to figure out the order of an equation from an expression of the equation itself.
- What are examples of questions we can answer using the Laplace equation, the heat equation, and the wave equation respectively?
- Outline the basic strategy for solving a PDE.
- What is the Laplace operator? Write it out as a sum of partial derivatives.
- Copy down the heat equation from the slides. Write out the process of solving this equation using Forward Euler integration.
- Copy down the wave equation from the slides. Write out the process of solving this equation using Forward Euler integration.

Lecture 24: Physically Based Animation and PDEs

- Be able to use the grid version of the Laplacian to do smoothing on a grid.