

CS1316: Digital Representations of the World

Mark Guzdial

October 1, 2003

1 Introduction: Overview of the Course

Pictures and sounds can represent how the world looks and sounds, but for many applications, we need to represent more than the surface. Movie makers, videogame designers, database administrators, and spreadsheet designers create *models* to represent the structure and behavior of the world. By creating working models, we can represent how the world works, or even how alternative or virtual worlds work. Rather than defining all the surface details (e.g., the color of every pixel in every frame of a movie), we can allow the details to be computed by the model as a *simulation* of the world. You don't have to draw every scene of the stampeding animals in *The Lion King* or the waving villagers in *Hunchback of Notre Dame* if you can model the characters, start the simulation, and simply capture (film) the model running on the computer.

To represent structure means to represent the things in the world, the parts of those things, the relationship between those parts such that the things work, and the relationship between the various things in the world so that the world works. Representing structure requires going beyond surface representation to describe the basic principles of how we describe the physical world, e.g., some things are *within* others, some things are *next to* others.

Some relationships are more dynamic. Things in the world respond to one another. To represent those relationships, we have to go beyond physical structure to talk about how objects respond to one another—that is, define the *behavior* of the objects.

This class is an introduction to how we can represent digitally some of the deeper structure and behaviors in the world. It's an introduction to data structures and algorithms, but also simulation and interface, to describe what we do with our digital representations and how we manipulate them usefully.

2 Topics of the Course

- **Representing the structure of a scene.** We can describe the linear relationships between objects (i.e., “the dog is to the right of the boy, and the boy is to the right of the tree”) using a *list*. If we have a list of graphical operations that draw the objects, we can change their order and appearance, by deleting and inserting elements in the list.

We can describe more complex scenes in more dimensions by using *hierarchical lists* or *trees*. Trees have more than one dimension so we have options how we *traverse* the

trees. Different traversals of lists of graphical operations lead to different *renderings* of the scenes. Examples of applications that we might use here include Geographical Information Systems (GIS).

Students will be learning basic list and tree operations as ways of describing relationships between graphical elements. Operations such as insert, deletion, and even tree balancing will have a direct and concrete correspondence to items in the rendered scene.

- **Representing the structure of music.** We can similarly use lists and trees to describe the elements of music. We can easily represent chorus and verse structure using lists, and we can change these lists to change the resultant music. By moving to a multi-dimensional representation, we can make our representation of music more complex, including multiple tracks and motifs. Different traversals through the music structure leads to different music.

The computer science content of the music section will not differ dramatically from the graphical scene context, but it establishes the power of the abstract concepts to use them in more than context. And like the graphical examples, manipulation of the music lists and trees will result in concrete changes.

- **Describing behavior.** Things in the world have more than just structure—they have behaviors. There are many ways of describing behavior. Functions are just one way. A common way of describing behavior is as sets of *rules*. Using rules, we can describe *Web search agents* (e.g., shopping agents—computer *robots* that shop for you on the Web), video game behaviors, and even the responses of interactive textual agents such as the virtual psychiatrist *Eliza*.

We will discuss some simple automata concepts so that we can define regular expression matching, and build rules around those for creating interactive textual agents and Web search agents. We will also have the opportunity to discuss automata more broadly and contrast what can be written with rules as compared with what can be written as functions.

- **Designing objects.** Once we have both structure and behavior, simple models of top-down and bottom-up design become inadequate to explain how to describe systems of multiple objects all with their own structures and behaviors. There are mechanisms and notations for analyzing and designing systems of objects.

While we won't be introducing the complete Rational design process and UML, it will be useful to introduce concepts list “user stories” (or Use Cases, in UML terms) and CRC cards as ways of analyzing object systems and designing solutions.

- **Simulating the World.** A collection of objects representing things in the world with structures and behaviors defines a *model* of the world. When we execute this model, allowing the objects to react to one another according to their behaviors and to use their structures to support their behaviors, we have a *simulation* of the world.

Simulations are one of the earliest and still most powerful uses of computers. We can define simulations of a physical system (like the motion of a projectile in space)

and with enough objects, even describe complex systems whose behavior is hard to represent and understand without a computer, such as cars in a traffic jam and ants in an ant hill. A video game is essentially a simulation of a virtual world.

Simulations don't *require* object systems. There are many powerful simulations that represent the world with numbers, such as population simulations. We might even simply use Excel to create such simulations.

An extended case study to be used in the course is the development and use of a package to create discrete event simulations. Discrete event simulations don't execute in real-time. Instead, *events* are scheduled to occur in *simulation time*, which means that events are placed in an *event queue* which is *sorted* so that events are executed in the right order. This presents an opportunity to discuss sorting and complexity of sorting algorithms.

- **Defining a Virtual World.** The modern computer itself appears to the user as a virtual world (e.g., one that is not meant to be a representation of the physical world and only exists within the computer) with structures (buttons, text areas, list boxes, scrollbars) and behaviors associated with these structures (e.g., click the “Insert” button to add a new item to a list). The structures that we saw were used to represent structure in scenes and music are also used to represent the relationships between things in the virtual world. The course covers creation of simple windows-based interfaces and how concepts introduced earlier in the course appear in the user interface context, like events, event loops, and traversing a scene tree.
- **Interfacing Worlds.** Finally, we can use the virtual world (user interface components) to allow the user to control the representations that we've made of the real world, so that users can define, simulate, and explore our models through text areas, buttons, and windows.