Engaging Early Programming Students with Modern Assignments Using BRIDGES*

Allie Beckman[1], Matthew Mcquaigue[1], Alec Goncharow[1]
David Burlinson[1], Kalpathi Subramanian[1]
Erik Saule[1], Jamie Payton[2]
[1]Computer Science, UNC Charlotte
{abeckma2, mmcquaig, agoncha1, dburlins, krs, esaule}@uncc.edu
[2]Computer and Information Sciences, Temple University
payton@temple.edu

Abstract

Early programming courses, such as CS1, are an important time to capture the interest of the students while imparting important technical knowledge. Yet many CS1 sections use contrived assignments and activities that tend to make students uninterested and doubt the usefulness of the content. We demonstrate that one can make an interesting CS1 experience for students by coupling interesting datasets with visual representations and interactive applications. Our approach enables teaching an engaging early programming course without changing the content of that course. This approach relies on the BRIDGES system that has been under development for the past 5 years; BRIDGES provides easy access to datasets and interactive applications. The assignments we present are all scaffolded to be directly integrated into most early programming courses to make routine topics more compelling and exciting.

1 Introduction

Computational literacy and problem-solving skills are crucial facets of an increasingly tech-driven economy and world. While enrollments in computing

majors have grown in recent years, particularly high attrition rates in these degree programs hamper the rate at which colleges and universities contribute to the modern workforce. Students in introductory and second-year courses are most susceptible, and much work has been done to investigate and address the factors contributing to the erosion of this student population [2].

One of the primary factors in maintaining interest among computing majors is their level of engagement with the course material. The perceived relevance of the material to the students' own lives and careers is crucial for kindling a desire to learn how to solve more complex problems down the road. Unfortunately, this is an area of weakness in many computing programs: the introductory courses are packed full of students, and it is enticing for universities to prioritize scalability over quality and rigor by relying on graduate students or more automated tools and paradigms to teach and evaluate the basic content.

Programming assignments in introductory courses have traditionally been contrived to teach basic structures like logical branching and loops using toy datasets, with basic command line input and output comprising students' interaction with the program. More engaging, modern approaches generally involve socially or culturally relevant data, simple graphical libraries, and more gamification of the material. All these features are prioritized in our work.

We present in this paper a lean educational framework that makes student engagement central to the content of introductory programming courses, without changing or compromising the rigor, content, and learning goals of these courses. Our BRIDGES framework uses a mix of real-world datasets from different domains, simple games and interactive applications as engagement tools to emphasize and reinforce core concepts in introductory programming. We present a set of assignments and their relationships to programming concepts, and we show how they were deployed in introductory CS courses and perceived by students of these classes. The framework and assignments are available online (`https://bridgesuncc.github.io/`).

2 Related Works

What CS1 typically look like. The typical first course in computer science, or CS1, introduces students to programming using a high level programming language such as Java or Python. The course goal is to introduce the basic constructs of the programming language, such as variables and expressions, control structures, functions and simple data structures like lists and strings. These courses train students to solve nontrivial problems using these constructs (e.g., see classic CS1 exemplar [11]).

There are wide variations in how the basic concepts are taught, in terms of the learning environments, tools, pedagogical approaches, and the student pop-

ulation (majors, non-majors) and demographics. Many introductory courses have begun to incorporate graphics, GUIs, and visualizations, as a creative output produced in projects [8, 4] or to illustrate key aspects of the underlying objects or algorithms [7, 3]. Furthermore, a rising awareness of the multi-disciplinary value of computing literacy has encouraged some institutions to experiment with different flavors of introductory programming courses based on game development [1], robotics [6], and data science [5].

What Makes Students Engaged. Ultimately, the goal of courses and curriculum is the overall education and academic success of the learners. To that end, materials that capture the imagination of incoming students and reinforce their interest and motivation in computing are particularly valuable. Popular assignment repositories (Nifty Assignments [14], EngageCSEdu [13], etc.) tend to include the 'fun' factor, as do game-themed assignments [16]. Usage of real-world and large datasets in course projects have also proven successful [12, 4], in contrast to using tiny, contrived or toy examples that fail to engage students.

In recent years, active learning techniques have been implemented in classrooms to promote student engagement, and include any combination of lab-based instruction, flipped classroom settings, gamification, peer-learning, and use of multimedia content [15, 9, 10].

3 The BRIDGES System

The BRIDGES system [4] is relevant to the goals of introductory CS courses such as CS1 and CS2. It has the capability to provide easy access to external datasets that can be readily used in course assignments. Secondly, it provides a 2D abstraction of a grid that can be used for games and image processing. The system provides bindings for the commonly used languages in early CS courses (Java, C++, and Python). Finally, results from assignments are highly visual and can be shared with friends and family, thanks to web-based rendering.

Dataset Access. BRIDGES provides simple APIs to access external data sources such as USGS Earthquake data, Wikidata, IMDB actor/movies, Genius' Song Lyrics, and OpenStreet Maps. A single function call returns data from a specific source, typically a list of objects, that can then be directly used. By using interesting datasets from different domains, assignments can be made more real and relevant to students.

Visualizing Bitmap Images. BRIDGES supports bitmap images through a 2D grid abstraction which can be the basis for assignments on images and

image processing. This also directly relates to 2D arrays and array addressing, which are central to CS1 and CS2.

Game API. BRIDGES supports a simple Game API that forms the basis for a number of 2D games that are readily usable and aligned with the goals of early CS courses. The game API has 4 core functions, (1) Reading Inputs, (2) Updating the game state using customized game mechanics, (3) Rendering to screen, and, (4) Maintaining a frame rate of 30 frames per second. In order to maintain simplicity of the API for new programmers and ensure smooth rendering, 2D games are implemented as 2D grids with a maximum of 1024 cells where each can be assigned a color and one of 256 predefined sprites, and 10 input keys for interaction. These constraints still enable the construction of many games and applications, and the *(student) programmer can focus on implementing the game logic and updates to the game state, while the display output and graphics are the responsibility of the system.* This lets the student focus on the course-level goals and not on the tool-level details. The user is responsible for implementing two functions: (1) initialize(), which is run once at the beginning of the game, and (2) gameLoop(), which contains all of the game logic and is called for each frame of the game.

This API is simple yet expressive enough to enable the implementation of a number of 2D games such as Bugstomp, Snake, 2048, Infinite Runner, Minesweeper, and Racing Car; as well as many of the Nifty [14] assignments such as Falling Sand, Spreading of Fire, and Hurricane Tracker. Each of these can be scaffolded to align with the learning outcomes of an early CS courses.

Students will typically run the code in their IDE and interact with the game through a web browser. However, our system also supports exporting games as standalone Android applications.

4 A Set of Engaging CS1 Assignments

We now describe a sequence of engaging assignments using BRIDGES that can be the basis for a CS1 curriculum. Table 1 illustrates the assignments and topics these assignments are aligned with. Assignments are scaffolded (available on our website) so that the students will see the specific functions that are to be implemented, in line with the objectives of the assignment. Instructors may request solutions for planning their course.

Etch a Smile. The student is given the task of drawing a smiley face. There are two versions of this assignment, Students write single lines of code which fill specific cells on a 2D grid using x and y values with a color of their choice. Students can also draw symbols on a cell. Alternately, they can use loops to

Assignment	Topics	Engagement
Etch A Smile	Functions	creativity, fun, visual output
BugStomp	Conditionals	Game Interaction
Tic Tac Toe	Conditionals, Std. I/O	Game Interaction
Song Lyrics	2D arrays, loops, conditionals, strings	real data, explore personal choices
Image Proc.	2D arrays, loops, File I/O	real data, visual output
Mountain Paths	2D arrays, loops, greedy algorithms	real data, visual output

Table 1: Example CS1 Projects, Topic Mappings and engagement factors

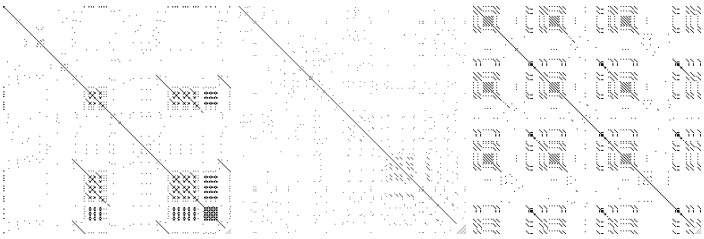
fill areas of the 2D grid with specific colors to create a face.
Where does it fit? Students will gain experience with generating a visually pleasing output using code that is more interesting than vanilla "hello world" function. They will also gain an understanding of coordinate grids from a programmers perspective. Familiarity with loops can be used to augment this assignment. Fig. 1 illustrates some examples.

Figure 1: Etch A Smile Face Example

Bugstomp. The game involves moving a sprite around a 2D grid to step on randomly generated bug sprites. The assignment involves moving around a 2D grid, generate random positions to place bugs in the grid and ensure that the player and bug are always within the grid.
Where does it fit? Moving a sprite within a 2D grid requires using input keys, updating the game board, and using conditional statements to ensure all positions stay within the grid, and checking if the character stepped on a bug.

Tic Tac Toe. An old assignment with a new look: students create a 3 x 3 board where they take turns placing their symbol and trying to get three in a row, column, or diagonal. Students can play against friends or develop a human vs computer experience. The board displays user chosen symbols for the players, and arrow keys are used to select a move.
Where does it fit? Tic-Tac-Toe is an excellent assignment for learning conditional statements, loops, and data management. Students can compare strings or values in an array to update the board. The game grid object of BRIDGES also provides options for students who have not seen arrays yet.

Analyzing Repetition in Song Lyrics. The student selects a song from an external data source (supported by BRIDGES) and parses each word. A 2D matrix is created using the words of the song that records whether the i-th word of a song is the same as the j-th word and creates unique patterns on a grid depending on the words repetitions of the selected song. See a highly repetitive song in Fig. 2c and one with little repetition in Fig. 2b.
Where does it fit? Students will have gained experience parsing and comparing strings using tokens. They will also have an improved understanding of loops, 2D array processing, conditional statements.



(a) "Radio Gaga" by Queen (b) "Guns and Ships" by L.-M. Miranda (c) "Satisfaction" by The Rolling Stones

Figure 2: Song Lyrics: Repetition Pattern for different songs

Image Processing. The student reads images given in some text format (e.g., PPM RGB images) into a 2D array and performs simple image processing operations such as image flipping, color flipping, removing a color primary. Students learn to work with images and how to manipulate colors.
Where does it fit? This is an excellent assignment for 2D array processing, understanding image structures: storage of the array data in a linear sequence, relationship between 1D and 2D addresses, and exercising loops and conditions.

Mountain Paths. This is an adaptation of a Nifty assignment [14] but it uses BRIDGES's visualization capabilities for displaying outputs. Students are provided with an elevation image of a mountain as a text file. Starting from a random pixel on the left edge of the image, the goal is to find a path that takes the least amount of effort to get to the right edge of the image. A simple greedy algorithm is used to make a local decision to move from one pixel to the next, such that the pixel with the least difference in elevation between the current position and the next position is chosen. The selected set of pixels are drawn in color to illustrate the path, as can be seen in Fig. 3.
Where does it fit? This assignment is a very nice introduction to greedy

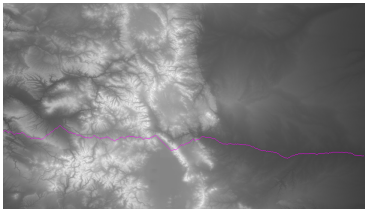


Figure 3: Path of Least Elevation Example

algorithms using a simple and highly engaging real world dataset. Students learn file I/O, implement a greedy algorithm, practice using conditionals, loops and 2D array processing.

5 Student Feedback

The projects detailed in Section 4 have been deployed in sophomore and junior level CS courses. These projects have been used often to introduce a new programming language (C++), or more frequently, as early 'warm-up' projects early in the course, or to illustrate a specific concept, such as object design. Table 2 illustrates the deployment of several of these projects over the past 3 years. We conducted project surveys and student reflections on the assignment that included the following questions:

1. What are the essential concepts that were learned by completing the assignment? [Short Answer]
2. Why is this important? [Short Answer]
3. How does the concept contribute to understanding how to program? [Short Answer]
4. The assignment was relevant to my career goals [5 point Likert scale].
5. The assignment increased my interest in computing [5 point Likert scale]

The reflection survey asked students the following questions:

1. Rate the difficulty of the module (5 point scale)
2. Roughly what percent of the module did you complete [25, 50, 75, 100]
3. Did you find the tasks engaging and meaningful? (4 point scale)
4. What did you like and not like about the assignment? [Short Answer]

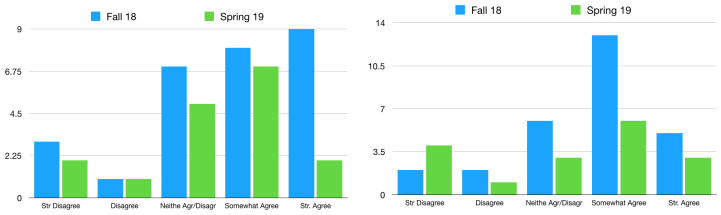
The class enrollment varied, ranging from 30-50 students across the different semesters. Participation was high (a small amount of credit was assigned for completing the surveys); however, only students who completed a substantial portion of the project were counted towards the survey participation.

Assignment	Semester/Year	Data Collection
Etch A Smile	F18,Spr.19	Student Reflection
Tic Tac Toe	F17,F18	Surveys, Student Reflection
Snakes & Ladders	Spr. 19	Student Reflection
Mountain Paths	F17,F18,Spr.19	Surveys, Student Reflection
Image Proc	F17,F18,Spr.19	Surveys, Student Reflection

Table 2: Student Feedback - Data Collection

Etch a Smile. Nearly all students found the project easy and completed it. The themes that emerged from the student reflections included the freedom to create a shape of their own choosing and play around with the color grid in creating a face. Examples of student quotes included '...allows creativity in a new way!', 'experiment and mess around with the coding', 'enjoyed the module and ... eager to learn more', 'enjoyed the assignment, it was engaging'.

Tic Tac Toe. This familiar game was also well received with over 80% completing the assignment and found it engaging. Student reflections were overall very positive, appreciated the moderate challenge, though several found it difficult as they were new to C++. Example free form responses included 'enjoyed making tic tac toe ... later plan to modify it to run on an arduino', 'game is a good way to demonstrate understanding of programming', 'liked the assignment, enjoyed creating the game'.



(a) *The assignment increased my interest in Computing.* (b) *The assignment was relevant to my career goals.*

Figure 4: Project Surveys (Fall 18 and Spring 19): Mountain Paths Assignment

Mountain Paths. This was a harder assignment, with 65-80% of the class completing 75% or more of the assignment. Almost all found the assignment engaging. The major themes from the reflection surveys were on the challenge of the assignment ('very challenging and I learned quite a bit', 'feel challenged, but also feel satisfied', 'was not prepared'), appreciation of the flexibility in the assignment ('loved the assignment allowed for different inputs', 'make a color object and amend it on each loop - it worked'), choice of the assignment

('excellent practical example of greedy algorithm') and the visualizations ('liked seeing the visualizations', 'enjoyable to finally see the best path line').

Fig. 4 shows the results of a survey of student's attitude towards computing. The results from two semesters show that the student were engaged by an assignment they perceived as relevant.

Image Processing This assignment continued from the Mountain Paths assignment. Students overall found the assignment challenging (roughly half in Fall 2018, 70% in Spring 19 found it difficult), and about 60-70% completed 75% or more of the assignment. Over 90% found the assignment engaging. Students appreciated the similarity of this assignment to the previous assignment ('last assignment was necessary to prepare me'), appreciation for the assignment ('good assignment with fairly clear instructions'), enjoyment ('liked the image processing portion', 'really liked seeing how easily implement some simple image processing with this assignment', 'liked manipulation of the image'), assessing success/failure ('my procrastination bit me...')

6 Conclusions

We have presented a set of highly engaging assignments that meet the principal goals of a typical introductory programming course like CS1. The assignments contain important elements of engagement, such as the use of real-world data, visualizations to see the final outputs, and interactive games and applications. We deployed and tested these assignments and collected student feedback. Overall, the student responses have been very positive: They find the assignments interesting, fun, and yet challenging.

Although these assignments have not yet been deployed in CS1, the topics and the assignments are at the level of a CS1. We have begun working with CS1/CS2 instructors using BRIDGES as part of their course. It would be interesting to do a comparison of the student responses with those deployed in a dedicated CS1 course using these engagement principles.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. DUE-1726809 and CCF-1652442.

References

[1] Jessica D Bayliss and Sean Strout. Games as a flavor of CS1. In *ACM SIGCSE Bulletin*, volume 38, pages 500–504, 2006.

[2] Theresa Beaubouef and John Mason. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106, 2005.

[3] Sarah Buchanan, Brandon Ochs, and Joseph J LaViola Jr. CSTutor: a pen-based tutor for data structure visualization. In *Proc. ACM SIGCSE*, pages 565–570, 2012.

[4] David Burlinson, Mihai Mehedint, Chris Grafer, Kalpathi Subramanian, Jamie Payton, Paula Goolkasian, Michael Youngblood, and Robert Kosara. Bridges: A system to enable creation of engaging data structures assignments with real-world data and visualizations. In *Proc. ACM SIGCSE*, pages 18–23, 2016.

[5] Sarah Dahlby Albright, Titus H Klinge, and Samuel A Rebelsky. A functional approach to data science in CS1. In *Proc. ACM SIGCSE*, pages 1035–1040, 2018.

[6] Amy Delman, Adiba Ishak, Lawrence Goetz, Mikhail Kunin, Yedidyah Langsam, and Theodore Raphan. Development of a system for teaching CS1 in C/C++ with lego NXT robots. In *FECS*, pages 396–400, 2010.

[7] Prasun Dewan. How a language-based GUI generator can influence the teaching of object-oriented programming. In *Proc. ACM SIGCSE*, pages 69–74, 2012.

[8] Ira Greenberg, Deepak Kumar, and Dianna Xu. Creative coding and visual portfolios for CS1. In *Proc. ACM SIGCSE*, pages 247–252, 2012.

[9] Mark Guzdial. A media computation course for non-majors. In *Proc. ITICSE*, pages 104–108, 2003.

[10] Diane Horton, Michelle Craig, Jennifer Campbell, Paul Gries, and Daniel Zingaro. Comparing outcomes in inverted and traditional CS1. In *Proc. ITICSE*, pages 261–266, 2014.

[11] Joint Taskforce on ACM Curricula. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM/IEEE Computer Society, 2013.

[12] Lucas Layman, Laurie Williams, and Kelli Slaten. Note to self: Make assignments meaningful. In *Proc. ACM SIGCSE*, SIGCSE '07, pages 459–463, 2007.

[13] Alvaro Monge, Beth A. Quinn, and Cameron L. Fadjo. EngageCSEdu: CS1 and CS2 materials for engaging and retaining undergraduate CS students. In *Proc. ACM SIGCSE*, pages 271–271, 2015.

[14] Nick Parlante. Nifty assignments, 2018.

[15] Johanna Pirker, Maria Riffnaller-Schiefer, and Christian Gütl. Motivational active learning: Engaging university students in computer science education. In *Proc. ITICSE*, pages 297–302, 2014.

[16] Kelvin Sung, Rebecca Rosenberg, Michael Panitz, and Ruth Anderson. Assessing game-themed programming assignments for CS1/2 courses. In *Proc. of GDCSE*, GDCSE '08, pages 51–55, 2008.