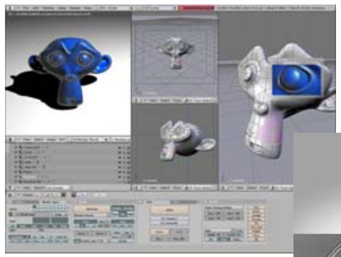
COMPUTER GRAPHICS
CS-314: Fall 2012
Instructor (me): Alla Sheffer

http://www.ugrad.cs.ubc.ca/~cs314

What is Computer Graphics ?

- Generation of virtual visual content



- Encompasses many (sub-)disciplines
  - (defined by what and not by how)

Copyright A. Sheffer, 2012, UBC

What is CG used for?

What is CG used for: Movies

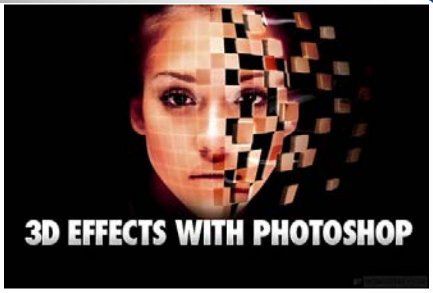
Animation

Special Effects (mixed reality)

Copyright A. Sheffer, 2012, UBC

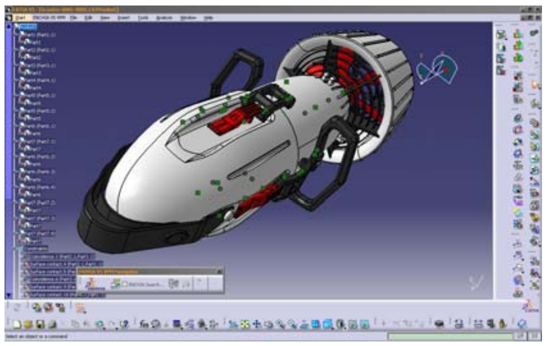
What is CG used for: Games

What is CG used for: Digital Media

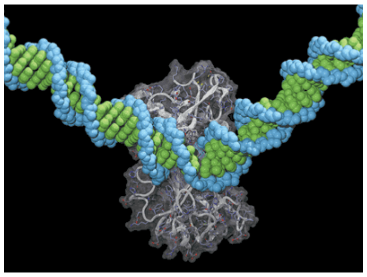


Copyright A. Sheffer, 2012, UBC

What is CG used for: Everything ☺



Engineering (CAD)



Natural Sciences (Visualization/Simulation)

The Science of CG: Components

- Content Creation (3D):
  - Modeling - Representing object properties
    - Geometry: polygons, smooth surfaces etc.
    - Materials: reflection models etc.
  - Animation - Making geometric models move and deform
- Rendering - Generation of 2D images from (3D) models
  - Interactive rendering
  - Global (offline) methods: Ray-tracing, etc...
- Imaging – manipulation of 2D images

The Science of CG : Brief history

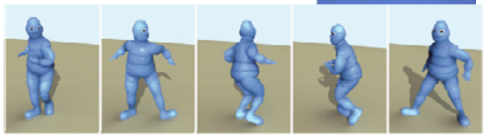
Then (1980s)

Now (trailer from SIGGRAPH'12)

The science of CG (circa 2012)

- Modeling

- Animation



- Rendering

- Imaging

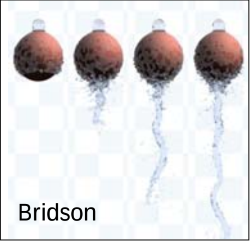
Copyright A. Sheffer, 2012, UBC

CG Research at UBC

Van de Panne

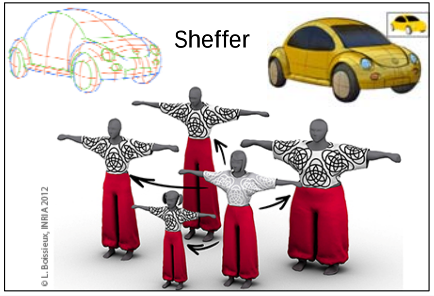
(a) A forward roll transformed to a dive roll.
(b) A cartwheel retargeted to an Asimo-like robot.
(c) A walk transformed onto a balance beam.

Heidrich



Bridson

Pai

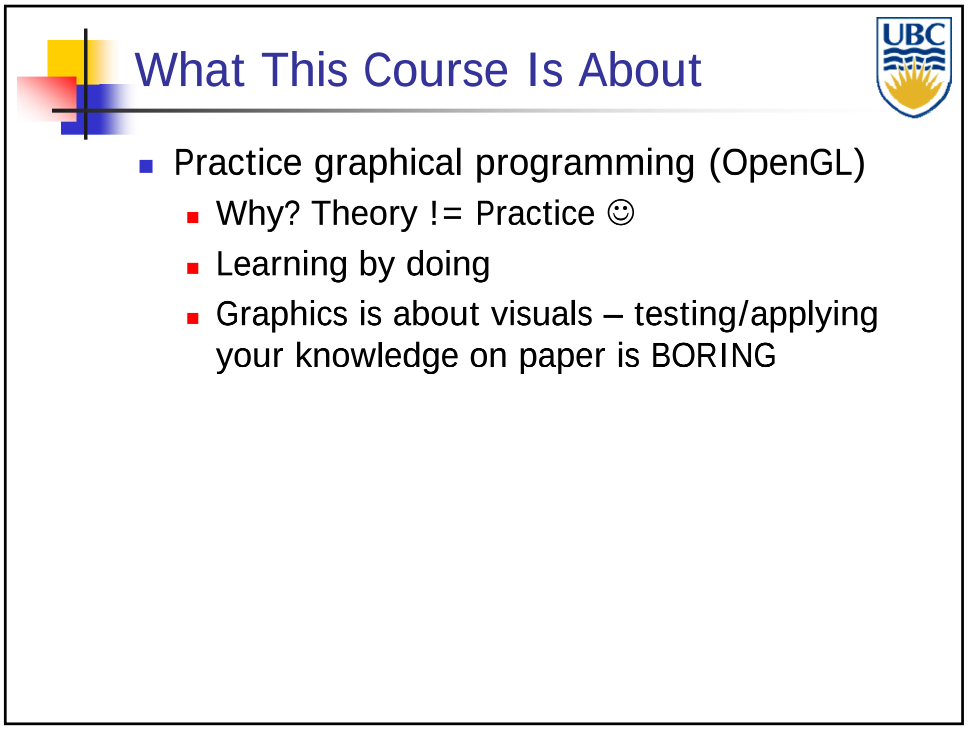


Sheffer

What This Course Is About

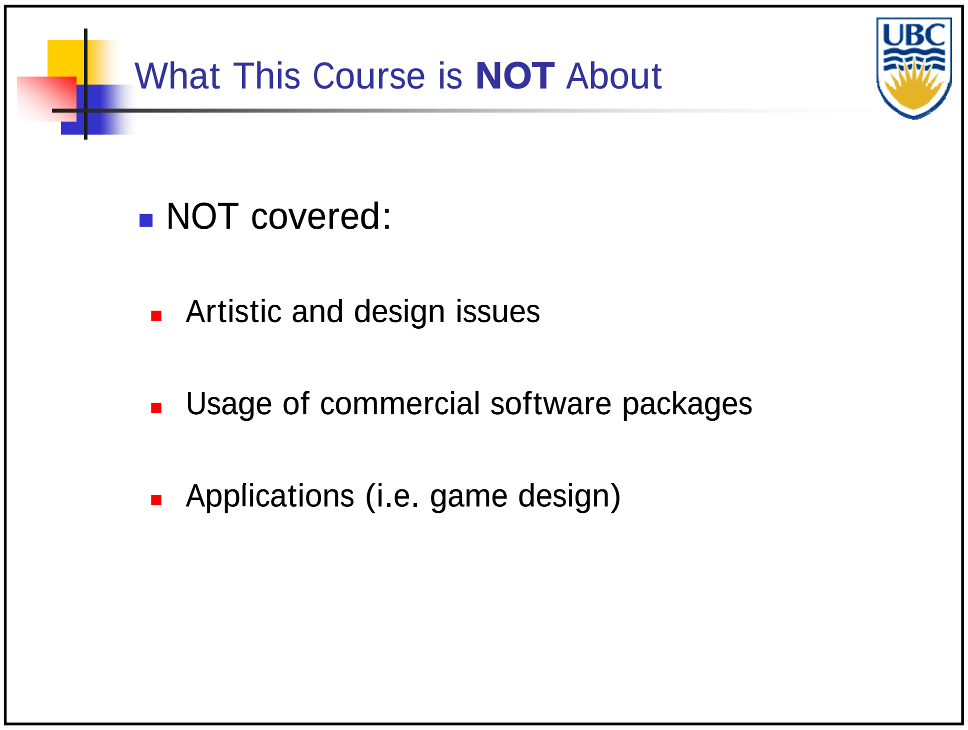
- Fundamental algorithms of computer graphics
- Course Focus: Rendering
  - Why?
    - Critical core graphics component
    - Lots of ideas/methods shared with other CG components
- Content creation addressed in detail by follow-up courses
  - Modeling – 424 (next term)
  - Animation – 426 (next year)
  - Grad level (rendering, modeling,animation)

Copyright A. Sheffer, 2012, UBC



What This Course Is About

- Practice graphical programming (OpenGL)
  - Why? Theory != Practice ☺
  - Learning by doing
  - Graphics is about visuals – testing/applying your knowledge on paper is BORING



What This Course is **NOT** About

- NOT covered:
  - Artistic and design issues
  - Usage of commercial software packages
  - Applications (i.e. game design)

Copyright A. Sheffer, 2012, UBC

Why study CG?

- It is fun – create visually appealing results
- Opens doors to lots of job opportunities
- Gain programming experience
  - Not “just” programming – lots of math, theory, intuition
- Warning: Not at easy course
  - heavy math
  - heavy programming

Policies (boring stuff): http:www.ugrad.cs.ubc.ca/~cs314

Copyright A. Sheffer, 2012, UBC

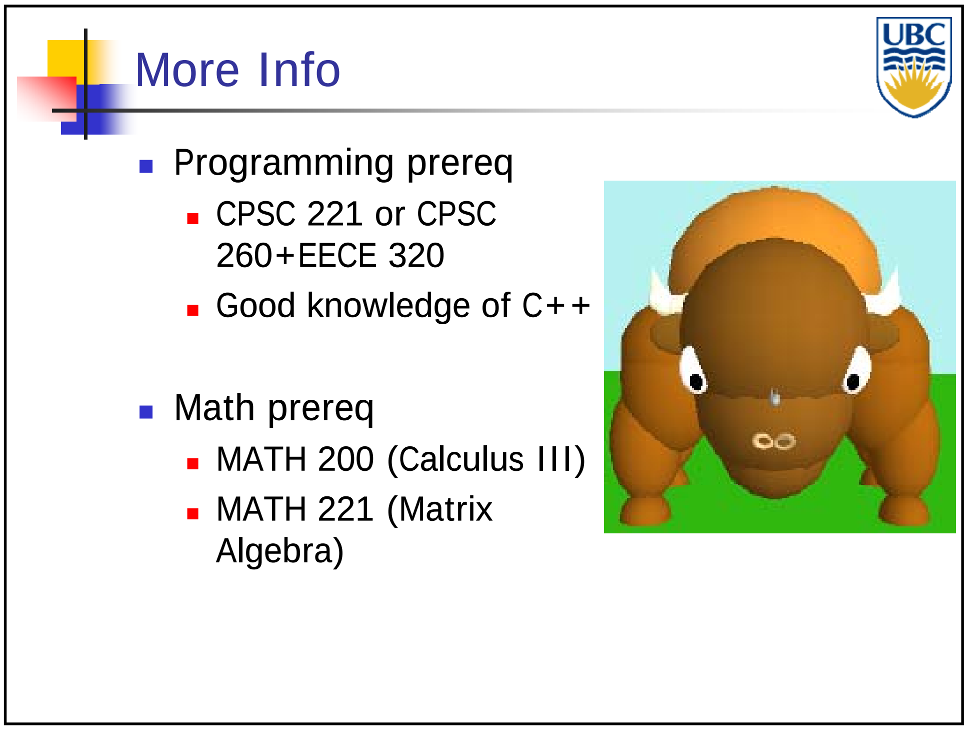
Teaching Staff

- Instructor: Alla Sheffer
- Office hours (in X651):
  - Monday 10-11AM
  - Tuesday 3-4PM
  - Wed 10-11AM
- Contact: sheffa@cs.ubc.ca
  - **Use discussion board (piazza) for questions relevant to other students**
- TAs: Crawford Doran (cdoran@cs.ubc.ca), Anil Mahmud (anil_mahmud@uap-bd.edu) Mikhail Bessmeltsev (bmpix@cs.ubc.ca)

Course Information

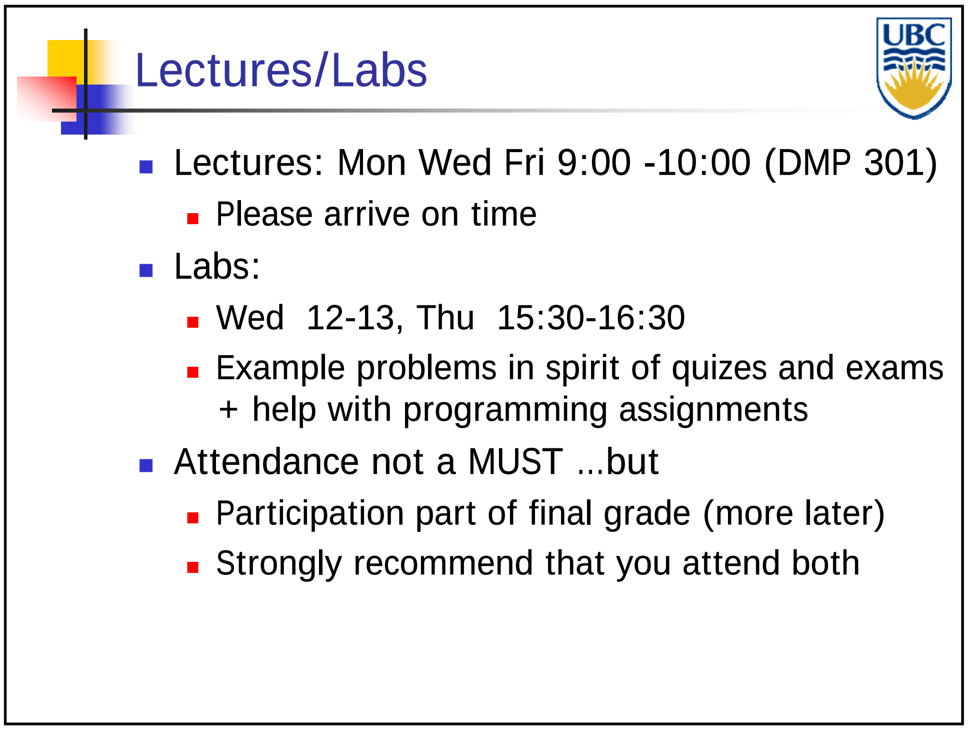
- Up-to-date information:
  - http://www.ugrad.cs.ubc.ca/~cs314
    - updated often, reload frequently
  - Discussion board on *Piazza* - to set up account follow link from course home page
- I assume that once information is posted on board or web-page students know it
  - within 2 workdays

Copyright A. Sheffer, 2012, UBC



More Info

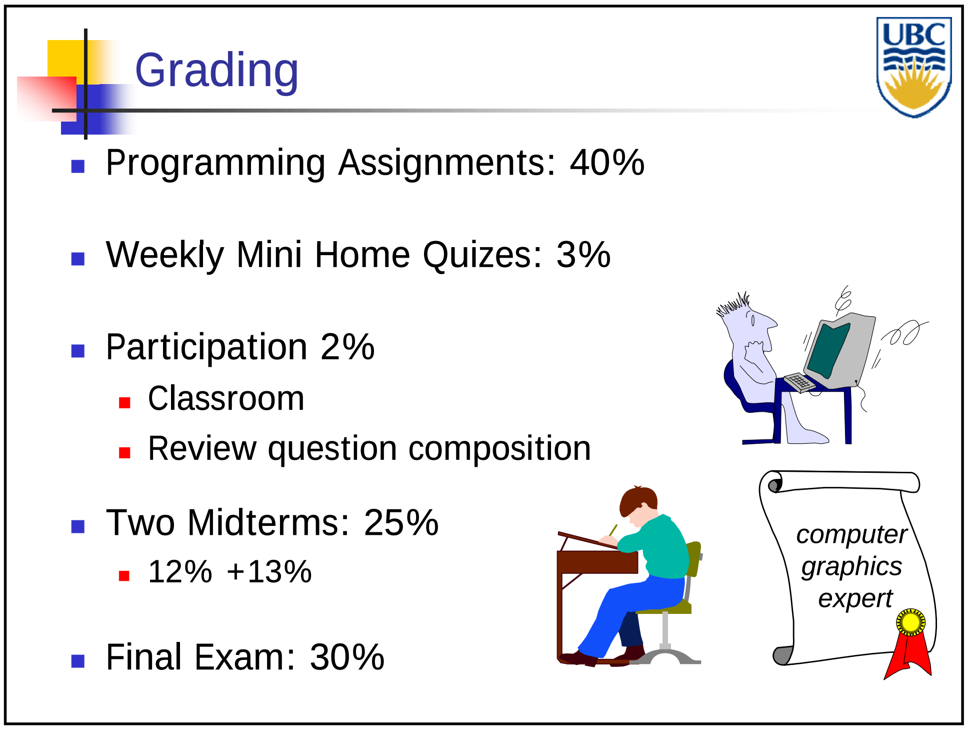
- Programming prereq
  - CPSC 221 or CPSC 260+EECE 320
  - Good knowledge of C++
- Math prereq
  - MATH 200 (Calculus III)
  - MATH 221 (Matrix Algebra)



Lectures/Labs

- Lectures: Mon Wed Fri 9:00 -10:00 (DMP 301)
  - Please arrive on time
- Labs:
  - Wed 12-13, Thu 15:30-16:30
  - Example problems in spirit of quizes and exams + help with programming assignments
- Attendance not a MUST ...but
  - Participation part of final grade (more later)
  - Strongly recommend that you attend both

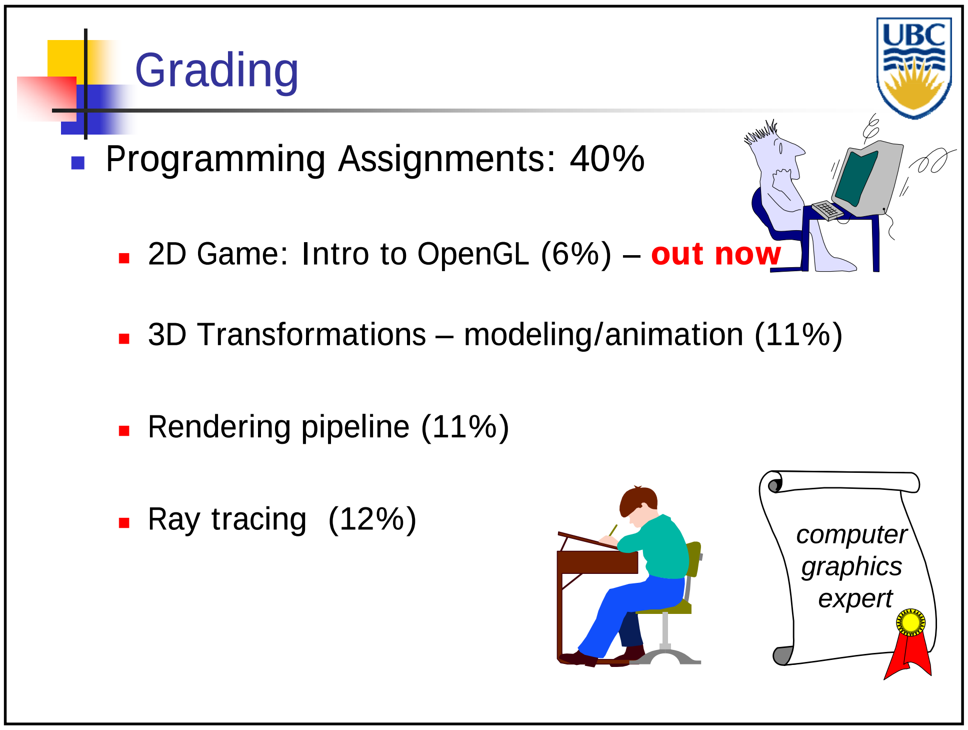
Copyright A. Sheffer, 2012, UBC



Grading

- Programming Assignments: 40%
- Weekly Mini Home Quizes: 3%
- Participation 2%
  - Classroom
  - Review question composition
- Two Midterms: 25%
  - 12% +13%
- Final Exam: 30%

computer
graphics
expert

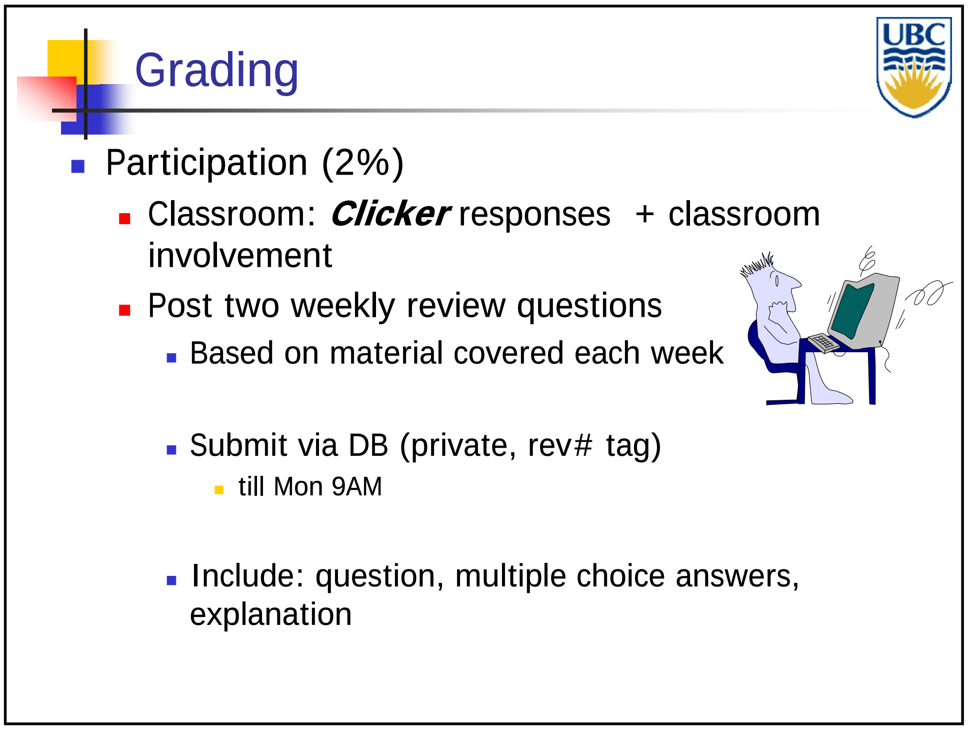


Grading

- Programming Assignments: 40%
  - 2D Game: Intro to OpenGL (6%) – **out now**
  - 3D Transformations – modeling/animation (11%)
  - Rendering pipeline (11%)
  - Ray tracing (12%)

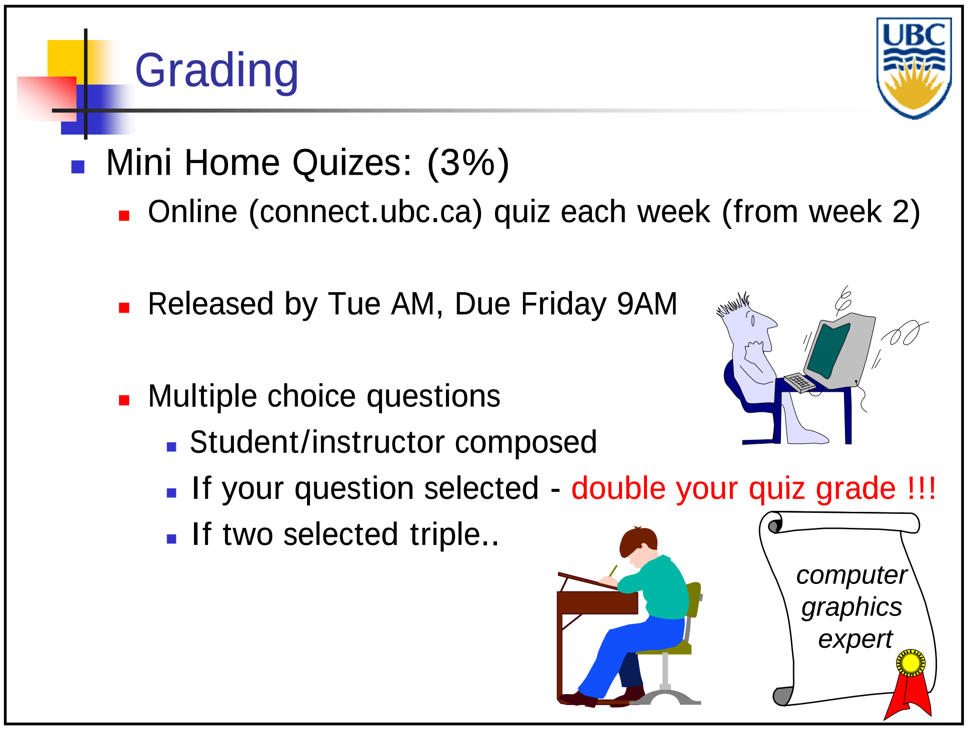
computer
graphics
expert

Copyright A. Sheffer, 2012, UBC



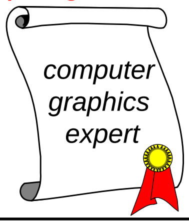
Grading

- Participation (2%)
  - Classroom: ***Clicker*** responses + classroom involvement
  - Post two weekly review questions
    - Based on material covered each week
    - Submit via DB (private, rev# tag)
      - till Mon 9AM
    - Include: question, multiple choice answers, explanation



Grading

- Mini Home Quizes: (3%)
  - Online (connect.ubc.ca) quiz each week (from week 2)
  - Released by Tue AM, Due Friday 9AM
  - Multiple choice questions
    - Student/instructor composed
    - If your question selected - double your quiz grade !!!
    - If two selected triple..

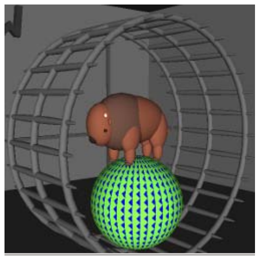


Copyright A. Sheffer, 2012, UBC

Important Dates

- Assignment 1 due: Sep 21
- Assignment 2 due: Oct 12
- Assignment 3 due: Nov 2
- Assignment 4 due: Nov 30

- Midterm 1: Oct 19
- Midterm 2: Nov 9



Course Organization

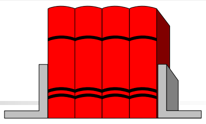
- Programming assignments:
  - C++, Windows or Linux
    - **Tested on department Linux machines**
  - OpenGL graphics library / GLUT for user interface
- **Face to face grading in lab**
  - Opportunity to show all the “cool” extra stuff
  - Test that you do *know* what every piece of your code does
- Hall of fame – coolest projects from 2002 on

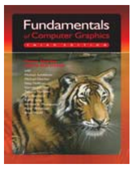
Copyright A. Sheffer, 2012, UBC

Late/Missing Work

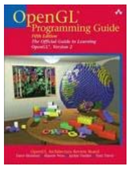
- Programming Assignments:
  - 3 grace days **TOTAL**
    - for unforeseen circumstances
    - strong recommendation: don’t use early in term
    - handing in late uses up automatically unless you tell us
- Home Quizes/Review Question Sets
  - Can miss *two* of each
- Exception: severe illness/crisis, as per UBC rules
  - MUST
    - Get approval from me ASAP (in person or email)
    - Turn in proper documentation

Literature (optional)

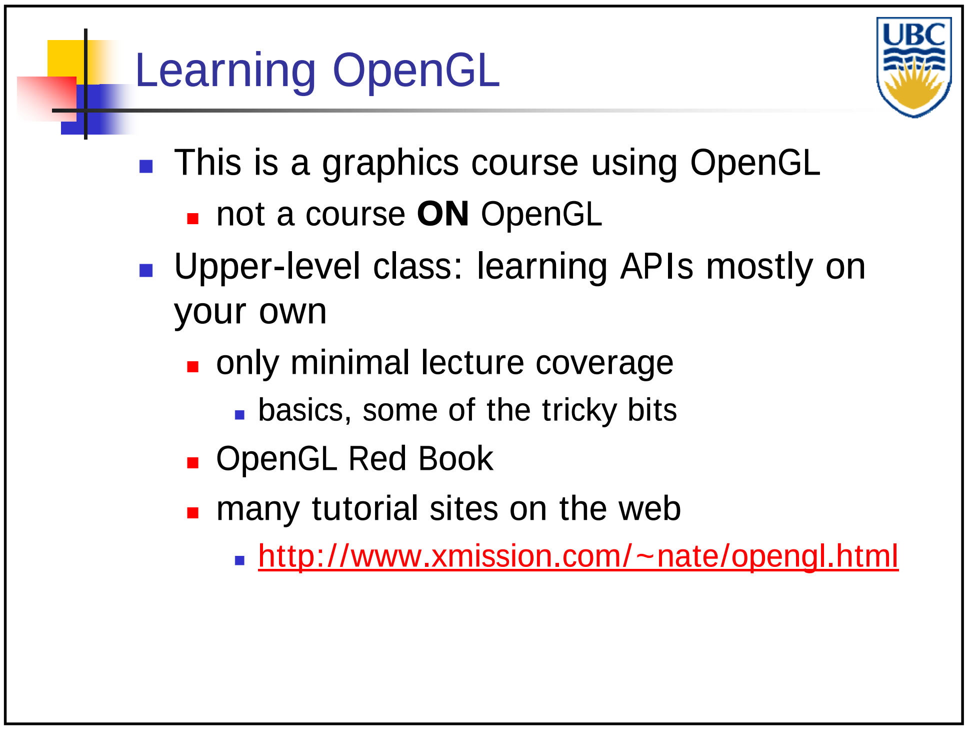




- Fundamentals of Computer Graphics
  - *Third edition (second is OK too – but note syllabus changes)*
  - Peter Shirley, A.K. Peters

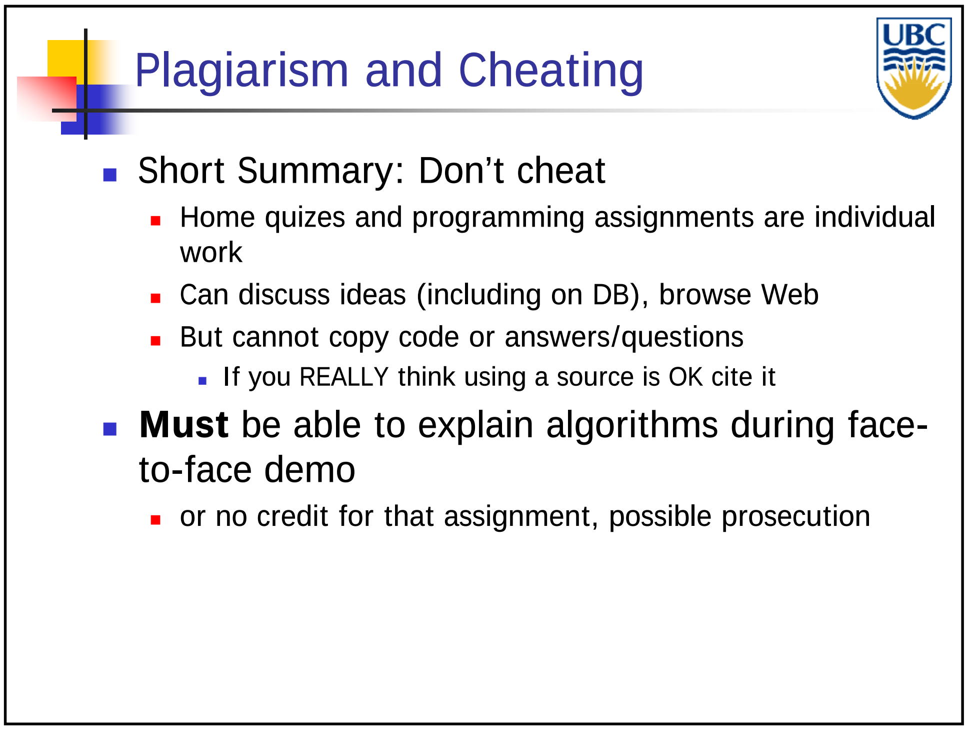


- OpenGL Programming Guide
  - J. Neider, T. Davis and W. Mason, Addison-Wesley



Learning OpenGL

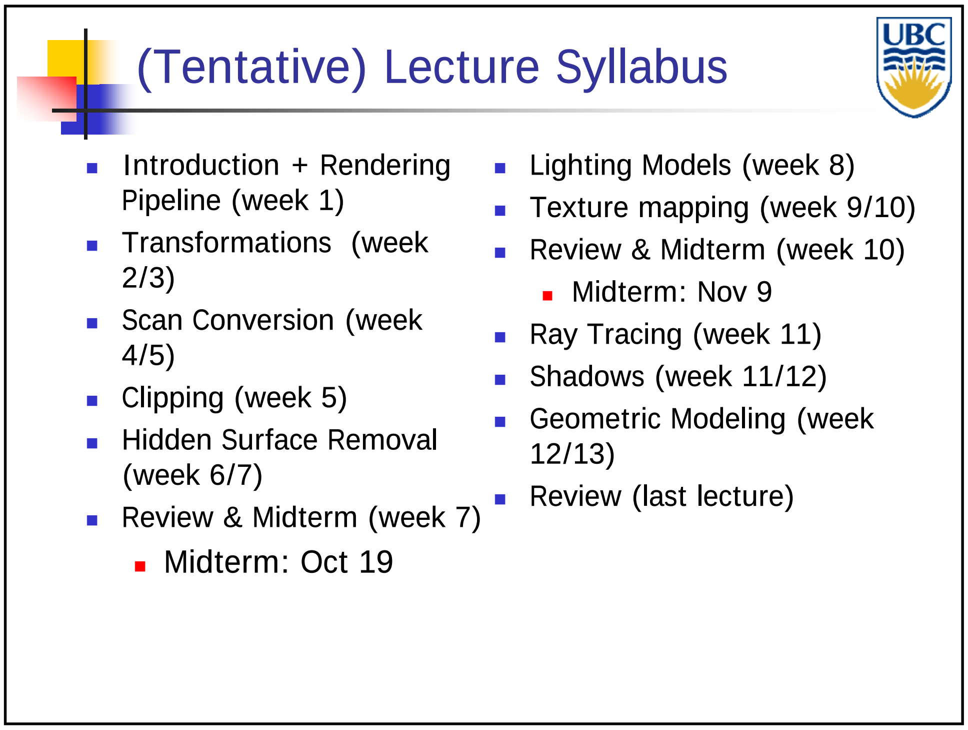
- This is a graphics course using OpenGL
  - not a course **ON** OpenGL
- Upper-level class: learning APIs mostly on your own
  - only minimal lecture coverage
    - basics, some of the tricky bits
  - OpenGL Red Book
  - many tutorial sites on the web
    - http://www.xmission.com/~nate/opengl.html



Plagiarism and Cheating

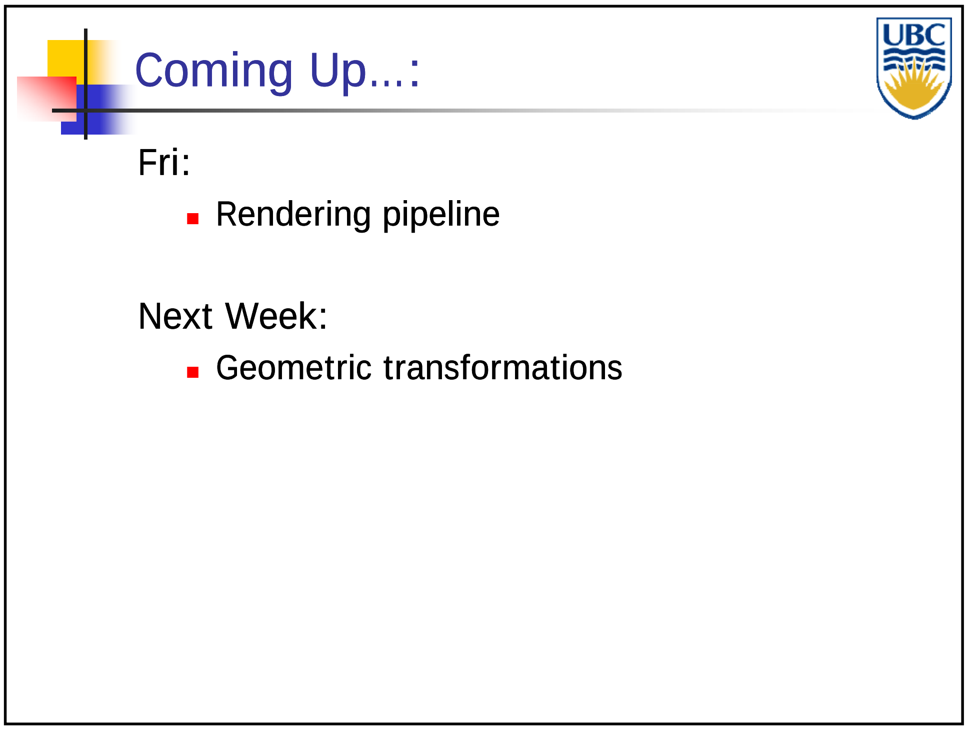
- Short Summary: Don't cheat
  - Home quizes and programming assignments are individual work
  - Can discuss ideas (including on DB), browse Web
  - But cannot copy code or answers/questions
    - If you REALLY think using a source is OK cite it
- **Must** be able to explain algorithms during face-to-face demo
  - or no credit for that assignment, possible prosecution

Copyright A. Sheffer, 2012, UBC



(Tentative) Lecture Syllabus

- Introduction + Rendering Pipeline (week 1)
- Transformations (week 2/3)
- Scan Conversion (week 4/5)
- Clipping (week 5)
- Hidden Surface Removal (week 6/7)
- Review & Midterm (week 7)
  - Midterm: Oct 19
- Lighting Models (week 8)
- Texture mapping (week 9/10)
- Review & Midterm (week 10)
  - Midterm: Nov 9
- Ray Tracing (week 11)
- Shadows (week 11/12)
- Geometric Modeling (week 12/13)
- Review (last lecture)



Coming Up...:

Fri:

- Rendering pipeline

Next Week:

- Geometric transformations

Copyright A. Sheffer, 2012, UBC

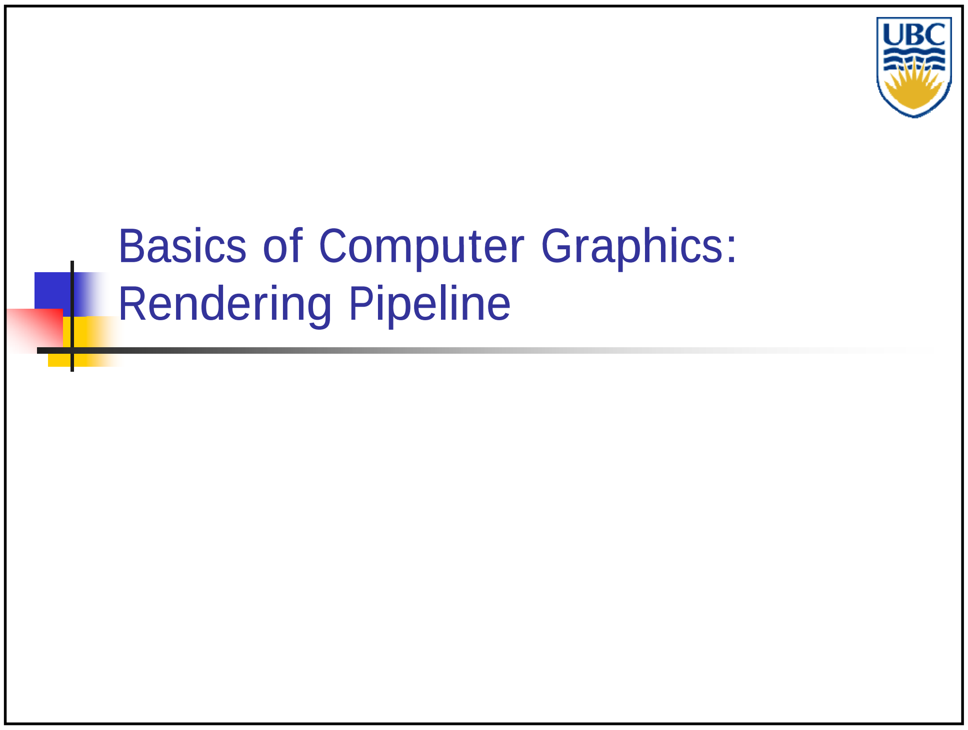
Your tasks for the week

- Piazza Discussion Group:
  - Register
  - Post review questions by Mon 9AM
    - Use private option, rev1 tag
- Assignment 1
  - Test programming environment on lab computers/Set laptop environment (optional)
  - **Come to lab today !!!**

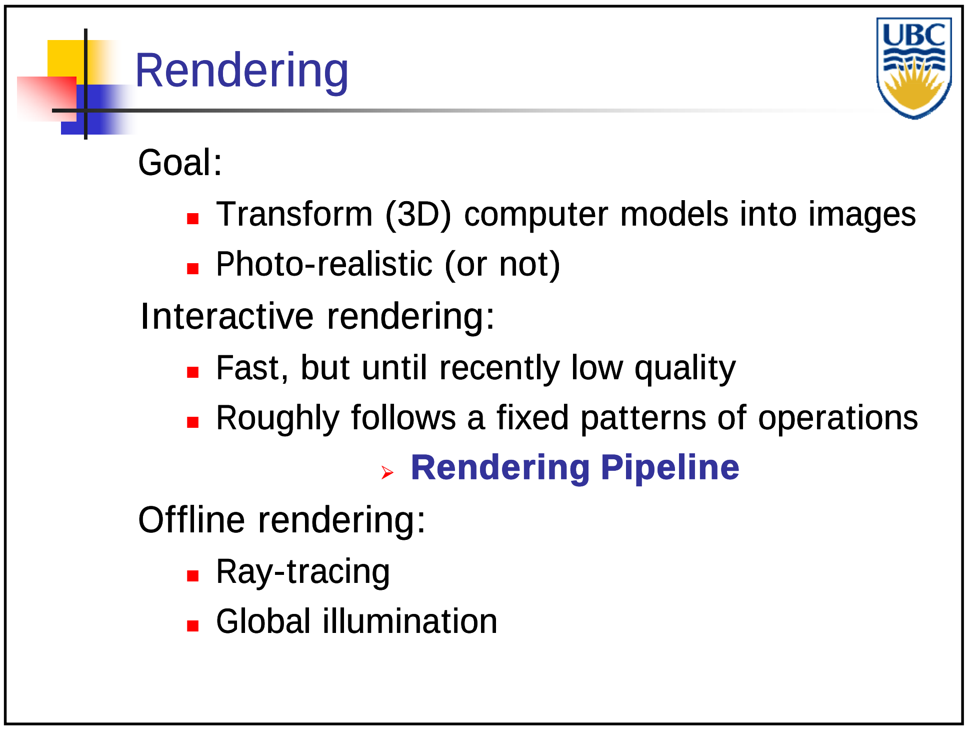
Your tasks for the week

- Sign and Submit Plagiarism Form
  - http://www.ugrad.cs.ubc.ca/~cs314/Vsep2012/plag.html
- Reading (in Shirley: Introduction to CG)
  - Math refresher: Chapters 2, 4
    - **Lots of math coming in the next few weeks**
  - Background on graphics: Chapter 1

Copyright A. Sheffer, 2012, UBC



Basics of Computer Graphics: Rendering Pipeline



Rendering

Goal:

- Transform (3D) computer models into images
- Photo-realistic (or not)

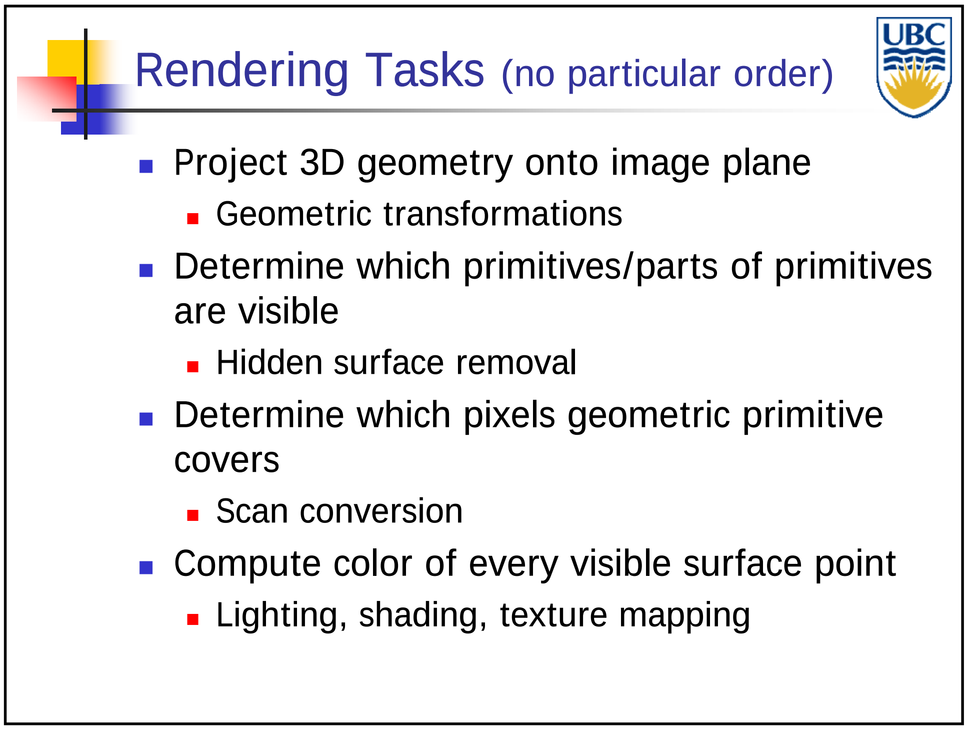
Interactive rendering:

- Fast, but until recently low quality
- Roughly follows a fixed patterns of operations
    - **Rendering Pipeline**

Offline rendering:

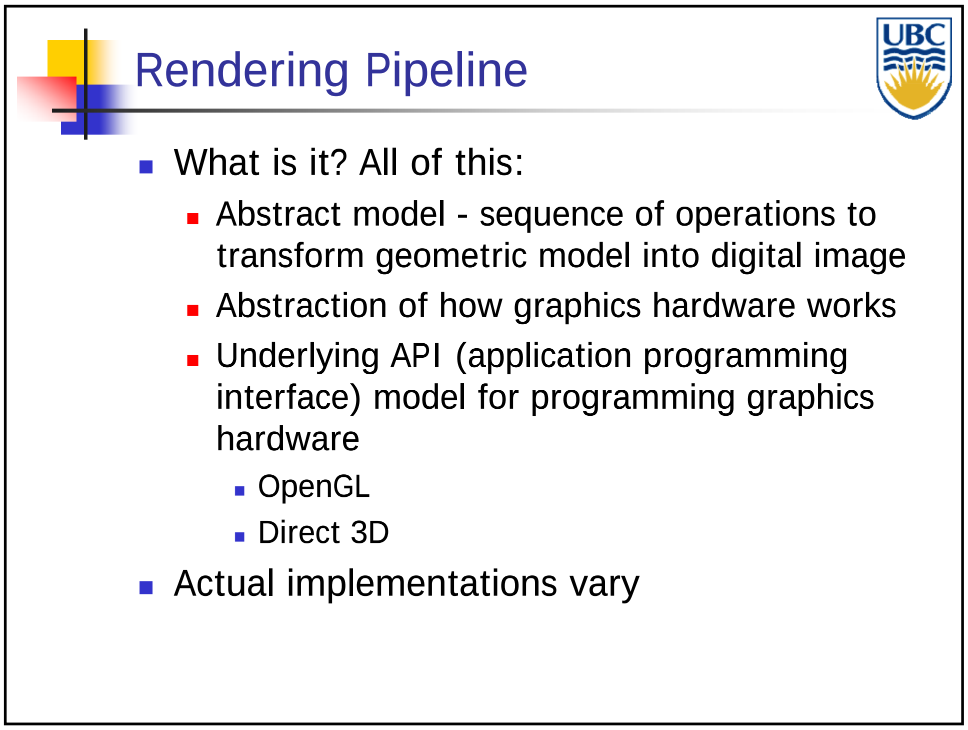
- Ray-tracing
- Global illumination

Copyright A. Sheffer, 2012, UBC



Rendering Tasks (no particular order)

- Project 3D geometry onto image plane
  - Geometric transformations
- Determine which primitives/parts of primitives are visible
  - Hidden surface removal
- Determine which pixels geometric primitive covers
  - Scan conversion
- Compute color of every visible surface point
  - Lighting, shading, texture mapping

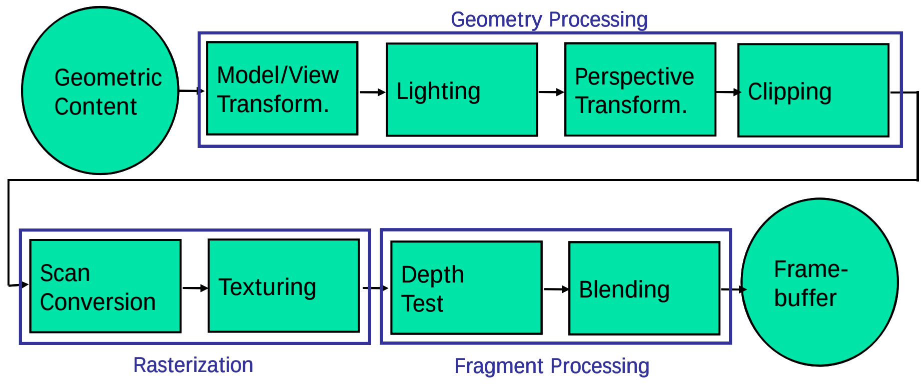


Rendering Pipeline

- What is it? All of this:
  - Abstract model - sequence of operations to transform geometric model into digital image
  - Abstraction of how graphics hardware works
  - Underlying API (application programming interface) model for programming graphics hardware
    - OpenGL
    - Direct 3D
- Actual implementations vary

Copyright A. Sheffer, 2012, UBC

The Rendering Pipeline

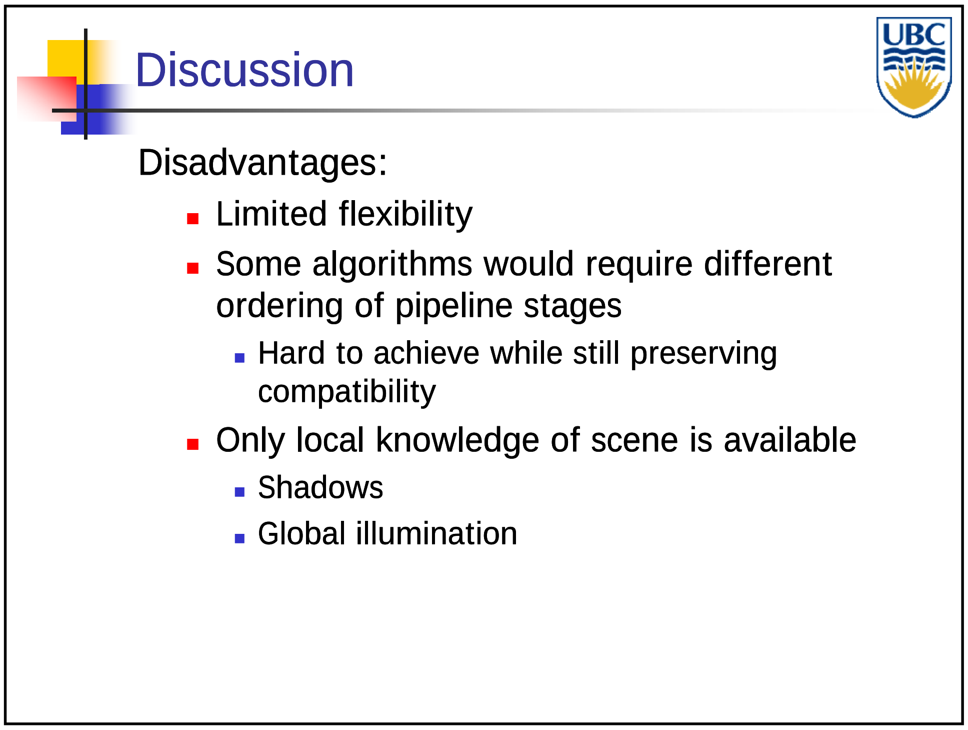


Discussion

Advantages of pipeline structure

- Logical separation of different components, modularity
- Easy to parallelize:
  - Earlier stages can already work on new data while later stages still work with previous data
  - Similar to pipelining in modern CPUs
  - But much more aggressive parallelization possible (special purpose hardware!)
  - Important for hardware implementations!
- Only local knowledge of the scene is necessary

Copyright A. Sheffer, 2012, UBC



Discussion

Disadvantages:

- Limited flexibility
- Some algorithms would require different ordering of pipeline stages
  - Hard to achieve while still preserving compatibility
- Only local knowledge of scene is available
  - Shadows
  - Global illumination

Copyright A. Sheffer, 2012, UBC