Computational thinking across the curriculum: A conceptual framework

May 18, 2009

1 An introduction to computational thinking and our project

The development of computer technologies and computer science has been largely motivated by a desire to support, extend and amplify the human intellect. The first general purpose computer, ENIAC, was built in 1946 to calculate artillery firing tables to support the task of artillery crews. At SRI in the mid-1960s, Doug Engelbart, in a project aptly entitled "Augmenting Human Intellect: A Conceptual Framework" [Engelbart 1962], invented the mouse, the GUI (Graphic User Interface), hypertext (e.g. HTML), networked computers, and collaborative software tools such as tele- and video-conferencing. Today, the Internet, the WWW, a myriad of computer applications, and computational concepts and techniques are available for the purpose of augmenting a person's productivity, and more, in practically all human endeavors.

In order to make an effective use of computer applications and techniques in his/her field, a person needs to have certain skills. One skill is the ability to use basic computer applications such as an editor and a web or file-system browser; this skill is often described as computer literacy. Another skill is a high level understanding of the workings of a computer system, often defined as computer fluency. While computer literacy and fluency are certainly necessary, neither is sufficient for fully realizing the potential that computing can have in augmenting a person's productivity in their field. The third, critical, skill set is the intellectual and reasoning skills that a professional needs to master in order to apply computational techniques or computer applications to the problems and projects in their field, whether the field is economics, art, a science, the humanities, or the social sciences.

This third skill was given the name computational thinking in a recent 2006 CACM article by Jeannette Wing [Wing 2006]. Computational thinking is not new, however. Many of its elements are as old as mathematics itself (e.g. Euclid's 500 B.C GCD algorithm). Computational thinking has been fleshed out and used by computer scientists in the context of computer application development for decades. Computational thinking has also been applied to fields other than computer science for years. For example, computer scientists, psychologists, sociologists,

anthropologists, and biologists have all contributed to applying computational concepts and processes to the field of cognitive science [Gardner 1987]. The application of computational thinking within computer science and related fields has been implicit, because it is the natural approach to problem solving in the field. The application of computational thinking to other fields has also usually been implicit, sometimes without an explicit recognition of the reasoning skills involved. What is different about the recent attention on computational thinking is the emphasis on explicitly defining what it is and explicitly using it to gain new insights into problems in fields outside of computer science. As Wing argues in her seminal article, "the ideas of abstraction, layering of abstractions, and automation, to name a few, are fundamental computer science concepts that have already yielded new insights in the natural sciences and hard social sciences such as economics" [Wing 2006]. She argues that computational thinking is an emerging basic skill that should become an integral part of education, together with reading, writing, critical thinking, and problem solving.

In this project we are developing a framework for implementing Wing's vision in the context of undergraduate education. More specifically, we focus on using liberal studies courses – part of the education of the vast majority of undergraduates – as a vehicle for the teaching of computational thinking. In order to achieve this, a broad consensus among faculty from diverse areas must be reached. In this project we are working toward achieving this consensus by developing a framework that faculty without formal training in information technology can use to understand and integrate computational thinking into their liberal studies courses.

2 A practical definition of computational thinking

Since Wing's article many, including Wing, have attempted to pin down a definition of computational thinking. Wing uses many examples from computer science and "hard sciences" to define the term in her article. Unfortunately, she does not give examples of computational thinking outside the "hard sciences". Wing's article is also not ideal for explaining computational thinking to professionals outside of computing. Part of the contribution of our project is to uncover examples of computational thinking more broadly and to explain computational thinking to faculty in all academic disciplines.

One way to decide whether some analytical process is computational thinking is if it is – or can be – described using computational thinking keywords and whether it can be categorized as employing a fundamental "computing principle". In order to define keywords and categories, we define some terminology:

- **Algorithm** (or procedure, function): An algorithm is a set of rules that describe how to do something, or how to solve a problem. An algorithm may be described as a program, pseudo-code or a less formal step-by-step explanation (even a recipe).
- **Data** (or variable, database, queue): Data is the information that is part of the problem/question, as well how the information is organized and how it is accessed. Distances

between neighboring cities is the data for the problem of computing distances between cities, for example.

- **Abstraction** (or conceptualization, modularization): Abstraction is the pulling out of important properties and the generalization of relationships.
- **Iteration** (or loops, recursion): Iterations involve the repetition of a procedure until a desired goal is reached. In math, iterations occur in long division. In science, it occurs in repeating a step of an experiment, until the desired condition is achieved. In game design and development, it happens in the construction of game versions, from initial prototype to the beta.
- **Object**: An entity that has certain properties and can perform certain actions. A human being, a car, or a calculator software application are all objects.
- **Process**: The execution of an explicit or implicit algorithm. A process could be a human being, a community, or a virus performing some actions. It can also be an actual running program or computer application.
- **System**: A system is a group of processes or objects that interact. A system could be computer network, a flock of birds, a social group including a virtual community, as well as larger entities like economies and biological systems.

2.1 Principles of computing and keywords

Computation is a broad term that encompasses different tasks, concepts and techniques. Similarly, computational thinking involves a broad set of approaches and skills, with applications in many disciplines. In this project we find it useful to define different categories of computational thinking. Using these categories we can understanding computational thinking by distinguishing differences and finding similarities between specific examples in different fields.

In order to understand and organize computational thinking, we will use the categories defined by Denning in his "Great Principles of Computing" project. The goal of Denning's project is to articulate the fundamental principles of computing. Of particular interest to us is one of Denning's motivations: "To establish a new relationship with people from other fields by offering computing principles in a language that shows them how to map the principles into their own fields." He claims that the "principles of computing can be organized into seven categories, each emphasizing a unique perspective on computation." The Great Principles of Computing, according to Denning, are: *computation, communication, coordination, recollection, automation, evaluation, and design.*

These seven "principles" form a foundation that is useful to recognize, organize, and categorize instances of computational thinking and build a framework that can translate computational thinking to contexts outside of computer science. The categories should be seen as a starting point, subject to modification as necessary. For example, abstraction is viewed by Denning

as belonging to the design principle, when it could be argued that abstraction is a principle on its own. Below is our definition of each principle. Note that the definitions are somewhat different from Denning's because we are defining then in a context larger than computer science. For each principle, we also define a list of keywords typically used when expressing concepts or processes that the principle embodies.

- **Computation** is the execution of an algorithm, a process that starts from an initial state containing the algorithm and input data, and goes through a sequence of intermediate states until a final, goal state is reached.

  Keywords: *state and state transitions, algorithm, program, exhaustive search, backtracking, recursion and iteration, decision tree, randomization, problem complexity, computability.*

- **Communication** is the transmission of information from one process or object to another.

  Keywords: *information and its representations, messages, sender/receiver, communication protocol, message compression, message encryption, error correction, communication channel, encoder/decoder, noise, authentication.*

- **Coordination** is control (through communication, for example) of the timing of computation at participating processes in order to achieve a certain goal.

  Keywords: *interacting processes/agents, inter-process protocols including communication protocols, synchronization, events and event handling, flow and sharing dependencies, concurrency.*

- **Recollection** is the encoding and organization of data in ways to make it efficient to search and perform other operations.

  Keywords: *storage media, data hierarchy and organization, data manipulations including data insertion/query/removal and their efficiencies, data locality and caching, virtual representations, naming system, relative and absolute references.*

- **Automation** is the mapping of computation to physical systems that perform them.

  Keywords: *mapping of algorithm to physical computing object, mechanization, applies to repetitive processes and provides error-free, consistent, fast, cost-efficient executions.*

- **Evaluation** is the statistical, numerical, or experimental analysis of data.

  Keywords: *visualization, data analysis, statistics, data mining, simulation, recommender system, computational experiment.*

- **Design** is the organization (using abstraction, modularization, aggregation, decomposition) of a system, process, object, etc.

  Keywords: *abstraction, layers of abstraction, modeling, modularity, information hiding, class, architecture, aggregation, pattern, underlying structure.*

2.2 What benefits does computational thinking bring?

When reaching out to people who do not work in the area of computing, it is important to emphasize the possible benefits of applying computational thinking. There is a tendency to be (justifiably) skeptical of applying new approaches, particularly when existing approaches have yielded satisfactory results and when the new approaches come from an area outside of one's expertise.

Computational thinking:

1. Facilitates new ways of seeing existing problems
2. Emphasizes creating knowledge rather than using information [Phillips 2008]
3. Presents possibilities for creatively solving problems [Phillips 2008]
4. Facilitates innovation

Examples illustrating these benefits are provided below.

2.2.1 New views of existing problems

Interpreting DNA strands as data or code is the classic example of computational thinking that was responsible for a revolution in molecular biology. Abstracting the complexity of organic chemistry to linear strings of 4 characters allowed researchers to view DNA in a completely new way. DNA strands encode instructions to be used in the development of the organism and this encoding is an example of data organization. Genetic mutations can be viewed as randomized computation and the cellular growth and interactions between cells may be viewed using the ideas of coordination and communication. By taking this new view of DNA, researchers have been able make dramatic advances in the area of molecular biology, such as the completion of the Human Genome Project in which the all the genes in human DNA were identified, the sequences making up human DNA were determined, and tools for analyzing the information were developed [Genome 2008].

2.2.2 Creating knowledge

Laboratory experiments in psychology find that media violence increases aggression in the short run [Anderson etal 2003]. Because of this effect, there exist a variety of restrictions on American children's ability to view violent movies and purchase violent games. Little understanding of the issue outside of the lab was known, however, until two researchers used crime statistics and movie databases to study the correlation between the release of a violent movie and actual violence on a given weekend. The researchers report "like the laboratory experiments, we find indirect evidence that movie violence increases violent crime; however,

this effect is dominated by the reduction in crime induced by a substitution away from more dangerous activities" [Dahl 2008]. The attendance of violent people at violent movies results in less crime because violent people were engaged in activities that made it more difficult to engage in crime, namely watching a movie. Applying computational thinking techniques such as data analysis and searching, allowed these researchers to gain new insight into the actual effects in the real world of a phenomenon observed in the laboratory.

2.2.3 Creatively solving problems

Origami, the Japanese art of paper folding, is something that has engaged artists, mathematicians, and computer scientists for many centuries. Recently the field of computational origami has arisen, in which algorithms and theory devoted to the solution of origami problems by mathematical means are developed [Lang 2008]. In order to construct an origami representation of an object, the object's features may be abstracted using graph theory. Once an abstract graph model is obtained, algorithms that construct the sequence of folding have been developed to construct the origami representation of the object. This in turn implies that the origami construction can be automated. By abstracting and automating this process, new and incredibly complex origami structures can be constructed [Lang 2008]. The process of automating origami has also allowed people in the field to teach the underlying concepts of origami, with a particular focus on creating original origami designs [Lang 2003]. For a video demonstrating computational origami, see:
http://www.ted.com/index.php/talks/robert_lang_folds_way_new_origami.html

2.2.4 Innovation

Writing music is a very creative activity, but aspects of it are certainly open to automation. Various music recommender systems employ social tags that have been created by users of the system to classify and recommend new music, and some people have even considered the problem of automatically creating social tags for music [Eck etal 2007]. The fact that social tags can be constructed at all suggests that musical styles are able to be classified, which suggests that they can be analyzed in more depth using computational thinking. An interesting final project completed by a student at the University of Southern California considers the issue of producing an automatic accompaniment system [Chuan 2007]. The system "assist(s) music lovers in writing a complete song with sophisticated chord progressions. The system takes melody as input, and harmonizes it with style-specific accompaniments" [Chuan 2007]. In addition to building the system, Chuan also conducts two experiments. In the first, he harmonizes a Beatles song using the styles of three different bands (Radiohead, U2, and Green Day). In the second, he uses the system to generate a new Radiohead song based on other songs of the band. Whether the results are good music is a matter of subjective taste, but computational techniques allowed this student to create musical innovations.

3 Our approach to the first year: Solving the chicken and egg problem

In order to build the framework, we need the participation of a large, diverse faculty spanning many academic disciplines. The non-computing faculty, however, will not be familiar with the concept of computational thinking. Without an extensive set of computational thinking examples, best practices, and assessment tools across diverse areas, they are naturally reluctant to join the project.

To generate the necessary artifacts to be able to move our project into a variety of disciplines, we have taken a bootstrapping approach to the first year of our project. We have generated examples, best practices, and assessment tools by first focusing on Liberal Studies courses closer to computing, i.e. courses taught by faculty members in the School of Computing and Digital Media (CDM). CDM is an academic unit of DePaul University that offers of a unique combination of technical computing and digital arts and media degree programs. CDM includes faculty trained in computer science and related fields as well as faculty trained in interactive media, animation and cinema.

The reason for starting with courses offered by CDM is simple: the computational thinking skills taught in these courses are more explicit and easier to understand and the faculty who teach these courses are more familiar with computing as a discipline. By doing so, we have leveraged the unique nature of CDM where technology unites faculty with training in computer science and faculty whose focus lies outside of computer science. Our goal is that the materials produced while working together will enable CDM faculty to make computational thinking accessible to faculty across the university and in other institutions.

Now that the project is moving into its second year and the framework is populated with examples, we are ready to move to courses further from computing and taught by faculty outside of CDM. We will use the examples already in the framework to uncover computational thinking skills implicitly taught in these courses and/or discover how computational thinking can be applied in the courses in novel and effective ways.

3.1 An overview of the results from the first year

Below we provide a high-level overview of the examples that have been developed by CDM faculty during the first year of the project. We give a brief description of each instance of computational thinking, grouping the instances by the computing principle that they illustrate. For each instance, we also provide the title of the relevant course and the Liberal Studies learning domain the course belongs to.

3.1.1 Automation

- Airline order processing system (ECT 250, Internet, Commerce and Society; Scientific Inquiry Domain)

  On a typical online retailing website, orders are taken through online forms, payment information is collected, inventories are updated, and products are made ready for shipping. The entire order fulfillment process is automatically handled by computers with minimum human intervention. Not every process can be automated however and deciding whether a process can or cannot be automated requires an understanding of the mapping between the algorithm underlying the process and the computing system potentially executing it.

- Scripting for hand-drawn animation (ANI 201 Animation I; Arts & Literature Domain)

  If digital video is analyzed at the level of the frame, it can be abstracted into large amounts of discrete, incrementally changing data. Using scripts in applications such as Photoshop, After Effects and Maya, manipulations of one frame can be automated and applied to any number of other frames; the automation of changes enables freer creative experimentation with an efficient feedback process. A student who recognizes the way that these processes can be separated and adapted to multiple uses and software will be able to work faster and with more creative control.

3.1.2 Communication

- Communication protocols (IT 130, The Internet and the Web; Scientific Inquiry Domain)

  When two entities communicate, and particularly if they do so in an automated way, they follow a protocol that specifies, at any given step, the state of each entity (and, in particular, who is listening/receiving and who is talking/sending a message), messages that can be received/sent, and a state transition function that maps (state, received message) pairs to (new state, sent message) pairs. A student who understand how to break down the structure of communication protocols will be a more effective user of them.

3.1.3 Computation

- Defining sub-goals and recursive thinking (CSC 235, Problem Solving; Scientific Inquiry Domain)

  The technique aims at defining a simpler problem (or problems) than the original problem, whose solution leads to a solution to the original problem. Recursive thinking is a core computational thinking approach that can significantly simplify the solution of a problem.

- Cryptography and the hardness of computation (CSC 233, Codes and Ciphers; Scientific Inquiry Domain)

  All modern cryptosystems are based on the assumption that certain mathematical problems are intractable, that is, hard to solve by a computer. While traditional systems use estimates on the size of the keyspace as a rough measure of complexity, the hardness of breaking modern systems can often be tied more closely to the difficulty of computing a mathematical value. For example, Diffiee/Hellman is based on the assumption that a variant of the discrete logarithm problem is hard to compute, and RSA is based on the hardness of factoring large numbers. Students realize that in modern cryptography computation is more than a tool, but a meta-tool: the very fact that some task is computationally hard, can be exploited to build a feasible solution to another problem (public-key cryptography).

- Searching and pruning (CSC 235, Problem Solving; Scientific Inquiry Domain)

  The "search and prune" technique uses heuristics and exhaustive search methods to arrive at a solution. It is usually used to solve problems that do not possess structure and "easy solutions". For such problems, one may need to perform a systematic search to check all possible choices that may lead to a solution.

- Modularization in complex 3D Modeling (ANI 230, 3D Modeling)

  Because most 3D model are inspired by or have a foundation in the world we are familiar with, there is an inherent complexity in what a 3D modeler is expected to achieve in order to give their models an accepted level of believability. On the other hand, the production time and the computer processing of 3D models require them to be simple, not complex. Techniques such as abstraction, modularization, automation, and randomization are necessary to create realistic models that can be efficiently designed and processed.

3.1.4 Coordination

- Cybernetic Systems (GAM 224, Introduction to Game Design; Arts and Literature Domain)

  A game can be modeled as a cybernetic system. In a cybernetic system, the "encouragement" of movement in a certain direction is called a positive feedback loop. A positive feedback loop has the effect of creating exponential growth or decline in some aspect of the system. On the other hand, the "discouragement" of movement in a certain direction is called a negative feedback loop, and its effect is to stabilize a system or maintain equilibrium. In order to create a game that is balanced, students must understand how the alteration of the game state through positive and negative feedback loops affects the player's perception of the fairness and competitiveness of the game. Students will model and understand the internal state of a game system, the mechanisms detecting the game state, and the mechanisms altering the game state.

3.1.5 Design

- Abstracting properties into classes (HCI 201, Multimedia and the World Wide Web; Scientific Inquiry Domain)

  A class is the abstraction of key properties of objects and patterns. It is used to hide less relevant or variable properties of the object or pattern and to provide a way to create multiple instances (of objects or patterns) sharing the same properties. The ability to identify and abstract certain properties of a group of items into a class is a key computational thinking skill.

- Representation of Game Rules (GAM 224, Introduction to Game Design; Arts and Literature)

  Game rules can be categorized into three types: constituative, operational, and implicit. Operational rules are the guidelines players require in order to play, such as the rules printed on the box of a board game. Constituative rules are the underlying logical and mathematical structures in the game. Implicit rules are the "unwritten" rules of the game, such as rules about decorum. Important here are the first two types of rules: constituative and operational. Two games are considered to be the same if there is a 1-1 relationship between the constituative rules of the two games, so that if you can find a winning strategy in one game you can use the mapping to find a winning strategy in the other game. At the same time, the operational rules for two structurally identical games can vary significantly. While the operational rules are what make a game enjoyable to play, the constituative rules are the ones that more experienced players are using when they find winning strategies.

  Abstracting game rules in different ways allows students to see the relationship between different abstractions of rules, the modeling of game behavior, and the underlying structure of a game.

- Structure of screenplays (DC 201, Introduction to Screenwriting; Arts and Literature Domain)

  In a properly structured movie, the story consists of six basic stages, which are defined by five key points in the plot. Not only are these points always the same; they always occupy the same positions in the story. So what happens at the 25% point of a 90-minute comedy will be identical to what happens at the same percentage of a three-hour epic. (These percentages apply both to the running time of the film and the pages of a screenplay.) Students should demonstrate competency in abstracting the structure for a narrative screenplay by breaking down the three-act structure and five plot points to successful and disappointing motion pictures.

3.1.6 Evaluation

- Constructing a histogram and visualizing data (CSC 239, Personal Computing; Scientific Inquiry Domain)

The construction of a histogram is an important first step in any statistical analysis. The visual appearance of the histogram can be used to determine if there are outliers and what the appropriate statistics are for reporting center and spread. For more complex statistical analysis, the shape of the histogram can be used to determine what analyses are appropriate and which are not. However, the creation of a histogram is not a simple process; it is iterative and benefits greatly from automation.

- The Substitution Cipher (CSC 233, Codes and Ciphers; Scientific Inquiry Domain)

  In a substitution cipher each letter is replaced by some other (unique) letter of the alphabet. As such, a substitution cipher does not change the frequency distribution of the letters, it just disguises it by renaming the letters. This observation, with a little bit of statistical mathematics, can be used to automate the exhaustive breaking of the shift cipher, that is, the process can be done by a computer (or simulated by hand) without any human involvement. For the students to understand and recreate this process, they need to understand one of the most basic principles in computational thinking: when trying to understand a process (be it a shift cipher, a sorting algorithm, or some other more complex procedure) it is difficult to argue about what changes; indeed, it is necessary to understand what a particular process does not change. In the case of the shift cipher, one such property is the frequency profile.

3.1.7 Recollection

- Tree data structure (IT 130, The Internet and the Web; Scientific Inquiry Domain)

  Trees are used as a structure for hierarchical data organization. The big idea is: organize large amounts of items into groups and continue partitioning each group into subgroups, recursively. A tree data structure allows efficient insertion and search. Understanding and using a tree data structure is a fundamental computational thinking skill.

- Absolute and relative references (IT 130, The Internet and the Web; Scientific Inquiry Domain)

  A setting or a location can be specified relative to a current state or in terms of an absolute specification. For example, a location of a building can be specified relative to a current location (e.g. 3 blocks north of here) or by an absolute location (e.g. 25 S. Elm St.,). The concept of relative and absolute references are commonly used in IT applications and include file name references, cell references in a spreadsheet and configuration settings, such as those used in page formatting.

- Caching (IT 130, The Internet and the Web; Scientific Inquiry Domain)

  Caching replicates data originating from a remote resource locally, so it is close to the consumer of the data. The goal of caching is to improve data access latencies. What data is cached will depend on assumptions about future data accesses which are usually not deterministic. Understanding the usefulness of caching and the ability to reason about what data to cache makes a user of software applications more efficient.

4 Computational thinking instances

In this section we provide the details for the computational thinking instances that have been developed and categorized by CDM faculty during the first year of the project. The instances are grouped by computational principle, and the entry for each instance includes a high-level description of the instance, computational thinking learning goals for the instance, discussion questions to guide the introduction of computational thinking ideas, and assessments to evaluate how well students have mastered the learning goal(s).

4.1 Automation

4.1.1 An Online Order Processing System

On a typical online retailing website, orders are taken through online forms, payment information is collected, inventories are updated, and products are made ready for shipping. The entire order fulfillment process is automatically handled by computers with minimum human intervention. Not every process can be automated however and deciding whether a process can or cannot be automated requires an understanding of the mapping between the algorithm underlying the process and the computing system potentially executing it.

Learning Goal

Students are able to analyze a process and argue whether it has the properties (e.g. can be described using a clear algorithm, exhibits repetitions, etc.) that make it amenable to automation; students understand, in context, the benefits (error-free, consistency, speed, low-cost) automation brings.

Case Description

Dell is one of the top sellers of PC computers. In the first quarter of 2007, Dell shipped about 8 million units based on Gartner and IDC (International Data Corporation) reports. It is evident that the number of orders was huge. How did Dell process the orders? Let's look at Dell's order processing system at http://www.dell.com/. Discussion Questions

Discussion questions

Q1. What characteristics did you notice during the order process? (standardized procedure, repetitive, etc.)

Q2. Did you perceive any human intervention during this process? Are humans good at processing this type of orders? Why? (benefits of human information processing?) Why not? (limits of human information processing)

Q3. Who actually handled all the orders? (computers). Define the term "automation" in computational thinking.

Q4. What are the benefits of automation (consistent, error free, fast, efficient, cost-effective, etc.)

Q5. Can you think of some other examples of automation?

Q6. What characteristics of order processing make automation helpful? (Large number of orders, standardized procedure, repetitive, etc.)

Assessment

The following question will be added to an assignment: The registrar's office at a community college is short of people handling course registrations due to the recent surge of student enrollment. This results in students waiting in a line for hours and employees working for long hours. They decide to develop a Web-based course registration system to automate this process. Please answer the following questions: 1) Please use a flow chart and your own language to describe a common registration process. 2) What characteristics of the registration process make it suitable for automation? 3) What are the benefits of using automation?

4.1.2 Scripting for hand-drawn animation

If digital video is analyzed at the level of the frame, it can be abstracted into large amounts of discrete, incrementally changing data. Using scripts in applications such as Photoshop, After Effects and Maya, manipulations of one frame can be automated and applied to any number of other frames; the automation of changes enables freer creative experimentation with an efficient feedback process. A student who recognizes the way that these processes can be separated and adapted to multiple uses and software will be able to work faster and with more creative control.

Learning Goal

Students should understand the pragmatic value of automation for digital media manipulation. They should be able to apply the principles of automation to a variable set of objectives in a production environment.

Case description

In ANI 201 Animation I students are introduced to hand-drawn animation, an art form that appears far removed from computation. Since its discovery in the late 1800's, however, technological innovation has been critical for the practical and economical production of hand-drawn animation due to the large number of drawings required (from 8-24 drawings per second). One aspect of the process, the cleaning up and resizing of the scanned drawings, can be made simpler by using Photoshop's Action and Batch Processing tools.

After completing approximately 60 drawings for the project, students input the drawings into Photoshop using a flatbed scanner that has been adapted to 3-pin registration (a method that allows accurate registration between successive drawings). In order for automation to work,

it's critical that all the drawings are correctly registered to each other and are of equal size when scanned.

Discussion Questions

Q1. What are the limitations of the Adobe batch processing system? What can't it do?

Q2. How does the creation of a batch processing script differ from creating a computer program?

Q3. How could the ideas of batch processing be applied to other animation/video software, such as Flash, After Effects, Maya or Final Cut?

Q4. Show and discuss a simple example of using the scripting system in After Effects.

Assessment

Students will adapt the script created in class to other situations and they will be assessed on whether they effectively apply it to enable flexibility, experimentation, and creativity. For example:

1. Students will choose a step in the automated process and batch process the same group of files using different settings. Possibilities include: resizing the target frame, changing the levels settings, applying a sharpen filter, cutting a section of the image that contains an unwanted scanning artifact, etc.

2. Students will be asked to figure out how to best tackle a production problem using batch processing. For example:

   > You have a set of 100 high resolution images that are to be processed for use on a website. You are asked to create one set of the images resized to 600 pixels wide with high quality JPEG compression, and another set resized to 300 pixels wide with medium quality compression. Describe the steps you would take to accomplish the tasks efficiently using batch processing.

4.2 Communication

4.2.1 Communication protocols

When two entities communicate, and particularly if they do so in an automated way, they follow a protocol that specifies, at any given step, the state of each entity (and, in particular, who is listening/receiving and who is talking/sending a message), messages that can be received/sent, and a state transition function that maps (state, received message) pairs to (new state, sent message) pairs. A student who understand how to break down the structure of communication protocols will be a more effective user of them.

Learning Goal

Students understand that automated communication requires a precise communication protocol, including a specification of the set of states the communicating entities can be in and, for each state, the set of messages the entities can receive/send.

Case description: TCP/IP

The Internet provides two fundamental communication protocols: TCP for connection-oriented reliable communication and IP for best-effort connection-less communication. Sending an IP message is like sending a letter through the mail: the USPS will do its best to deliver it, without guarantees. TCP uses IP for message transport but augments it with the requirement that the receipt of every message must be acknowledged (using an IP ACK message) to the sender. An unacknowledged message will be resent.

Discussion questions

Q1. What states/messages are required for the IP protocol? What is the advantage of the IP protocol? What is the disadvantage?

Q2. What state/message types are needed, at a minimum, for the TCP protocol?

Q3. What state transitions can you describe?

Q4. What is the advantage of TCP? What is its cost?

Case description: HTTP protocol

Web browsers and web servers communicate using the HTTP protocol. The web server is waiting for client requests in a listening state. A click of a hyperlink in a client-side browser will create a TCP connection between client and server. This connection will be used by the client browser to send a HTTP "GET" message to the server along with the url of the requested resource. The server will respond with a reply code and, if the request was successful, the requested content.

Discussion questions

Q1. What happens when we click on a hyperlink? How does the browser connect to the web server hosting the resource? Should this be a reliable or unreliable connection?

Q2. What information should the request message contain?

Q3. What information should the reply message contain? How are errors handled?

Assessment

Web browsers cache recently visited content. Future requests for the same content can be served from the cache. How long should the browser be doing this? Well, it depends on the content and the content owner (web server) determines this. How can you augment the HTTP protocol to let the web content server inform the web browser how long should the cached content be considered fresh?

4.3 Computation

4.3.1 Defining Subgoals

This technique aims at defining a simpler problem (or problems) than the original problem, whose solution leads to a solution to the original problem.

Learning goal

The students should understand the Defining Subgoals technique, and they should be able to apply it. In particular, given an appropriate problem, they should be able to identify the subproblem within it whose solution leads to a solution of the original problem. They also need to illustrate the process of going from a solution of the subproblem to a solution for the original problem.

Case description: Towers of Hanoi

The Tower of Hanoi problem has the following setup. Given are 3 pegs labeled A, B, C and n disks (of different sizes) placed on top of each other around peg A in a decreasing order of their size (largest at the bottom). The problem is to transfer the disks from peg A to peg C under the restriction that 1) only one disk can be moved at a time, 2) a picked up disk must be released around some peg before another disc is picked up, and 3) no disk can be placed on top of a smaller one.

Discussion questions

Q1. Demonstrate a solution to the Towers of Hanoi problem with 4 disks. How many steps does your solution take?

Q2. Your ultimate goal is to transfer the n disks from peg A to peg C. Define an appropriate subgoal.

Q1. Show how using the subgoal defined above you can achieve the ultimate goal of transferring the n disks from peg A to C.

Assessment

The student will, for example, be assigned the problem of constructing a straight-line passing through a given point and parallel to a given straight-line. They will be asked to describe

how, using a compass and a straight-edge ruler, they can do this. They will be asked to define an appropriate subgoal, and show how achieving this subgoal leads to achieving the original goal.

4.3.2 Cryptography and the hardness of computation

All modern cryptosystems are based on the assumption that certain mathematical problems are intractable, that is, hard to solve by a computer. While traditional systems use estimates on the size of the keyspace as a rough measure of complexity, the hardness of breaking modern systems can often be tied more closely to mathematical problems. For example, Diffie/Hellman is based on the assumption that a variant of the discrete logarithm problem is hard, and RSA is based on the hardness of factoring large numbers. So in modern cryptography computation is more than a tool, it has become a meta-tool: the very fact that some task is computationally hard, can be exploited to build a feasible solution to another problem (public-key cryptography).

Learning Goal

Understand the role of computation, modular arithmetic, randomness and well-designed cryptographic protocols in modern cryptography;

Case Description: Public-Key Cryptography

In public-key cryptography two parties (typically called Alice and Bob) can communicate securely over a public channel, even if they have never met before or exchanged any secret information: they can establish secure communication between them while other parties are listening (the other person is always known as Eve, the eavesdropper). Intuitively, this seems impossible: how could you exchange a secret with somebody you don't know over the phone, say, in the presence of other people listening in on your communication? However, public-key cryptography shows that this is indeed possible, and the mathematics involved for the easier systems is not very difficult and can be understood by the students.

Discussion questions

Q1. Do an example of a Diffiee/Hellman key exchange. How does this make public-key cryptography possible?

Q2. Do an example of the three-way protocol. Why can this system not be broken using logarithms? What about discrete logarithms?

Q3. How can we calculate the average salary of everybody in the room without anybody learning anybody else's salary? Hint: the problem can be solved using a trusted third party (Trent), but what if we do not even allow that? We can assume that everybody is honest, but we do not want to share our salary information.

Q4. Do an example of bit-commitment using the hardness of the discrete log problem. How can we use bit-commitment to convince somebody that we have found a legal coloring of a graph?

Assessment

Students will set up a Diffiee/Hellman system for key-exchange and set up a secret-sharing theme requiring randomness.

4.3.3 Searching and Pruning

The "search and prune" technique uses heuristics and exhaustive search methods to arrive at a solution. It is usually used to solve problems that do not possess structure and easy solutions". For such problems, one may need to perform a systematic search to check all possible choices that may lead to a solution.

Learning goal

The student should learn how to apply systematic search procedures and understand how they can be automated. Given an appropriate problem, students should be able to illustrate how to apply the process of searching and pruning to arrive at a solution.

Case description: Queens of Chess

The problem asks to place 8 queens on a chessboard (or, in general, to place n queens on an nn chess board) such that no two queens can attack each other.

Discussion questions

Q1. Find a solution to the problem on an $8x8$ chessboard by trial and error.

Q2. Suppose that, in your search, you were able to place the i-th queen successfully on the board but you couldn't place the $i + 1$-st queen. How do you modify your search?

Q3. Can you devise a systematic method to list all solutions to the problem?

Assessment

The students will be give na maze and asked to answer the following:

1. Show how to model the maze using a graph. What are the vertices and edges?
2. Apply depth-first search to the graph to find a solution to the maze.

4.3.4 Modularization in complex 3D modeling

Because most 3D model are inspired by or have a foundation in the world we are familiar with, there is an inherent complexity in what a 3D modeler is expected to achieve in order to give their models an accepted level of believability. On the other hand, the production time and the computer processing of 3D models require them to be simple, not complex. Techniques such as abstraction, modularization, automation, and randomization are necessary to create realistic models that can be efficiently designed and processed.

Learning Goal

Students are able to identify visual patterns in a complex environment or object in order to break it into groups of repetitive modular components. They are then able to use automation and randomization to efficiently design a realistic reconstruction of the environment or model in 3D space.

Case description: Environment Modeling

When one considers the problem a modeler faces when she is required to create a field of grass, it quickly becomes apparent that modeling each individual blade, texturing it, and placing each in a scene would be a completely inefficient use of time and even then may not result in a usable finished model. Using modularization and automation, a 3D modeler would instead create a single blade of grass which would then be duplicated to fill the required area. The organic" seeming placement, rotation, scale, and relative color of individual blades can then be achieved either by a simple randomization script or (more often than not) some random clicks of the mouse.

Students are required to model an interior warehouse environment out of simple polygon primitives. Along with modeling a simple I-beam skeleton, they must create a believable space by modeling three different kinds of inventory and then by arranging them appropriately within the warehouse.

Discussion questions

Q1. What challenges are presented when trying to fill the warehouse space with inventory?

Q2. Describe some ways in which those challenges can be met? What are the advantages and disadvantages to each solution?

Q3. Do the solutions change depending on whether the warehouse is modeled for a film as opposed to a video game?

Q4. What issues arise from duplicating model groups? How can these issues be addressed?

Q5. What types of model attributes can randomness be applied to? Which ones give desirable results?

Q6. Does using an automated script for randomness offer any benefits over creating it by hand? Are there benefits to doing it by hand?

Assessment

Students will be required to turn in both 3D models and rendered images of a finished warehouse space. Both images and model will be evaluated for the student's ability to:

- Assemble efficient groups of modular components.
- Create the illusion of a complex space through the duplication of modular components.
- Break the patterns of repetition of warehouse inventory with randomization.

4.4 Coordination

4.4.1 Understanding Cybernetic Systems

A game can be modeled as a cybernetic system. In a cybernetic system, the "encouragement" of movement in a certain direction is called a positive feedback loop. A positive feedback loop has the effect of creating exponential growth or decline in some aspect of the system. On the other hand, the "discouragement" of movement in a certain direction is called a negative feedback loop, and its effect is to stabilize a system or maintain equilibrium. In order to create a game that is balanced, students must understand how the alteration of the game state through positive and negative feedback loops affects the player's perception of the fairness and competitiveness of the game. It requires students to model and understand the internal state of a game system, the mechanisms detecting the game state, and the mechanisms altering the game state. Students would in these case use coordination and evaluation computational thinking skills.

Learning goal

Students will be able to describe what a negative or positive feedback loop is, how such a feedback loop can be incorporated into an existing game, and how that feedback loop impacts the experience of both winning and losing players in the game.

Case description: Classifying feedback loops

A cybernetic system is one in which the behavior of the system is controlled by a negative feedback loop, a positive feedback loop, or in some cases, several of each. A negative feedback loop acts to move the state of the system in the direction of its previous state. A positive feedback loop acts to move the state of the system in the direction it is currently moving.

Discussion questions

Students will consider game rules found in common board games and be consider the following questions for each rule:

Q1. Classify each rule as a positive feedback loop, negative feedback loop, or neither.

Q2. If positive, indicate what direction the game state is moving in and how the rule aids the movement of the state of the system.

Q3. If negative, indicate what state the rule is trying to preserve and how the rule keeps the game in that state.

Q4. If neither, indicate why.

Q5. If negative or positive, describe the impact that the rule has on a player who is winning and the impact that the rule has on a player who is losing. How would each type of player perceive the rule?

Game rules that may be considered are:

1. You must reach the final/winning square (square 100) in Chutes and Ladders with an exact roll. For example, if you are on square 96 and you roll a 6, you do not advance but instead lose your turn.
2. In Checkers, if a piece reaches the far end of the board, then it becomes a king". A king" is allowed to move and jump diagonally backward and forward, unlike ordinary pieces which may only move and jump diagonally forward.
3. In Candyland, penalty spaces on the board cause players to remain stuck in a certain position until they draw a specific card that frees them from the space.
4. Players landing on properties owned by another player in Monopoly must pay rent. A player who owns an entire color group of properties may charge double the rent for any unimproved property in that color group.

Case description: Describing feedback loops

Pick one example of a positive feedback loop and one example of a negative feedback loop in a computer game you have played. For each of the positive and negative feedback loop examples you describe:

Q1. Describe the direction that the game state is moving when the positive feedback loop begins, and how the positive feedback loop enhances that game state.

Q2. Describe the state that the situation or rule is trying to preserve, and how the negative feedback loop causes the game state to be stabilized.

Q3. Describe how the positive feedback loop enhances the game play. Be specific about how the positive feedback loop modifies game play, paying special attention to its impact on both the winning and losing players.

Q4. Describe how the negative feedback loop enhances the game play. Be specific about how the negative feedback loop modifies game play, playing special attention to its impact on both the winning and losing players.

Assessment

Consider an existing board, card, or computer game. Modify the game to include a positive or negative feedback loop not already present in the game. In doing so, describe:

- Precisely where in the game (under what circumstances and when) you are introducing the negative or positive feedback loop
- How the negative or positive feedback loop works, and what game state it is either enhancing (in the case of a positive feedback loop) or stabilizing (in the case of a negative feedback loop)
- How your change to the game makes it more fun. What balance issues is the change to the game addressing? How will the change to the game affect winning and losing players? Do you anticipate any potential problems that your change could introduce?

4.5 Design

4.5.1 Class Membership

A class is the abstraction of key properties of objects and patterns. It is used to hide less relevant or variable properties of the object or pattern and to provide a way to create multiple instances (of objects or patterns) sharing the same properties. The ability to identify and abstract certain properties of a group of items into a class is an example of computational thinking that belongs to the Design category.

Learning goal

Students can identify a group of (e.g. visual) elements that should all have the same property and abstract them into a class; they should then be able to specify class-wide (e.g. visual) properties that achieves the desired uniformity of the visual elements.

Case description: Style classes

A style rule (using CSS) may specify visual formatting properties for a class. These formatting properties are then applied to all HTML tags that have the same class name as the style rule.

Discussion questions

Q1. How does the use of a style class promote a consistent design across multiple web pages?

Q2. Can the class name by any sequence of characters? What are the advantages of giving it a meaningful name?

Assessment

The students will write a CSS rule that will format all HTML tags of a specific class so that their content appears in a particular color.

4.5.2 Representation of Game Rules

Game rules can be categorized into three types: constituative, operational, and implicit. Operational rules are the guidelines players require in order to play, such as the rules printed on the box of a board game. Constituative rules are the underlying logical and mathematical structures in the game. Implicit rules are the unwritten" rules of the game, such as rules about decorum. Important here are the first two types of rules: constituative and operational. Two games are considered to be the same if there is a 1-1 relationship between the constituative rules of the two games, so that if you can find a winning strategy in one game you can use the mapping to find a winning strategy in the other game. At the same time, the operational rules for two structurally identical games can vary significantly. While the operational rules are what make a game enjoyable to play, the constituative rules are the ones that more experienced players are using when they find winning strategies.

Abstracting game rules in different ways is an example of computational thinking; it allows students to see the relationship between different abstractions of rules, the modeling of game behavior, and the underlying structure of a game.

Learning Goal

Students will be able to abstract the operational rules of a simple board game to find the underlying constituative rules for the game and use the constituative rules to comment on strategies that may exist for the game.

Case description: Tic-Tac-Toe and 3-to-15

Students will consider the well-known rules of Tic-Tac-Toe as well as the rules of 3-to-15 described as follows:

1. Two players alternate turns
2. On your turn, pick a number from 1 to 9. You may not pick a number that has already been picked by either player.
3. The first person to obtain a set of exactly 3 numbers that sum to 15 wins the game

4. If all numbers between 1 and 9 have been chosen and no player has a subset that sums to 15, the game ends in a draw

Discussion Questions

Q1. What strategies exist for Tic-Tac-Toe? Include both strategies for winning and for preventing the other player from winning.

Q2. What strategies exist for 3-to-15? Include both strategies for winning and for preventing the other player from winning.

Q3. Are the two games the same? Why?

Q5. Translate a strategy for Tic-Tac-Toe into a strategy for 3-to-15. For example, describe a blocking strategy for 3-to-15 derived from a blocking strategy for Tic-Tac-Toe.

Q6. Which game is easier to play using its operational rules? Why?

Case description: Chutes and Ladders

The Chutes and Ladders children's board game will be considered. The goal of this activity is to find the set of constituative rules of Chutes and Ladders.

Discussion questions

Q1. How can you represent the spinner? How can you represent the player's movement on the board without using a board? How will the chutes" be represented? Explicitly list all of the constituative rules that must be included for the chutes" in the game. How will the ladders" be represented? Explicitly list all of the constitutative rules that must be included for the ladders" in the game.

Q2. How do you handle the winning condition using this model?

Q3. Does the purely constituative version of Chutes and Ladders have the same feel as the original game?

Q4. Are there any strategies that the constituative rules make clear to you?

Assessment

Consider the board game Candyland. Develop a set of constituative rules for Candyland by:

1. Constructing a table that represents the positions on the board for each of the color blocks and picture cards. The table should include a representation for the two shortcuts (gumdrop pass and rainbow trail).

2. Describing a way to randomly produce each of the card combinations from the deck.

3. Describing how each of the 3 penalty spaces (gooey gumdrops, lost in the lollypop woods, and stuck in the molasses swamp) will be handled in your rules.

4. Describing the winning condition for the game.

After you have constructed the constituative rules, describe any strategies that the rules make clear. If there are no strategies that your constituative rules illuminate, explain why that is. Is it a property of your representation? Or is it a property of the game?

4.5.3 Screenplay structure

In a properly structured movie, the story consists of six basic stages, which are defined by five key points in the plot. Not only are these points always the same; they always occupy the same positions in the story. So what happens at the 25% point of a 90-minute comedy will be identical to what happens at the same percentage of a three-hour epic. (These percentages apply both to the running time of the film and the pages of a screenplay.)

PLOT POINT #1 Opportunity or Inciting Incident Ten percent of the way into a screenplay, the hero must be presented with an opportunity, or some sort of incident which turns his/her world upside down and creates a new, visible desire, which will start the character on his/her journey.

PLOT POINT #2 Change of Plans or Reversal Something must happen to the hero one-fourth of the way through the screenplay that will transform the original desire into a specific, visible goal with a clearly defined end point. This is the scene where the story concept is defined, and the hero's outer motivation is revealed.

PLOT POINT #3 The Point of No Return At the exact midpoint of the screenplay, the hero must fully commit to their goal. Up to this point, he/she had the option of turning back, giving up on the plan, and returning to the life he/she was living at the beginning of the film. But now the hero must cross the point of no return.

PLOT POINT #4 Hits Bottom Around page 90 of your screenplay, something must happen to the hero that makes it seem to the audience that all is lost. These disastrous events leave the hero with only one option: he/she must make one, last, all-or-nothing, do-or-die effort as he/she enters

PLOT POINT #5 The Climax Several things must occur at the climax of the film: the hero must face the biggest obstacle of the entire story; he/she must determine his/her own fate; and the outer motivation must be resolved once and for all.

Learning Goal

Students should demonstrate competency in abstracting the structure for a narrative screenplay by breaking down the three-act structure and five plot points to successful and disappointing motion pictures.

Case description: The Feature

The structure of the three-act narrative creates the feature-length script's spine, which the student can then flesh out in any way he or she is so inclined to do so. These rules of structure also provide students an opportunity to use abstraction when thinking of major turning points or plot points within the screenplay. Knowing that the inciting incident or turning point #1 must occur at page 15 to 18 (10% into the script) means a student has a very limited amount of time to set up the three basic components of the script: character, desire and conflict.

Understanding the stages and turning points provides screenwriting students with a powerful tool for developing and writing their feature screenplay. Having these basic sets of rules in place, allows a student greater freedom to explore more personal and creative elements of their screenplay.

Discussion questions

Q1. Is there an inciting incident?

Q2. Is the story concept defined at the one-quarter mark?

Q3. Have they fully introduced the hero before presenting him/her with an opportunity around page 10?

Q4. Is there a turning point at the end of Act I?

Q5. Does the hero hit bottom" at the end of Act II?

Q6. Does she/he suffer a major setback 75% of the way into the script?

Assessment

Students will compar3 Flops and Blockbusters to Reveal Structure:

- Students will take the screenplays to three successful motion pictures (100 million dollars or more at the box office) and three box office flops all of the same genres and break them down using the THREE ACT STRUCTURE as well as the FIVE PLOT POINTS.
- Students will then compare the six scripts and plot them out using the structure formula outlined in class. They will then draw some conclusions on the theme, metaphor, premise, moral and hero's journey from their script comparisons.

- Through this breakdown and comparison students will discover which screenplays follow the three-act structure and five plot points and which ones do not. In comparing the different plot points, paying close attention to the inciting incident, for comedies and dramas students will use Abstraction in determining why some motion pictures are successful (box office hits) while others are not despite being from similar genres.
- This assessment is intended to allow students to show that they have a clear understanding of the stages, turning points and overall structure of narrative cinema.

4.6 Evaluation

4.6.1 Constructing a histogram

The construction of a histogram is an important first step in any statistical analysis. The visual appearance of the histogram can be used to determine if there are outliers and what the appropriate statistics are for reporting center and spread. For more complex statistical analysis, the shape of the histogram can be used to determine what analyses are appropriate and which are not. However, the creation of a histogram is not a simple process; it is iterative and benefits greatly from automation.

Case description

There are computer programs that completely automate the construction of a histogram. However, since the histogram is a visual statistic, it is best constructed using human input; in particular the choice of the number of bins is critical in generating a histogram that best provides visual information about the distribution of the underlying data. A histogram is critical in determining whether the mean and standard deviation or the mode and quartiles are better for summarizing the center and spread of the data.

Learning goal

Students can use a statistics package to generate multiple histograms from a single data set and choose the most meaningful visualization of the symmetry or skew of the data.

Case discussion

Students are shown the algorithm for constructing a histogram:

1. Determine the minimum and maximum of the data
2. Create a number of bins for aggregating data
3. Create a distribution table for the data based on the bins
4. Graph the distribution table

Students will consider the following questions:

Q1. If the resulting histogram is too spiky, how do we improve it?

Q2. If the resulting histogram is too shapeless, what do we do?

Assessment

Students are given a data set and are asked to:

- generate histogram bins that include all the data and are of equal width.
- generate the best" number of bins, i.e the one with the number of bins that best visualizes the data (i.e. patterns in the data set are made explicit and reasonable hypotheses can be made about the data set).

As this activity is done for many problem sets, the students have many opportunities to mean the objectives, and also to see different visualizations for different data sets (e.g. small data sets versus large ones; symmetric distributions versus skewed ones.)

4.6.2 The Substitution Cipher

In a substitution cipher each letter is replaced by some other (unique) letter of the alphabet. As such, a substitution cipher does not change the frequency distribution of the letters, it just disguises it by renaming the letters. This observation, with a little bit of statistical mathematics, can be used to automate the exhaustive breaking of the shift cipher, that is, the process can be done by a com- puter (or simulated by hand) without any human involvement. For the students to understand and recreate this process, they need to un- derstand one of the most basic principles in computational thinking: when trying to understand a process (be it a shift cipher, a sorting algorithm, or some other more complex procedure) it is dicult to argue about what changes; indeed, it is necessary to understand what a particular process does not change. In the case of the shift cipher, one such property is the frequency prole.

Learning Goal

Use statistical analysis and trial and error to break a monoalphabetic substitution cipher.

Case description

Some (overly optimistic) texts claim that substitution ciphers are easily solved by frequency analysis; when trying this by hand, the students will quickly nd out that this is at best a half- truth. Students realize that an approach can be heuristic, i.e. only lead to partial or suboptimal solutions which might not even be fully correct. The question then is how to improve results.

A form of exhaustive search, trial and error can be used but it quickly becomes painful for substitution ciphers; students realize that additional understanding of the syntactic nature

of English text is required before this approach is meaningful. Indeed, a combination of frequency cliques, bigrams, and other information about English words leads to a collection of rules which together with frequency analysis can be used to solve substitution ciphers in which word separations have been maintained.

Discusion questions

Q1. Perform some substitution cipher encryptions and decryptions.

Q2. Can we try all possible keys to break the cipher? How many keys are there?

Q3. How long would it take to try all keys by hand? On a computer? How would we recognize English plaintext?

Q4. Does the frequency approach used for the shift cipher work?

Q5. Does the frequency approach lead to a mechanizable solution of sub- stitution ciphers? How could it be extended to work?

Q6. What information about English words would be helpful in solving a substitution cipher by hand?

Learning Goal

Use statistical analysis and trial and error to break a monoalphabetic substitution cipher.

Assessment

The students will get a (individual) monoalphabetic substitution cipher to break.

4.7 Recollection

4.7.1 Tree Structure

Trees are used as a structure for hierarchical data organization. The big idea is: organize large amounts of items into groups and continue partitioning each group into subgroups, recursively. A tree data structure allows efficient insertion and search. Understanding and using a tree data structure is a computational thinking instance that belongs in the Recollection category.

Learning Goal

Students recognize the application of a tree structure to organize data; they are able to take advantage of the structure to efficiently locate/reference data in the tree.

Case 1 description: File System Structure/URLs

URLs consist, basically, of a hostname and a pathname. The pathname is just the file system (relative) pathname of the file with the content identified by the URL. The file system pathname uses the hierarchical terminology to reference a file on a computer.

Discussion Questions

Q1. Do you know anyone with 50 files on their desktop? How easy is it to find a file with this organization?

Q2. How are folders useful for organizing and finding files?

Q3. What's a good number of folders? If we have a large number of files, then we might end up with too many folders How do we continue organizing the folders?

Q4. If I have 1000 files and 5 folders at each level. How many levels do I need?

Q5. How do we search for a file in a file system?

Q6. How can we represent a file location as text? How do URLs do this?

Case 2 description: Site Maps

A web site is often organized as a tree. Web sites exist to offer content that can be partitioned into groups and, recursively, into subgroups. This organization is to help users find the content they are looking for.

Discussion Questions

Q1. How does a tree structure aid users in finding needed content on a web site?

Q2. How are web site categories organized within a tree?

Q3. How is navigating through a web site like searching through a tree?

Q4. Examine the web site for your college or university. Discuss how pages and links match a link structure? Are there links that do not match a tree structure?

Case 3 description: Domain Name Servers

The Domain Name System is a computer application that connects many servers around the globe to provide the following service: translating a hostname to an IP address. (The destination host IP addresses is required to send information to it.) Domain name system is hierarchically organized to allow for efficient search of the DNS server that can provide a current IP number for a given a hostname.

Discussion Questions

Q1. Why have IP addresses? Why have hostnames?

Q2. How does the domain name system find an IP number?

Q3. How does the hierarchical structure of the Domain Name System reduce the number of DNS queries for finding an IP number?

Q4. How does this hierarchical structure allow for distributed managing of the hostname/IP address pairs?

Assessment

Students will be given a visual representation of a simple file system tree and will answer several questions:

- Give the absolute pathname of file X
- Give the pathname of file Z relative to ancestor folder Y
- If moving folder D into folder C, list the files that will move as well.

4.7.2 Relative vs. Absolute References

A setting or a location can be specified relative to a current state or in terms of an absolute specification. For example, a location of a building can be specified relative to a current location (e.g. 3 blocks north of here) or by an absolute location (e.g. 25 S. Elm St.,). The concept of relative and absolute references are commonly used in IT applications and include file name references, cell references in a spreadsheet and configuration settings, such as those used in page formatting. Being able to understand and use absolute and relative references is an example of computational thinking.

Learning goal

Students can find the relative reference given an absolute reference and state; vice versa, they can reconstruct an absolute reference based on a state and a relative reference.

Case description: Referencing URL/File location

Items in a tree structure can be referenced through an absolute pathname, i.e. the sequence of groups containing the item, from largest to smallest. In many cases, these structures can be iterated through and a relative pathname to an item is defined from the current group (containing the item).

Discussion Questions

Q1. Given a web page and a hyperlink to another web page at the same web site, describe absolute and relative references

Q2. Why use relative references? How is a relative hyperlink reference useful if you transfer your files to a web server? What if you had used an absolute reference and then moved your files?

Q3. Consider a file finder application: which user actions would make use of absolute references?

Q4. Consider a file finder application: Which are relative to current location?

Case description: Style Property Settings

Many style properties can be specified by an absolute setting or relative to the current setting. For example, a margin setting may be given in terms of an absolute amount (1 inch margins) or relative to the current setting (increase current margin setting by 5em).

Discussion Questions

Q1. When is it useful to specify a relative setting?

Q2. When is it more useful to specify an absolute setting?

Assessment

Students will be given a visual representation of a simple file system tree and will answer several questions:

- Give the absolute pathname of file X
- Give the pathname of file Z relative to ancestor folder Y
- If moving folder D into folder C, list the files that will move as well.

4.7.3 Caching

Caching replicates data originating from a remote resource locally, so it is close to the consumer of the data. The goal of caching is to improve data access latencies. What data is cached will depend on assumptions about future data accesses which are usually not deterministic. Understanding the usefulness of caching and the ability to reason about what data to cache is example of computational thinking that belongs in the Recollection category.

Learning Goal

Students understand the purpose of caching; they understand the issue of limited cache capacity and usability, and that the choice of what to cache is based on assumptions about future data access; they understand that cached data may be stale and will need to be re-synced with the original data.

Case 1 description: Web Browser Caching

Web browsers cache recently visited web pages to improve response times. The assumption is that future web page request will match recent web pages requests.

Discussion Questions

Q1. What web sites do you get back to quite often?

Q2. Do you mind waiting for a web page to load?

Q3. How can you minimize waiting for web pages you visit often?

Q4. Your solution raises issues of cache consistency. How do we deal with it?

Case 2 description: DNS caching

DNS servers cache recently obtained ¡hostname, IP address¿ pairs to improve response times. The assumption is that future DNS requests will match recent ones.

Discussion Questions

Q1. How many DNS servers need to be queried in order to obtain an IP address for a web host in Sweden?

Q2. What penalty do we pay when we move from web page to web page on this Swedish host?

Q3. How do we using the caching concept to shorten the latency of web page downloads from this web site?

Assessment

Student will be given the example of phone number caching on cell phones and will be asked the following questions:

- Why do cell phones cache recently dialed or received numbers?
- Why not cache every number ever seen? What limits the cache capacity? Is it just memory capacity or other factors?
- Give a reasonable argument for what numbers should be cached. Most recent? Dialed most often? Both?
- Is there an issue of cached data staleness in this example?

References

[1] C. Anderson, L. Berkowitz, E. Donnerstein, Huesmann, L. Rowell, J. Johnson, D. Linz, N. Malamut, and E. Wartella. The Influence of Media Violence on Youth, *Psychological Science in the Public Interest*, 4, pp. 18-110, 2003.

[2] C. Chuan. Automatic Style-Specific Accompaniment, University of Southern California, http://www-scf.usc.edu/˜ise575/c/projects/chuan/, 2007.

[3] G. Dahl and S. DellaViqna. Does Movie Violence Increase Violent Crime, *National Bureau of Economic Research*, NBER Working Paper 13718, http://www.nber.org/papers/w13718, 2008.

[4] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic Generation of Social Tags for Music Recommendation, http://www.iro.umontreal.ca/˜eckdoug//papers/2007_nips.pdf, 2007.

[5] D. Engelbart. Augmenting Human Intellect: A Conceptual Framework, Summary Report AFOSR-3233, Stanford Research Institute, Menlo Park, CA, 1962.

[6] H. Gardner. *The Minds New Science: A History of the Cognitive Science Revolution*, Basic Books, 1987.

[7] The Human Genome Project, http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml, accessed December 2008.

[8] R. Lang. http://www.langorigami.com/, accessed December 2008.

[9] Robert J. Lang. *Origami Design Secrets*, 2003.

[10] P. Phillips. http://www.cs.cmu.edu/˜CompThink/resources/ct_pat_phillips.ppt, accessed December 2008.

[11] J. Wing. Computational Thinking, *Communications of the ACM*, 49:3, pp. 33-35, March 2006.