Computer Graphics - Week 8

Questions about Last Week?
Comments about the Assignment

- Deadline for 2nd assignment extended to Friday March 12 at 5:30 pm
  - Regular rules for late submission apply

- Post-mortem about 1st assignment
  - We will subtract points for bad Readme files
  - Programs have to run!!
    - At least tell us they don't work.
  - Programming should involve some upfront thinking
    - Drawing each planet separately is somewhat inelegant
    - Formatting and Comments!!!
  - Questions are encouraged...
    - but read the assignment and read our answers to your questions
  - If not explicitly stated otherwise, teamwork is not permitted

Overview of Week 3

- Aliasing and Anti-aliasing

- Summary of raster graphics pipeline

- VRML
Anti-aliasing

- Aliasing are artifacts introduced by sampling the object geometry, e.g. staircasing, broken up polygons, blinking objects in animations etc.

- First, a short overview of sampling theory to further the understanding of sampling artifacts

- Then, we will study different techniques to limit or eliminate aliasing effects: anti-aliasing

- Last, we will look at implementation techniques to implement anti-aliasing, e.g. using the alpha channel

Aliasing: Examples

- Staircasing
- Broken-up objects
- Blinking (missed) objects
Aliasing: Root Cause

- Aliasing is caused by sampling objects with insufficient sampling frequency
  - Shannon's sampling theorem states that reconstruction of a sampled signal can only be accomplished if the signal is sampled with a frequency at least with twice the highest frequency component of the signal.
  - In practice, even higher sampling frequency is necessary.

- Most objects in computer graphics contain infinite frequency components
  - At the edges the signal changes instantaneously
  - Therefore, aliasing is a ubiquitous problem in graphics

Anti-aliasing: Super-sampling

- An obvious remedy to aliasing is to increase the sampling frequency
  - This means to render images at a higher pixel resolution
  - Yet, super-sampling is still a point sampling process

- Obviously, this will alleviate but not solve the problem
  - Super-sampling misses fewer objects + creates smaller staircases
  - However it does not ensure proper sampling of all features
Anti-aliasing: Area Sampling (1)

- Point sampling is a hit-or-miss proposition
- Instead of that binary decision, we prefer to have a sliding scale that indicates how much an object contributes to a pixel
- This approach is called area sampling

Anti-aliasing: Area Sampling (2)

- Compute the contribution an object makes to a pixel
  - Primitives are assumed to cover a finite area, even points and lines
- Computation either analytical and by oversampling
  - Analytical approaches
    - Compute the exact pixel area covered by a primitive for all pixels contributing to the anti-aliasing filter
    - Weight the area by that filter function
    - Write the pixel
  - Oversampling, approximates true area-sampling
    - Sample the image at a higher frequency
    - Compute a weighted average of all subpixels within the filter area
    - Write the pixel
Anti-aliasing: Filtering (1)

Once the primitives contribution to the pixel has been determined, it must be translated into a pixel intensity.

The simplest approach is to simply use the coverage percentage as a scale factor for the object’s intensity:
- Creates non-linear transitions between pixels
- Distance of the object from the pixel center is ignored
- Box filter

Anti-aliasing: Filtering (2)

For a given pixel, the contributions of objects in a neighborhood around the pixel must be combined using a weighted average.

The averaging method must satisfy several criteria:
- This neighborhood is larger than 1 pixel
- An object contributes less if it is farther away from the pixel center
- The total contribution of an object to all affected pixels is constant
Anti-aliasing: Filtering (3)

- Filter shapes with a fall-off away from the pixel center and extending beyond the pixel borders satisfy these criteria.

Anti-aliasing: Filtering (4)

- Different filter shapes satisfy these criteria
  - Triangular filter, Gaussian filters, ...

- However the actual image quality still differs between these filters
  - In general the more complicated the filter, the better the results

- This observation is explained by sampling theory
Sampling Theory: Overview

- Sampling theory investigates and describes the effects of sampling continuous signals.

- Examples of signals:
  - Time-variant, e.g. sound
  - Space-variant, e.g. images
  - One-dimensional, e.g. scanline
  - Two-dimensional, e.g. image

- The signal is considered either in the time/spatial domain of the frequency domain.

Sampling Theory: Sampling Process

- Signal sampled at intervals:
  - The size of these intervals determines the sampling frequency.
  - The signal is multiplied by a comb function, consisting of several pulses (a.k.a. Dirac function or Delta function).

- The sampled signal consists of unconnected (discrete) values.

- The reconstructed signal constructs a new signal that approximates the original signal.
  - Reconstruction is done using a filter applied to the sampled signal.
Sampling Theory: Scan Conversion

- Scan Conversion samples the input geometry
  - Determine the visible object and its attributes at the pixel center
  - Pixel centers form a two-dimensional array of Delta functions

Sampling Theory: Frequency Domain

- Fourier Analysis allows to decompose a signal into a mix of contributing sine waves
  - Each sine wave has amplitude and phase
  - For periodic signals:
    - Fourier series, i.e. (infinite) sum of discrete frequencies
  - For aperiodic signals:
    - Fourier transformation, i.e. (infinite) integral various frequencies

- The total of all frequencies forming a signal is called the spectrum
**Sampling Theory: Notation**

- We denote the signal with lower case letters
  - For example, $f(x)$ or $g(x,y)$

- The Fourier transform is shown in capital letters
  - For example, $F(u)$ or $G(u,v)$
  - The variable $u$ is the frequency

**Sampling Theory: Signal and Spectrum**

- Fourier transformation connects signal and frequency domain
  - Fourier transform returns amplitude and phase
  - Often, only the amplitude spectrum is of interest
  - DC value is $F(0)$

- Typically spectrum falls off rapidly towards high frequencies
  - Higher frequencies indicate higher energy
Sampling Theory: Filters

- A Filter modifies the frequency spectrum
  - Scales every component of the spectrum
  - \( G(u) = F(u) \ast H(u) \)

- Several filters are frequently used
  - Low-pass, eliminate high frequencies
  - High-pass, eliminate/reduce low frequencies
  - Band-pass, attenuate all frequencies outside a certain range

Sampling Theory: Convolution (1)

- Fourier transformation relates multiplication in one domain with convolution in the other domain

- Convolution of signal \( f \) and filter \( h \) at a point \( x \) is the product of \( f \) and \( h \) centered at \( x \).

\[
g(x) = f(x) \ast h(x) = \int_{-\infty}^{\infty} f(t) \cdot h(x-t) \cdot dt
\]
Sampling Theory: Convolution (2)

- Convolution computes an average of the signal $f$ around $x$ weighted by the filter function $h$.

Sampling Theory: Sampling (1)

- Sampling is multiplying the original signal with a comb function.
- This corresponds to a convolution of the signal with Fourier transform of the comb function.
- The Fourier transform of a comb function is another comb function.
  - The distance between the Dirac pulses is determined by the sampling frequency $\omega = 2\pi / t_s$. 

**Sampling Theory: Sampling (2)**

- Convolving the signal with a comb function replicates the spectrum of the signal centered around the Dirac pulses.

- If the sampling frequency is too low, the replicated spectra overlap. This generates aliasing.

**Sampling Theory: Nyquist Frequency**

- Nyquist frequency is the minimum sampling frequency where spectra do not overlap.

- Therefore:

\[ f_0 = \frac{1}{t_0} \geq 2 \cdot f_{\text{max}} \]
Sampling Theory: Low-pass Filter (1)

- In order to eliminate the extra copies of the spectrum, a low-pass filter is applied.

- If the copies of the spectrum do not overlap, the signal can be fully reconstructed.

- Otherwise, the signal is only approximated.
  - High frequencies are lost and the signal appears blurred.

Sampling Theory: Low-pass Filter (2)

- Filtering
  - Multiplication in the frequency domain
  - Convolution in spatial domain

- Ideal low-pass filter
  - A.k.a. box filter
  - Sharp cut-off frequency
  - Corresponds to sinc function in the spatial domain
  - \( \text{sinc}(x) = \frac{\sin(x)}{x} \)
Sampling Theory: Sinc function

- Amplitude $A$ and zero crossings of the sinc function are determined by the sampling frequency $W$
  - $A = 2W$ will preserve the energy of the signal
  - Lower sampling frequency widens the sinc function and reduces its amplitude

- Sinc is problematic
  - Very wide non-zero support
  - Negative lobes

Sampling Theory: Other Filters (1)

- Box filter
  - Sinc in frequency domain
  - Wide non-zero, negative support
  - Unweighted average, very simple, but large errors

- Triangular filter
  - $\text{Sinc}^2$ in frequency domain
  - Smaller, positive support
  - Weighted average, fairly simple

- Gaussian filter
  - Gaussian in frequency domain
  - Small, positive support
  - Better weighted average
  - Computationally expensive
Sampling Theory: Other Filters (2)

- **Windowed filters**
  - E.g. windowed sinc function, i.e.
  
  \[
  sw(x) = \begin{cases} 
  \text{sinc}(x) & \text{if } |x| < \text{const.} \\
  0 & \text{if } |x| \geq \text{const.} 
  \end{cases}
  \]

- **Requirements for any filter functions**
  - The integral over the filter function is 1.
    This ensures that the signal is not attenuated or amplified
  - The filter function should have finite support in the spatial domain.
    This makes it practical to compute the filter function.

Sampling Theory: Summary

- **Scan conversion** forms a sampling process with subsequent reconstruction

- **Sampling theory** tells us that reconstruction of a sample signal can only be accomplished if the sampling frequency is above the Nyquist frequency
  - Then the copies of the signals spectrum do not overlap
  - To meet this condition, the signal must be band-limited

- **Reconstruction is accomplished with a low-pass filter**
  - Low-pass filtering means to compute a weighted average in the spatial domain
  - The spatial domain is larger than one pixel!
  - Different low-pass filters can be used.
Anti-aliasing: Implementation

- Proper filtering of the rendered primitives can be difficult and computationally expensive

- Various techniques approximate the proper calculation of the area contributions
  - Super-sampling
  - Alpha channel for image compositing

- Additional problems arise if hidden-surface removal is combined with anti-aliasing
  - How much does the object cover the pixel?
  - How much of the object is visible in the pixel?

Anti-aliasing: Super-sampling (1)

- The area covered by the object is computed by point sampling the pixel itself
  - Sample locations are known as sub-pixels
  - Sub-pixel coverage approximates the true coverage
  - Simple to compute, applicable to all primitive types

- Filtering weights and combines the sub-pixel values according to the filter function to the final pixel value
Anti-aliasing: Super-sampling (2)

- Determine visibility only at pixel center
  - Creates wrong results if visibility changes in the pixel
  - Requires only standard z-buffer and frame buffer

- Determine visibility at every sub-pixel
  - Proper resolution of visibility within the pixel
  - Adjust depth value to sub-pixel location
  - Requires high-resolution z-buffer for sub-pixel color / z

---

Anti-aliasing: Compositing (1)

- Compositing was developed for combining images
  - Images are combined pixel by pixel
  - First described by Porter & Duff in 1984

- At every pixel we store in the alpha channel the current coverage information

- For a new fragment we compute new color and alpha information based on pixel and fragment color/alpha

- We will now look at compositing two fragments, each of which has only one edge crossing its pixel area
  - More edges per pixel are rare cases and not treated explicitly
  - No edges are covered by setting \( \alpha = 1 \)
Anti-aliasing: Compositing (2)

- The alpha value does not provide information about how the object is oriented within a pixel
  - There is no clear definition on how to combine current pixel value and new fragment value
  - Several possibilities for how two fragments might interact

- For image synthesis and anti-aliasing only one is relevant: A over B
  - See Foley et al. chapter 17.6 for other operations

\[
\alpha_A = 0.3, \quad \alpha_B = 0.5
\]

Anti-aliasing: Compositing (3)

- A over B
  - Determine which objects is in front of the other (visibility)
  - Both fragments contribute
  - Assume random orientation of edges within the pixel
  - Then, on average \((1 - \alpha_A)\) of fragment B is visible

\[
C = \alpha_A \cdot A + (1 - \alpha_A) \cdot \alpha_B \cdot B
\]
\[
\alpha = (1 - \alpha_A) \cdot (1 - \alpha_B)
\]
Anti-aliasing: Compositing (4)

Notice, that the fragment colors A and B are always used weighted by the coverage values $\alpha$

To avoid computing the term $\alpha^*A$ at every pixel, colors are stored premultiplied by $\alpha$
- Instead of $(R,G,B,\alpha)$ we store $(\alpha R, \alpha G, \alpha B, \alpha)$

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

Anti-aliasing: OpenGL (1)

Anti-aliasing in OpenGL may differ for different OpenGL implementations
- Quality of anti-aliasing controlled via "hints" (glHint)

Point and Line anti-aliasing
- glEnable(GL_POINT_SMOOTH) + glEnable(GL_LINE_SMOOTH)

Polygon anti-aliasing
- If drawn as points or lines, point/line anti-aliasing applies
- Otherwise, glEnable(GL_POLYGON_SMOOTH). This will generate coverage information in a fragment's alpha value
- Then setup the alpha-blending function to blend between incoming fragment and stored pixel value
Anti-aliasing: OpenGL (2)

- Supersampling is supported in OpenGL via the Accumulation Buffer
- The accumulation buffer is a very deep frame
- Successive rendering passes are accumulated in the accumulation buffer
- At the end the pixel values are divided by the number rendering passes (averaging)
- Supersampling is implemented by jittering the scene position slightly (subpixel distance) between passes

Anti-aliasing: Summary

- Aliasing result of under-sampling the primitives
  - Geometric primitives introduce infinite frequencies along edges
- Sampling theory indicates that the signal must be band-limited before sampling to allow full reconstruction of the signal
  - The ideal low-pass filter is hard to implement
  - Approximations mean the computation of a weighted average of the pixel area covered by a primitive (area sampling)
- Area sampling is hard to perform accurately
  - Supersampling and compositing approximate correct area sampling and resolution of visibility between several fragments partially covering into a pixel
VRML

- VRML = Virtual Reality Modeling Language
  - Derived from SGI's OpenInventor
  - Version 2.0 has been adopted by ISO as VRML'97 standard

- VRML was designed for definition, interaction and transmission of 3D models in a web-environment
  - Based on a scene-graph model
  - Supports several non-graphics constructs to build virtual worlds, e.g. audio, event routing, customizable nodes (prototypes), concept of time, scripting interface

- We will focus here on the graphics-specific elements of VRML, i.e. scene graphs and basic event processing
  - Also see paper handed out in class
VRML: Overview

- Scenes are constructed by building a scene-graph
  - Basically identical to the model hierarchies we discussed earlier
  - Different nodes represent objects, transformations, attributes and grouping semantics
  - More formally nodes are classified as geometry (shape) nodes, appearance nodes, geometric property nodes and grouping nodes

- Each node has none, one or several fields that set relevant node parameters
  - Each node field has a name and a default value
  - Fields not explicitly set will assume their default value

VRML: Files

- VRML files have the extension ".wrl" for "world"

- VRML files
  - Very brief header
  - Description of the world

```
#VRML V2.0 utf8
Shape
{
  appearance Appearance
  {
    material Material
    {
      diffuseColor 1.0 0.0 0.0
      specularColor 1.0 1.0 1.0
      shininess 10
    }
  }
  geometry Cone
  {
    bottomRadius 1.0
    height 2.0
  }
}
```
VRML: Nodes

- Nodes compose the scene graph describing the world

- Syntax
  - Unnamed nodes:
    ```xml
    <nodeType> { <body> }
    ```
  - Named nodes:
    ```xml
    DEF <name> <nodeType> { <body> }
    ```
  - Named nodes can be instantiated anywhere a node is expected:
    ```xml
    USE <name>
    ```
  - The body of a node specifies the values of its fields and events it can send and receive

```
#VRML V2.0 utf8
Shape
{
  appearance Appearance
  {
    material Material
    {
      diffuseColor 1.0 0.0 0.0
      specularColor 1.0 1.0 1.0
      shininess 10
    }
  }
  geometry Cone
  {
    bottomRadius 1.0
    height 2.0
  }
}
```

VRML: Fields

- Fields set a node's parameters
  - All fields have default values
  - Default values are used if no specific value is assigned

- Syntax
  - Single valued field (prefix SF)
    ```xml
    <fieldName> <fieldValue>
    ```
  - Multi-valued field (prefix MF)
    ```xml
    <fieldName> [<fieldValues>]
    ```
  - Multiple values are separated by commas
  - The brackets may be omitted if only one value is assigned

```
#VRML V2.0 utf8
Shape
{
  appearance Appearance
  {
    material Material
    {
      diffuseColor 1.0 0.0 0.0
      specularColor 1.0 1.0 1.0
      shininess 10
    }
  }
  geometry Cone
  {
    bottomRadius 1.0
    height 2.0
  }
}
```
VRML: Field Types

- Every field can only assume values of a specific type
- Types include
  - Bool, Int32, Float, Vec2f, Vec3f, String,
  - Color, Image, Time, Rotation
  - Node
- Most node types are available as single-valued and multi-valued versions (except Bool and Image)

VRML: Node Types (1)

- Shapes and Geometry
  - Box, Cone, Coordinate, Cylinder, ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, Normal, PointSet, Shape, Sphere, Text
- Appearance
  - Appearance, Color, FontStyle, ImageTexture, Material, MovieTexture, PixelTexture, TextureCoordinate, TextureTransform
- Grouping
  - Anchor, Billboard, Collision, Group, Inline, LOD, Switch, Transform
- Environment
  - AudioClip, Background, DirectionalLight, Fog, PointLight, Sound, Spotlight
VRML: Node Types (2)

- **Viewing**
  - NavigationInfo, Viewpoint

- **Animation (Interpolators)**
  - ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator, ScalarInterpolator, TimeSensor

- **Interaction (Sensors)**
  - CylinderSensor, PlaneSensor, ProximitySensor, SphereSensor, TouchSensor, VisibilitySensor

- **Miscellaneous**
  - Script, WorldInfo

VRML: Shape and Geometry Nodes (1)

- **Shape Node**
  - Container node to collect geometry and appearance data
  - Use various other nodes to specify these fields

- **3D Primitives**
  - Box: axis-aligned box with specified dimensions
  - Cone: cone around Y axis, centered at origin
  - Sphere: obvious
VRML: Shape and Geometry Nodes (2)

- Indexed Face Set
  - Used to build shapes as a collection of planar polygons
  - Polygons are defined from indices into arrays of vertices, colors, normals, and texture coordinates

- The node contains at least
  - a list of vertex coordinates (field 'coord')
  - a list of vertex indices defining the faces (field 'coordIndex')

- Optionally, attributes can be defined at the vertices
  - Colors, normals, texture coordinates

- IndexedLineSet is a similar construct for constructing a collection of lines

---

VRML: Shape and Geometry Nodes (3)

- Example: Pyramid

```vrlml
#VRML V2.0 utf8
Shape [
  geometry IndexedFaceSet [
    coord Coordinate [
      point [-1 -1 0, 1 -1 0, 1 1 0, -1 1 0, 0 0 2]
    ]
    color Color [
      color [1 0 0, 1 0 1, 0 0 1, 0 1 1, 1 1 0]
    ]
    colorPerVertex TRUE
    coordIndex [0 3 2 1 -1, # bottom 0 1 4 -1, 1 2 4 -1, 2 3 4 -1, 3 0 4 -1]
  ]
]```

---
VRML: Grouping Nodes (1)

- Simplest one is Group
  - Collects several children into a group
  - Field 'children' contains those node

- Transformation node is similar to Group node
  - Scales, rotates and translates (in that order) its children

- Billboard
  - Always orients the local Z-axis to point to the viewer by rotating all children around specified axis
  - Supports the display of objects that always face the viewer, e.g. "flat trees", signs, etc.

VRML: Grouping Nodes (2)

- Anchor node
  - Loads another URL when the user clicks on any of the children nodes
  - Replaces the current world

- Inline node
  - Reads content of another URL and merges its nodes into the current world (similar to #include in C preprocessor)
  - Does not replace the current world
VRML: Grouping Nodes (3)

- **Switch**
  - Selects one amongst its children for display
  - The 'whichChoice' field can be changed via events
  - Useful e.g. for simple animation or state changes

- **LOD (level of detail)**
  - Selects from several representations (levels) of the same object
  - Switching depends on the distance from the viewer
  - Switching thresholds are specified for each level
  - Useful to increase rendering efficiency by displaying simpler representation for distant objects

VRML: Appearance Nodes

- **Appearance nodes are container nodes for various surface properties**
  - Materials
  - Textures

- **Textures are specified as images**
  - PixelTexture: specifies the image in the VRML file
  - ImageTexture: specifies the image by reference to an external file
  - MovieTexture: time dependent texture loaded from an external MPEG-1 file
VRML: Lighting Models and Materials

- Lighting models and material properties are very similar to OpenGL
  - Lights are placed in the scene like regular objects. The subject to standard transformations
  - Check out the following nodes: DirectionalLight, PointLight, SpotLight, Fog, and Material

VRML: Viewing and Navigation

- Viewpoint selection and navigation is primarily controlled by the browser

- Viewpoints can be specified in the VRML file
  - Viewpoint node allows to predefine several nodes
  - Viewpoints can be selected by the browser
  - Viewpoints can manipulated automatically by changing the parent transformation node

- Navigation specifies how the browser manipulates the viewpoint
  - NavigationInfo node supports various modes, e.g. WALK, FLY, EXAMINE
VRML: Authoring and Viewing

- Obviously, creating VRML files using a text editor is (at best) inconvenient
  - The text-based format makes it easy to generate VRML through authoring applications, e.g. 3D editors, scientific visualizations etc.

- VRML files are viewable through various tools
- Most popular viewers are known as VRML browser
  - Plug-ins available for most Web-browser

Summary

- Aliasing and Anti-aliasing
  - Route cause is sampling below Nyquist frequency
  - Area-sampling is an implementation of low-pass filtering the image
  - Super-sampling and compositing

- VRML
  - Basic concepts
  - Syntactical elements
  - We have not covered advanced VRML features
Homework

- Prepare for the midterm exam

- The exam is a closed book exam. You are allowed to bring 1 letter-sized sheet with notes.

- Read the lecture notes and the corresponding chapters in the textbook
  - You must demonstrate understand of the material and can apply it
  - Fully understand the polygon raster pipeline, i.e. sequence, function, and algorithms
  - You should be able to apply algorithms etc. Except in very simple cases, you do not need to derive formulae.
  - Exam questions will query knowledge, understanding and ability to use what you learned

Assignment

- Create a VRML cityscape
  - Streets, buildings, cars, people, trees, ...

- Although not the preferred way, create the world from scratch, i.e. using a text editor

- There are very few requirements

- So ...
  - go out, be wild, have some fun

- And ...
  - come back with a cool VRML world
Next Week ...

▶ Spring break !!!