Beauty and Joy of Computing: Overview of Curriculum

Overarching Goals

The goal of AP Computer Science Principles is to provide a broad, inspiring overview of computer science that is appropriate for all students who have completed Algebra 1.

The Beauty and Joy of Computing (BJC) is an adaptation of the University of California, Berkeley computer science breadth course designed to meet the requirements of the AP CS Principles Curriculum Framework and to be broadly accessible to a diverse population of high school students.

In this course, students will

- create programming projects using the Snap*!* language,
- learn some of the most *powerful ideas* of computer science,
- be *creative*, and
- investigate the social implications of computing.

The BJC curriculum is available at http://bjc.edc.org and the Teacher Guide is available at http://bjc.edc.org/bjc-r/teacher.

Description of Curriculum

BJC covers the entire CS Principles Framework and addresses the seven Big Ideas in the framework with a primary emphasis on programming (Big Idea 5) and, closely linked with programming, on abstraction (Big Idea 2). As much as possible, BJC uses programming as the vehicle to tell other parts of the story; for example by presenting data (Big Idea 3) not through commercial database software but by writing programming projects that manipulate data sets as lists.

The secondary emphasis of BJC is on the social implications of computers (Big Idea 7, Global Impact). Social topics are included in every unit, not just one. Students are encouraged to think critically about each application of technology. The Big Ideas of creativity (Big Idea 1) and algorithms (Big Idea 4) are addressed throughout the units, and there is particular attention to the Internet (Big Idea 6) in Unit 4.

Snap*!*, the programming language used in BJC, was developed specifically for this curriculum. Its visual, drag-and-drop design is based on that of Scratch, so that it is accessible to a wide audience and not intimidating, but the language, itself, is extended with the abstraction mechanisms needed for serious computer science: first class procedures for control abstraction and first class lists for data abstraction. These capabilities are embodied in carefully chosen visual metaphors so that ideas traditionally considered difficult can be understood and enjoyed by beginners.

The course is divided into seven units that cover:

1. sequential programming and loops
2. conditionals and functions

3. lists
4. the Internet and introduction to data
5. algorithms and data
6. recursive commands
7. recursive functions

The actual AP exam comes some time in Unit 6; the last two units primarily cover topics beyond the CS Principles Framework. Note that the seven units do *not* correspond one for one with the seven Big Ideas of the Framework, although Units 4 and 5 focus on Big Ideas 3, 4, and 6. However, all seven Big Ideas of the Framework are addressed in BJC.

Recursion and functional programming are two programming techniques that go beyond the Framework requirements, but are at the heart of what makes BJC unique. Unit 6 is about recursive commands, mainly fractals; Unit 7 is about recursive functions, combining the ideas of recursion, from Unit 6, and functional programming, introduced in Unit 3 with the higher order functions on lists. One highlight of the course is the implementation by students of three key list operations: Map, Keep, and Combine.

These ideas are important to include because they help students come to see computer programs themselves—not just the effects produced by the programs—as things of beauty. A key moment in developing that sense comes when students understand how a short recursive procedure can generate a deeply complex computational process.

Computational Thinking Practices

The BJC course includes two different kinds of online lab pages for students: Programming Labs, in which students create and analyze programs and learn about the technical aspects of computing, and Social Implications Labs, in which students consider, discuss, and write about the human aspects of computing. Below we describe how each of the computational thinking practices is addressed in the BJC curriculum.

P1: Connecting Computing—Primarily in the Social Implications Labs throughout the course, but also in the Programming Labs of Unit 4: The Internet and Global Impact, students consider the social impacts of computing, including connections between technological advances and impacts on society.

P2: Creating Computational Artifacts—Throughout the BJC course, students create many computer programs, the very foundation of computational artifacts. These include specific assigned tasks as well as major projects entirely invented and developed by students. Other kinds of artifacts are developed as required by the AP CS Principles performance tasks.

P3: Abstracting—Abstraction, the central idea of computer science and of this curriculum, is also highlighted throughout the year. BJC includes both control abstraction (through writing procedures, generalizing patterns by adding inputs, and using and building higher order functions) and data abstraction (writing constructors, selectors, and mutators for abstract data types).

P4: Analyzing Problems and Artifacts—Students analyze and debug programs provided in the curriculum and also critique their own work. They discuss *how* a

particular program works or *why* a program does not work and what it would take to fix it.

P5: Communicating—Students are regularly asked to discuss questions and topics with their pair programming partner or (for more complex topics) another pair. In addition, whole class discussion prompts are included in the Teacher Guide for every unit. The first five units of BJC include topics in the social implications of computing, which are treated through whole class and small group discussions and writing prompts.

P6: Collaborating—Throughout the curriculum, students use pair programming. They take turns at the keyboard and resolve programming bugs and discuss issues together. Following the AP Create task requirements, large programming projects are designed and partly implemented in pairs.

Big Ideas

Below we describe how each of the Big Ideas in the AP CS Principles Framework is addressed in the BJC curriculum.

Big Idea 1: Creativity—BJC's primary attention under this heading is creativity as expressed in computer programming. In addition to the Create performance task, throughout the curriculum there are labs in which students are given the core of a program and then encouraged to embellish and vary it.

Big Idea 2: Abstraction—BJC considers abstraction to be the central idea of computer science, and it is the central idea of this curriculum. Early in the year, students write procedures to draw a square, draw an equilateral triangle, draw an equilateral hexagon, and then generalize the pattern into a general polygon procedure by adding an input for the number of sides. Several more examples of this kind appear in the first two units. Unit 3 introduces data abstraction, using lists to represent various abstract data types by writing constructor and selector functions. (Later examples introduce mutators too.) Unit 4 explains the Internet as a multi-layered abstraction over local network interface hardware. And later, students generalize patterns of functions over lists by writing higher order functions such as map and filter.

Big Idea 3: Data and Information—Unit 3 introduces lists, the primitive data aggregation mechanism in Snap*!*. This unit models operations on data using small data sets built into project frameworks. In Units 4 and 5, the same techniques are used on larger data sets found on the Internet. Snap*!* allows CSV data to be imported into lists of lists. The primitive list operations make it easy to carry out selections, slices, or joins. Unit 4 also teaches how to scrape HTML pages programmatically; the main example is for students to find the local temperature and chance of rain by

localizing the computer's IP address and using the result to construct a query to a weather-reporting site.

Big Idea 4: Algorithms—Students learn to develop and use algorithms to solve problems right at the outset in BJC, and they refine their understanding throughout the year. Unit 5 is where the ideas about asymptotic analysis of algorithms and computability are taught.

Big Idea 5: Programming—Programming is at the heart of BJC. Most of the learning objectives in this category are addressed from the beginning of the year.

Big Idea 6: The Internet—Unit 4 is about the Internet. It explains the specific details under Big Idea 6 and also uses data read from the Internet in Snap*!* to introduce Big Idea 3.

Big Idea 7: Global Impact—The Social Implications labs connect the ideas of each unit to the human experience. Students discuss innovations, benefits and harmful effects, and cultural contexts of computing.

Required Computational Tools

University of California, Berkeley. Snap*!* 4.0. http://snap.berkeley.edu/run.

Snap*!* is a visual, drag-and-drop programming language. It is a significantly extended reimplementation of Scratch (a project of the Lifelong Kindergarten Group at the MIT Media Lab) that features first class lists, first class procedures, and continuations, and allows users to create recursive blocks. These added capabilities make it suitable for a serious introduction to computer science for high school or college students.

Snap*!* is implemented in JavaScript, so it will run on any platform with a browser, with no explicit installation required. (Standalone versions for most platforms are coming soon.) Because the language is developed by the some of the same team as the BJC curriculum, teacher-reported problems are solved quickly, and curriculum-related enhancements get priority development.

Software and Hardware Requirements

Computers (preferably) or tablets with a browser new enough to support HTML5 Canvas. (The BJC and Snap*!* development teams generally use Chrome or Firefox but also test on Safari, Opera, IE, and Edge.)

School Network Requirements

Schools must whitelist berkeley.edu, edc.org, and miosoft.com (the Snap*!* cloud storage provider) to access all BJC course materials.

Other Resources

The curriculum itself is also accessed through a browser. The BJC curriculum is available at http://bjc.edc.org and the Teacher Guide is at http://bjc.edc.org/bjc-r/teacher. All materials, including the textbook *Blown to Bits,* are available free online (Creative Commons licensed).