Graphics and Interactive Techniques (GIT)

Preamble

Computer graphics is the term used to describe the computer generation and manipulation of images and can be viewed as the science of enabling visual communication through computation. Its applications include machine learning; medical imaging; engineering; scientific, information, and knowledge visualization; cartoons; special effects; simulators; and video games. Traditionally, graphics at the undergraduate level focused on rendering, linear algebra, physics, the graphics pipeline, interaction, and phenomenological approaches. Today's graphics courses increasingly include physical computing, animation, and haptics. At the advanced level, undergraduate institutions are increasingly likely to offer one or more courses specializing in a specific graphics knowledge unit: e.g., gaming, animation, visualization, tangible or physical computing, and immersive courses such as AR/VR/XR. There is considerable overlap with other computer science knowledge areas: Algorithmic Foundations, Architecture and Organization, Artificial Intelligence; Human-Computer Interaction; Parallel and Distributed Computing; Specialized Platform Development; Software Engineering, and Society, Ethics, and Professionalism.

In order for students to become adept at the use and generation of computer graphics, many issues must be addressed, such as human perception and cognition, data and image file formats, hardware interfaces, and application program interfaces (APIs). Unlike other knowledge areas, knowledge units or topics within Graphics and Interactive Techniques may be included in a variety of elective courses ranging from a traditional Interactive Computer Graphics course to Gaming or Animation. Alternatively, graphics topics may be introduced in an applied project in Human Computer Interaction, Embedded Systems, Web Development, etc. Undergraduate computer science students who study the knowledge units specified below through a balance of theory and applied instruction will be able to understand, evaluate, and/or implement the related graphics and interactive techniques as users and developers. Because technology changes rapidly, the Graphics and Interactive Techniques subcommittee attempted to avoid being overly prescriptive. Where provided, examples of APIs, programs, and languages should be considered as appropriate examples in 2023. In effect, this is a snapshot in time.

Graphics as a knowledge area has expanded and become pervasive since the CS2013 report. Machine learning, computer vision, data science, artificial intelligence, and interfaces driven by embedded sensors in everything from cars to coffee makers use graphics and interactive techniques. The now ubiquitous smartphone has made the majority of the world's population regular users and creators of graphics, digital images, and the interactive techniques to manipulate them. Animations, games, visualizations, and immersive applications that ran on desktops in 2013, now can run on mobile devices. The amount of stored digital data grew exponentially since 2013, and both data and visualizations are now published by myriad sources including news media and scientific organizations.

Revenue from mobile video games now exceeds that of music and movies combined.[1] Computer Generated Imagery (CGI) is employed in almost all films.

It is critical that students and faculty confront the ethical questions and conundrums that have arisen and will continue to arise because of applications in computer graphics—especially those that employ machine learning, data science, and artificial intelligence. Today's news unfortunately provides examples of inequity and wrong-doing in autonomous navigation, deepfakes, computational photography, and facial recognition.

Changes since CS 2013

In an effort to align CC2013's Graphics and Visualizations areas 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 to the original list of knowledge units: Fundamental Concepts, Visualization, Basic Rendering (renamed Rendering), Geometric Modeling, Advanced Rendering (renamed Shading), and Computer Animation:

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

Core Hours

Knowledge Unit	CS Core	KA Core
Fundamental Concepts	4	
Rendering		18
Geometric Modeling		6
Shading		6
Computer Animation		Animation KA - 6
Visualization		Visualization KA - 6
Immersion (MR, AR, VR)		Immersion KA - 6
Interaction		Interaction KA - 6

[1] Jon Quast, Clay Bruning, and Sanmeet Deo. "Markets: This Opportunity for Investors Is Bigger Than Movies and Music Combined." retrieved from https://www.nasdaq.com/articles/this-opportunity-for-investors-is-bigger-than-movies-and-music-combined-2021-10-03.

Image Processing		Image Processing KA - 6
Tangible/Physical Computing		Tangible/Physical Computing KA - 6
Simulation		Simulation KA - 6
Total	**4**	

Knowledge Units

GIT-Fundamentals: Fundamental Concepts

For nearly every computer scientist and software developer, 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.

CS Core:

1. Uses of graphics and interactive techniques and potential risks and abuses
    a. Entertainment, business, and scientific applications: examples include visual effects, machine learning, computer vision, user interfaces, video editing, games and game engines, computer-aided design and manufacturing, data visualization, and virtual/augmented/mixed reality.
    b. Intellectual property, deep fakes, facial recognition, privacy (See also: SEP-Privacy, SEP-IP, SEP-Ethics)
2. Graphic output
    a. displays (e.g., LCD)
    b. printer
    c. analog film
    d. resolution
        i. pixels for visual displays
        ii. dots for laser printers
    e. aspect ratio
    f. frame rate
3. Human vision system
    a. tristimulus reception (RGB)
    b. eye as a camera (projection)
    c. persistence of vision (frame rate, motion blur)
    d. contrast (detection, Mach banding, dithering/aliasing)
    e. non-linear response (dynamic range, tone mapping)
    f. binocular vision (stereo)
    g. accessibility (color deficiency, strobing, monocular vision, etc.) (See also: SEP-IDEA, HCI-User)
4. Standard image formats
    a. raster
        i. lossless (e.g., TIF)

    ii. lossy (e.g., JPG, PNG, etc.)
  b. vector (e.g. SVG, Adobe Illustrator)
5. Digitization of analog data
  a. rasterization
  b. resolution
  c. sampling and quantization
6. Color models: additive (RGB), subtractive (CMYK), and color perception (HSV)
7. Tradeoffs between storing image data and re-computing image data.
8. Applied interactive graphics, e.g., graphics API, mobile app
9. Animation as a sequence of still images

Illustrative Learning Outcomes:

CS 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 graphical 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. Compute the memory requirements for a multi-second movie (lasting *n* seconds) displaying at a specific framerate (*f* frames per second) at a specified resolutions (*r* pixels per frame)
11. Compare and contrast digital video to analog video.
12. Describe the basic process of producing continuous motion from a sequence of discrete frames (sometimes called “flicker fusion”).
13. Describe a possible visual misrepresentation that could result from digitally sampling an analog world.
14. Compute memory space requirements based on resolution and color coding.
15. Compute time requirements based on refresh rates and rasterization techniques.

GIT-Visualization

KA Core

1. Data Visualization and Information Visualization
2. Visualization of:
  a. 2D/3D scalar fields
  b. Vector fields and flow data
  c. Time-varying data

d. High-dimensional data
e. Non-spatial data

2. Visualization techniques (color mapping, isosurfaces, dimension reduction, parallel coordinates, multi-variate, tree/graph-structured, text)
3. Direct volume data rendering: ray-casting, transfer functions, segmentation.
4. Common data formats (e.g., HDF, netCDF, geotiff, raw binary, CSV, ASCII to parse, etc.)
5. Common Visualization software and libraries (e.g., R, Processing, D3.js, GIS, Matlab, IDL, Python, etc.)
6. Perceptual and cognitive foundations that drive visual abstractions
   a. Visual communication
   b. Color theory
   7. Visualization design
   a. Purpose (discovery, outreach)
   b. Audience (technical, general public)
   c. Ethically responsible visualization
      i. Avoid misleading visualizations (due to exaggeration, hole filling, smoothing, data cleanup).
      ii. Even correct data can be misleading, e.g., aliasing, incorrectly moving or stopped fan blades.
8. Evaluation of visualization methods and applications
9. Visualization bias
10. Applications of visualization

Illustrative Learning Outcomes

1. Compare and contrast data visualization and information visualization.
2. Implement basic algorithms for visualization.
3. Evaluate the tradeoffs of visualization algorithms in terms of accuracy and performance.
4. Propose a suitable visualization design for a particular combination of data characteristics, application tasks, and audience.
5. Analyze the effectiveness of a given visualization for a particular task.
6. Design a process to evaluate the utility of a visualization algorithm or system.
7. Recognize a variety of applications of visualization including representations of scientific, medical, and mathematical data; flow visualization; and spatial analysis.

GIT-Rendering

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 most graphics-related courses.

KA Core (See SPD-Game 2 & 3c)

1. Object and scene modeling
   a. Object representations: polygonal, parametric, etc.
   b. Modeling transformations: affine and coordinate-system transformations
   c. Scene representations: scene graphs

2. Camera and projection modeling
    a. Pinhole cameras, similar triangles, and projection model
    b. Camera models
    c. Projective geometry
3. Radiometry and light models
    a. Radiometry
    b. Rendering equation
    c. Rendering in nature – emission and scattering, etc.
4. Rendering
    a. Simple triangle rasterization
    b. Rendering with a shader-based API
    c. Visibility and occlusion, including solutions to this problem (e.g., depth buffering, Painter's algorithm, and ray tracing)
    d. Texture mapping, including minification and magnification (e.g., trilinear MIP mapping)
    e. Application of spatial data structures to rendering.
    f. Ray tracing
    g. Sampling and anti-aliasing

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 rendering pipeline.
3. Compare and contrast how forward and backwards rendering factor into the graphics pipeline.
4. Create a program to display 2D shapes in a window.
5. Create a program to display 3D models.
6. Derive linear perspective from similar triangles by converting points (x, y, z) to points (x/z, y/z, 1).
7. Compute two-dimensional and 3-dimensional points by applying affine transformations.
8. Apply the three-dimensional coordinate system and the changes required to extend 2D transformation operations to handle transformations in 3D.
9. Explain the concept and applications of texture mapping, sampling, and anti-aliasing.
10. Compare ray tracing and rasterization for the visibility problem.
11. Implement simple procedures that perform transformation and clipping operations on simple two-dimensional shapes.
12. Implement a simple real-time renderer using a rasterization API (e.g., OpenGL, webGL) using vertex buffers and shaders.
13. Compare and contrast the different rendering techniques.
14. Compare and contrast the difference in transforming the camera vs. the models.

GIT-Modeling: Geometric Modeling

KA Core

1. Basic geometric operations such as intersection calculation and proximity tests on 2D objects
2. Surface representation/model
    a. Tessellation

   b. Mesh representation, mesh fairing, and mesh generation techniques such as Delaunay triangulation, and marching cubes/tetrahedrons
   c. Parametric polynomial curves and surfaces
   d. Implicit representation of curves and surfaces
   e. Spatial subdivision techniques
3. Volumetric representation/model
   a. Volumes, voxels, and point-based representations.
   b. Signed Distance Fields
   c. Sparse Volumes, i.e., VDB
   d. Constructive Solid Geometry (CSG) representation
4. Procedural representation/model (See also: FPL-Translation 1.a)
   a. Fractals
   b. L-Systems
5. Multi-resolution modeling (See also SPD-Game)
6. Reconstruction, e.g., 3D scanning, photogrammetry, etc.

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-Shading

Topics:

1. Solutions and approximations to the rendering equation, for example:
   a. Distribution ray tracing and path tracing
   b. Photon mapping
   c. Bidirectional path tracing
   d. Metropolis light transport
2. Time (motion blur), lens position (focus), and continuous frequency (color) and their impact on rendering
3. Shadow mapping
4. Occlusion culling
5. Bidirectional Scattering Distribution function (BSDF) theory and microfacets
6. Subsurface scattering
7. Area light sources
8. Hierarchical depth buffering
9. Image-based rendering
10. Non-photorealistic rendering
11. GPU architecture (See also AR-Heterogeneous Architectures 1 & 3)

12. Human visual systems including adaptation to light, sensitivity to noise, and flicker fusion (See also: HCI-Accessibility, SEP-IDEA)

Illustrative 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-Animation: Computer Animation

KU Core:

1. Principles of Animation: Squash and Stretch, Timing, Anticipation, Staging, Follow Through and Overlapping Action, Straight Ahead Action and Pose-to-Pose Action, Slow In and Out, Arcs, Exaggeration, and Appeal
2. Types of animation
    a. 2- and 3-dimensional animation
    b. motion graphics
    c. motion capture
    d. motion graphics
    e. stop animation
3. Key-frame animation
    a. Keyframe Interpolation Methods: Lerp / Slerp / Spline
4. Forward and inverse kinematics (See also: SPD-Robot)
5. Skinning algorithms
    a. Capturing
    b. Linear blend, dual quaternion
    c. Rigging
    d. Blend shapes
    e. Pose space deformation
6. Motion capture
    a. Set up and fundamentals
    b. Blending motion capture clips
    c. Blending motion capture and keyframe animation
    d. Ethical considerations (e.g., accessibility and privacy)
        i. Avoidance of "default" captures - there is no typical human walk cycle.

Illustrative Learning Outcomes:

1. Using a simple open-source character model and rig, illustrate why each of the principles of animation is fundamental to realistic animation.
2. Compute the location and orientation of model parts using a forward kinematic approach.
3. Compute the orientation of articulated parts of a model from a location and orientation using an inverse kinematic approach.
4. Compare the tradeoffs in different representations of rotations.
5. Implement the spline interpolation method for producing in-between positions and orientations.
6. Use off-the-shelf animation software to construct, rig, and animate simple organic forms.

GIT: Simulation

Simulation has strong ties to Computational Science. In the graphic domain, however, simulation techniques are re-purposed to a different end. Rather than creating predictive models, the goal instead is to achieve a mixture of physical plausibility and artistic intention. To illustrate, the goals of “model surface tension in a liquid” and “produce a crown splash” are related, but different. Depending on the simulation goals, covered topics may vary as shown below.

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

- Particle systems
    - Integration methods (Forward Euler, Midpoint, Leapfrog)
- Rigid Body Dynamics
    - Particle systems
    - Collision Detection
        - Tri/point, edge/edge
- Cloth
    - Particle systems
    - Mass/spring networks
    - Collision Detection
        - Tri/point, edge/edge
- Particle-Based Water
    - Integration methods
    - Smoother Particle Hydrodynamics (SPH) Kernels
    - Signed Distance Function-Based Collisions
- Grid-Based Smoke and Fire
    - Semi-Lagrangian Advection
    - Pressure Projection
- Grid and Particle-Based Water
    - Particle-Based Water
- Grid-Based Smoke and Fire
    - Semi-Lagrangian Advection
    - Pressure Projection
- Grid and Particle-Based Water
    - Particle-Based Water
    - Grid-Based Smoke, and Fire

KU Core:

1. Collision detection and response
    a. Signed Distance Fields
    b. Sphere/sphere
    c. Triangle/point
    d. Edge/edge
2. Procedural animation using noise
3. Particle systems
    a. Integration methods (Forward Euler, Midpoint, Leapfrog)
    b. Mass/spring networks
    c. Position-based dynamics
    d. Rules (boids/crowds)
    e. Rigid bodies
4. Grid-based fluids
    a. Semi-Lagrangian advection
    b. Pressure Projection

- Heightfields
    a. Terrain: Transport, erosion
    b. Water: Ripple, Shallow water.
- Rule-based systems, e.g., L-Systems, Space-colonizing systems, Game of Life, etc.

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. Contrast the basic ideas behind fluid simulation methods for modeling ballistic trajectories, for example for splashes, dust, fire, or smoke.
3. Implement a smoke solver with user interaction

GIT-Immersion

KU Core: See also: SPD-Games, SPD-Mobile, HCI-Design

1. Define and distinguish VR, AR, and MR
2. Define and distinguish immersion and presence
3. 360 Video
4. Stereoscopic display
    a. Head-mounted displays
    b. Stereo glasses
5. Viewer tracking
    a. Inside out vs Outside In
    b. Head / Body / Hand / tracking
6. Time-critical rendering to achieve optimal motion to photon (MTP) latency
    a. Multiple levels of details (LOD)
    b. Image-based VR

    c. Branching movies
7. Distributed VR, collaboration over computer network
8. Applications in medicine, simulation, training, and visualization
9. Safety in immersive applications
    a. Motion sickness
    b. VR obscures the real world, which increases the potential for falls and physical accidents

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)
7. Identify the most important technical characteristics of a VR system/application that should be controlled to avoid motion sickness and explain why.

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.

KU Core:

1. Event Driven Programming (See also: FPL-Event-Driven)
    a. Mouse or touch events
    b. Keyboard events
    c. Voice input
    d. Sensors
    e. Message passing communication
    f. Network events
2. Graphical User Interface (Single Channel)
    a. Window
    b. Icons
    c. Menus
    d. Pointing Devices
3. Gestural Interfaces (See also: SPD-Games)
    a. Touch screens
    b. Game controllers
4. Haptic Interfaces
    a. External actuators
    b. Gloves
    c. Exoskeletons

5. Multimodal Interfaces
6. Head-worn Interfaces (AI)
    a. brainwave (EEG type electrodes)
    b. headsets with embedded eye tracking
    c. AR glasses
7. Natural Language Interfaces (See also: AI-NLP)
8. Accessibility (See also: SEP_IDEA)

Illustrative Learning Outcomes:

1. Create a simple game that responds to single channel mouse and keyboard events
2. Program a circuit to respond to a variable resistor
3. Create a mobile app that responds to touch events
4. Use gestures to control a program
5. Design and implement an application that provides haptic feedback
6. Design and implement an application that responds to different event triggers

GIT-Image: 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. It is the cornerstone of Computer Vision which is a KU in AI.

KU Core:

1. Morphological operations
    a. Connected components
    b. Dilation
    c. Erosion
    d. Computing region properties (area, perimeter, centroid, etc.)
2. Color histograms
    a. Representation
    b. Contrast enhancement through normalization
3. Image enhancement
    a. Convolution
    b. Blur (e.g., Gaussian)
    c. Sharpen (e.g.,Laplacian)
    d. Frequency filtering (e.g.,low-pass, high-pass)
4. Image restoration
    a. Noise, degradation
    b. Inpainting and other completion algorithms
    c. Wiener filter
5. Image coding
    a. Redundancy
    b. Compression (e.g., Huffman coding)
    c. DCT, wavelet transform, Fourier transforms
    d. Nyquist Theorem
    e. Watermarks

6. Connections to deep learning (e.g., Convolutional Neural Networks) (See AI-ML 7)

Illustrative Learning Outcomes:

1. Use dilation and erosion to smooth the edges of a binary image.
2. Manipulate the hue of an image
3. Filter an image using a high-pass filter (advanced: in frequency domain)
4. Restore missing parts of an image using an in-paint algorithm (e.g., Poisson image editing)
5. Enhance an image by selectively filtering in the frequency domain

GIT-Physical: Tangible/Physical Computing

Cross-cutting with Specialized Platform Development and HCI

KU Core:

1. Interaction with the physical world
   a. Acquisition of data from sensors
   b. Driving external actuators
2. Connection to physical artifacts
   a. Computer-Aided Design (CAD)
   b. Computer-Aided Manufacturing (CAM)
   c. Fabrication (See also HCI-Design)
      i. HCI prototyping (see HCI)
      ii. Additive (3D printing)
      iii. Subtractive (CNC milling)
      iv. Forming (vacuum forming)
3. Internet of Things (See also SDP-Interactive)
   a. Network connectivity
   b. Wireless communication

Illustrative Learning Outcomes:

1. Construct a simple switch and use it to turn on an LED.
2. Construct a simple system to move a servo in response to sensor data
3. Use a light sensor to vary a property of something else (e.g. color or brightness of an LED or graphic)
4. Create a 3D form in a CAD package
   a. Show how affine transformations are achieved in the CAD program
   b. Show an example of instances of an object
   c. Create a fabrication plan. Provide a cost estimate for materials and time. How will you fabricate it?
   d. Fabricate it. How closely did your actual fabrication process match your plan? Where did it differ?
5. Write the G- and M-Code to construct a 3D maze, and use a CAD/CAM package to check your work
6. 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.
7. Distinguish between the different types of fabrication and describe when you would use each.

GIT-SEP: Society, Ethics and Professionalism

KU Core:

1. Physical Computing
   a. Privacy (e.g., health and other personal information)
2. Accessibility in immersive applications (See also: SEP-IDEA)
   a. Accessible to those who cannot move
   b. Accessible to those who cannot be moved
3. Ethics/privacy in immersive applications. (See also: SEP-Privacy, SEP-Ethics, and SEP-Security)
   a. Acquisition of private data (room scans, body proportions, active cameras, etc)
   b. Can't look away from immersive applications easily.
   c. Danger to self/surroundings while immersed
4. Bias in image processing
   a. Deep fakes
   b. Applications that misidentify people based on skin color or hairstyle
5. Copyright law and watermarked images

Illustrative Learning Outcomes:

1. Discuss the security issues inherent in location tags.
2. Describe the ethical pitfalls of facial recognition. Can facial recognition be used ethically? If so, how?
3. Discuss the copyright issues of using watermarked images to train a neural network.

Professional Dispositions

1. Self-directed: self-learner, self-motivated. It is important for graphics programmers to keep up with technical advances.
2. Collaborative: team player: Graphics programmers typically develop on teams composed of people with differing specialties.
3. Effective communication is critical.
   a. oral
   b. written
   c. code

Math Requirements

Required:

1. Linear Algebra:
   a. Points (coordinate systems & homogeneous coordinates), vectors, and matrices
   b. Vector operations: addition, scaling, dot and cross products
   c. Matrix operations: addition, multiplication, determinants
   d. Affine transformations
2. Calculus

a. Continuity

Desirable:

1. Linear Algebra
   a. Eigenvectors and Eigen decomposition
   b. Gaussian Elimination and Lower Upper Factorization
   c. Singular Value Decomposition
2. Calculus
   a. Quaternions
3. Probability

Necessary and Desirable Data Structures:

1. Data Structures necessary for this knowledge area include:
   a. Directed Acyclic Graphs
   b. Tuples (Points / vectors / matrices of fixed dimension)
   c. Dense 1d, 2d, 3d arrays.
2. Data Structures desirable for this knowledge area include:
   a. Array of Structures vs. Structure of Arrays
   b. Trees (e.g., K-trees, quadtrees, Huffman Trees)

Shared Topics and Crosscutting Themes

Shared Topics:

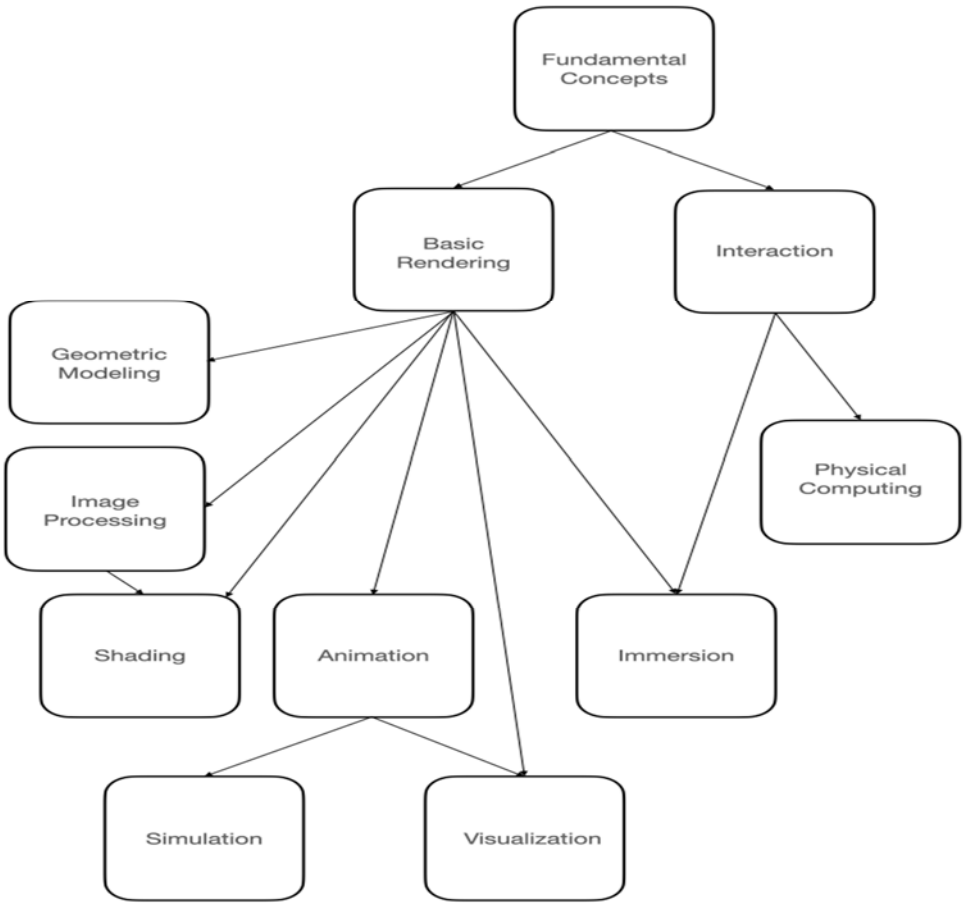
- GIT-Immersion □ □ HCI
- GIT-Interaction □ □ ~~HCI/GUI Programming~~
- Graftals: GIT-Geometric Modeling □ □ FPL-H: Language Translation and Execution
- Simulation
- Visualization in GIT, AI, and Specialized Platform Development Interactive Computing Platforms (Data Visualization)
- Image Processing in GIT and Specialized Platform Development Interactive Computing Platforms (Supporting Math Studies)
- Tangible Computing in GIT and Specialized Platform Development Interactive Computing Platforms (Game Platforms)
- Tangible Computing in GIT and Specialized Platform Development Interactive Computing Platforms (Embedded Platforms)
- Tangible Computing and Animation in GIT and Specialized Platform Development Interactive Computing Platforms (Robot Platforms)
- Immersion in GIT and Specialized Platform Development Interactive Computing Platforms (Mobile Platforms)
- Image Processing in GIT and Advanced Machine Learning (Graphical Models) in AI
- Image Processing and Physical Computing in GIT and Robotics Location and Mapping and Navigation in AI
- Image Processing in GIT and Perception and Computer Vision in AI

- Core Graphics in GIT and Algorithms and Application Domains in PD
- GIT and Interactive Computing Platforms in SPD
- GIT and Game Platforms in SPD
- GIT and Imbedded Platforms in SPD

Crosscutting themes:

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

Course Packaging Suggestions



Interactive Computer Graphics to include the following:

- GIT-Rendering: 40 hours

- SEP-C: Professional Ethics: 4 hours

Pre-requisites:

- CS2
- Affine Transforms from Linear Algebra
- Trigonometry

Skill statement: A student who completes this course should understand and be able to create basic computer graphics using an API. They should know how to position and orient models, the camera, and distant and local lights.

Tangible Computing to include the following:

- GIT-Tangible/Physical Computing (27 hours)
- SPD-E: Embedded Platforms (10 hours)
- HCI-User: Understanding the User (3 hours)
- SEP-Privacy: Privacy and Civil Liberties and SEP-K: Diversity, Equity and Inclusion (4 hours)

Pre-requisites:

- CS1

Skill statement: A student who completes this course should be able to design and build circuits and program a microcontroller. They will understand polarity, Ohm's law, and how to work with electronics safely.

Image Processing to include the following:

- GIT-Image Processing (30 hours)
- GIT-Rendering: Basic Rendering: (10 hours)
- SEP-Privacy: Privacy and Civil Liberties,SEP-IDEA: Inclusion, Diversity, Equity and Accessibility and SEP-IP: Intellectual Property (4 hours)

Pre-requisites:

- CS2
- Linear Algebra
- Probability

Skill statement: A student who completes this course should understand and be able to appropriately acquire, process, display, and save digital images.

Data Visualization to include the following:

- GIT-Visualization (30 hours)
- GIT-Rendering: Basic Rendering (10 hours)
- HCI-User: Understanding the User (3 hours)
- SEP-E: Privacy and Civil Liberties and SEP-IDEA: Inclusion, Diversity, Equity and Accessibility, SEP-Ethics: Professional Ethics (4 hours)

Pre-requisites:

- CS2
- Linear Algebra

Skill statement: A student who completes this course should understand how to select a dataset; ensure the data are accurate and appropriate; design, develop and test a visualization program that depicts the data and is usable.

Simulation to include the following:

- GIT-Simulation (30 hours)
- GIT-Rendering (10 hours)
- SEP-Ethics: Professional Ethics (4 hours)

Pre-requisites:

- CS2
- Linear Algebra

Skill statement: A student who completes this course should understand and be able to create directable simulations, both of physical and non-physical systems.

Committee

Chair: Susan Reiser, UNC Asheville, Asheville, USA

Members:

- Erik Brunvand, University of Utah, Salt Lake City, USA
- Kel Elkins, NASA/GSFC Scientific Visualization Studio, Greenbelt, MD
- Jeff Lait, SideFX, Toronto, Canada
- Amruth Kumar, Ramapo College, Mahwah, USA
- Paul Mihail, Valdosta State University, Valdosta, USA
- Tabitha Peck, Davidson College, Davidson, USA
- Ken Schmidt, NOAA NCEI, Asheville, USA
- Dave Shreiner, UnityTechnologies & Sonoma State University, San Francisco, USA

Contributors:

- Ginger Alford, Southern Methodist University, TX, USA
- Christopher Andrews, Middlebury College, Middlebury, VT, USA
- AJ Christensen, NASA/GSFC Scientific Visualization Studio – SSAI, Champaign, IL
- Roger Eastman, University of Maryland, College Park, MD, USA
- Ted Kim, Yale University, New Haven, CT, USA
- Barbara Mones, University of Washington, Seattle, WA, USA
- Greg Shirah, NASA/GSFC Scientific Visualization Studio, Greenbelt, MD
- Beatriz Sousa Santos, University of Aveiro, Portugal
- Anthony Steed, University College, London, UK

Appendix: Core Topics and Skill Levels

KU	Topic	Skill	Core	Hours

Core	• Applications • Human vision system • Digitization of analog data • Standard media formats • Color Models • Tradeoffs between storing data and re-computing data • Animation as a sequence of still images • SEP related to graphics	Explain	CS	4
Basic Rendering	• Graphics pipeline. • Affine and coordinate system transformations. • Rendering in nature, e.g., the emission and scattering of light and its relation to numerical integration. • Texture mapping, including minification and magnification (e.g., trilinear MIP-mapping). • Sampling and anti-aliasing. • Visibility and occlusion, including solutions to this problem such as depth buffering, Painter's algorithm, and ray tracing.	Explain	KA	10
	• Forward and backward rendering (i.e., ray-casting and rasterization). • Polygonal representation. • Basic radiometry, similar triangles, and projection model. • Ray tracing. • The rendering equation. • Simple triangle rasterization. • Application of spatial data structures to rendering. • Scene graphs.	Explain	KA	5
	• Generate an image with a standard API	Implement	KA	3

Visualization KA Core	• Visualization of:
	  ○ 2D/3D scalar fields: color mapping
	  ○ Time-varying data
		Explain and Implement	KA	3

	• Visualization techniques (color mapping, dimension reduction)	Explain and Implement	KA	2
	• Perceptual and cognitive foundations that drive visual abstractions.	Explain	KA	1
	• Visualization Bias	Evaluate	KA	1

Modeling KA Core	• Surface representation/model
	  • Mesh representation, mesh fairing, and mesh generation techniques such as Delaunay triangulation, marching cubes
	  • Parametric polynomial curves and surfaces
		Explain and Use	KA	2

	• Volumetric representation/model
	  • Volumes, voxels, and point-based representations.
	  • Constructive Solid Geometry (CSG) representation
		Explain and Use	KA	2

	• Procedural representation/model
	  • Fractals
	  • L-Systems, cross referenced with programming languages (grammars to generated pictures).
	  • Generative Modeling
		Explain and Use	KA	2

Shading KA Core	• Time (motion blur) and lens position (focus) and their impact on rendering • Shadow mapping • Occlusion culling • Area light sources • Hierarchical depth buffering • Non-photorealistic rendering	Explain and Use	KA	6
Computer Animation KA Core	• Principles of Animation (Squash and Stretch, Timing, Anticipation, Staging, Follow Through and Overlapping Action, Straight Ahead Action and Pose-to-Pose Action, Slow In and Out, Arcs, Exaggeration, and Appeal) • Key-frame animation	Explain and Use	KA	2
	• Forward and inverse kinematics	Explain and Use	KA	2

	• Transforms:
	  • Translations
	  • Scale / Shear
	  • Rotations
	• Camera animation
	  • Look at
	  • Focus
		Explain and Implement	KA	2

Simulation KA Core	• Particle systems	Explain and implement	KA	2
	• Collision detection and response	Explain and Implement	KA	2
	• Grid based fluids	Explain and Implement	KA	2
Immersion KA Core	• Define and distinguish VR, AR, and MR • Applications in medicine, simulation, and training, and visualization	Explain	KA	1

	• Stereoscopic display
	• Viewer tracking
	  ○ Inside out vs Outside In
	  ○ Head / Body / Hand / tracking
	• Visibility computation
		Explain and Implement	KA	3

	• Safety in immersive applications • Accessibility in immersive applications. • Ethics/privacy in immersive applications.	Explain and Evaluate	KA	2
Interaction KA Core	• Event Driven Programming (Shared with SPD Interactive, SPD Game Development) • Graphical User Interface	Explain and Implement	KA	4
	• Accessibility in GUI Design (shared with HCI)	Explain and Evaluate	KA	2
Image Processing	• Convolution filters	Explain and implement	KA	2
	• Convolutional Neural Networks.(shared with AI:Machine Learning)	Explain and Implement	KA	2
	• Histograms • Fourier and/or Cosine Transforms	Explain and Implement	KA	2

Physical Computing KA Core	• Communication with the physical world (shared with SPD/Embedded Systems, PL/Embedded Systems)
	  ○ Acquisition of data from sensors
	  ○ Driving external actuators
		Explain and Implement	KA	1

	• Event driven programming (shared with GIT: Interaction)	Explain and Implement	KA	1

	● Connection to physical artifacts ○ Computer Aided Design ○ Computer Aided Manufacturing ○ Fabrication ■ prototyping (shared with HCI) ■ Additive (3D printing) ■ Subtractive (CNC milling) ■ Forming (vacuum forming)	Explain, evaluate, and Implement an example	KA	3
	● Internet of Things ○ Network connectivity ○ Wireless communication	Explain	KA	1