Keith Alcock
EDU 285

Prepare an authentic classroom assessment at the grade level and subject area you intend to teach. Include:

1) grade level and subject area,
2) activity or learning task description,
3) materials needed,
4) goal of the assessment,
5) method of assessment,
6) standards and criteria for evaluation,
7) use of assessment results, and
8) feedback to students [and parents].

Goal of assessment: How will this assessment contribute to my students' ability to learn or my ability to teach them?
Method of assessment: How will I gather this information?
Evaluation: What standards and criteria will I use to interpret the results of my assessment or observation?
Use: How will I use the results?
Feedback: What feedback will I give students? Parents?

The table below outlines a year-long authentic assessment consisting of multiple parts which correspond roughly to the units in a sample computer programming textbook. The assessment for one particular unit, classes, is presented in some detail thereafter. Included are a worksheet and two rubrics for the unit.

Grade level and subject area
Second and third year C++ and Java computer programming students in grades 10 through 12
Activity or learning task description
Students work on a long-term, realistic programming project in parallel with their short-term, "toy" lab assignments. Project topics are rotated yearly so that students studying both languages have different topics the second year. The project topic proposed for my first year is the student's choice of teacher-approved card game. (The approval requirement is to disallow gambling, drinking, etc. games and to monitor for games that are obviously too simple or complex.) Approved games include Go Fish, UNO, War, and forms of Solitaire, but for most students probably not complicated Bridge or 500. Worksheets for each unit explain how the material might be applied to a generic project and the teacher models this with the teacher's card game. Students then relate the material to their projects and apply it. Example units and connections are as follows: • Variables: score of the game, count of cards in each player's hand • Constants: possible face values on a card, the number of players • Arithmetic expressions: add the value of the first card plus twice the value of the next • Logical expressions: if the face value of one card is greater, then it takes the trick, or else in the case of a tie, the result is determined by the next cards • Iterative statements: take each card in the deck and distribute it to alternating players • Arrays: the deck of cards, each player's hand • The switch statement: user selection of menu items • Strings: names of players, prompts, error messages • Classes: card, suit, player, game • Searching: find the card with highest face value in a player's hand, look for a match in the opponent's hand • Sorting: display players' cards in a logical order, show scores highest to lowest These particular units are based on the table of contents for the book C++ for You++. The corresponding Java book, Java Methods A & AB, is organized similarly.
Materials needed
• Computers with installed Java (e.g., JCreator, Eclipse, JBuilder, NetBeans) or C++ (e.g., CodeWarrior, Visual Studio, C++ Builder, XCode) development systems. This is standard equipment for the course. • Internet access and list of websites with example card games that are appropriate for students and include rules. Example sites are The House of Cards Card Games Thanos Card Games Card Game Rules • Card game reserved for the teacher which can be used to model the assessment activities. Magic square tic-tac-toe is my pick. Two players take turns choosing cards from a collection of nine having values one through nine. The first player to collect

three cards whose values add to fifteen wins. From this simple description I can show how arithmetic and logical expressions, arrays, and iteration relate to the project.

- A version control system (e.g., Subversion) is used to manage multiple versions of documents and code. Since it is seldom a component of a standard high school computer lab, it will need to be installed on an appropriate server. Student computers will need clients provided as extensions to the development system or as separate applications. All are freely available. (Version control is optional, but important for project authenticity and for project management and grading.)

Goal[s] of the assessment: How will this assessment contribute to my students' ability to learn or my ability to teach them?

The assessment has manifold goals including the suggested increase in students' ability to learn and my ability to teach them. Firstly, however, it is intended to increase what students do learn, not just what they are able to learn.

Actual learning

- Students practice software engineering activities such as software design and project management in addition to learning computer programming concepts like algorithms and data structures.
- As students apply the abstract concepts they have learned to their concrete project, concepts become more familiar from the review and more and more tangible from the application.
- The project may redirect some of students' gaming enthusiasm toward designing, evaluating, and programming a game rather than just playing one. They may spend more time on their card game and less on gaming or simply game with more attention to computer science issues.
- Textbooks usually emphasize the what and how of a technique, but students need to face questions like why, when, and where.
- Since lab assignments are the same for each student and they often collaborate (and indeed may be required to), the project offers a chance for independent work and requires students to prove themselves against the material.
- If students complete their lab work, they have an outlet for as much time and energy as they would like to devote to their programming project.

Ability to learn

- The project affords students influence over their assigned work and allows them to pursue a personal interest. The topic restriction provides a suitably orienting framework.
- As we discuss new programming topics, students can consider how the content relates in the context of their familiar project. The project provides scaffolding and scripting opportunities.
- Lab exercises are typically mundane and uninspiring, but an entertaining project that can be presented to parents and demoed to other students may provide substantial motivation.
- Throw away assignments discourage extra effort. A project is worth preserving. Students take their work home to show family and present code samples to potential employers.

- We snapshot projects over time so that students can better observe and evaluate how they are learning and improve on the process.

Ability to teach

- The project better allows me to show individual attention to students than the standard assignments do.
- I can employ more practical programming experience and better hold students' attentions with real world examples, rules of thumb, and "stories from the trenches."

Method of assessment: How will I gather this information?

- At the beginning of the year, students first write out responses to prompts rather than record their ideas directly in code, since their initial unfamiliarity with the programming language may limit their expressiveness. The priority is on conceptual understanding rather than syntactic detail. Written answers serve as rough design specifications for students' programs and are also gathered for evaluation. They are judged according to a rubric and feedback is given to promote proper coding.
- As students' planning skills and programming language fluency improve, they are allowed to code first and then document the result. This sequence can thwart "analysis paralysis." Documentation explains in retrospect what was done to the program rather than prospectively what is to be done. The documentation is similarly collected for evaluation.
- The code itself constitutes the main deliverable. Students check their code into the version control system and label it for easy identification. The teacher reads through changes incorporated since the last checkpoint for evaluation. If necessary, verbal explanations are elicited from students or demos are requested. Progress is graded. Students are always allowed to modify their programs in places where progress has been inadequate.
- Twice each quarter students evaluate their performance with a self assessment, which the teacher evaluates.
- Students have the opportunity either to demonstrate their programs to classmates at the end of the year or to present a short talk during the year on a specific technique they exploited which might be of interest to other students.
- At the end of the year, students participate individually in targeted code walkthroughs designed to demonstrate understanding of their work.

Standards and criteria for evaluation: What standards and criteria will I use to interpret the results of my assessment or observation?

- Rubrics are customized for each unit's version of the assessment. Several Computer Science rubrics and advice that primarily address coding can be found online:
  - AP Computer Science: Rubric for Grading Programs
  - Computer Programming Grading Rubric
  - Criteria
  - Computer Science Rubrics
  - Computer Science Rubric
  - AP Computer Science – Program Implementation
- The section above has identified rubrics for written work (either plans or documentation), program code, student self-assessments, a demonstration or short talk, and a code walkthrough.

Use of assessment results: How will I use the results?
• Assessment results, particularly the per unit assessments, are primarily formative. Since this is a long term project, it is especially important that gaps in learning be identified as soon as possible so that the remainder of the project (and future projects) is not endangered. The results will be used for remediation. • As the project progresses, it accumulates knowledge that the students have acquired over the year. The end of year activities, the demonstration and walk through, constitute a summative evaluation of the project and will be used largely for grading. • If multiple students require remediation, then I can diagnose a problem with my teaching. The results therefore represent a diagnostic evaluation of the teacher.
Feedback to students [and parents]: What feedback will I give students? Parents?
• Students receive not only numeric, rubric-based feedback for each unit, but also detailed comments to guide them in revising their programs. • Parents are invited to see their child's work in action on parent night, statically during parent-teacher conferences, or by appointment. These opportunities complement the standard feedback mechanisms of grades and progress reports. • I am investigating how we can most easily convert console based games into HTML forms or Java applets so that parents and peers can play these games from student web pages. We may include a feedback form to collect data from third parties. Constructive criticism will be forwarded to students.

Grade level and subject area
Second year C++ computer programming students in grades 10 through 12
Activity or learning task description
Students identify classes needed for the implementation of their card games. Students use the previously collected game rules and their requirements analyses to identify noteworthy nouns indicating potential classes. They will have performed much the same feat in labs, but not on as large a scale, as open ended an assignment, or as independently. To find member functions of these classes, students next search for verbs associated with one of more of the nouns. They decide which of the nouns, if any, takes responsibility for the action. Before coding the classes and member functions, students document their proposed design. After receiving feedback from the teacher and taking any corrective action, students program their classes and functions. For C++ students this includes writing the class declarations in the class.h files and method stubs in the class.cpp files. Java students would produce class.java files with method stubs and unit test cases that exercise the methods.
Materials needed
Except for the worksheet and rubrics particular to this unit (see below), no materials are needed beyond those required for the project or course or that weren't completed for a previous unit.
Goal[s] of the assessment: How will this assessment contribute to my students' ability to learn or my ability to teach them?
The primary goals for this unit's assessment are to ensure that students are capable of rudimentary object oriented design and of encoding the design in their programming language. Other goals are consistent with the project's goals.
Method of assessment: How will I gather this information?
Students hand their written work to me on worksheets I provide. Code is checked into the version control system, which alerts me that files have been updated. When the student applies a particular label, e.g., beta version, I know that an evaluation is needed. This particular worksheet instructs students to highlight text on printouts of their game rules, so the highlighted rules must be submitted as well.
Standards and criteria for evaluation: What standards and criteria will I use to interpret the results of my assessment or observation?
For this unit, the general project rubrics have been customized. Two appear below for the written work and code.
Use of assessment results: How will I use the results?
Use aligns with that of the project's general assessment results.
Feedback to students [and parents]: What feedback will I give students? Parents?
Detailed comments of the written work include identification of hidden nouns and verbs that students may have missed and objects that were misidentified as classes. If any inheritance (single or multiple in the case of C++, single plus interfaces in Java) is foreseen, students are advised to pay particularly close attention during the upcoming unit. Source code is evaluated not just in generic terms, but on the fidelity of translation from the written work. Related feedback is provided.

Computer Programming in C++
Classes Worksheet

1. As we have practiced, identify likely classes involved in your card game. Highlight them on a copy of the game rules and record them here along with a brief description in your own words of what they represent. Also highlight any objects you find, create an appropriate class, document it, and list its instances. Hand in your highlighted game rules with this worksheet.

2. Identify likely methods involved in your card game. Highlight them in a different color on a copy of the game rules and record them here along with the classes of objects they might take as arguments or might return. Include a brief explanation in your own words of what each method does. Hand in your highlighted game rules with this worksheet.

3. Group methods by the class they should belong to and then by their accessibility. Present your results in outline form ready for coding.

4a. Submit your written work for approval. For more details see the rubric for written work on classes.

4b. After your written work has been approved, make any of the requested changes before continuing.

5. Write the C++ code corresponding to the classes and methods you have identified. In the class.h files create the class declarations. In the corresponding class.cpp files create stub implementations of all methods so that the code compiles and links cleanly.

6. When you are satisfied with your code, check it into the version control system. Label the release "Unit Classes, Version Beta 1" when you are ready for the teacher to check it. For more details see the rubric for programming work on classes.

Computer Programming in C++
Rubric for Written Work on Classes

Category	Excellent (3)	Good (2)	Satisfactory (1)	Unsatisfactory (0)	Score
Classes	Nearly all classes have been identified with few extraneous cases and little class/object confusion.	Most classes have been identified with some false hits or class/object mistakes.	Only the most important classes have been identified. Classes and objects are confused.	Few or no classes or objects have been identified.	
Methods	Nearly all methods have been identified and provided with parameters of the correct type.	Most methods have been identified and provided with reasonable parameters.	Only the most important methods have been identified and may lack parameters.	Few or no methods have been identified.	
Mapping	All methods have been mapped to the correct classes	The mapping is slightly off. Some methods may be unassigned.	The majority of methods are mapped reasonably.	Methods appear to be assigned randomly or aren't mapped at all.	
Completeness	All nouns and verbs corresponding to classes and methods have been highlighted. No additional words were.	Most words were correctly highlighted with few mistakes.	Many words were highlighted, but notable problems exist.	No words were highlighted or those that were don't match the written list.	
Documentation	Comments on all classes and methods make program functionality clear.	Comments aid substantially to an understanding of the program.	Comments at least help the author understand intentions.	Comments are absent or more confusing than helpful.	
Delivery	The written work is delivered on time.	The written work is delivered 1-2 days late.	The written work is delivered 3-7 days late.	The written work is delivered more than 7 days late.	
Total					

Computer Programming in C++
Rubric for Programming Work on Classes

Category	Excellent (3)	Good (2)	Satisfactory (1)	Unsatisfactory (0)	Score
Correctness	All classes and methods match the written specification.	Most classes and methods match the written specification.	Many classes and methods match the written specification.	Hardly any classes and methods match the written specification.	
Modularity	Each class is contained in its own file with minimal dependencies on other files.	Very few classes share a file and there are very few unnecessary dependencies.	Several classes share a file and there are unnecessary dependencies.	Many classes are thrown together into the same file.	
Efficiency	No classes or methods are duplicated in code.	Very few classes or methods are duplicated in code.	Several classes or methods are duplicated in code.	Many classes or methods are duplicated in code.	
Presentation	White space and punctuation are consistent and add to legibility.	White space and punctuation are mostly consistent and legible.	White space and punctuation are present but inconsistent or confusing.	White space and punctuation are absent or used randomly.	
Documentation	All names used in the program are obvious or documented. Files contain appropriate headings.	Most names used in the program are obvious or documented. All files contain headings.	Many names used in the program are obvious or documented. Some files are absent headings.	Hardly any names used in the program are obvious or documented. Few files contain headings.	
Delivery	The program work is delivered on time.	The program work is delivered 1-2 days late.	The program work is delivered 3-7 days late.	The program work is delivered more than 7 days late.	
Total					