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, and phenomenological approaches. 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: Artificial Intelligence; Human Computer Interaction; Parallel and Distributed Computing; Specialized Platform Development; and Society, Ethics and Professionalism.

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 and cognition, data and image file formats, hardware interfaces, and application program interfaces (APIs). 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 2022. 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, AI, and the inclusion of embedded sensors in everything from cars to coffee makers utilize graphics and interactive techniques. 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 ran on desktops in 2013, now can 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 and 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.[1]

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

[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.

examples of inequity and wrong-doing related to autonomous navigation, deep-space imaging, computational photography, deep fakes, and facial recognition.

Changes since CS 2013: 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 to the original list of knowledge units (Fundamental Concepts, Visualization, Basic Rendering, Geometric Modeling, Advanced Rendering renamed Advanced Shading, Computer Animation):

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

Core Hours

Knowledge Unit	CS Core	KA Core
Fundamental Concepts	4	
Basic Rendering		18
Geometric Modeling		6
Advanced Shading		6
Computer Animation		Animation KA - 6
Visualization		Visualization KA - 6
Immersion (MR, AR, VR)		Immersion KA - 6
Interaction		Interaction KA - 6
Image Processing		Image Processing KA - 6
Tangible/Physical Computing		Tangible/Physical Computing KA - 6
Simulation		Simulation KA - 6
Total	**4**	

Knowledge Units

GIT/Fundamental Concepts
[4 CS Core hours]

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.

Topics:

[Core]

- 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.
- 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 with SEP, HCI)
- Digitization of analog data
    - rasterization vs vector representations
        - example: polygon vs volume vs actual object
    - resolution
        - pixels for visual display
        - dots for laser printers
    - sampling and quantization
- Standard media formats
    - raster
        - lossless
        - lossy
    - vector
- Color models: additive (RGB), subtractive (CMYK) and color perception (HSV)
- Tradeoffs between storing data and re-computing data as embodied by vector and raster representations of images
- Animation as a sequence of still images
- SEP issues: deep fakes, facial recognition, privacy, intellectual property

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 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.
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/Visualization

Visualization is the process of creating graphical representations of 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:

- Data Visualization and Information Visualization
- Visualization of:
    - 2D/3D scalar fields
    - Vector fields and flow data
    - Time-varying data
    - High-dimensional data
    - Non-spatial data
- Visualization techniques (color mapping, isosurfaces, dimension reduction, parallel coordinates, 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 and applications.
- Visualization Bias
- Applications of visualization.

Illustrative Learning Outcomes:

1. Implement basic algorithms for 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/Basic 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.

Topics:

- Graphics pipeline.
- 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 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.

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 2-dimensional and 3-dimensional points by applying affine transformations.
8. Apply the 3-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 2-dimensional images.
12. Implement a simple real-time renderer using a rasterization API (e.g., OpenGL) using vertex buffers and shaders.
13. Compare and contrast the different rendering techniques.
14. Compute space requirements based on resolution and color coding.
15. 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.

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, i.e., VDB
    - Constructive Solid Geometry (CSG) representation
- Procedural representation/model
    - Fractals
    - L-Systems, cross referenced with programming languages (grammars to generated pictures).
- 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/Shading

Topics:

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

Topics:

- 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
    - 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., accessibility 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/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. 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 (Forward Euler, Midpoint, Leapfrog)
    - Mass/spring networks
    - Position based dynamics
    - Rules (boids/crowds)
    - Rigid bodies
- Grid based fluids
    - Semi-Lagrangian advection
    - Pressure 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 (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

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

Topics:

- Define and distinguish VR, AR, and MR
- 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, training, and visualization
- Safety in immersive applications
    - Motion sickness
    - VR obscures the real world, which increases the potential for falls and physical accidents
- Accessibility in immersive applications
    - Accessible to those who cannot move
    - Accessible to those who cannot be moved
- 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 immersive applications 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)
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.

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
    - Gloves
    - Exoskeletons
- Multimodal Interfaces
- Immersive Interfaces (AI)
    - brainwave (EEG type electrodes)
    - headsets with embedded eye tracking
    - AR glasses
- Accessibility (cross-reference with SEP)

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. It is the cornerstone of Computer Vision which is a KU in the AI.

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
- SEP issues
    - Deep fakes
    - Applications that misidentify people based on skin color or hairstyle

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)
- Enhance an image by selectively filtering in the frequency domain
- Describe the ethical pitfalls of facial recognition. Can facial recognition be used ethically? If so, how?

GIT/Tangible/Physical Computing

Cross-cutting with Specialized Platform Development and HCI

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)
        - Additive (3D printing)
        - Subtractive (CNC milling)
        - Forming (vacuum forming)
- Internet of Things (reference Networking)
    - Network connectivity
    - Wireless communication
- SEP issue
    - Privacy

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.

Professional Dispositions

- 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
- Accountable for decisions and their ramifications
- The ability to give and receive code reviews
- Effective communication
    - oral
    - written
    - code

Math Requirements

Required:

- 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

Desirable:

- Linear Algebra
    - Eigenvectors and and Eigen decomposition
    - Gaussian Elimination and Lower Upper Factorization
    - Singular Value Decomposition
- Calculus
    - Quaternions

Necessary and Desirable Data Structures

Data Structures necessary for this knowledge area includes:

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

Data Structures desirable for this knowledge area includes:

- Array of Structures vs Structure of Arrays
- Trees (e.g. K-trees, quadtrees, Huffman Trees)

Shared Concepts and Crosscutting Themes

Shared Concepts:

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

Competency Specifications

- **Task 1:** Select the correct image format.
- **Competency Statement:** Select the image format most appropriate for 1) a photograph, 2) a cartoon, and 3) a cut path for a water jet.
- **Competency area:** Application / Theory
- **Competency unit:** Evaluation
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Core
- **Required skill level:** Understand and Explain
- **Core level:**

- **Task 2:** Write a white paper describing the rendering pipeline.
- **Competency Statement:** In general terms explain what happens between the graphics API and the frame buffer.
- **Competency area:** Application
- **Competency unit:** Communicate
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
- **Required skill level:** Understand and Explain
- **Core level:**

- **Task 3:** Animate a bouncing ball.
- **Competency Statement:** Animate a realistically bouncing kickball. Pay attention to the principles of animation.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Animation
    - KA-GIT / KU Advanced Shading
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 4:** Create an organic-looking striped pattern.

- **Competency Statement:** Using your favorite programming language, write a program to implement a reaction diffusion model to produce stripes. Design the user interface so that the parameters can be manipulated in real time.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Core
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Advanced Shading
    - KA-GIT / KU Simulation
    - KA-GIT / KU Interaction
    - KA-HCI / KU-System Design
    - KA-HCI / KU-Understanding the User
    - KA-SDF / KU-Development Methods
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 5:** Create a digital fireworks display customizable by the user.
- **Competency Statement:** Using your favorite programming language, write a program to simulate fireworks using a particle system. Design the user interface so that the majority of parameters are adjustable in real time.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Core
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Simulation
    - KA-GIT / KU Interaction
    - KA-HCI / KU-System Design
    - KA-HCI / KU-Understanding the User
    - KA-SDF / KU-Development Methods
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 6:** Simulate impaired vision.
- **Competency Statement:** Use Gaussian filtering of varying kernel sizes to simulate near-sightedness.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment / Consumer Acceptance
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Core

    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Image Processing
    - KA-SDF / KU-Development Methods
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 7:** Enhance an image lacking contrast.
- **Competency Statement:** Use histogram normalization on a three-channel color image.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Image Processing
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 8:** Create a tic-tac-toe app.
- **Competency Statement:** Create a tic-tac-toe mobile app for both iphone and Android platforms designed for people 6 years of age and older.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU Interaction
    - KA-HCI / KU-System Design
    - KA-HCI / KU-Understanding the User
    - KA-SPD / Mobile Foundations
    - KA-SPD / Mobile Platforms
    - KA-SDF / KU-Development Methods
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

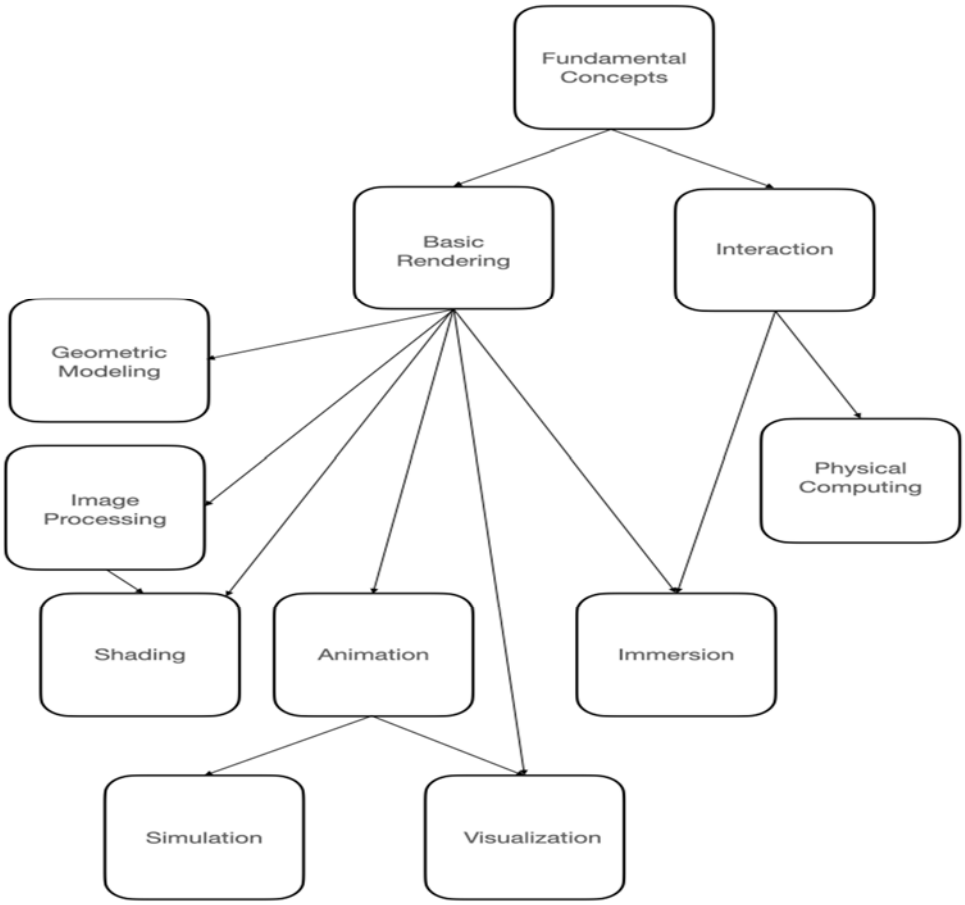
- **Task 9:** Visualize a region's temperature
- **Competency Statement:** Given weather data, design and implement an animation depicting temperature changes for a region over time.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment / Evaluation / Consumer Acceptance
- **Required knowledge areas and knowledge units:**

    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU-Visualization
    - KA-HCI / KU-System Design
    - KA-HCI / KU-Understanding the User
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 10:** Visually compare the COVID 19 infections over time in two locations.
- **Competency Statement:** Given COVID 19 data, design and implement an animation depicting the number of infections in two locations over time so that they can be compared.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment / Evaluation / Consumer Acceptance
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU Fundamental Concepts
    - KA-GIT / KU Basic Rendering
    - KA-GIT / KU-Visualization
    - KA-HCI / KU-System Design
    - KA-HCI / KU-Understanding the User
- **Required skill level:** Apply / Evaluate / Develop
- **Core level:**

- **Task 11:** Create a device to turn on an LED in low light.
- **Competency Statement:** Using a microcontroller and standard electronic components, create an electrical circuit and program the microcontroller to turn on an LED in low light.
- **Competency area:** Application
- **Competency unit:** Requirements / Design / Development / Testing / Deployment
- **Required knowledge areas and knowledge units:**
    - KA-GIT / KU-Fundamental Concepts
    - KA-GIT / KU-Physical Computing
    - KA-SPD / KU-Embedded Platforms
    - KA-HCI / KU-System Design
    - KA-SEP, KU-Privacy
- **Required skill level:** Explain / Apply / Evaluate / Develop
- **Core level:**

Course Packaging Suggestions



Interactive Computer Graphics to include the following:

- GIT KU Basic Rendering: 40 hours
- SEP KUs 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 KU Physical Computing: 30 hours
- SPD KU Embedded Platforms: 10 hours
- SEP KU Privacy and DEI: 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 KU Image Processing: 30 hours
- GIT KU Basic Rendering: 10 hours
- KUs DEI, Privacy, Intellectual Property Law from SEP Knowledge Area: 4 hours

Pre-requisites:

- CS2
- Linear Algebra

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 KU Visualization: 30 hours
- GIT KU Basic Rendering: 10 hours
- HCI, KU Understanding the User: 3 hours
- KUs DEI, Privacy, Ethics from SEP Knowledge Area: 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 KU Simulation: 30 hours
- GIT KU Basic Rendering: 10 hours
- SEP KUs 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.

Course Packaging Suggestions

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, Unity, San Francisco, USA

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
- Ginger Alford, Southern Methodist University, TX, USA

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