CSE251
Basics of Computer Graphics
Module: Preliminaries

Avinash Sharma

Spring 2017

Overview

Introduction

History

Course Organization

Preliminary Concepts

Next Class: Geometric Transformations

What is Computer Graphics?

- Techniques and tools to generate realistic images on the computer
- How?
  - Create representations and “models” of the world
  - Create algorithms to produce ultra-realistic images.
  - Do these fast.
- What is the **Computational Process**?

Abstract – Represent – Process

- Success of computers: Applying this successfully to different application areas!

Digitial or Computer Revolution

- Changed the world greatly in the past 20-30 years!
- How? **Digitize** different things/concepts/ideas/...
  - Digital preseravation, replication, etc., are very cheap
- Initially: Ease tediums or difficulty of activities
  - Aircraft design, payroll, Efficient book-keeping, etc.
- Later: Improve and transform the process
  - Electronic account-books to networked banks to online banking to virtual money to ...
- Enablers: Digital Representation, Efficient Processing and Manipulation, Quick Communication
- And ... reversing the digitization process

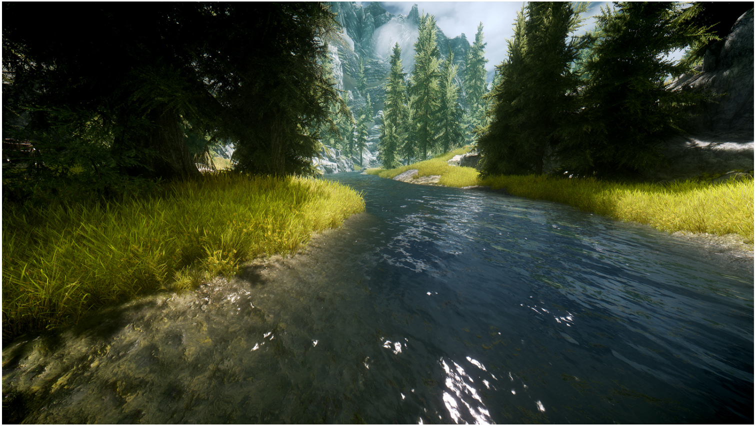
Some Computational Processes

- **Music:** Digitize using microphones and analog-to-digital conversion, process to remove noise, store/transmit as MP3 files, playback using D-to-A and speakers
  - Similarly, Video, Skype, etc.
- **Weather Prediction:** Capture parameters from locations, apply metereological models, process at different levels of detail, predict
  - Drug design, molecular dynamics, more science
- **Computer Games:** World and its rules set by designer, some aspects controlled by players, interaction with objects according to rules, show results to players
  - Several simulations, Virtual Reality, etc.

What's this ?

MX SW 6.5 mm
Tracer
99 | 4
300 m
HE Grenade
x2
2 m
6: Grenadier (SwoXx)
3: "Grid 027011"
3: "I am ready."
6 [Will]: "Grid 027011"
6 [Will]: "Waiting."
6 [Will]: "Awaiting orders."

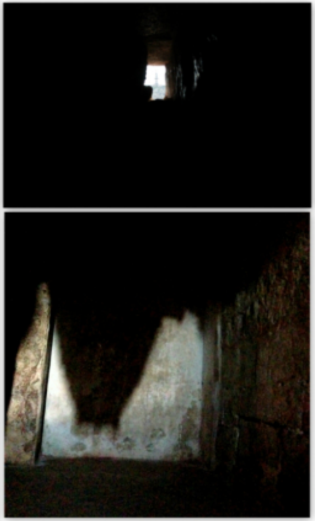
What's this ?



Producing Realistic Images

- Represent physical world using basic "primitives"
  - Basic geometric shapes and objects
  - Efficiency vs utility: Use simple and useful primitives
  - Break up complex scenes into available components
  - Efficiency: **Smallness in size** and **ease of operation**
- Process of image generation from the representation
  - Ape the best that we know: Human eyes
  - (Digital) Cameras approximate the eye in our world
  - Pin-hole camera model approximates the eye conceptually
  - Mathematics of pin-hole cameras known
  - Apply pin-hole camera computationally

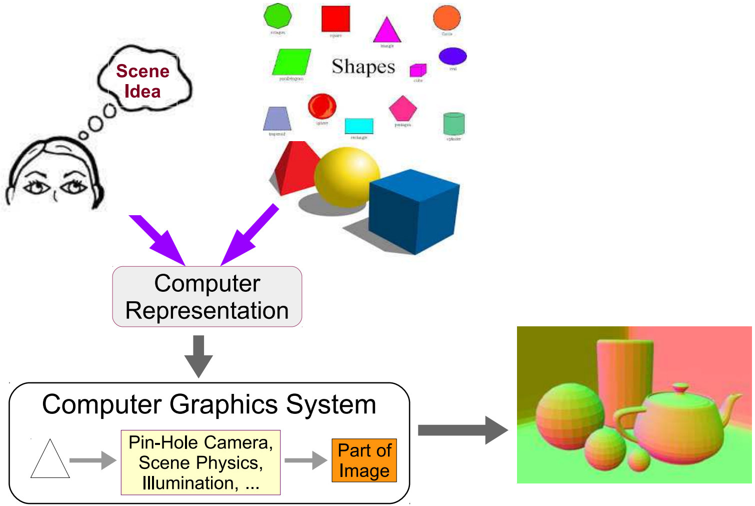
Pin-hole Camera Model



Producing Realistic Images

- Transform a primitive to a camera image correctly
  - Apply geometrical pin-hole camera model on the primitive
  - A series of computations to map primitive to image
  - Paint the picture based on physical properties of objects
- Apply the same to the scene consisting of primitives
  - Evaluate how multiple primitives interact or interfere
  - Paint the physically correct picture of the whole scene
- Do all this efficiently
  - Millions of primitives. High resolution images
  - Complex objects with fine structure and properties
  - Update image fast for application. Real-time for games!

Graphics Process



Application Areas

- User interfaces
- Computer aided design (Civil/Mech/VLSI)
- Visualization of scientific & engineering data
- Art
- Virtual Reality
- Entertainment: Great computer games!
- Special effects in movies. **Whole movies themselves!!**

Quick History

- Whirlwind Computer (1950) from MIT had computer driven CRTs for output.
- SAGE air-defense system (mid 50s) had CRT, lightpen for target identification.
- Ivan Sutherland's Sketchpad (1963): Early interactive graphics system.
- CAD/CAM industry saw the potential of computer graphics in drafting and drawing.
- GE's DAC system (1964), Digitek system, etc.
- Systems were prohibitively expensive and difficult to use.

Quick History (cont.)

- Special display processors or image generators were used for high-end graphics.
- Workstations by Silicon Graphics: early eighties.
- Graphics was expensive, escoteric, and hence rare!
- **A parallel:** Computing became “popular” only after mass-produced personal computers became a reality in mid 80s. Before that, bulky, expensive, and rare devices.
- **Circle of Computing Revolution:** *More users* lead to *greater revenues/returns* which affords *more research* which result in *better/cheaper computers* which in turn bring *yet more users*. And this continues!!

Popular Graphics

- Graphics became "popular" only after mass-produced *Graphics Processing Units* (**GPU**s) or *graphics accelerators* came into existence.
- Graphics Accelerators: on board hardware to speed up graphics computations.
- Accelerators were expensive until end nineties!
- Very high end performance is available economically today. Getting part of the CPU chip these days.
- **Computer Games** provide the fuel for fast growth

Graphics Programming

- Device dependent graphics in early days.
- 3D Core Graphics system was specified in SIGGRAPH 77. (Special Interest Group on Graphics)
- GKS (Graphics Kernel System): 2D standard. ANSI standard in 1985.
- GKS-3D: 1988.
- PHIGS: Programmer's Hierarchical Interactive Graphics System. (ANSI 1988)

Graphics Programming (cont.)

- **OpenGL**: current ANSI standard.
  - Evolved from SGI's GL (graphics library).
  - Window system independent programming.
  - GLUT (utility toolkit) for the rest.
  - Popular. Many accelerators support it.
- DirectDraw/Direct3D: Microsoft's attempt at it!
- **WebGL:** OpenGL to be used for web programming that is now gaining popularity
- **OpenGL ES:** Slightly reduced version for mobile devices, which will be the prime computing platform
- Desirable: High level toolkits.

Course Content

- 2D & 3D Graphics: Concepts, Mathematics, Hierarchical Modelling, Algorithms. Practice in OpenGL.
- Representation: Lines & Curves, Surfaces, Solids.
- Drawing algorithms: Primitives, visibility, efficienty
- Lighting and Shading: Simluating the physics of image generation
- Ray Tracing: If we get time

Background Required

- Good programming skills in C/C++.
- Geometry/Linear Algebra: Points, vectors, matrices, transformations, etc.
- Trigonometry basics.
- Data structures.
- Java for Web or Mobile graphics
- **Good imagination. Ability to visualize in 3D**

Text Books and Reference

- **Computer Graphics with OpenGL** by Hearn and Baker, Third edition. Indian Edition available.

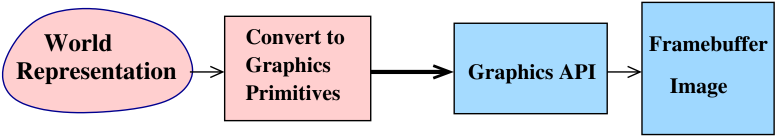
Additional books:

- Fundamentals of Computer Graphics by Peter Shirley.
- Computer Graphics: Principles & Practice by Foley, van Dam, Feiner, Hughes. Indian Edition available.
- Interactive Computer Graphics: A Top-Down Approach Using OpenGL, Fifth edition by Edward Angel.
- OpenGL Programming Guide by Neider, et. al.

Course Management

- Homework assignments, Programming assignments, lab test, mid-term tests, final exam
- Weightages of different components:
  50-60% for the two exams.
  30-40% for programming assignments
  10% for (Written assignments, Bonus, Quiz, etc.)

Graphics Process



- **Model** the desired world in your head.
- **Represent** it using natural structures in the program. Convert to standard primitives supported by the API
- **Processing** is done by the API. Converts the primitives in stages and forms an image in the framebuffer
- The image is displayed automatically on the device

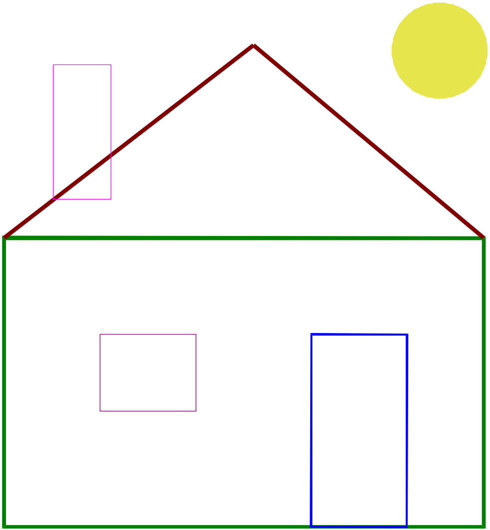
How to Draw A House?

- Compose out of basic shapes

```
drawRectangle(v1, v2, v3, v4);     // Main part
drawTriangle(v2, v3, v5);          // Roof
drawRectangle( ... );              // Door
drawRectangle( ... );              // Window
drawRectangle( ... );              // Chimney
drawCircle( ... );                 // Sun
```

- That's all, really!

Resulting House



Graphics Primitives

- Points: 2D or 3D. (x, y) or (x, y, z).
- Lines: specified using end-points
- Triangles/Polygons: specified using vertices
- Why not **circles, ellipses, hyperbolas**?

Graphics Attributes

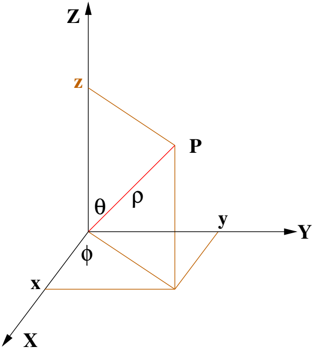
- Colour, Point width.
- Line width, Line style.
- Fill, Fill Pattern.

Point Representation

- A point is represented using 2 or 3 numbers $(x, y, [z])$ that are the projections on to the respective coordinate axes.
- Fundamental shape-defining primitive in most Graphics APIs. Everything else is built from it!
- Represented using byte, short, int, float, double, etc.
- The scale and unit are application dependent. Could be angstroms or lightyears!
- Points undergo transformations: **Translations, Rotations, Scaling, Shearing.**

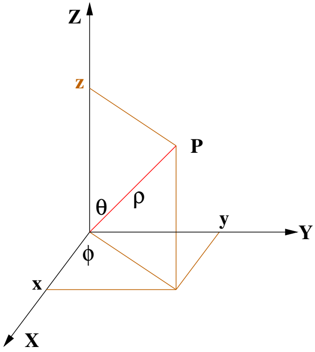
3D Coordinates

- Cartesian: (x, y, z).
- Polar: (ρ, θ, ϕ)
- $z =$
  $y =$
  $x =$
- $\rho =$
  $\phi =$
  $\theta =$
- Elevation: θ, Azimuthal: ϕ



3D Coordinates

- Cartesian: (x, y, z).
- Polar: (ρ, θ, ϕ)
- $z = \rho \cos\theta$
  $y = \rho \sin\theta \sin\phi$
  $x = \rho \sin\theta \cos\phi$
- $\rho^2 = x^2 + y^2 + z^2$
  $\phi = \tan^{-1}(y/x)$
  $\theta = \tan^{-1}(\sqrt{x^2+y^2}/z)$
- Elevation: θ, Azimuthal: ϕ



Translation

- Translate a point $P = (x, y, [z])$ by $(a, b, [c])$.
- Points coordinates become $P' = (?, ?, ?)$.
- In vector form, $P' = ?$.

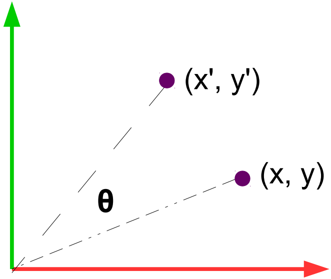
Translation

- Translate a point $P = (x, y, [z])$ by $(a, b, [c])$.
- Points coordinates become $P' = (x + a, y + b, [z + c])$.
- In vector form, $P' = P + T$, where $T = (a, b, [c])$.
- Distances, angles, parallelism are all maintained.

2D Rotation

- Rotate about origin CCW by θ.
- $x' = ?, \quad y' = ?$
- Matrix notation: $P' = R\ P$

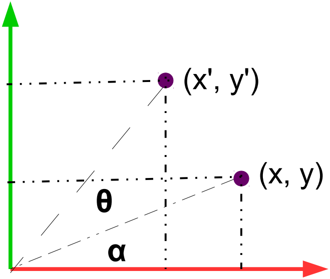
$$\begin{bmatrix} x \\ y \end{bmatrix}' = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2D Rotation

- Rotate about origin CCW by θ.
- $x' = ?, \quad y' = ?$
- Matrix notation: $P' = R\, P$

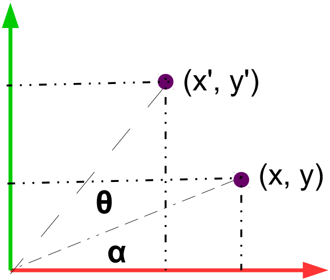
$$\begin{bmatrix} x \\ y \end{bmatrix}' = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



2D Rotation

- Rotate about origin CCW by θ.
- $x' = x\cos\theta - y\sin\theta$,
  $y' = x\sin\theta + y\cos\theta$.
- Matrix notation: $P' = R\ P$

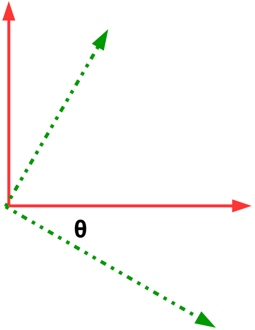
$$\begin{bmatrix} x \\ y \end{bmatrix}' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

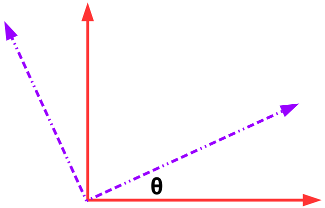


2D Rotation: Observations

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- Orthonormal: $R^{-1} = R^T$
- Rows: vectors that **rotate to** coordinate axes
- Cols: vectors coordinate axes **rotate to**
- Invariants: distances, angles, parallelism.





Next Class

- Basic and Composite Geometric Transformations.