# CG2 SS2011, Eduard Gröller

Onur Dogangönül

November 29, 2011

## Contents

# 1 Graphical Programming

## 1.1 Introduction

- Process graphical objects and data.
- Early standards:
  - GKS (Graphical Kernel System, 1977)
    - 1st low-level ISO standard
    - Provides a set of drawing features for 2-dim vector graphics
  - PHIGS
    - Programmer's Hierarchical Interactive Graphics System (ISO/IEC 9592)
    - Defines a scenegraph-API
  - Both standards declined with the rise of OpenGL.
- X Window System
  - Software/network protocol provides basis for GUI for networked computers

  - HAL (hardware abstraction layer): software is written using generalized set of commands
  - Device independence/reuse of programs on computers that implement X

- OSF/Motif: GUI specification/widget toolkit for building applications that follow X Window System

- Open Inventor:
  - High level 3D graphics API
  - Scene graph oriented
  - E.g. engines, sensors, manipulators

- VRML
  - Virtual Reality Modeling Language
  - Std file format for representing 3dim interactive vector graphics

- X3D
  - ISO standard XML-based file format representing 3D computer graphics
  - Successor of VRML

- OpenGL .
  - HW independent interface to graphics hw
  - No rendering context or user input management
  - Interface between OS and OpenGL (GLFW etc..)

### 1.2 Java3D

- Builds upon OpenGL/DirectX layer

- Scenegraph API
  - Viewing subtree: definition and parameters for rendering and projection.
  - Content subtree: defines the scene (geometry and transformations).

- Renderer
  - Double buffering (RGBA)
  - Infinitely traversing SG

  - Takes care of transparency
  - Influence of fog, lights, sound
  - Optimized for rendering

- Node States (Capabilities):
  - Detached: After creation, all ops legal
  - Live
    * Node in SG
    * Set permission via setCapability(..)
  - Compiled: Optimized by renderer (mods restricted)

- Node Types
  - Group nodes
    * BranchGroup: grouping of shapes (only detachable/relocable node)
    * TransformGroup: t applied to all children
  - Shape3D
    * Defines object within scene
    * Geometry: polygon related info
    * Attributes: material definition, rendering mode, ...
  - Geometry nodes: coords, normals, RGB(A) color, tex coords, indices (for indexed types) ...

- Helper Classes
  - com.sun.j3d.utils.*
    * SimpleUniverse: fast viewing model setup
    * Mouse transform mapping for interactions
    * Simple geometrical Objects: cube, ...
  - javax.vecmath.*: Tuple, Point, Vector, Matrix, Quaternions with their ops
  - javax.media.j3d.Transform3D: Orth/Persp. projection, scale, rotate, transalte, ...

- Appearance: defines attributes for rendering
  - Polygon mode (points, wireframe, fill)
  - Culling
  - Rendering attrib: Depth buffer, alpha blending
  - Transp/Color if not per vertex

    - Material (reflex coeffs) and texture images

- Lighting:
    - Require bounds delimit area of infl
    - Ambient, Direcitonal (inf dist), pointlight

- Behaviors
    - Events for SG
    - Actions executed if criteria met: time trigger, kbd/mouse event, object picking, collision, frames elapsed, ...
    - Boolean comb of criteria
    - Area of relevance
    - Influence transformation, geometry, ...

- Viewing model
    - Supports multiple Canvas3D objects (stereo rendering, cave).
    - Supports tracking
    - Detailed description of viewer's eyes/ears

# 2 Advanced Modelling

## 2.1 Motivation

- Real world phenomena comprise: complex geometry, large defs, topological changes, fuzzy objs bounds
- Hard to model with meshes: E.g. Smoke, fire, fluids, fur, hair, grasss

## 2.2 Particle Systems

- Modelling of objs changing over time
- Rain, snow, clounds, smoke, fire, waterfalls, water spray, grass,...
- Render certain num of particles
- Changing particle params: Loc, speed, appearance
- Die after a time
- Particle shapes: spheres, boxes or arb models
- Rnd processes gen objs within defined region
- Shapes may be small spheres, ellipsoids, boxes, ...
- Size, color, transparency, movement can vary randomly.

## 2.3 Implicit Modelling

- No fixed shape/topology
- Molecular structs, water droplets, metling objs, muscle shapes
- Shape/Topology change in motion/proximity to other objs
- No seams, oriented surface (in/outside def), differentiable, closed, continuous
- Imp eq doesn't express a var in terms of another var
- Surface of imp model is set of points fullfilling given imp eq
- Also called level curves, iso contours or contour lines.

### 2.3.1 Blobby Objects

- Modelling as distri functions over region of space
- E.g. comb of gaussian density funcs
  - $f(x, y, z) = be^{-ar^2}$
  - r ... distance to center point

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

  - k gaussian bumps at k control points $(x_k, y_k, z_k)$ are comb

$$f(x, y, z) = \sum_k b_k e^{-a_k - r_k^2} - T = 0$$

  - $r_k^2 = (x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2$
  - Param T as threshold, params $a_k, b_k$ adjust shape of ind blobs
  - Neg $a_k, b_k$ model dents
- Metaball model/Soft-object model use density functions that fall of to 0 in finite interval

### 2.3.2 Superquadrics

- Quadrics: cruve in plane, plane in 3d space or hyperplane in nd space etc (given by quadratic func)
- Generalization of quadric reps (Add params to quadric eq)
- Num of add params equal to dimension of obj

- Increased flex for obj shapes
- Superellipse
  - Imp eq: $f(x, y) = \left(\frac{x}{r_x}\right)^{\frac{2}{s}} + \left(\frac{y}{r_y}\right)^{\frac{2}{s}} = 1$
  - $r_x, r_y$ params of ellipse, s param of superellipse (e.g. s = 1, normal ellispe)
  - Also parametric eq for superellipse in reader
- Superellipsoid
  - Imp eq: $f(x, y, z) = \left[\left(\frac{x}{r_x}\right)^{\frac{2}{s_2}} + \left(\frac{y}{r_y}\right)^{\frac{2}{s_2}}\right]^{\frac{s_2}{s_1}} = 1$
  - E.g. normal ellipsoid for s = 1
  - Parametric for superellipsoid in reader
  - Superquadric shapes comb for more complex structs

## 2.4 Procedural Modeling

- Complex geometry build with set of production rules
- Simple primitive and repetitive structs
- Often params that change the appearence.
- Sweeps
  - For trans, rot or other symmetries
  - Spec two dim shape (circle, rect, spline-curves, ...), and curve to move along
  - Pros/Cons
    * Generates complex shapes
    * Hard to render
    * Difficult modelling

## 2.5 Structure Deforming Transformations

- Used in comb with other modeling tech
- Tapering: non-linear scaling, where scaling factors are functions (see matrix in reader)
- Twisting: non-linear rot, where rot params depend on coords of input point
- Bending: non-linear rot, where rot params depend on coords of input point

### 2.6 Other

- Cellular Texture Generation
  - Cellular particle system with cell state, programs and extracellular environments
  - Cell: grp of polygons ith texture and transparency maps
  - Cell state: pos, orient, shape, chemical concentrations (reaction-diffusion), ...
  - Cell programs: go to surface, die if too far away, align, adhere to other cells, divide until surface covered
  - Extra cellular env: neighbor orientiation, concentration, ...
  - E.g. fur - cells similarly oriented like neighbors
- Visualization of Knitwear
  - Sim of 3d struct with instanced volume elements
  - 2d cross section swept along parametric curve
  - Rendering with raycasting (curved rays)
- Fractal Terrain Sim
- Plant Rendering Population spec: Explicit spec (survey, interactive) or procedural gen

# 3 Advanced Modelling 2

### 3.1 Parametric Curves/Surfaces Overview

- Param curves
  - $c : c(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}$ , $u \in [a, b] \subset \mathbb{R}$
  - $x(u), y(u), z(u)$ are diff functions in u
  - Tangent vec: $t(u) = \frac{d}{du}c(u)$
  - c regular in $c(u_0) \iff t(u_0) \neq 0$
  - c regular $\iff$ c can be param that all curve points are regular
- Param surfaces
  - $s : s(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}$ , $(u, v) \in [a, b] \times [c, d] \subset \mathbb{R}^2$

- x,y,z are diff in u and v
- Tangent plane: $t_0(l, m) = s(u_0, v_0) + ls_u(u_0, v_0) + ms_v(u_0, v_0)$
- Normal vec: $n(u_0, v_0) = s_u(u_0, v_0) \times s_v(u_0, v_0)$
- s regular in $s(u_0, v_0) \iff n(u_0, v_0) \neq 0$
- s regular $\iff$ s can be param so that surface points regular

## 3.2 Curves

### 3.2.1 Bezier Curves

- Given n + 1 points $b_0, ..., b_n \in \mathbb{E}^3$ and arb $t \in \mathbb{R}$
- Set $b_i^r := (1-t)b_i^{r-1} + tb_{i+1}^{r-1}$ with $b_i^0 := b_i$
- $b_0^n$ is a curve point
- Points $b_0^{n-1}$ and$b_1^{n-1}$ determine tangent line in $b_n^n$
- Exp parametric rep: $b(t) = \sum_{i=1}^n b_i B_i^n(t)$, $B_i^n(t$ are Bernstein polynoms of nth deg
- Properties
  - Affine invariance: affine transform of control points is affine trans of curve
  - Convex hull: curve lies inside convex hull of control points
  - Enpoint interpolation: first/last control points lie on curve
  - Linear precision: all control points lie on line, bezier curve is a line
  - Variantion diminishing: curve has no more intersection with plane than its bezier polygon
- Disadvantes
  - Changin one control points changes whole curve
  - Degree of curve depends on number of control points: Higher flexibility - more control points
- Important algorithms
  - Degree eval: Add new control points for higher degree without changing shape
  - Subdivision: Increase flex without increasing complexity
- Evaluation: horner sheme like forms for faster eval

### 3.2.2 B-Spline Curves

- Piecewise polynomial curves of k-1 degree
- Degree almost ind from num control points
- Allow local control
- Given n+1 control points (boor points) and knot vector $U = (u_0, ..., u_{n+k})$
- B-spline curve s(u) is piecewise poly curve of order k (degree k-1) with $1 \leq k \leq n + 1$ of form

$$s(u) = \sum_{i=0}^{n} d_i N_i^k(u)$$

- N ... normalized b-spline basis funcs (recursive def)
- Properties
  - Affine inv
  - Convex hull
  - Variation diminishing
  - Local support: changes only in local segments etc..
- Special types
  - Form of knot vector determines special cases
  - Open
  - Closed: Kontrol points closed path
  - Uniform: Spacing between knot values constant (uniform B-spline), fitts better for algorithms due to periodic basis funcs
- Evaluating B-Spline Curves (de Boor algorithm)
  - Generalized de Casteljau, same principle of linear interpolation
  - To evaluate s(u) do recursion (see reader)
  - Direct evaluation (alternative)
    * Find knot span in which parameter value lies
    * Compute non zero basis func
    * Mult value of basis func with corresp control point
- Knot insertion/Knot refinement
  - Knot insertion does not change shape, but refines segmetation
  - Increase flexi of curve, compute derivs, split curves or eval curve
  - de Boor alg also knot insertion alg
  - Algs with simultaneously knot insertion (knot refinement)

### 3.2.3 Rational Curves

- Conic sections and quadrics have rational parametrization
- $c(u) = \frac{1}{w(u)} \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix}, u \in I \subset \mathbb{R}$
- More elegant useful rep using homogeneous coords in $\mathbb{E}^4$
- Original curve interpreted as projection of curve on hyperplane $w(u) = 1$ in $\mathbb{E}^4$
- Rational Bezier Curves
  - $b(u) = \frac{\sum_{i=0}^{n} w_i b_i B_i^n(u)}{\sum_i^n = 0 w_i B_i^n(u)}$
  - If $w_i = 1$ same as polynomial bezier curve
  - Properties
    * Same properties as non rational
    * Projective invariant
    * Does not lie inside control poly if weights neg
    * Weights as additional design param
  - Same algs can be applied for rational curves with homogeneous rep
- Non Uniform Rational B-Spline Curves (NURBS)
  - Most imp and flexi design elements provided in CAD systems
  - Polynomial/Rational Bezier and B-spline curves are subsets of NURBS
  - Properties same as rational bezier curves
  - Chaning weights affects only segment

## 3.3 Surfaces

- Tensor product surfaces
  - Surface as moving curve that changes
  - Changing curve means changing control points
  - Leads to tensor product surface
- Bezier surfaces
  - Tensor product bezier surfaces analog properties to bezier curves
  - Apply algorithm on curves

- B-spline surfaces: Analog properties, again all algs apply
- Bezier Triangles: see reader
- Subdivision Surfaces
  - Generated from base mesh, iteratively smoothing
  - Mesh converges to a limit surface through iterations
  - Charles Loop subdivision
    * For triangle meshes only
    * Split each triangle in 4 smaller
- For more see reader

# 4 Advanced 3D Data Structures

## 4.1 Introduction

- Motivation
  - Diff data sources and apps need diff data reps
  - Requirements must be met
- 3D data structs requirements
  - Exact rep of general obj
  - Comb of objs
  - Linear trans
  - Interaction
  - Fast spatial searches
  - Mem capacity
  - Fast rendering

## 4.2 Point Cloud

- Usage: For fast and simple preview (from digitizer or 3d scanner)
- Specs
  - Data structure: list of points (spatial coords)
  - Transformations: t whole cloud by applying t on each point
  - Combinations: combine clouds
  - Memory Consumption: high mem for accurate reps

  - Rendering
    * Simple points rendering
    * Depth buffer for correct occlusion
    * Nowadays high complexity: polygons often smaller than screen-pixel (point cloud more efficient)

- Pros
  - Fast/easy rendering
  - Exact repwith correct occlusion
  - Fast/easy transformations

- Cons
  - Many points for complex surfaces
  - High memory for that case
  - Limited comb ops

### 4.3 Wireframe Model

- Edges rep
- Used for constr of buildings
- Specs
  - Data structure: list of edges, each edge has 2 verts, each vert has spatial coords
  - Transformations: same as point cloud
  - Combinations: Append point and edge lists
  - Memory Consumption
    * Vert may be starting point of many edges
    * Lots of edges for curved surfs
  - Rendering
    * For devices which only draw lines (vector displays/plotters)
    * Not possible to eliminate hidden edges
    * No hidden line method out of bare wireframe model

- Pros
  - Fast rendering/Easy transformations
  - Gen models by digitizing

- Cons

- High memory for curved surfaces
- Restricted comb possis
- Inexact due to line approximation of curves (no surfs, no occlusion)

### 4.4 Boundary Representation (B-Rep)

- List of faces, all objs rep by boundaries, closed unit
- Add face info for fast rendering (normals, tangents,...)
- Not exact rep since polygons
- Usually only planar polygons
- Find neighbor polygons fast
- Similar alternative: Winged-edge data structure (edges central)
  - Starts with edges, each edge has pointer to 2 faces and pointer to succ and pred edges of face
  - Fast neighboring searches possible, but more pointers
- Specs
  - Data structure
    - List of faces.
    - Face has list edges.
    - Edge has two vertices.
  - Transformations: Same but add per face info also transformed
  - Combinations
    - Simple clipping with plane
    - Advanced ops (union, intersection,difference) addsematic needed
    - Each B-Rep object closed (normals pointing outward)
    - Possible combination algorithm:
      - Cut polygons of A with polygons of B
      - Speed up with bboxes (rough intersection test), exact int only when needed
      - Convex polygons only is better
      - Classifly polygons of A (inside, outside, on surface) to polygons of B
      - Inside/Outside test: shoot ray along normal
      - Nearest polygon is considered inside/outside depending on normal dir

        - Points of faces have same classification until border points appear
        - Add pointers for classification inheritance
        - Depending on the operation remove polygons/merge both objects
  - Memory Consumption
    - Add memory consumption due to faces small
    - Subdivision for curved surfaces: high mem
    - Add links in vert list between neighbor points with smale classification
      - Parametrizable surfaces (free form surfaces, quadrics, superquadrics): param space is subdivided into pieces, approxby planar polygons (high mem for high subdiv)
      - Non-parametrizable surfaces (non-planar polygons): Non-planar polygons are tesselated recursively until approx is good (threshold for normal vector diff between faces)
  - Rendering: Needs alg which handles hidden lines/surfs

- Pros
  - Trans easy
  - Arbitrary closed objects

- Cons
  - High mem const when tesselatd/subdivided
  - Combs not easy/cheap
  - Curved surfaces need approx

### 4.5 Binary Space Partitioning Tree (BSP tree)

- Special B-Rep type for static scenes, polygons def sep plane which partition the space
- Simplifies rendering by easier/faster sorting of polygons
- Drawn back to front (correct occlusion, transparency)
  - Data structure
    - Pointer to root node (planar polygon), node points to two child nodes
    - Left/right subtree: in front of poly/behind poly
    - Intersecting polys are cut

    * Inclusion test of vertex easy, traverse tree: if last step is right step point in object
    * Construction
        - Linear list if convex obj
        - Else convert face list of B-rep into BSPtree
        - Search poly which intersects the fewest other polys with sep plane (rnd select polys for that)
        - Subdivide face list
        - Poly from step 1 is root, others are the left and right subtrees
- Transformation: All vertices, per face info plus plane equations transformed
- Combinatons
    * Either combine B-rep of obj and constr BSP tree or combine directly (faster)
    * Polygon $P_A$ (root node of object A) intersected with object B
        - $P_A^{in}$ .. part of $P_A$ inside object B
        - $P_A^{out}$ .. part of $P_A$ outside object B
    * Separation plane of $P_A$ is used to separate object B into two parts
        - $B^-$ .. Part of B behind the separation plane
        - $B^+$ .. Part of B in front of the separation plane
    * Root of the new BSP tree object is the separation plane and depending on the operation $P_A^{in}$ (intersection) or $P_A^{out}$ (union and differnce) as polygon
    * The subtrees of C are generated recursivly
        - $C_l = A_l$ op $B^-$
        - $C_r = A_r$ op $B^+$
    * Recursion stops if one of the two operands represents a homogenous subtree
- Memory comparable to B-Rep since same lists
- Fast rendering with correct occlusion
    * Painter's algorithm (back to front rendering)
    * Draw all polygons not in half-space of the viewpoint (left or right subtree)
    * Draw root and then other subtree
- Pros
    * Fast rendering with correct occlusion,

    * Easy transformations
    * Arbitrary objects
    * Combinations faster than B-Reps
    * Model gen on digitized models possible
    * Fast search
  - Cons
    * Curved surfaces approximated
    * Only convex polygons used
    * High memory for curved/complex objects

### 4.6 kD Tree

- Special case of BSP tree (axis-aligned subdivision)
- Inner nodes are axis altering sep planes, polygons only in leaf nodes (may be cut in pieces)
- Faster kD Tree gen due to axis aligned sep planes but simple inside/outside test of BSP tree lost

### 4.7 Octree

- Rep volume in data structure
- Fast/easy comb of objects, not necessary in the rendering process (also faster/easier)
- Each octree node is region in space, if nevessary region subdivided into 8 subspaces
- Subdivision done through center of space along main axes
- Each leaf is a block (empty white (W) block, full black (B) block)
- If rep not accurate (gray (G) block), further subdivision
- Gray blocks are subdivided until they are good enough
- Specs
  - Diff transformation, tree rebuild after every trans (specific t like 90deg rot simple)
  - Fast/easy comb
  - High memory for curved and complex surfaces
  - Rendering

    * Renders farthest subspace first until closest subspace (correct occlusion)
    * If W node, do nothing
    * If F node, draw cube
    * If further subdiv, order leaf nodes relativ to view point and render recursively
  - Pros
    * Fast/easy comb
    * Easy rendering
    * Fast spatial searches
    * Model gen through digitization possible
  - Cons
    * High mem
    * Hard transformations
    * No exact rep(only cubes rendered)

#### 4.7.1 Extended Octree

- New leaf nodes: Face, edge, vertex nodes
- Allows more details without further subdiv
- Octree gen out of faces and edges (similar to B-Rep)
  - Separate lists in 8 sublists for each octant by clipping with borders between octants
  - For each octant
    * If both lists empty: node is either full or empty
    * If vert list contains one vertex: gen vert node
    * If face list contains two faces: gen edge node
    * If face list contains one face: gen face node
    * Else subdivde octant
- Specs
  - Data struct: Full, empty, edge, vert or face node or p to 8 subnodes
  - Trans: Convert into B-Rep, t and convert back
  - Comb
    * If nodes are either full or empty: apply operation
    * If nodes are subdivided then all subnodes are combined

    - If node of A is either E,V or F and B is G then subnodes of B are combined with the node of A
    - If nodes of A and B are either E, V or F then the resulting combination is calculated geometrically
  - Memory: Lower for complex objs than usual octree
  - Rendering: Convert back to B-Rep
  - Pros
    - Memory consumption lower
    - Better rep of arb objects due to new nodes
  - Cons:
    - Conversion to B-Rep
    - Combination of two extended octrees more complicated
- Octree as spatial dirs
  - Often used as search struct for diff objs combined in a scene
  - Important for computer games (collision detection)
  - Nearest neighbor to curr obj fast

### 4.8 Constructive Solid Geometry Tree (CSG Tree)

- Binary tree, with logical (comb) intermediate nodes, leafs contain primitives (spheres, blocks, sweeps, ...) and their trans
- Specs
  - Transformation: Add trans to the leaf node
  - Comb: Attach both trees as subtrees to a new root node with desired operation
  - Memory: Efficient exact representation of complex objs
  - Rendering
    - Directly not possible
    - Convert to B-Rep (exact rep gets lost)
    - Ray tracing for rendering (slower)
  - Pros
    - Low mem
    - Exact rep
    - Easy comb and trans
    - Fast spatial searches
  - Cons
    - Rendering more complicated
    - Not easy to gen a CSG tree out of digitized data

### 4.9 Bintree

- Binarytree used for spatial searches
- Recursive perpendicular subdivision (altering axes)
- Optimized irregular subdivison, fewer nodes than octree
- Intermediate nodes contain subdiv pos (values $\in [0,1]$), leaf nodes geometry

### 4.10 Grid

- Regular subdivision
- Only one lew (non-tree struct), cells addressed directly (neighboring rel)
- Flexible data types
- Cell size important (accuracy)
- Hierarchical grids: Base struct consists of large cells, finer cells where needed
- Specs
  - Trans: difficult, must be recomputed (90deg rot simple,...)
  - Comb
    * No geometric info, but MIN,MAX,DIFF ops available
    * Grid resolution must fit
  - Memory Consumption: depends on grid res
  - Rendering: Suitable for algs which frequently sample neighboring cells during rendering process (raycasting)
  - Pros
    * Constant processing time of cells
    * Mem dependent on res
    * Can be used to rep digitized objects
  - Cons
    * No geometrical info about object
    * High mem for accurate rep
    * Hard trans

# 5 Sampling

## 5.1 Signals and Functions

- Signal as function (often temporal/spatial domain) that conveys info
- Continuous/discrete signals (analog/digital sources)
- Selecting finite set of values from signal is sampling
- Attempt to recreate orig continuous signal is reconstruction.

## 5.2 Nyquist frequency

- Uniform sampling big errors, what is best sampling freq?
- Nyquist frequency: Sampling above two times the highest freq of function
- Aliasing
  - High frequencies masquerading as low frequencies
  - Use low-pass filter before sampling process to reduce nyquist freq (also sampling data reduced)
- Sampling in CG: Projection of 3d data on 2d image plane
- Point Sampling
  - Select point for each pixel, evaluate signal at point and assign value to pixel
  - Supersampling: sampling at a higher rate.
- Area sampling
  - Unweighted area sampling
    * Define grid
    * On each grid point evaluate average intensity of square
    * Problems with small objects in the square area
  - Weighted area sampling: Objects contribution weighted accord to dist

### 5.3 Sampling Theory

- Frequency domain: signal as sum of sine waves (phase shiften, diff freqs and amplitudes)
- Fourier analysis (FA): determine which sine waves must be used
- Fourier transform (FT): switch from spatial to frequency domain vice versa
- F(u): function in freqdomain, F(u) how much the freq u appears in orig signal (amplitude, phase shift)
- F(u) is the rep of f(x) in the frequency domain
- FT of a continuous, integrable signal f(x) is defined by:

$$F(u) = \int_{-\inf}^{+\infty} f(x)[\cos 2\pi ux - i \sin 2\pi ux]\delta x$$

- For each u, the value F(u) is a complex number $R(u)+iI(u)$, encoding phase shift and amplitude of the freq u component of signal
- Amplitude (magnitude) of F(u) is defined by: $|F(u)| = \sqrt{R^2(u) + I^2(u)}$
- Phase shift (phase angle) is given by:

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]$$

- Choosing high sampling rate for discrete FT, good approximation of continuous FT is obtained
- Fast FT may be used to convert to the frequency domain.
- Alternatives: Hartley transform, wavelet transform.

### 5.4 Low-Pass Filtering

- Remove problematic high frequencies from the orig signal to recon orig signal from finite samples
- Bandwidth limiting, band limiting, low-pass filtering means lower nyquist freq

### 5.4.1 Convolution and Convolution Theorem

- Two functions in spatial domain are convolved their frequency domain rep are multiplied and vice versa
- $f_1 * f_2 \equiv F_1 F_2$ and $F_1 * F_2 \equiv f_1 f_2$
- FT and multiplication can be much cheaper than convolution in spatial domain
- Perfect low-pass filter in frequency domain is multiplication with a box filter, which is conv with sinc function in spatial domain
- Sinc has infinite support, therefor non-ideal truncated sinc alternatives are used

## 5.5 Reconstruction

- Sampling
  - Mult with comb func (1 in sampling points) in spatial domain
  - Conv with FT of comb in freq domain
  - FT comb is another comb with teeth at other posis
- Sampling below nyquist freq means the teeth of comb are too far apart in spatial domain and to close together in frequency domain (overlapping - aliasing)
- Reconstruction means: eliminate shadow spectra and retain original spectrum (multiplication with box filter in freq domain)
- In spatial domain this corresponds to a convolution with the sinc filter (ideally)
- Non-ideal reconstruction causes two types of aliasing
  - Orig spec not entirely coverd (high freqs damped)
  - Shadow spec not fully eliminated (erroneous high freqs)
- Used recon filters
  - NN
  - Linear int
  - Symmetric cubic filters
  - Windowed Sinc

# 6 Texturing

## 6.1 Introduction

- Diff material props: Color, Reflection, Gloss, Transp, Bump
- Texture gradient 3 props: size, shape, density (defines amount of deformation of pattern)
- Texture mapping: map tex coords on obj space coords, finally obj space coords mapped to img space
- Parametrization must be defined
- Perspective correctness
  - Affine tex mapping directly interpolates texture coords between two endpoinds
  - Perspective correct mapping interpolates after dividing by depth value, then uses interpolated reciprocal to recover correct coords

## 6.2 Types of Textures

- Mostly 2d regular grid storing texels (bitmap, etc..)
  - High mem, slow tex fetch, parametrization needed, aliasing
- Also general texture defs: 1d, 2d, 3d (mostly volume rendering),...
- Scanline Oriented Mapping
  - Scanline oriented rendering calcs img line by line
  - Texture loopup of line needed (texture scanning)
- Procedural and Solid Texturing
  - Gen tex on the fly per fragment
  - Fast eval, but limited variety
  - 3d parameter space, therefore no parametrization needed, undeformed texture
  - Solid texturing: gen function evaluated over $\mathbb{R}^3$ at surf
- Environment Mapping
  - Tex def as surrounding env
  - Project env map on surrounding hull/cube/sphere/cylinder of obj (view rays from center of obj)

  - Polar coord def: better texels access
  - For reflective surf, intersection of reflected ray def surf value
  - For dif surf lowpass filter needed (scattered light of diffuse reflec)
  - Projection step different for each environment map
    * Cube map: perspective distortion is taken into account
    * Sphere: raytracing (no reflected rays)

- Bump Mapping
  - Tex spec as height field, new normals derived
  - Does not change geometry of obj, just influence shading

- Horizon Mapping
  - For each texel n horizon values calced before and interpolated when shading
  - Point in obj space only lit when dir to light src higher than precomputed horizon
  - Bumps are casting shadows, bumps below shadow horizon are not falsly lit

- Parallax Mapping
  - Displace text coord at surf point by a func of view angle in tangent space (angle rel to surf normal) and val of height map at that point
  - At steeper view angles, tex coords displaced more (effect of depth due to parallax)
  - Texture loopup pos altered
  - Occlusions cannot be modelled

- Relief Mapping
  - More accurate alternative to parallax mapping
  - Short-dist raytracing done in pixel shader
  - Self occlusion, self shadowing, view-motion parallax and silhouettes
  - Acceleration techniques possible

### 6.3 Anti-aliasing

- Texture mapping, deformation of size, moire-patterns or pixelization artefacts
- Pixel covered by many texels
  - Choosing just one texel false
  - Weighted averaging better (direct convolution during texel eval, or prefiltering)
- Direct convolution
  - Simple covering shapes used for averaging
  - Computationally expensive but superior quali than NN
- Mip-mapping
  - Prefiltering, diff resolution of texture precalced (half size each step)
  - One third additional storage
  - Mipmap level must be determined when mapping
  - Trilinear int: interpolation between mip map levels
- Summed area table
  - Sum of rects calced before
  - Access sum of certain rect region in constant time
  - Not used for high precision texts (sum becomes to big)

## 7 Illustrative Visualization

### 7.1 Introduction

- Bild mit Absicht Wissen zu vermitteln (Komplexe Zusammenhänge von Strukturen oder Abläufen)
- Abstraktion(Verzerrung Realität/Model) um visualle Überladung zu verhindern
- Verwendet Stilmittel und fokusiert auf Thema (Focus/Context)
- Verschiedene Abstraktionebenen (Illustrator wählt Abstraktionmittel)

## 7.2 Abstraktions Ziele

- Darstellung Form/Struktur
- Hervorstreichung/Verminderung von Objekten
- Vereinfachung
- Künstlichkeit darstellen
- Sichtbarkeit von Objekten sicherstellen
- Räumlichen Überblick
- So detailiert wie nötig, so einfach wie möglich

## 7.3 Low-Level Abstraktionstechniken

- Wie wird etwas dargestellt?
- Stilisierung
  - Silhouetten/Konturen
  - Bleistift/Tusche
  - Punktiert (stippling)
  - Strichliert (hatching)
  - Metallisch/Farbton shading
  - Echtzeit strichlierung
  - Suggestive contours
  - Curvature-based ridge and valley enhancement
- Volumen rendering, style transfunc ordnet voxel farbe und transp zu
- Lit Sphere Maps kann belichtung ersetzen

## 7.4 High-Level Abstraktionstechniken

- Was soll dargestellt werden ("Smart visibility")
- Sichtbarkeit durch Relevanz bestimmt
- Bsp
  - View-dependent transp
  - Cutaways and breakaways(verborgene Strukturen zeigen)
  - Volume splitting: Kopf
  - Hybrid Visibility Composting

- Importance-driven feature enhancement
    - Relevanz Spez, Relevanz Composting, Opacity niveau
    - Betonung explizit gewählten features
  - Illustrative Context-Perserving Exploration
    - Implizite Betonung von relevanten Features
    - Ghosting ("Magic lamp"): innere und äußere gleichzeitig

- Explosionsdarstellung
  - Techn. illustration, Bauanleitung
  - Transformieren von verdeckenden Strukturen
  - Ablauf
    - Focus selection
    - Def von Teilstrukturen
    - Layoutgenerierung
    - Rendering
  - Automated Gernation of Interactive 3d exploded view diagrams
  - Explosion mit/ohne Randbedingungen

- VolumeShop: Interaktive App Illustrationen aus Volumendaten

- Multi-Object Volume Rendering
  - 3 Obj: Auswahl (gewählte, evtl transformierte region), Ghost (gewählte region in orig pos), Hg
  - Darstellungen von einzelnen Obj und Überschneidungen mittels 2dim Transferfunktionen

- Illustrative Beleuchtung: Verschiedene Stile mittels Beleuchtungstransferfunktion, Stile für Hg, Auswahl und Ghost

- Selektive Illustration
  - Smart vis: view-dep cutaways und ghosting
  - Interactive importance-driven vol rendering
  - Ghosting: opazität von verdeckenden Strukturen selektiv vermindert
  - Pfeile, Fächer, Annotationen

# 8 Visualisierung

- Siehe Visualisierung Zusammenfassung (Visualisierung Gröller, Onur Dogangönül)