Graphics and Interactive Techniques (GIT)

Computer graphics is the term commonly used to describe the computer generation and manipulation of images. It is the science of enabling visual communication through computation. Its applications include medical imaging; engineering; scientific, information, and knowledge visualization; cartoons; special effects; and video games. Traditionally, graphics at the undergraduate level focused on rendering, linear algebra, physics, the graphics pipeline, and phenomenological approaches. At the advanced level, undergraduate institutions are increasingly likely to offer one or more courses specializing in a graphics knowledge unit: e.g. gaming, animation, visualization, and tangible computing. In order for students to become adept at the use and generation of computer graphics, many implementation-specific issues must be addressed, such as human perception, data and image file formats, hardware interfaces, and application program interfaces (APIs). These knowledge units change rapidly, and the description that follows attempts to avoid being overly prescriptive. Where provided, examples of API's, programs, and languages should be considered as appropriate examples in 2022, in effect, a snapshot in time..

With a balance of theory and applied instruction, computer science students who learn content from the knowledge units specific below will be able to understand, evaluate, and/or implement the related graphics and interactive techniques as users and developers.

Graphics as a knowledge area has expanded and become more pervasive since the CS2013 report. The now ubiquitous cell phone has made the majority of the world's population regular users and creators of graphics, digital images, and immersive and interactive techniques. Animations, games, visualizations, and immersive applications that in 2013 had to run on desktops, can now run on mobile devices. The amount of data grew exponentially since 2013, and both data and visualizations are now published by myriad sources including news media as well as scientific organizations. Revenue from mobile video games now exceeds that of music and movies combined. Computer Generated Imagery (CGI) is employed in almost all films.

In an effort to align CC2013's Graphics and Visualizations area with SIGGRAPH we have renamed it Graphics and Interactive Techniques (GIT). To capture the expanding footprint of the field, the following knowledge units have been added:

- Immersion (MR, AR, VR)
- Interaction
- Image Processing
- Tangible/Physical Computing
- Simulation

Allocation of Hours

	Core-Tier1 hours	Core-Tier2 hours	Includes Electives (TBD)
GIT/Fundamental Concepts	**4**		**Y**
GIT/Basic Rendering			**Y**
GIT/Geometric Modeling			**Y**
GIT/Advanced Rendering			**Y**
GIT/Computer Animation			**Y**
GIT/Visualization			**Y**
GIT/Immersion (MR, AR, VR)			**Y**
GIT/Interaction			**Y**
GIT/Image Processing			**Y**
GIT/Tangible/Physical Computing			**Y**
GIT/Simulation			**Y**

GIT/Fundamental Concepts

[4 Core hours]

For nearly every computer scientist and software developer, an understanding of how humans interact with machines is essential. While these topics may be covered in a standard undergraduate graphics course, they may also be covered in introductory computer science and programming courses. Note that many of these topics are revisited in greater depth in later sections.

Topics:

[Core]

- Entertainment, business, and scientific applications: examples include visual effects, user interfaces, video editing, games and game engines, computer-aided design and manufacturing, data visualization, and virtual reality.
- Human vision system
    - tristimulus reception (RGB)
    - eye-as-camera (projection)
    - persistence of vision (frame rate/motion blur)
    - contrast detection (Mach banding/dithering/aliasing)
    - non-linear response (tone mapping)
    - binocular vision (stereo)
    - accessibility. (color deficiency, strobing, monocular vision etc). (Cross-reference SEP, HCI)
- Digitization of analog data.
    - rasterization vs vector representations
        - eg: polygon vs volume vs actual object
    - resolution
        - pixels for visual display
        - dots for laser printers
    - sampling and quantization (HCI Foundations)
- Standard media formats
    - raster
        - lossless
        - lossy
    - vector

- Additive and subtractive color models (CMYK and RGB) and why these provide a range of colors
- Tradeoffs between storing data and re-computing data as embodied by vector and raster representations of images
- Animation as a sequence of still images

Illustrative Learning Outcomes:

[Core]

1. Identify common uses of digital presentation to humans (e.g., computer graphics, sound).
2. Explain how analog signals can be reasonably represented by discrete samples, for example, how images can be represented by pixels.
3. Compute the memory requirement for storing a color image given its resolution.

4. Create a graphic depicting how the limits of human perception affect choices about the digital representation of analog signals.
5. Design a user interface and an alternative for persons with color perception deficiency.
6. Construct a simple user interface using a standard API.
7. When should you use each of the following common graphics file formats: JPG, PNG, MP3, MP4, and GIF? Why?
8. Give an example of a lossy and a lossless image compression technique found in common graphics file formats.

9. Describe color models and their use in graphics display devices.
10. Describe the tradeoffs between storing information vs. storing enough information to reproduce the information, as in the difference between vector and raster formats.

[Elective]

11. Compute the memory requirements for an *n* second movie based on *f* frames per second and *r* resolution.
12. Describe the basic process of producing continuous motion from a sequence of discrete frames (sometimes called "flicker fusion").
13. Describe a possible visual mis-representation that could result from digitally sampling an analog world.

GIT/Basic Rendering

[Elective]

This section describes basic rendering and fundamental graphics techniques that nearly every undergraduate course in graphics will cover and that are essential for further study in graphics. Sampling and anti-aliasing are related to the effect of digitization and appear in other areas of computing, for example, in audio sampling.

Topics:

- Rendering in nature, e.g., the emission and scattering of light and its relation to numerical integration.
- Forward and backward rendering (i.e., ray-casting and rasterization).
- Polygonal representation.
- Basic radiometry, similar triangles, and projection model.
- Affine and coordinate system transformations.
- Ray tracing.
- Visibility and occlusion, including solutions to this problem such as depth buffering, Painter's algorithm, and ray tracing.
- The forward and backward rendering equation.
- Simple triangle rasterization.
- Rendering with a shader-based API.
- Texture mapping, including minification and magnification (e.g., trilinear MIP-mapping).
- Application of spatial data structures to rendering.
- Sampling and anti-aliasing.
- Scene graphs and the graphics pipeline.

Illustrative Learning Outcomes:

1. Diagram the light transport problem and its relation to numerical integration i.e., light is emitted, scatters around the scene, and is measured by the eye.
2. Describe the basic graphics pipeline.
3. Compare and contrast how forward and backwards rendering factor into it.

4. Create a program to display 3D models of simple graphics images.
5. Derive linear perspective from similar triangles by converting points (x, y, z) to points (x/z, y/z, 1).
6. Compute 2-dimensional and 3-dimensional points by applying affine transformations.
7. Apply the 3-dimensional coordinate system and the changes required to extend 2D transformation operations to handle transformations in 3D.
8. Explain the concept and applications of texture mapping, sampling, and anti-aliasing.
9. Compare ray tracing and rasterization for the visibility problem.
10. Implement simple procedures that perform transformation and clipping operations on simple 2-dimensional images.
11. Implement a simple real-time renderer using a rasterization API (e.g., OpenGL) using vertex buffers and shaders.
12. Compare and contrast the different rendering techniques.
13. Compute space requirements based on resolution and color coding.
14. Compute time requirements based on refresh rates and rasterization techniques.

GIT/Geometric Modeling

Geometric modeling includes the representation, creation and manipulation of 2D shapes and 3D forms.

[Elective]

Topics:

- Basic geometric operations such as intersection calculation and proximity tests
- Surface representation/model
    - Tessellation
    - Mesh representation, mesh fairing, and mesh generation techniques such as Delaunay triangulation, marching cubes
    - Parametric polynomial curves and surfaces
    - Implicit representation of curves and surfaces
    - Spatial subdivision techniques
- Volumetric representation/model
    - Volumes, voxels, and point-based representations.
    - Signed Distance Fields
    - Sparse Volumes, ie: VDB
    - Constructive Solid Geometry (CSG) representation
- Procedural representation/model
    - Fractals
    - L-Systems, cross referenced with programming languages (grammars to generated pictures).
- Approximation techniques such as polynomial curves, Bezier curves, spline curves and surfaces, and non-uniform rational basis spines (NURBS , and level set method.
- Procedural models such as fractals, generative modeling
- Elastically deformation and freeform deformable models.
    - Quasi-static methods
    - Bi-harmonic capture/deform
- Multiresolution modeling.
- Reconstruction.

Illustrative Learning Outcomes:

1. Contrast representing curves and surfaces in both implicit and parametric forms.
2. Create simple polyhedral models by surface tessellation.
3. Generate a mesh representation from an implicit surface.
4. Generate a fractal model or terrain using a procedural method.
5. Generate a mesh from data points acquired with a laser scanner.
6. Construct CSG models from simple primitives, such as cubes and quadric surfaces.
7. Contrast modeling approaches with respect to space and time complexity and quality of image.

GIT/Advanced Rendering

[Elective]

Topics:

- Solutions and approximations to the rendering equation, for example:
    - Distribution ray tracing and path tracing
    - Photon mapping

    - Bidirectional path tracing
    - Reyes (micropolygon) rendering
    - Metropolis light transport
- Time (motion blur), lens position (focus), and continuous frequency (color) and their impact on rendering
- Shadow mapping
- Occlusion culling
- Bidirectional Scattering Distribution function (BSDF) theory and microfacets
- Subsurface scattering
- Area light sources
- Hierarchical depth buffering
- The Light Field, image-based rendering
- Non-photorealistic rendering
- GPU architecture
- Human visual systems including adaptation to light, sensitivity to noise, and flicker fusion

Learning Outcomes:

1. Demonstrate how an algorithm estimates a solution to the rendering equation.
2. Prove the properties of a rendering algorithm, e.g., complete, consistent, and unbiased.
3. Analyze the bandwidth and computation demands of a simple algorithm.
4. Implement a non-trivial shading algorithm (e.g., toon shading, cascaded shadow maps) under a rasterization API.
5. Show how a particular artistic technique might be implemented in a renderer.
6. Explain how to recognize the graphics techniques used to create a particular image.
7. Implement any of the specified graphics techniques using a primitive graphics system at the individual pixel level.
8. Implement a ray tracer for scenes using a simple (e.g., Phong's) BRDF plus reflection and refraction.

GIT/Computer Animation

[Elective]

Topics:

- Key-frame animation
    - Keyframe Interpolation Methods: Lerp / Slerp / Spline
- Forward and inverse kinematics
- Skinning algorithms
    - Capturing
    - Linear blend, dual quaternion
- Rigging
- Blend shapes
    - Pose space deformation
- Transforms:
    - Translations
    - Scale / Shear
    - Rotations
        - Euler angles
        - Quaternions
        - Angle/axis, exponential map
    - Transformation Order: SRT / XYZ
- Camera animation
    - Look at

    - Focus
- Motion capture
    - Set up and fundamentals
    - Ethical considerations (e.g,. race and privacy)
        - Avoidance of “default” captures - there is no typical human walk cycle.

Learning Outcomes:

1. Compute the location and orientation of model parts using a forward kinematic approach.
2. Compute the orientation of articulated parts of a model from a location and orientation using an inverse kinematic approach.
3. Compare the tradeoffs in different representations of rotations.
4. Implement the spline interpolation method for producing in-between positions and orientations.
5. Use common animation software to construct simple organic forms using metaball and skeleton.

GIT/Visualization

[Elective]

Visualization is the process of creating graphical representations of information and data. Visualization has strong ties to Human Computer Interaction as well as Computational Science. Readers should refer to the HCI and CN KAs for additional topics related to user population and interface evaluations.

Topics:

- Visualization of:
    - 2D/3D scalar fields: color mapping, isosurfaces.
    - Vector fields and flow data
    - Time-varying data
    - High-dimensional data: dimension reduction, parallel coordinates,
    - Non-spatial data: multi-variate, tree/graph structured, text
- Direct volume data rendering: ray-casting, transfer functions, segmentation.
- Common data formats (HDF, netCDF, geotiff, raw binary, CSV, ASCII to parse, etc.)
- Common Visualization software and libraries (R, Processing, D3.js, GIS, Matlab, IDL, Python, etc.)
- Perceptual and cognitive foundations that drive visual abstractions.
    - Visual communication
    - Color theory
- Visualization design.
    - Purpose (discovery, outreach).
    - Audience (technical, general public).
    - Ethically responsible visualization
        - Avoid misleading visualizations (exaggeration, hole filling, smoothing, data cleanup).
        - Even correct data can be misleading - eg, aliasing, incorrectly moving or stopped fan blades.
- Evaluation of visualization methods.
- Applications of visualization.

Illustrative Learning Outcomes:

1. Implement basic algorithms for scalar and vector visualization.
2. Evaluate the tradeoffs of visualization algorithms in terms of accuracy and performance.

3. Propose a suitable visualization design for a particular combination of data characteristics, application tasks, and audience.
4. Analyze the effectiveness of a given visualization for a particular task.
5. Design a process to evaluate the utility of a visualization algorithm or system.
6. Recognize a variety of applications of visualization including representations of scientific, medical, and mathematical data; flow visualization; and spatial analysis.

GIT/Simulation

[Elective]

Simulation has strong ties to Computational Science, especially with respect to the CN KA Modeling and Simulation. In the graphic domain, however, simulation techniques are re-purposed to a different end. Rather than creating predictive models, the goal instead is some mesh of physical plausibility and artistic intention. The goals of "model surface tension in a liquid" and "produce a crown splash" are related but different.

Topics by Subject:

- Collision detection and response
  - Signed Distance Fields
  - Sphere/sphere
  - Triangle/point
  - Edge/edge
- Procedural animation using noise
- Particle systems
  - Integration methods
  - Mass/spring networks
  - Position based dynamics
  - Rules (boids/crowds)
  - Rigid bodies
- Grid based fluids
  - Semi-lagrangian advection
  - Projection
- Heightfields
  - Terrain: Transport, erosion
  - Water: Ripple, Shallow water.
- Rule-based system
  - LSystems.
  - Space-colonizing systems.
  - Game of Life

Prerequisite Data Structures:

- Dense Volumes
- Sparse Volumes
- Adaptive Volumes
- Points with Attributes
- Triangle Soups
- Heightfields

Goals (Given a goal, which topics should be used):

- Particle systems
    - Integration methods
- RBD
    - Particle systems
    - Collision Detection
        - Tri/point, edge/edge
- Cloth
    - Particle systems
    - Mass/spring networks
    - Collision Detection
        - Tri/point, edge/edge
- Water: Particle
    - Integration methods
    - SPH Kernels
    - Signed Distance Collision
- Smoke/Fire - Grid
    - SL Advection + Projection
- Water - Grid + Particle
    - Water - Particle + Smoke/Fire

Illustrative Learning Outcomes:

1. Implement algorithms for physical modeling of particle dynamics using simple Newtonian mechanics, for example Witkin & Kass, snakes and worms, symplectic Euler, Stormer/Verlet, or midpoint Euler methods.
2. Discriminate between and evaluate the basic ideas behind some methods for fluid dynamics for modeling ballistic trajectories, for example for splashes, dust, fire, or smoke.
3. Implement a smoke solver with user interaction

GIT/Immersion
[Elective]

Topics:

- Stereoscopic display
- Viewer tracking
    - Inside out vs Outside In
    - Head / Body / Hand / tracking
- Visibility computation
- Time-critical rendering, multiple levels of details (LOD) Image-based VR system
    - Motion to Photon latency
- Distributed VR, collaboration over computer network Interactive modeling
- Applications in medicine, simulation, and training

- Making immersive applications accessible.

    - Accessible to those who cannot move

    - Accessible to those who cannot be moved. (Motion sickness)

- Ethics/privacy in immersive applications. (cross-reference with SEP)
    - Acquisition of private data (room scans, body proportions, active cameras, etc)
    - Can't look away from Immersion easily.
    - Danger to self/surroundings while immersed

Illustrative Learning Outcomes:

1. Create a stereoscopic image.
2. Summarize the pros and cons of different types of viewer tracking.
3. Compare and contrast the differences between geometry- and image-based virtual reality.
4. Judge and defend the design issues of user action synchronization and data consistency in a networked environment.
5. Create the specifications for an augmented reality application to be used by surgeons in the operating room.
6. Evaluate an immersive application's accessibility (cross-reference with HCI)

GIT/Interaction

Interactive computer graphics is a requisite part of real time applications ranging from the utilitarian like word processors to virtual and/or augmented reality applications.

Students will learn the following topics in a graphics course or a course that covers HCI/GUI Construction and HCI/Programming.

Topics:

- Event Driven Programming
    - Mouse or touch events
    - Keyboard events
    - Voice input
    - Sensors
    - Message passing communication
    - Network events
    - Interrupt event processing
- Graphical User Interface (Single Channel)
    - Window
    - Icons
    - Menus
    - Pointing Devices
- Gestural Interfaces
    - Accessibility - other approaches if gesture not possible (Inject "thumbs up" without a thumb)
- Haptic Interfaces
    - External actuators
- Multimodal Interfaces

Illustrative Learning Outcomes:

- Create a simple game that responds to single channel mouse and keyboard events
- Program a circuit to respond to a variable resistor
- Create a mobile app that responds to touch events

- Use gestures to control a program
- Design and implement an application that provides haptic feedback
- Design and implement an application that responds to different event triggers

GIT/Image Processing

Image Processing consists of the analysis and processing of images for multiple purposes, but most frequently to improve image quality and to manipulate imagery.

Topics:

- Morphological operations
    - Connected components
    - Dilation
    - Erosion
    - Computing region properties (area, perimeter, centroid, etc.)
- Color histograms
    - Representation
    - Contrast enhancement through normalization
- Image enhancement
    - Convolution
    - Blur (e.g., Gaussian)
    - Sharpen (Laplacian)
    - Frequency filtering (low-pass, high-pass)
- Image restoration
    - Noise, degradation
    - Inpainting and other completion algorithms
    - Wiener filter
- Image coding
    - Redundancy
    - Huffman coding
    - DCT, wavelet transform, Fourier transforms
    - Nyquist Theorem
    - Watermarks
        - Ethical considerations
- Emerging area:
    - Convolutional Neural Networks
    - Transformers

Illustrative Learning Outcomes:

- Use dilation and erosion to smooth the edges of a binary image.
- Manipulate hue in an image
- Filter an image using a high-pass filter (advanced: in frequency domain)
- Restore missing part of an image using an inpaint algorithm (e.g., Poisson image editing)
- Compress image by selectively filtering in the frequency domain

GIT/Tangible/Physical Computing

Topics:

- Communication with the physical world
    - Acquisition of data from sensors
    - Driving external actuators
- Event driven programming (see Interaction topic)
- Connection to physical artifacts
    - Computer Aided Design
    - Computer Aided Manufacturing
    - Fabrication
        - HCI prototyping (see HCI topic)
        - Additive (3D printing)
        - Subtractive (CNC milling)
        - Forming (vacuum forming)
- Internet of Things
    - Network connectivity
    - Wireless communication

Illustrative Learning Outcomes:

- Construct a simple switch and use it to turn on an LED.
- Construct a simple system to move a servo in response to sensor data
- Use a light sensor to vary a property of something else (e.g. color or brightness of an LED or graphic)
- Create a 3D form in a CAD package
    - Show how affine transformations are achieved in the CAD program
    - Show an example of instances of an object
    - Create a fabrication plan. Provide a cost estimate for materials and time. How will you fabricate it?
    - Fabricate it. How closely did your actual fabrication process match your plan? Where did it differ?.
- Write the G- and M-Code to construct a 3D maze, use a CAD/CAM package to check your work
- If you were to design an IoT pill dispenser, would you use Ethernet, WiFi, Bluetooth. RFID/NFC, or something else for Internet connectivity. Why? Make one.
- Distinguish between the different types of fabrication and describe when you would use each.

Desirable Professional Dispositions

Professional dispositions are part of the competency model promised in our vision statement. Professional dispositions are "cultivated behaviors" desirable in the workplace. They are malleable and observable. The dispositions listed in earlier curricular volumes include:

- *Proactive: take initiative, self-starter, independent*
- *Self-directed: self-learner, self-motivated*
- *Goal-driven*
- *Professional: exercise discretion, behave ethically*

- *Adaptable: Flexible*
- *Collaborative: team player*
- *Responsive*
- *Meticulous: attentive to detail, thorough*
- *Inventive: look beyond simple solutions*

- *The ability to give and receive code reviews*

- *Effective communication*
    - *oral*
    - *written*
    - *code*

This is not a comprehensive list. Please feel free to add your own, e.g., "Follow instructions" is important for beginning programmers. Professional dispositions differ from skills in that they involve willingness and agency, e.g., it is not sufficient for one to have the skills to collaborate with others. One must also have the willingness to collaborate, the latter being a disposition.

Necessary and Desirable Math

Please enumerate and be specific. Necessary Math is the Math without which a student cannot understand concepts in this knowledge area. Desirable Math helps the student tackle elective or advanced concepts in this knowledge area and/or do well.

The Math necessary for this knowledge area includes:

- Linear Algebra:
    - Points (coordinate systems & homogeneous coordinates), vectors, and matrices
    - Vector operations: addition, scaling, dot and cross products
    - Matrix operations: addition, multiplication, determinants
    - Affine transformations
- Calculus
    - Continuity

The Math desirable for this knowledge includes:

- Calculus
    - Eigenvectors
    - Quaternions

Necessary and Desirable Data Structures

Please enumerate and be specific. Necessary Data Structures are the Data Structures without which a student cannot understand concepts in this knowledge area. Desirable Data Structures helps the student tackle elective or advanced concepts in this knowledge area and/or do well. The intent is these are known in the language that is taught.

The Data Structures necessary for this knowledge area includes:

- Directed Acyclic Graphs
- Tuples (Points / vectors / matrices of fixed dimension)
- Dense 1d, 2d, 3d arrays.

The Data Structures desirable for this knowledge area includes:

- Array of Structs vs Struct of Arrays
- Trees

Shared and Crosscutting Concepts

Working definition: Shared concepts are specific concepts shared with one or some other knowledge areas. Crosscutting concepts are abstract concepts that are shared with most of the knowledge areas. Crosscutting concepts include abstraction, encapsulation, generalization, specialization, reliability, efficiency, tradeoff, modeling, simulation, composition, usability, and approximation.

This knowledge area shares the following concepts with other knowledge areas:

- GIT Immersion and HCI
- GIT Interaction and HCI/GUI Programming and CN/Interactive Visualization
- Graftals (GIT/Modeling) with Programming Languages (PL/BNF grammars)
- Simulation

Crosscutting concepts that apply to this knowledge area include:

- Efficiency
- Ethics
- Modeling
- Programming
- Prototyping
- Usability

Subcommittee

For each person, please include name, affiliation, city, and country

Chair: Susan Reiser, UNC Asheville, NC, USA

Subcommittee members:

- Erik Brunvand, University of Utah, UT, USA
- Kel Elkins, NASA, MD, USA
- Jeff Lait, SideFX, Toronto, Ontario, Canada
- Amruth Kymar,
- Paul Mihail, Valdosta State University, GA, USA
- Tabitha Peck, Davidson College, NC, USA
- Ken Schmidt, NCEI, NC, USA
- Dave Shreiner, Unity, CA, USA

Other contributors:

- Greg Shirah, NASA/GSFC Scientific Visualization Studio, Greenbelt, MD
- AJ Christensen, NASA/GSFC Scientific Visualization Studio – SSAI, Champaign, IL
- Barbara Mones, University of Washington, Seattle, WA, USA
- Beatriz Sousa Santos, University of Aveiro, Portugal
- Ted Kim, Yale University, CT, USA