If Memory Serves: Towards Designing and Evaluating a Game for Teaching Pointers to Undergraduate Students

Monica M. McGill
mmmcgill@knox.edu
Knox College
Department of Computer Science
Galesburg, IL, USA

Chris Johnson
johnch@uwec.edu
University of Wisconsin
Department of Computer Science
Eau Claire, WI, USA

James Atlas
atlas@cis.udel.edu
University of Delaware
Computer and Information Sciences
Newark, DE, USA

Durell Bouchard
bouchard@roanoke.edu
Roanoke College
Math, Computer Science, & Physics
Salem, VA, USA

Chris Messom
christopher.messom@monash.edu
Monash University
Faculty of Information Technology
Melbourne, Australia

Ian Pollock
ian.pollock@csueastbay.edu
California State University, East Bay
Department of Art
Hayward, CA, USA

Michael James Scott
michael.scott@falmouth.ac.uk
Falmouth University
Games Academy
Penryn, Cornwall, UK

ABSTRACT

Games can serve as a valuable tool for enriching computer science education, since they can facilitate a number of conditions that can promote learning and instigate affective change. As part of the 22nd ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2017), the Working Group on Game Development for Computer Science Education convened to extend their prior work, a review of the literature and a review of over 120 educational games that support computing instruction. The Working Group builds off this earlier work to design and develop a prototype of a game grounded in specific learning objectives. They provide the source code for the game to the computing education community for further review, adaptation, and exploration. To aid this endeavor, the Working Group also chose to explore the research methods needed to establish validity, highlighting a need for more rigorous approaches to evaluate the effectiveness of the use of games in computer science education.

This report provides two distinct contributions to the body of knowledge in games for computer science education. We present an experience report in the form of a case study describing the design and development of *If Memory Serves*, a game to support teaching pointers to undergraduate students. We then propose guidelines to validate its effectiveness rooted in theoretical approaches for evaluating learning in games and media. We include an invitation to the computer science education community to explore the game's potential in classrooms and report on its ability to achieve the stated learning outcomes.

CCS CONCEPTS

•**Applied computing → Education;** •**Information systems →** *Multimedia information systems;*

KEYWORDS

games; educational; serious; learning; pointers; computer memory; design; development; research methods; validation framework

ACM Reference format:
Monica M. McGill, Chris Johnson, James Atlas, Durell Bouchard, Chris Messom, Ian Pollock, and Michael James Scott. 2017. If Memory Serves: Towards Designing and Evaluating a Game for Teaching Pointers to Undergraduate Students. In *Proceedings of the 22nd Annual ACM Conference on Innovation and Technology in Computer Science Education, Bologna, Italy, July 03–05, 2017 (ITiCSE'17)*, 22 pages.
DOI: https://doi.org/10.1145/3059009.3059037

1 INTRODUCTION

Educators have long used games as platforms for teaching and learning, with some of the earliest works on the subject appearing in 1987 [4, 64]. Games tend to possess qualities not typically found in traditional learning tasks. They often contextualize problems and situate them within a compelling alternate reality that unfolds through intriguing narrative, draw more upon a player's intrinsic motivations than extrinsic ones, provide structure for deliberate practice, scaffold activities effectively, while, among other things, also promote a spirit of free experimentation [18, 28, 99].

Computer science educators face a number of challenges with respect to educational attainment [49]. Some of these include high failures rates, particularly in introductory programming [13, 118], relatively poor retention [11, 46], and a prominent lack of diversity [20]. Game-based learning experiences seem to have the potential to help resolve some of these challenges [50, 85, 116]. However, the territory is relatively unexplored, and many of the advantages that game evangelists claim remain unrealized in the computer science education context.

At ITiCSE 2016, a working group convened to survey the landscape of existing digital games that have been used to teach and learn computer science concepts. The group identified and played over 120 games that had either been released or described in the literature as means for learning computer science concepts, and each was analyzed, classified, and mapped to the learning objectives outlined in the 2013 ACM/IEEE Computer Science Curricula [5]. While more games were found than initially expected, the working group report [60] highlights several key findings:

- Few games explicitly state their learning objectives;
- Most of the games cluster around one area of the computing curriculum: CS1;
- Most of the games were classified as belonging to one of two categories: short-lived proof-of-concept projects built by academics; or, closed-source games built by professional game developers;
- Most had not been formally evaluated in any way. Thus, there is little insight into the effectiveness and suitability of these games.

Gathering adequate data in order to determine the efficacy and suitability for use of all of these games is a challenging endeavor. However, supporting the generation of such insight is likely to provide considerable benefit to educators, researchers in the SIGCSE community, and future game developers. Thus, we chose to address these identified gaps in games for computing education in the second year. Explicitly, *the objective of this second year of the working group was to identify a suitable approach to the development and validation of games designed for use in computer science education.*

This objective was formed in conjunction with the following two overarching research questions:

RQ1 How can the design of a digital game be leveraged to scaffold student understanding of, and interest in, the abstract ideas in computer science education?

RQ2 How can the effectiveness of digital games for computer science education be validated?

To explore these questions, we developed the following 3 goals for this year's working group:

(1) Design and develop a prototype of a game for computing education that is engaging and relevant for the learners,
(2) Design and develop a prototype of a game for computing education built upon clearly defined and measurable learning objectives, and
(3) Provide a framework based on formal educational research theories for evaluating its capacity to meet those objectives.

Though not perfectly aligned with the overarching research questions, we note that goals 1 and 2 address (in part) research question 1, and goals 2 and 3 address (in part) research question 2.

The remainder of the paper is divided into four sections. Section 2 provides a brief background of the challenges and potential limitations of using games to teach computer science. Section 3 provides a concentrated report focused on our experiences designing and developing the prototype of the game, including a description of the game. Section 4 provides a set of hypotheses and evaluation methods we propose using to evaluate the game for effectiveness. This hypothesis and evaluation model can also serve as a potential model for others to use when designing research studies to evaluate this or similar games. Section 5 is our plan for future work on the game, with a call for other researchers to join us in adding to, remixing, and extending the game.

2 BACKGROUND

Given that the first year of this working group created an extensive literature review of games for learning and games for learning computing education, and provides a framework for designing and evaluating games for computing education, we refer the reader to our previous work for a more extensive analysis [60]. In this section, we provide a brief overview of games for learning and a brief overview of games for computer science education. We also provide a look at one underserved area of games in computing education: memory management and pointers.

2.1 Games for Learning

Despite the well-noted motivational pull of video games, a systematic review of pass rates in early programming classes suggests that, compared to lecture-lab courses, the incorporation of a game into a course does not guarantee improvement [101, 115]. It was found that games, in terms of absolute improvement, were only ranked 9th out of the 13 types of educational intervention included in the study. Instead, those classes with relatable content and collaborative elements helped to improve pass rates. It is important to recognize, however, that there is considerable variation in how games could be implemented in a course, which may explain the variance as well as the mix of positive and negative effects. Indeed, some of these implementations may even have contained relatable content and collaboration, which could have confounded the impacts.

The Clark-Kozma debate on media effects highlights and debates this assumption: that one medium will inherently be better or worse than another [23, 54, 62, 107]. A game-oriented learning experience, in and of itself, may not necessarily be superior to all other forms of instruction. Within educational games themselves, as artifacts, there is likely to be considerable variance in the effectiveness of one game design over another. The design of the learning experience is critical to consider. To this end, games offer advantages in several key areas such as active learning [41, 120] and formative feedback [10, 16, 81]. However, these aspects of a learning experience are not unique to the medium of games–hence, the need for rigorous validation.

Games are no silver bullet, since it is possible for them to possess weaknesses in their design. One notable challenge in some game designs is that the learning achieved within the game may not transfer to practically useful contexts. This is known as the Vegas Effect—"learning that happens in games stays in games" [36]. Avoiding this effect demands that transfer be considered in the design and evaluated using appropriate measurements.

Outside of the games for computing context, an educational game that exhibited such an effect was *Crystal Island* [108], a game intended to help students learn how pathogens work. Rigorous experimentation did not support the efficacy of the game [6]. The study claims the data does not support narrative discovery learning, in line with similar criticisms in other media contexts [70]. Extraneous story material and unduly onerous interaction methods can challenge learners with a high cognitive load [61]. The authors go on to caution against what Meyer [72] describes as a technological-oriented approach to educational technology, instead advocating a learner-centered approach.

Even with such a learner-centered approach, however, care must be taken in the design of games' dynamics. For example, *Spent Play* [97, 98, 100] has an underlying goal of promoting constructive attitudes about the poor. However, "if playing *Spent* in the role of a poor person leads the player to view his or her experiences as reflecting the experiences of the poor, performing well on the game should lead players to believe that poor people, like him or her, can lift themselves out of poverty just by making the right choices. The player's controllability beliefs should be intensified, leading to less liking of the poor".

Another pitfall can be the difficulty of the core game mechanic. This relates to poor scaffolding and may block the progression of a student beyond their particular zone of proximal development [117]. As such, it is important to consider domain-specific pedagogic content knowledge (see [51, 77]) to balance the challenge of the deliberate practice and its alignment with intermediate learning goals with playability.

Moreover, misalignment between learning objectives and activities within a game is another concern. Core game mechanics that are "exogenous" do not require any learning to have taken place in order to progress through the game. This is opposed to "endogenous" where there is direct constructive alignment [15] such that successful learning facilitates progression through the game [53].

Considering some of the features of existing games for computer science education [60], it would seem that many have such pitfalls. Unfortunately, authors such as O'Neil and Perez [84] make clear that "there is almost no guidance for game designers and developers on how to design games that facilitate learning.", and work conducted in recent years suggests such advice remains limited [18, 24, 28]. However, when designing an interactive learning experience, it is worth considering the volume of research on cognition [71, 103] and cognitive load [111] in various forms of multimedia [61, 73, 74, 78, 79], some of which is being extended in the games playing context [61].

2.2 Previous Review of Games for Computer Science Education

As part of our work undertaken during the 2016 ITiCSE Working Group, we reviewed over 120 games designed to teach computing concepts and reviewed several dozen papers related to game-based learning (GBL) for computing. Each review consisted of inspecting available documentation and commentary and, when possible, playing the game. Using this information, we categorized each game with respect to numerous pedagogical and game characteristics [5]. These games were published over the last several decades, ranging from 1982 to 2016. At the time the games were reviewed, 34 were available commercially, 51 were freely available, the remaining were either no longer available or could not be found. In this section, we provide a general summary of the games and a classification of the games in context of the ACM/IEEE Computer Science Curricula 2013, as shown in Table 1 [13].

Table 1: Games classified by CS knowledge areas

Knowledge Areas	Count
SDF (Software Development Fundamentals)	77
AL (Algorithms and Complexity)	29
CN (Computational Science)	11
GV (Graphics and Visualization)	10
AR (Architecture and Organization)	9
IAS (Information Assurance and Security)	6
SE (Software Engineering)	5
HCI (Human-Computer Interaction)	4
IS (Intelligent Systems)	4
SF (Systems Fundamentals)	3
DS (Discrete Structures)	3
PBD (Platform-Based Development)	2
SP (Social Issues and Professional Practice)	1
PL (Programming Languages)	1

We also noted that Hainey [57] found that there is "a dearth of empirical evidence in the fields of computer science, software engineering and information systems to support the use of GBL." Based on our review, our findings supported these claims–we were unable to find evidence identifying which of these games were most effective in meeting educational outcomes. Further, this also prevented us from analyzing and proposing game design frameworks for CS education that might be most effective across various demographics.

In addition to a summary of the games reviewed, we provided in the report a guide to developing digital games designed to teach knowledge, skills, and/or attitudes related to computer science. This summary was based on an extensive review of well known and respected game designers and educational researchers, including Gee, Hunicke, Lee, and dozens of others [2, 3, 44, 45, 59]. In the report, we also review the necessary focus on motivation and how to motivate learners, including a review of Ryan, Rigby and Przybylski's work, self-determination theory, and more [101]. We refer the reader to the working group report for a more-in depth literature review and how this review informed guidelines presented for designing a game for computer science education [60].

The three primary gaps, which we highlighted in section 1, were the lack of reporting on the effectiveness of the games, the lack of reporting on the initial planning of the game (no learning objectives, scaffolding discussions, etc), and the lack of games outside the Software Development Fundamentals (SDF) knowledge area.

2.3 Teaching Memory Management and Pointers

As we searched for an interesting learning objective to tackle within a game, we considered the topics that we teach as well as those that other instructors find challenging for learners. In addition to experiences of those in the group, pointers were commonly identified as a candidate threshold concept in surveys of computer science educators administered by Boustedt et al. [17]. Threshold concepts, as defined by Meyer and Lind [76], meet the following criteria: they transform how a student perceives the discipline, they help the student organize knowledge, and while often difficult to learn, they are difficult to unlearn. Based on a series of interviews with students, Boustedt et al. argue that pointers do indeed qualify for this designation.

A taxonomy of concepts associated with pointers is provided by Craig and Petersen [30]. They propose and evaluate a concept dependency map, which we have used to design the progression of levels in If Memory Serves. They also found students struggled the most with two concepts: 1) distinguishing between a pointer (a variable that holds an address) and the address itself and 2) understanding the relationship between pointers and arrays.

Allevato et al. [9] identify memory management as one of the most frequent issues their students encounter. Many programming tools, they argue, provide misleading or unhelpful error messages for dealing with pointer issues. They therefore developed Dereferee, an abstraction that wraps around pointers in C++ to detect mismanagement of memory and provide more helpful error messages. Their abstraction wraps around the standard pointer operations to identify 40 kinds of memory violations, including invalid pointer arithmetic, memory leaks, and dereferencing invalid pointers—perhaps because they are null, uninitialized, stale, or out of bounds. They found that Dereferee identified the exact source of 83% of bugs in a linked list assignment.

Adcock et al. [7] cite continued industry demand for experience in languages like C and C++ as reason for teaching memory management. To provide feedback to students, like Allevato et al., they model each pointer as a state machine that is used to detect and report illegal and unsafe operations. They found that the most common errors were dereferences of stale or null pointers and memory leaks.

2.4 Summary

Several gaps exist in many of the current games for computing education, including poor scaffolding, misalignment of learning objectives and in-game activities, lack of clearly defined learning objectives (or any learning objectives), and lack of evaluating the game to determine if it increased student learning. It is imperative that computing education games are designed to address these games, including the need to motivate and engage learners and to evaluate the games for effectiveness. To this end, we synthesized and extended existing guidelines (e.g., [26, 27, 32, 52, 68, 92, 102]—described further in [60]) and applied them to the creation of a new educational game for an under served area of computing education, memory management and pointers. The next section details our experience designing and developing a prototype of a game during ITiCSE 2017.

3 POINTER GAME CASE STUDY

In this section, we revisit research question 1:

RQ1 How can the design of a digital game be leveraged to scaffold student understanding of, and interest in, the abstract ideas in computer science education?

We present our experience designing and prototyping a game to address research question 1 as well as many of the concerns raised in the previous year's report. This experience report is presented in the form of a case study. It provides our rationale and thought processes throughout the concentrated time spent designing and developing a prototype of the game, *If Memory Serves* (`http://ifmemoryserves.org`). We include a summary of our team composition, early design considerations and decisions, a description of the game, including a summary of decisions related to designing and creating levels, and our consideration of orthogonal elements related to gameplay.

3.1 Team Composition

To provide context to our work, the seven members of our group have extensive experience teaching thousands of undergraduate and graduate students throughout a variety of computer science and game development courses. In addition to general introduction to computer science and interactive media courses, several in our group have taught courses covering computer architecture and data structures in C++, both of which include significant work with pointers.

Everyone on our team has also been actively involved in researching computer science education in either primary, secondary, or post-secondary education, including professional development of K-12 teachers. A couple of our team members have been involved with learning analytics, and several have conducted relevant research and have published in the areas of computing education and educational games. This includes one member who has been researching how to teach creative practitioners about computational thinking and coding for 20 years.

In addition to years of professional software engineering, analysis, and consulting experience, several members of our team have decades of experience researching, designing, developing, and publishing games and interactive playable media. This also includes teaching game design and development, game production, creation of art for games, narrative and storytelling for games. One team member has 20 years of experience teaching media arts students coding and design thinking at various universities. We have used and taught various tools for game development Unity3D Game Engine, Unreal Engine, Android, and Adobe Creative Suite, and more.

3.2 Game Overview

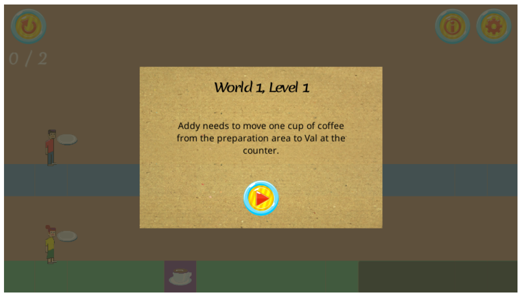
To provide context to subsequent sections, we provide here a general overview of *If Memory Serves.* The proposed game is a 2.5D, cooperative, resource management, puzzle game, in the spirit of *Diner Dash*, a resource management game that has been popular among both boys and girls. This, combined with a café theme, was carefully chosen for the familiarity in game play amount our target demographic as well as familiarity of its theme.

The world of the game is a top-down view of a café that uses pneumatic tubes to deliver food to customers. The player's view of the café includes both the kitchen and the dining area with a counter bisecting the two that the players' characters cannot cross. Players take the role of one of the Pointer sisters, either Val, the chef in the kitchen, or Addy, the server in the dining area. The chef can pick up and move dishes of food between the kitchen and the counter. This is an analogy for moving values between static memory (read-only, global memory) and automatic memory (the stack) in a computer program. The server can modify the pneumatic tubes that transport the dishes of food between the counter and the dining tables. This is an analogy for creating pointers between and dereferencing automatic memory (the stack) and dynamic memory (the heap).

The goal of the game is to serve customers by fulfilling orders by moving dishes of food to different areas. Serving food to a customer, clearing a table, and moving food from one place to another requires both the chef and the server to perform a particular sequence of actions. In particular, since the number of tubes is restricted, the movements of each player are restricted, and the abilities of each player are differentiated so they must work together to fulfill the order. Further, each of the players' actions corresponds to operations in computer memory that can be represented as C or C++ code. For example, World 1 (in Figure 2 depicts the act of copying values, only abstracted in the analogy of moving coffee from the preparation area to the serving area.

It is worth noting that the players are restricted from accessing all of the café to create a play mechanic that facilitates the creation of interesting puzzles and requires the cooperation of the two players. This restriction, however, introduces a false analogy because computer programs can access all three parts of memory without restriction. Thus, though we align the gameplay with tasks related to memory allocation, we recognize that some explanation may be necessary when explaining this analogy with students.

We present here several screen shots of the early prototype for If Memory Serves (`http://ifmemoryserves.org`) in its current 2D game view (see Figures 1, 2, 3, 4, and 5). Val is in the preparation area (bottom of scene) and Addy is in the serving area above that. The top area is reserved for customer tables (see Figures 2 and 3). The corresponding lines of code for player actions is currently set to visible and appears in the bottom right of the scene. At the time of this writing, there are currently six worlds with each world having between 2 and 4 levels.



World 1, Level 1

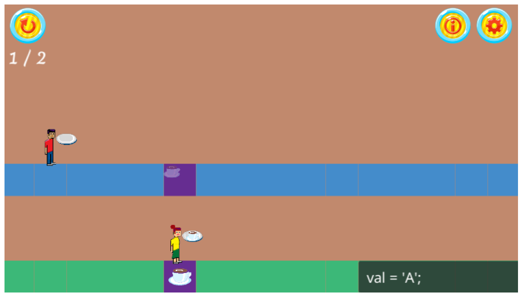
Addy needs to move one cup of coffee from the preparation area to Val at the counter.

Figure 1: Overlay for World 1, Level 1.

3.3 Early Design Considerations and Decisions

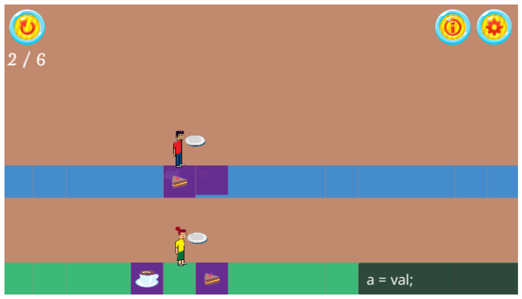
The key aim in our design process was to have the ideal learning outcomes influence the design of the game. With this focus in mind the three key components were the:

- learning objectives



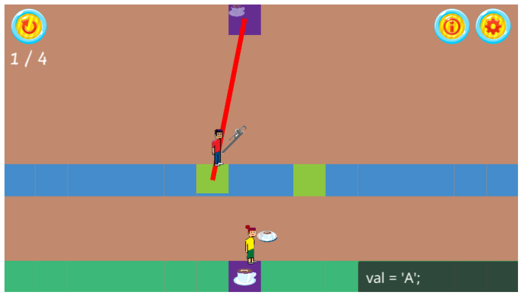
```
val = 'A';
```

Figure 2: World 1, Level 1. Players must move a cup of coffee from the preparation area to the serving area.



```
a = val;
```

Figure 3: Overlay for World 1, Level 4. Players need to help put a cup of coffee and a piece of cake on the counter.



```
val = 'A';
```

Figure 4: Overlay for World 2, Level 1. Addy can pick up a delivery pipe or attach a delivery pipe to a green square. Val can push or pull food through the pipe. For this level, players need to put a cup of coffee on the table.

- game elements (including the game's mechanics, dynamics, and aesthetics [59])
- learning objective-to-game element mapping

As part of our initial attempt at forming the criteria, we also considered how the learning objectives align with the ACM/IEEE Computing Curriculum categories [5]. Categorizing the game in terms of established and community-defined learning outcomes can provide a reference for instructors when searching for resources for teaching these topics. These are presented above in Table 2.

In this section, we describe the development process that we used to structure the planning and implementation of our game. Since game design is often a process of exploration and refinement, these

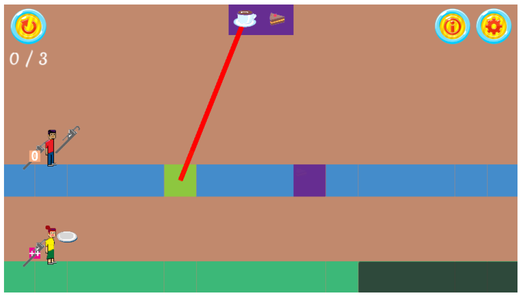


Figure 5: Overlay for World 4, Level 1. Addy's increment tool is used on a pipe's origin green square to rotate a pipe through positions at a table. Val has an offset tool which will temporarily redirect a pipe if Val is on top of the pipe's origin green square. It's initial value is 0 to indicate no offset, but when used on a pipe's origin green square will iterate over possible offset values at the table. Players must get a piece of cake on the counter.

Table 2: High level learning objectives of *If Memory Serves* mapped to ACM/IEEE CC2013 categories [5]

Category	Learning Outcomes
OS/ Memory Management	Review of physical memory and memory management hardware
PL/Language Translation and Execution	Memory management Manual memory management: allocating, de-allocating, and reusing heap memory
PL/ Runtime Systems	Dynamic memory management approaches and techniques
SDF/ Fundamental Data Structures	References and aliasing Linked lists
AR/ Memory System Organization and Architecture	Main memory organization and operation

steps were non-linear. As we worked through ideas, objectives, and criteria, the design and objectives morphed and shifted.

3.3.1 Identifying Learning Objectives. To begin the development process, we identified a set of learning objectives that:

(1) were generally accepted as important to the discipline of computer science,
(2) invoked concepts that students find difficult,
(3) were specific enough to be evaluated using short, targeted assessments, and
(4) were not already addressed by existing games.

Not all objectives found in other games for computing education satisfied these criteria. For example, we found many games in our prior work that could be classified as Logo puzzlers, or games in which movement commands are used to drive an agent along a path to a goal [60]. Such games are abundant and tend to have broad objectives that touch upon many aspects of programming, making it challenging to conduct a focused evaluation on the entire game. In these instances, the games could be classified as sandbox games and can be viewed more as a tool for programming in general rather than focused on one particular concept.

The initial list of broader categories we generated encompassed a variety of subjects, including debugging, functional decomposition, iteration, memory diagramming, and memory management. Pointer programming emerged as an ideal target problem that satisfied the four requirements above and were also previously identified in research as a topic in which students often struggle.

Since this was a core component in developing the criteria for the game, as our discussions progressed, we created and refined a table for the levels and what learning outcomes each level might address. As we worked through the topic, we generated a list of ideal learning outcomes, shown in Table 3.

3.3.2 Identifying Game Concepts. The next step in our process was to generate ideas that could be used to achieve one or more of our learning objectives. Several genres were offered, including match-3, word-puzzle and memory-style games, and the group converged on a rough idea of a puzzle game in which a player manipulated memory through pointers. This phase of the process also led to a list of secondary objectives that we hoped to achieve. In this section, we describe several of the factors that we considered as we generated ideas and criteria for the game.

3.3.3 Target Audience. We identified the target audience as undergraduate computer science students, most of whom are 18-24 years old. We chose this age group for several reasons. In addition to the reasons given earlier (see section 2.2) that pointers are an area that many students struggle, pointers are a more advanced concept often taught after introductory computing courses. As designers and developers, we would also have access to these groups for testing the game.

We were also intrigued by the prospect of making a game that was locally cooperative, meaning that two players would collaborate to achieve their goal using the same computer. Collaborative games are emerging as a popular form of game, where as well as developing collaborative skills, game players also build up a social bond with their partners [75, 94]. This is similar to some of the challenges and benefits that emerge with pair programming both in industry and academia. By so doing, students could play the game together in a lab setting and discuss strategies.

3.3.4 Game Thematics. Brainstorming resulted in several potential metaphors and themes that could be used for the game. A key feature of the metaphors is that they needed to have two spaces (metaphors for the stack, where values are manipulated and the heap, where pointers are manipulated). The most favorable metaphors included biological cells/environment, space/planets, brain/world and café (kitchen and restaurant). The café metaphor was chosen as it would be the most familiar metaphor for the target users and a theme in which many students could relate.

Although a powerful metaphor has been identified in the café game, the details of the analogies between the low level operations in pointer programming and kitchen/restaurant operation must be developed. Identifying these analogies helped us (as the game's designers) identify potential errors in understanding what can develop with false analogies and how they are likely to inhibit learning in the target domain.

Table 3: Ideal learning outcomes targeted through the game design

#	Learning Objectives	Assessment Criteria
A	Copying values (no pointers)	(1) know that values can be copied (2) develop skills to copy values
B	Create pointer, assign value, retrieve value, dereferencing	(1) create a pointer. (2) assign values into memory using pointers (3) dereference a pointer (get value stored there)
C	Swap values using pointers	(1) swap values using pointers and intermediate storage (2) identify errors in swapping pointers (e.g. not using intermediate storage)
D	Value versus pointers	(1) identify the difference between values and pointers (2) correctly identify when to use a value versus when to use a pointer
E	Pointer Arithmetic	(1) use pointer arithmetic to iterate through arrays of values (2) use pointer arithmetic to randomly access array locations (3) Identify errors in the use of pointer arithmetic
F	Double Pointers	(1) use pointers to pointers to create two-dimensional data structures (2) use pointer to pointers store and retrieve values from two-dimensional data structures (3) identify errors in the use of double pointers
G	Malloc/Free	(1) use malloc to allocate memory in the heap (2) use free to release memory (3) identify errors in the use of malloc and free
H	Data Structure	(1) use pointers to create data structure such as linked lists (2) use pointers to access and update data structure such as linked lists (3) Identify errors in the use of pointers in data structures

3.3.5 Game Mechanics. The mechanics of the game requires a keyboard and for each player to control one of the characters. In addition to the movement keys, each player uses 2-3 additional key functions to be able to move the food/beverages to the appropriate counters so the items could be served to the customers.

2D and 2.5D games potentially can have simple sliding tile mechanics; however, physic/kinematic based can also be beneficial, in particular when the key features of the game mechanic include collision. Constraining the players within the play areas is easily achieved using physic/kinematic based game mechanics and similarly collisions between players, food/beverages and kitchen benches/tables can be used to control the passing of objects to and from players to the game environment.

3.3.6 Game Representation. The proposed game representation is a 2.5D resource management puzzle game, but the prototype is pure 2D. To achieve both, we created both 2D placeholder art and 2.5D art assets for the early concept build of the game for the player characters as well as the objects in the game (Figure 6, 7). We intend to replace the placeholder assets with the 2.5D assets as we continue working on the game.

Both players view the world in third-person. The screen space is fixed, with no off-screen space or scrolling.

Flexibly-licensed sound assets were found that are consistent with a restaurant theme. These include background chatter, background music, and other ambient sounds and have been added to the game prototype.

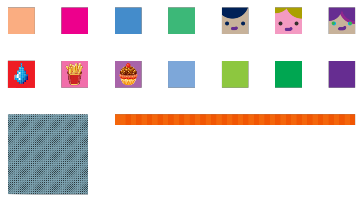


Figure 6: Sampling of 2D sprites

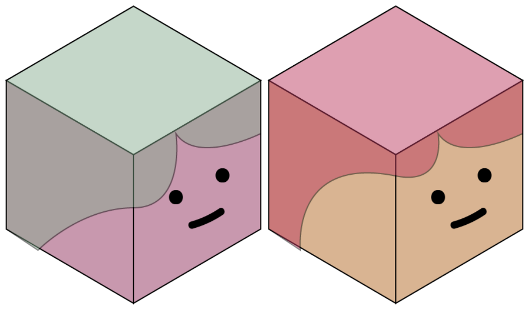


Figure 7: Sampling of stylized 2.5D sprites

3.3.7 Objective-Task Mapping. As mentioned above, once the target problem area was identified, we developed a set of ideal learning objectives, ordering them in a manner in which they were typically taught in courses. This was an organic process that originated in tandem with a rough breakdown of what and how we wanted to teach in the various levels. We aligned them to specific tasks that would be completed in the game, so the key game tasks were chosen to contribute to the intended learning outcomes, rather than add additional cognitive load for the student. This mapping between the learning outcomes and the in-game goals were mapped as shown in Table 4.

These learning outcomes were intended to be implemented within the game while at ITiCSE 2017, with a goal of creating levels 1-5 by the end of the conference and future refinement and new levels added thereafter.

3.3.8 Level Design. The specific tasks of the game are presented to players through its level design. As players complete the tasks in each level they progress to the higher levels. Normally there are several sub-levels of each level that players must complete. Some games will strictly enforce progression through the levels of the game; however, often it is possible to jump to the relevant level that a player is interested in using a menu structure.

Sometimes players will reach a point in a level where it is impossible to complete (e.g. when a value is overwritten in the classic pointer swapping task). In these scenarios the game must recognize that a solution is not possible, or a mechanism be provided for the player to repeat the level. In *If Memory Serves*, the player can reset the level by clicking on a reset level button.

The educational designer specifies a detailed learning task, for example, the classical swap example shown in Table 5.

Levels in the game are specified using a text-based level file that describes the initial configuration of the level as well as the target that must be achieved. A preliminary set of levels have been designed going from the most basic operations, through to the higher level learning outcomes that have been targeted for the game. These levels can also be extended easily by educational designers as the game mechanics do not need to be changed to make use of these new level, since they are dynamically created.

The game was prototyped using paper based mockups. Although player movement can be tedious using paper mockups, they are recommended for identifying potential pitfalls in the proposed game metaphors, themes, mechanics, and player tasks. The paper prototypes also gave us an opportunity to make sure we each understood how the mechanics would work and to look for the "fun." See Figure 8 for several examples of the early prototyping process.

3.4 Orthogonal Elements to Gameplay

There are many potential orthogonal elements to the pure (educational) game play that can enhance the engagement and motivation of the students who play. These include elements such as scoring and achievements (i.e., badges, leaderboards, and in-game rewards). These features can be developed independently of the raw game play and can be switched on and off as needed in the educational context where the game is used.

Many games include some scoring mechanism, either based on time or based on task completion. This provides a mechanism for giving external feedback on playing the game. As the player improves, a higher score is achieved. *Extrinsic rewards* such as these tend to not be as effective as *intrinsic rewards*, which is the satisfaction of actually learning and improving. In cases where self-reflection is difficult, the extrinsic reward system can be beneficial. We chose to loosely incorporate scoring to reward players when they are able to successfully complete a level. A higher score is given if the player meets or beats the par for that particular level.

In general, marking achievement in the game via, trophies, badges and leaderboards encourages the students self-efficacy and encourages students to use the game to improve their learning [33, 34]. Badges are used in games to recognize completion of tasks and sub-goals. These are visible mechanism for marking achievement of the player, which is maintained after a game is played. This is different from a game score that is reset the next time that the game is played. Again this is an extrinsic reward mechanism, however if there is good alignment between task completion and actual skills and knowledge, then this extrinsic marker is in fact a representation of the internal skill or knowledge acquired.

The collectivist approach used by Decker aligns well with several aspects of self-determination theory (i.e., senses of competency, relatedness, and autonomy) [33]. It can be very effective and, via organismic integration, help learners assimilate the external drivers. With this in mind, we have considered adding badges to create behaviors that wouldn't otherwise exist. Some research suggests they can also stifle behaviors that already exist as soon as the badges are no longer offered (potentially inhibiting forms of practice, depending on the badge). For this early prototype stage, badges are not incorporated; however, we anticipate future development of badges and other achievements as the game evolves.

4 VALIDATION FRAMEWORK

A key part of the educational game development process involves the evaluation of the game to determine its effectiveness. Such evaluation can be summative, to support claims of effectiveness, or formative, to set a road map for further work to improve the game. Sometimes both, to form a feedback loop that drives a develop-release cycle that is creatively iterative and/or makes incremental improvements in efficacy.

However, based on the findings of the group's prior work [60], it was deemed infeasible to conduct a rigorous evaluation of all or even a set of games previously reviewed the prior year in the available time-frame. Notably, most of the games identified did not have clear and granularly specified learning objectives nor were there suitable measurement instruments available; for example, in the case of *If Memory Serves*, as an aide in motivating students and helping them to learn about pointers.

Within the limited scope of the session at ITiCSE 2017, there was only sufficient time to identify appropriate learning objectives; design the game based on these objectives; iteratively develop a modest series of ludic sketches and paper mockups; and then implement and extend a digital prototype. Though members of the working group informally tested these prototypes, such user tests primarily addressed the mechanics and dynamics of the game design as well as methods of interaction and technical challenges.

Table 4: Initial mapping of learning outcomes including example tasks

Level #	LO	Objective	Example
1	A	Copying values	Demonstrate that values are copyable

2	B	Address-of / dereferencing, memory referencing/dereferencing, calling semantics	`clamp (lo, hi, *x) {`
			`  if (*x < lo) *x = lo;`
			`  else if (*x > hi) *x = hi;`
			`}`
			`clamp(0, 100, &val)`

3	C	Swap - temporary storage	`swap (*x, *y) {`
			`  int tmp = *x;`
			`  *x = *y;`
			`  *y = tmp;`
			`}`

4	D	Value versus address	

5	E	Pointer arithmetic, pointer/array duality	`for (int i=0; i < strlen(s); ++i) {  //  s[i] ; }`
			versus
			`for (char *p=s; c!=='\\0';++p)   { // *p ; }`

6†	F	Double pointers	A structure including a pointer member that is passed by reference to a function that modifies the member.
7†	G	Malloc/free	Instantiation of a collection of structures for which the number of simultaneous instances needed is not known until run time.
8†	H	Data structure	Linked list

† denotes a stretch goal

Given the need to provide a model for creating a new game with such clear aims within the context of computing education, such analysis is beyond the scope of this report. Yet, there remains a critical need to evaluate the effectiveness of this particular game as well as others currently in development. Revisiting research question two,

- ”[RQ2] How can the effectiveness of digital games for computer science education be validated?”

we were aware of the lack of measuring effectiveness with the previous games reviewed. Hence, we agreed to propose a validation framework, recommending research methods that would enable us as well as other educators and researchers to take the game, use it in their own institutions, and report their findings.

In this section, we establish a set of hypothesis, address epistemological challenges, present a design science approach, present a potential research model, and propose a set of instruments to use to measure each hypotheses.

4.1 Establishing Viable Hypothesis

While we created the prototype of the game, we were aware that we needed to align the learning objectives with measurable outcomes and provide a structured framework for validating these outcomes. To instantiate a validation framework, we first needed to determine a set of hypothesis for the game outcomes. The National Science Foundation's Office of International Science and Engineering (NSF ISE) defines five categories of educational technology impact that can be measured: knowledge; engagement; attitude; behavior; and skills [42]. Considering these alongside the original research questions, we propose these five initial hypotheses:

H_1 *Knowledge*: Students playing *If Memory Serves* in tandem with traditional learning activities (e.g., readings, lecture, etc.) will perform better on a test of conceptual knowledge on pointers (i.e., purpose, usage and syntax, tracing and prediction, debugging, etc.) than those students who only participate in the traditional learning activities.

H_2 *Engagement*: Students playing *If Memory Serves* in tandem with traditional learning activities (e.g., readings, lecture, etc.) will report a higher level of engagement with their learning activity on pointers than those students who only participate in the traditional learning activities.

H_3 *Attitude*: Students playing *If Memory Serves* in tandem with traditional learning activities (e.g., readings, lecture, etc.) will report greater confidence and positivity about their mastery of pointers than those students who only participate in the traditional learning activities.

H_4 *Behavior*: Students playing *If Memory Serves* in tandem with traditional learning activities (e.g., readings, lecture, etc.) will be observed to spend more time-on-task *and*

Table 5: Specifying level through educational design

Educational Designers Description of Task

Swap two values in heap using a temporary stack value. Initial Layout: Two heap values, A and B, one empty stack pointer, and one empty stack value.

Code Description of task and solution

```
// initial code
register char val;
char *p = new char('A');
char *q;
char a;
char *r = new char('B');
// solution code
q = p; // Addy creates a pointer from stack to A.
a = *q; // Val dereferences pointer to copy A to the empty stack value.
q = r; // Addy overwrites the pointer from stack to B.
val = *q; // Val dereferences pointer to copy B.
q = p; // Addy overwrites the pointer from stack to A.
*q = val; // Val moves copy of B to heap via pointer both heap values are B).
q = r; // Addy overwrites pointer from stack to other B.
*q = a; // Val copies the value A from the stack to the heap via the pointer.
```

Initial level specification

```
 ; Initial Layout
--      A B       --
--                --
--                --
--                --
-- &              --
--++++.+.+x+.++++++--
--    _     _     --
-- *              --
--------------------
```

Goal level specification

```
 ; Target Layout
--      B A       --
--                --
--                --
--                --
--                --
--++++.+.+?+.+++++--
--    _     _     --
--                --
--------------------
```

apply different learning strategies than those students who only participate in the traditional learning activities.

H_5 *Skills*: Students playing *If Memory Serves* in tandem with traditional learning activities (e.g., readings, lecture, etc.) will demonstrate more apt and coherent use of pointers in a programming assignment (measured by functional coherence, source code sophistication, and source code maintainability) than those students who only participate in the traditional learning activities.

Keeping these five hypotheses in mind, in the following sections we present a theoretical view of epistemological challenges, a proposed design science approach to conduct the research, and a research model before revisiting how to specifically measure each.

4.2 Epistemological Challenges

Proposing a single framework to validate *If Memory Serves* and other educational games that are being used to educate computing

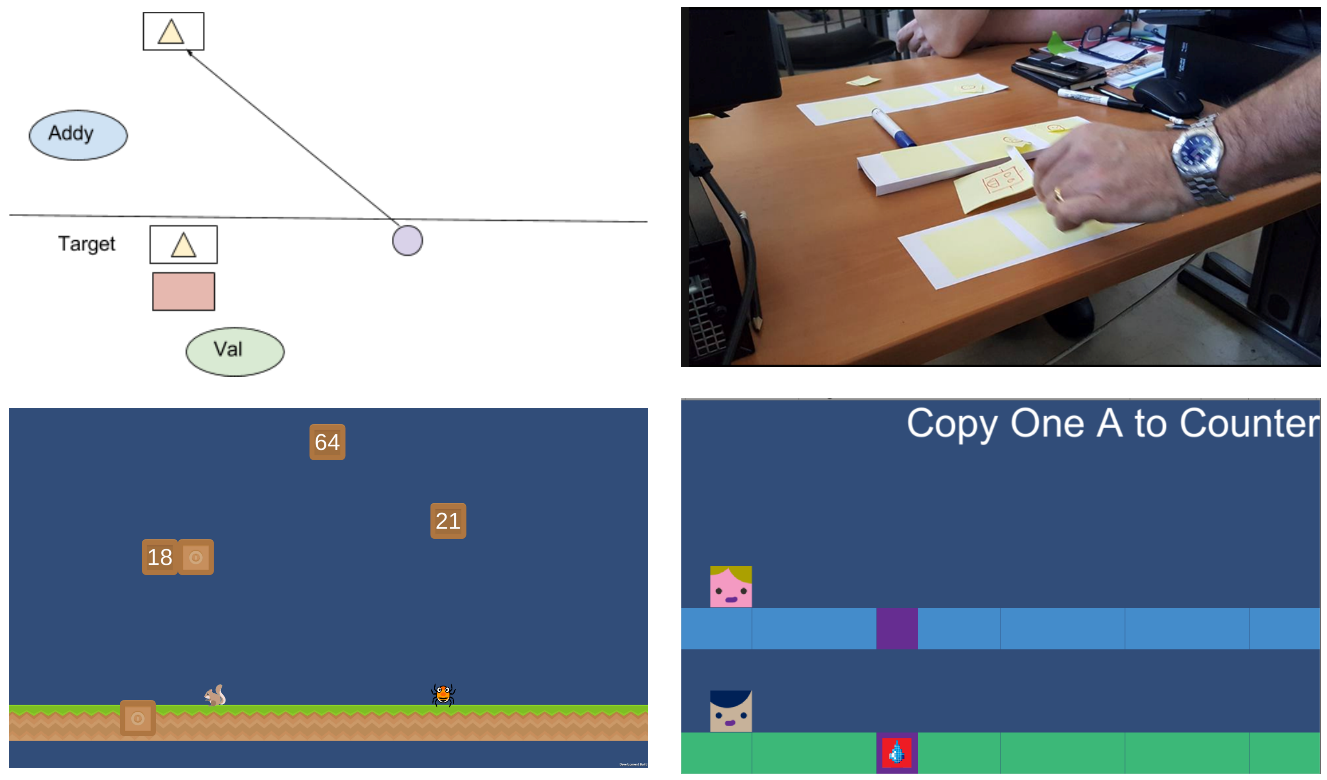


Figure 8: Prototyping a level in *If Memory Serves* including a softcopy mockup (top-left), a paper prototype (top-right), a ludic sketch (bottom-left), and an early digital prototype (bottom-right).

students presents an obvious challenge: What constitutes acceptable evidence of effectiveness? Different researchers will likely adopt different epistemological positions (i.e., the appropriateness of different methods for knowledge construction) and trends in knowledge construction can change as the paradigm of a particular research community shifts [63].

Within the SIGCSE community, researchers have levied many criticisms of the types of work being disseminated at its conferences as well as the methodologies utilized to make knowledge claims. Notable examples include Valentine's criticisms of *marco polo* papers and Randolph's review of the validity of experimental work being conducted [95, 96]. In more recent years, Tew & Dorn raise concern over the validity of measurement techniques [113] (also see Newton & Shaw's treatise on the topic [80]) as well as increased attention towards the use of rigorously validated concept inventories [112] (perhaps inspired by similar work on the development and evaluation of peer instruction in physics [38, 57]).

These criticisms are further compounded by a seeming lack of clarity on the minimally acceptable evidence as well as reference to prior work needed to make a knowledge claim within the SIGCSE community [65, 86], perhaps, driven through a lack of consensus on particular sub-fields. This means that there is no single dominating approach. While this means a plurality of work is being conducted, some of which make use of unusual yet enlightening approaches, it makes it challenging to situate a validation framework in the context of the existing literature.

Another complication, to this end, is the varied and disparate theoretical underpinning for much of the work in computer science education research [69]. Malmi et al. found that about half of the work published between 2005 and 2011 in prominent journals and research-focused conferences did not build on previous theoretical work and some were instead forming their own theoretical constructions. As these theories develop, they are likely going to have future impacts on any validation framework proposed. Particularly, as the nuances and peculiarities of teaching computer science topics becomes better understood as these could reveal new factors that educators need to consider when evaluating games for computing education.

It will also be important to consider not only *quantitative methods* in any validation framework proposed, but also *qualitative methods*, which have much to offer in terms of understanding impacts (rather than just detecting them) and relating them to the field's theory development [14, 56].

Despite these challenges, Pears et al. [87] note that there is a rich history of tool-development and evaluation in the field, of which, games should be considered a part—as learning tools. In particular, the development of tools to help students develop their programming skills forms a prominent part of the available literature. It is, perhaps, surprising not to see a dominant approach emerging. However, even within educational games as a sub-area of tool development, there is little evidence of consensus. Colleagues from across the Engineering Education community, particularly those with an interest in software engineering, have been facing similar challenges: Giessen conducts an overview of the effects of serious games, noting many cases of bad practice in methodology and a general lack of rigor [47]. Again, with respect to measuring quality in games, All notes a lack of good practice [8]. Petri et al. [89, 90] note that many attempts to evaluate games assume an *ad hoc* approach: "not [using] any well-defined model or method to conduct the evaluation of the educational game" going on to state that "evaluations are reported in an informal ad-hoc way, not providing an explicit definition of the evaluation objective, measures or data collection instruments."

It is clear that there are substantial differences in the ways researchers are approaching the evaluation of digital games and a lack of convergence on a particular set of methods suitable in evaluating games designed for computing education. Furthermore, existing methods are being placed under considerable scrutiny. While it is hoped that within this particular sub-field, game development for computer science education, the call to improve rigor will be heeded, the framework proposed in the next section is not intended to be a comprehensive or one-size-fits-all. Merely, it serves as an example of one such framework that addresses some of the challenges and pitfalls highlighted by other researchers doing work in this field.

4.3 Design Science Approach

Paving the way towards a rigorous model to evaluate games in the computer science context, the working group reviewed the criticism described in the previous section and, based on a synthesis of its insights and recommendations, propose a research design that would be suitable for the evaluation of *If Memory Serves* and other digital games being used in computing education by forming a validation framework. Of course, this is not the only approach to evaluating games, and will not claim to be comprehensive, but the group has aligned it with all of the NSF ISE impact categories [42] and the associated hypotheses outlined in the introduction.

The proposed approach borrows much from the school of design science [25, 37]. As noted in [37]:

> Design Science is "tough, analytic, partly formalizable, partly empirical, teachable doctrine." (Simon 1996, p.113). it formalizes the logic of design by prescribing statements of how artifacts "should become" (Pries-Heje and Baskerville 2008). The primary contribution of design science to knowledge is reflected in what the literature calls design principles which are rules that guide designers during their design of a certain class of artifacts (Hevner and Chatterjee 2010). The truthfulness of design principles can be evaluated using prototypes of the intended artifacts (Hevner and Chatterjee 2010).

The Double Diamond approach to artifact development is proposed as a general model by the British Design Council [29] and, broadly speaking, reflects some general practices in the games industry where developers tend to commence projects with a phase of pre-production and then commence production as a second stage [22] (see Figure 9).

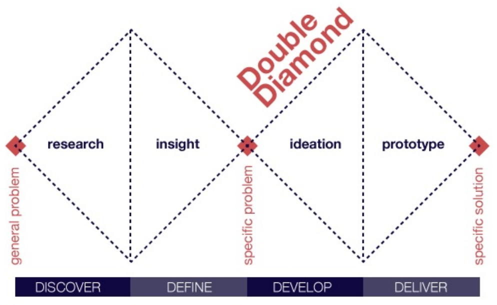


Figure 9: Double Diamond for Design [21]

To extend beyond this model of tool development into a model of tool development and evaluation, an additional phase of testing—the element of research—and a feedback loop. Thus, conducting more research to generate insight that is used to improve the prototype. This model corresponds to the approach described in [88]; however, we propose greater emphasis be made on agility (of the iteration) and a hypo-deductive empirical approaches to the research phase.

4.4 Research Model

The gold standard for validating whether a game-based learning intervention is effective is the randomized control trial (RCT) [121]. However, only 30 of 272 evaluations (~11%) identified in Connolly et al.'s systematic reviews used this approach [18, 28]. Further, only 19 of the 117 evaluations (~16%) that Petri and Wangenheim [89] had identified used an experimental design.

The validation framework, consequently, encourages the approach. It presents a fertile field of opportunities to verify prior work, establish effect sizes of game-based interventions in controlled conditions, and promote further insight generation.

Mahoney [67] serves as a convenient reference on different experimental designs. The simplest design for our context is the control group pre-test, post-test design (number 9, in table 1 on page 667). However, given the range of dependent variables, a slightly more complex design involving a delayed post-test is recommended.

The stages are as follows:

- Conduct a power analysis to determine minimum viable sample size given the hypothesized effect size (min. $d = 0.4$ [55])
- Recruit participants
- Capture demographic data
  - If available, obtain data on prior skill from relevant prior assessments
- Randomly allocate participants to two groups
  - If necessary, use a stratified random sample based on demographic data (e.g., gender, prior experience, etc.)
  - Ensure that participants are blinded to which experimental condition they are assigned to
- Capture knowledge of participants using the concept inventory
- Capture attitude of participants using a questionnaire
- Conduct experiment - intervention group and control group

Table 6: Extract of Supplementary Measurement Instruments (From [omit])

Measure	Available From
EGAMEFLOW	[43]
MEEGA+	[91]
PANAS	[119]
Game Usability Heuristic	[93]
SUS	[19]
TAM	[122]

  - Use tools such as OBS Studio to capture on-screen behavior
- Again, capture knowledge of participants using the concept inventory
- Again, capture attitude of participants using a questionnaire
- Interview participants following the mind-tape protocol (or similar)
  - Make use of footage captured by OBS studio
- Capture engagement of participants using self-report questionnaire
- Wait
- Again, capture knowledge of participants using the concept inventory
- Again, capture attitude of participants using a questionnaire
- Obtain data on skill from post-experiment assessments

At the first stage, we recommend keeping things as simple as possible: a game-only solo play group is appropriate to inform design until a more mature game is available. After conducting lab experiments, a follow-on pragmatic trial is appropriate (i.e., in a more natural setting to enhance external validity, like in studies of therapies once deployed in practice; c.f. [66]). We also note that care must be taken to ensure that both groups have the same amount of time on the task(s). If the experimental group has more time on task, we would expect the outcomes to be higher with or without a game.

4.5 Measures

Referring back to our original five hypothesis (see section 4.1), we propose a set of instruments to be used to measure each.

4.5.1 Knowledge.

4.5.1 Knowledge. The Knowledge hypothesis states:

> *Students playing* If Memory Serves *in tandem with traditional learning activities (e.g., readings, lecture, etc.) will perform better on a test of conceptual knowledge on pointers (i.e., purpose, usage and syntax, tracing and prediction, debugging, etc.) than those students who only participate in the traditional learning activities.*

This dependent variable will be measured through a custom-tailored concept inventory developed by the working group (see Appendix).

We reviewed several computer science assessment instruments, and one we found useful was the Foundational CS1 assessment instrument (FCS1) by Tew and Guzdial [1, 114]. From the descriptions provided for FCS1, we established three goals:

1. All items in the instrument should be multiple choice,
2. All items in the instrument should cover all learning outcomes, and
3. The instrument should contain three different types of questions:
   (a) *Definition questions:* In FCS1, the items seem to be vocabulary based and are intended to measure conceptual understanding. Using vocabulary questions with a pre- and post-test is difficult without repeating the questions, so the initial version of our instrument contains vocabulary questions that use code (i.e. which line of code has a variable dereference). This provides the capability for the researcher to change the code a different question that measures similar outcomes.
   (b) *Tracing questions:* These questions will gauge students' knowledge about what a given block of code will produce when it is executed. These are used to evaluate a student's ability to read code. We also included 'none of the above' answers and code with errors to create questions with common errors in an attempt to differentiate between pointer vs. value learning outcomes.
   (c) *Writing questions:* These questions will gauge the students' knowledge to identify the missing code that will correctly complete the program. These are to evaluate students' abilities to write code with pointers, while still using a multiple choice format.

The measurement instrument includes questions for each of the learning outcomes. Though the instrument was initially created in C++, it does not need to be restricted to it. In addition to refining the questions and piloting them with students, future considerations include:

- Would it make more sense to make a second C version or try to create questions that use pseudocode?
- Are there any other languages that use pointers that would be worth supporting?
- How to represent pointers in pseudocode in a manner that students understand?

4.5.2 Engagement. The Engagement hypothesis states:

> *Students playing* If Memory Serves *in tandem with traditional learning activities (e.g., readings, lecture, etc.) will report a higher level of engagement with their learning activity on pointers than those students who only participate in the traditional learning activities.*

This dependent variable will be measured through two approaches: (i) a self-report at the post-test; and (ii) measuring time-on-task from data obtained within the context of the experiment (i.e., screen capture and in-session metrics). This will permit triangulation on perception of interest while also afford opportunities to contextualize. The group propose the use of Keller's ARCS learning motivation model, which is popular in the examination of educational technology more generally. This has been incorporated into the measurement instrument validated and made available by Huang et al. [58].

4.5.3 Attitude. The Attitude hypothesis states:

> *Students playing* If Memory Serves *in tandem with traditional learning activities (e.g., readings, lecture, etc.) will report greater confidence and positivity about their mastery of pointers than those students who only participate in the traditional learning activities.*

This dependent variable will be measured at pre-test and post-test by self-report using an appropriate questionnaire. Depending on the context in which pointers are taught, this might include adaptations to existing instruments designed and validated to measure computing attitudes [35, 105]. Where appropriate, other supplementary measures could be used.

4.5.4 Behavior. The Behavior hypothesis states:

> *Students playing* If Memory Serves *in tandem with traditional learning activities (e.g., readings, lecture, etc.) will be observed to spend more time-on-task and apply different learning strategies than those students who only participate in the traditional learning activities.*

There are two aspects of this dependent variable which are of interest to the working group: (i) approach-to-task (related to engagement and time-on-task, but exploring the actual behaviors); and (ii) differences in the behaviors exhibited by participants between each of the experimental conditions. This measure is primarily qualitative, but depending on how data are processed and analyzed and presented could be quantized. Such qualitative data can also be incredibly useful in deriving new insights into a design, and helping to identify new things to model and predict in future experiments, but potentially forming recommendations for future developers—one of the aims of design research.

This data is collected via screen capture, perhaps using a tool such as OBS studio, and a follow-up interview conducted after the experiment. During the interview, a variation of the think-aloud protocol called *MindTape* [82, 83] is suggested (sometimes called a cognitive walk-through). This is a retrospective reflection approach, in which the screen capture is used as a guide for the interviewer and a memory prompt for the participant. This aids with clarifications of why particular behaviors may have been adopted, while aiding in the rejection of feasible, yet incorrect interpretations of behaviors and why they occurred. More sophisticated designs augment this by using in-session metrics or props (e.g. SubtleStone [48]) to draw attention to particular segments of a session.

4.5.5 Skill. The Skill hypothesis states:

> *Students playing* If Memory Serves *in tandem with traditional learning activities (e.g., readings, lecture, etc.) will demonstrate more apt and coherent use of pointers in a programming assignment (measured by functional coherence, source code sophistication, and source code maintainability) than those students who only participate in the traditional learning activities.*

Measuring this dependent variable can be challenging because the learning objectives of programs within and between institutions may differ. As such, the domain within which and/or the way in which a skill is applied may require a slightly different measure. It is known that there are differences in the approaches used to assess the application of knowledge in programming (see [40, 106, 109, 110], etc.).

It is, therefore, difficult to prescribe any single validated instrument or approach to measurement. Furthermore, caution is advised. It is important that any measure of skill is at least face-valid and self-consistent within the study. A typical approach may involve, in the first instance, the use of a combination of self-reported prior experience alongside evidence drawn from relevant prior assignment submissions. It is almost universally the case, per the principle of constructive alignment [15], that an assignment and/or exam will measure students' mastery of any particular point in a program. Thus, examining confers a suitable opportunity to capture skill at a later stage as a delayed post-test.

4.6 Data Analysis

The validation framework does not prescribe any particular method of analysis. However, as a suggestion for which statistical analysis techniques to apply to quantitative data, Analysis of Covariance (ANCOVA) is broadly considered appropriate for comparing data created through experimental trials. This is particularly the case when pre-test data is available [31]. For post-test only measures then a Wilcoxon-Mann-Whitney test or, if the data is normal, a t-test is appropriate [39].

Of note, however, is error inflation. This can be adequately accounted for using the Benjamini-Hotchberg correction [12]. This correction may increase sample size requirements.

We also note the importance of collecting adequate demographic information, including year in school, age, gender, and amount of time engaged in game. Since this is a collaborative game, being able to identify those participants who were paired for the game may also be important.

4.7 Ethics and Approval Processes

Full validation of educational games requires access to often confidential participant data (demographic information, educational task performance, academic grades etc), while participants are also required to complete various tasks that may then impact (either positively or negatively) their learning performance. All of these raise ethical issues which are governed by national and international laws and are set out in research ethics codes of conduct such as EU Ethics for Research document and the British Educational Research Association Ethics Guidelines. In Australia, the Australian Code for the Responsible Conduct of Research and the National Statement on Ethical Conduct in Human Research (2007) provide a legal framework govern the conduct of research with Human participants. In the European Union, all personal data is also subject to relevant data protection laws. These frameworks require universities and research institutions to have an ethics approval process in place to ensure only ethical research will be carried out.

The educational research required to validate educational games are generally not considered to be of high risk to participants in comparison to aggressive interventions in medical fields; however, research in the educational setting can also be high risk when the participants are from vulnerable groups, such as children, are unable to give informed consent, highly confidential information is accessed, there is a pre-existing and/or current teacher-student relationship with the researcher etc. In the case when educational research is high risk, these risks can be mitigated with relevant controls such as using participants that can give informed consent, de-identification of data and administering surveys outside of the teaching period.

Although many universities have a requirement for ethics approval to be granted before the research can commence, in cases of multi-institutional collaborations, normally one institution will submit an ethics application to that institution, which if approved will be then lodged with the other partner institutions.

5 LIMITATIONS AND FUTURE DIRECTIONS

We provide here a summary of the limitations of this study as well as a list of future directions and further call to action.

5.1 Limitations

Though no different than other working groups, our working group researched and developed game mechanics prior to ITiCSE 2017, developed the game prototype during ITiCSE 2017, and have continued to user test and evolve the game since meeting. In addition, we have developed the concept inventory and guide to evaluating the game. However, as with any active research project, there are several limitations. One major limitation is that the game is still a work in progress. Despite having a prototype of the game, the four days that the group convened during ITiCSE was insufficient time to review past limitations of games for computing education and create a completed game that meets these limitations. Our early discussions on days 1 and 2 focused primarily on design, choosing the areas of learning for the game, and defining the learning outcomes. Day 2 led to active discussions about game representation and how to choose a design that it appealing to a wide variety of demographics while remaining culturally relevant for undergraduate students. Days 3 and 4 were spent developing the game mechanics and an initial user interface, while refining how the game would play through its levels. Day 5 (the last day) was spent pulling together an initial draft of this report.

The Concept Inventory (presented in the Appendix) will be used and changed as needed based on piloting of the inventory and feedback from other instructors. We recognize that it is a starting point for evaluating the game in the future.

Given the ambitiousness of this project, ITiCSE provided us with a major step forward in providing a solid foundation for the game. However, we did not have the time nor access to a set of students to receive approval from an ethics board to conduct testing on any significant scale. Knowing this, we instead opted to create a framework for conducting this stage of measuring effectiveness of the tool, including the development of a concept inventory for pointers. This work remains to be completed and is discussed more in the next section.

5.2 Future Directions

This working group has initiated a game development and validation methodology that could be further developed and enhanced through national and international collaborations. Specifically, for *If Memory Serves*, we invite the community to consider modding the game in any of the following ways:

- Adapt and modify levels
- Adapt and modify art
- Adapt and modify narrative
- Integrate automated tutorial system
- Integrate achievements (e.g. badges)
- Conduct player testing
- Enhance the player experience
- Adapt the game for mobile devices
- Interpretive guides for instructors to enable them to easily contextualize the game with how pointers are used in their code

We invite the community to consider using the evaluation methods of the game as provided or changing/adding to these methods, including (but not limited to) the following:

- Provide additional resources for testing, including IRB materials
- Provide additional survey instruments for testing
- Provide methodology for conducting qualitative studies of effectiveness
- Provide an on-line system for collecting data on players
- Integrate data collection of game metrics into the game and provide an automated method for reporting those metrics

Neither of these are comprehensive lists, but they provide some direction into areas where we were unable to complete during the working group meeting. If you are able to collaborate or contribute to the development or evaluation of *If Memory Serves*, contact our working group for more information on resources and tools, including source code and survey tools.

A by-product of this work has the potential of being a repository of resources for educational research in educational games for computer science [104].

Clear areas for development include expansion of the current games platform including open-source tools chains. With open licensing it is likely that this platform will develop quickly and be available for educators and researcher around the world to use. With this expanded international community there will be further opportunities for broader-base testing and analysis of games based education in computer science.

6 CONCLUSIONS

This working group has developed a preliminary games education platform and a case study, *If Memory Serves*, that illustrates how a learning outcomes based approach can be used to facilitate educational games development within computer science. Games created for computer science should map to broader learning goals in common standards, such as the ACM/IEEE Computing Curricula, as well as more specific goals related to the subject(s) being taught or reinforced.

In addition to the more specific items above that relate to *If Memory Serves*, there are many further opportunities for collaboration and future work in the area of games for computing education, including:

- Psychological effects
- Psychometrics via game narratives and response systems
- Data mining for misconceptions or indicators of early-attrition
- Process analysis versus product analysis
- Learning analytics
- Learning visualization
- Motivational meta-game

Each of these areas represent a significant sub-discipline that has the potential to significantly influence the effectiveness of educational games as an instrument for supporting learning in the new generation of students.

REFERENCES

[1] db-SERC: Assessments - Computer Science. http://dbserc.pitt.edu/Assessment/Assessments-Computer-Science. (????). Accessed: 2017-08-15.

[2] 2011. Personifying programming tool feedback improves novice programmersfi learning. (2011), 109–fi?!116.

[3] 2012. Investigating the role of purposeful goals on novicesfi engagement in a programming game. (2012), 163–fi?!166.

[4] Clark C Abt. 1987. *Serious games.* University press of America.

[5] ACM/IEEE. 2013. *CS Joint Task Force on Computing Curricula. 2013.* Computer Science Curricula ACM Press and IEEE Computer Society Press.

[6] Deanne M Adams, Richard E Mayer, Andrew MacNamara, Alan Koenig, and Richard Wainess. 2012. Narrative games for learning: Testing the discovery and narrative hypotheses. *Journal of educational psychology* 104, 1 (2012), 235.

[7] Bruce Adcock, Paolo Bucci, Wayne D. Heym, Joseph E. Hollingsworth, Timothy Long, and Bruce W. Weide. 2007. Which Pointer Errors Do Students Make?. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07).* ACM, New York, NY, USA, 9–13. DOI: http://dx.doi.org/10.1145/1227310.1227317

[8] Anissa All, Elena Patricia Nunez Castellar, and Jan Van Looy. 2014. Measuring effectiveness in digital game-based learning: a methodological review. *International Journal of Serious Games* 2, 1 (2014), 3–20.

[9] Anthony Allevato, Stephen H. Edwards, and Manuel A. Pérez-Quiñones. 2009. Dereferee: Exploring Pointer Mismanagement in Student Code. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09).* ACM, New York, NY, USA, 173–177. DOI: http://dx.doi.org/10.1145/1508865.1508928

[10] Tiffany Barnes, H Richter, A Chaffin, A Godwin, E Powell, T Ralph, P Matthews, and H Jordan. 2007. The role of feedback in Game2Learn. In *CHI*, Vol. 2007. 1–5.

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

[12] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)* (1995), 289–300.

[13] Jens Bennedsen and Michael E Caspersen. 2007. Failure rates in introductory programming. *ACM SIGCSE Bulletin* 39, 2 (2007), 32–36.

[14] Anders Berglund, Mats Daniels, and Arnold Pears. 2006. Qualitative research projects in computing education research: an overview. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52.* Australian Computer Society, Inc., 25–33.

[15] John Biggs. 1996. Enhancing teaching through constructive alignment. *Higher education* 32, 3 (1996), 347–364.

[16] Paul Black and Dylan Wiliam. 2010. Inside the black box: Raising standards through classroom assessment. *Phi Delta Kappan* 92, 1 (2010), 81–90.

[17] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, Kate Sanders, and Carol Zander. 2007. Threshold Concepts in Computer Science: Do They Exist and Are They Useful?. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07).* ACM, New York, NY, USA, 504–508. DOI: http://dx.doi.org/10.1145/1227310.1227482

[18] Elizabeth A. Boyle, Thomas Hainey, Thomas M. Connolly, Grant Gray, Jeffrey Earp, Michela Ott, Theodore Lim, Manuel Ninaus, Claudia Ribeiro, and Joo Pereira. 2016. An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. *Computers & Education* 94 (2016), 178 – 192. DOI: http://dx.doi.org/10.1016/j.compedu.2015.11.003

[19] John Brooke and others. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[20] Tracy Camp. 1997. The incredible shrinking pipeline. *Commun. ACM* 40, 10 (1997), 103–110.

[21] MENA Design Research Centre. 2015. What Is Design Research? https://www.menadrc.org/research/. (2015). [Accessed 07-Sept-2017].

[22] Mark Cerny and Michael John. 2002. Game development. Myth vs. method. *Game Developer* (2002), 32–36.

[23] Richard E Clark. 1994. Media will never influence learning. *Educational technology research and development* 42, 2 (1994), 21–29.

[24] Richard E Clark, Kenneth Yates, Sean Early, Kathrine Moulton, KH Silber, and R Foshay. 2010. An analysis of the failure of electronic media and discovery-based learning: Evidence for the performance benefits of guided training methods. *Handbook of training and improving workplace performance* 1 (2010), 263–297.

[25] Allan Collins. 1992. Toward a design science of education. In *New directions in educational technology.* Springer, 15–22.

[26] Thomas Connolly, MH Stansfield, and Thomas Hainey. 2008. Development of a general framework for evaluating games-based learning. In *Proceedings of the 2nd European conference on games-based learning.* Universitat Oberta de Catalunya Barcelona, Spain, 105–114.

[27] Thomas Connolly, Mark Stansfield, and Thomas Hainey. 2009. Towards the development of a games-based learning evaluation framework. *Games-based learning advancements for multisensory human computer interfaces: Techniques and effective practices. Hershey PA: IGI Global* (2009).

[28] Thomas M Connolly, Elizabeth A Boyle, Ewan MacArthur, Thomas Hainey, and James M Boyle. 2012. A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education* 59, 2 (2012), 661–686.

[29] British Design Council. 2005. The Design Process: The 'Double Diamondfi Design Process Model. http://www.designcouncil.org.uk/about-design/how-designers-work/the-design-process/. (2005). [Accessed 03-July-2017].

[30] Michelle Craig and Andrew Petersen. 2016. Student Difficulties with Pointer Concepts in C. In *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '16).* ACM, New York, NY, USA, Article 8, 10 pages. DOI: http://dx.doi.org/10.1145/2843043.2843348

[31] Lee J Cronbach and Lita Furby. 1970. How we should measure" change": Or should we? *Psychological bulletin* 74, 1 (1970), 68.

[32] Sara De Freitas and Martin Oliver. 2006. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Computers & education* 46, 3 (2006), 249–264.

[33] Adrienne Decker and Elizabeth Lane Lawley. 2013. Life's a Game and the Game of Life: How Making a Game out of It Can Change Student Behavior. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13).* ACM, New York, NY, USA, 233–238. DOI: http://dx.doi.org/10.1145/2445196.2445269

[34] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11).* ACM, New York, NY, USA, 9–15. DOI: http://dx.doi.org/10.1145/2181037.2181040

[35] Brian Dorn and Allison Elliott Tew. 2013. Becoming Experts: Measuring Attitude Development in Introductory Computer Science. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13).* ACM, New York, NY, USA, 183–188. DOI: http://dx.doi.org/10.1145/2445196.2445252

[36] Brock Dubbels. Under Review. Serious Games Can Ensure Serious Learning. *Computers in Human Behaviour* (Under Review).

[37] Mazen El-Masri, Ali Tarhini, M Hassouna, and T Elyas. 2015. A Design Science Approach To Gamify Education: From Games To Platforms.. In *ECIS.*

[38] Adam P Fagen, Catherine H Crouch, and Eric Mazur. 2002. Peer instruction: Results from a range of classrooms. *The physics teacher* 40, 4 (2002), 206–209.

[39] Michael P Fay and Michael A Proschan. 2010. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys* 4 (2010), 1.

[40] Sue Fitzgerald, Brian Hanks, Raymond Lister, Renee McCauley, and Laurie Murphy. 2013. What are we thinking when we grade programs?. In *Proceeding of the 44th ACM technical symposium on Computer science education.* ACM, 471–476.

[41] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences* 111, 23 (2014), 8410–8415.

[42] Alan Friedman. 2008. Framework for evaluating impacts of informal science education projects. *Retrieved December* 8 (2008), 2008.

[43] Fong-Ling Fu, Rong-Chang Su, and Sheng-Chin Yu. 2009. EGameFlow: A Scale to Measure Learners' Enjoyment of e-Learning Games. *Comput. Educ.* 52, 1 (Jan. 2009), 101–112. DOI: http://dx.doi.org/10.1016/j.compedu.2008.07.004

[44] James P. Gee. 2000. Identity as an analytic lens for research in education. *Review of research in education* 25 (2000), 99–125.

[45] James P. Gee. 2003. What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE)* 1, 1 (2003), 20–20.

[46] Michail N Giannakos, Ilias O Pappas, Letizia Jaccheri, and Demetrios G Sampson. 2016. Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. *Education and Information Technologies* (2016), 1–18.

[47] Hans W Giessen. 2015. Serious games effects: an overview. *Procedia-Social and Behavioral Sciences* 174 (2015), 2240–2244.

[48] Judith Good, Jon Rimmer, Eric Harris, and Madeline Balaam. 2011. Self-reporting emotional experiences in computing lab sessions: An emotional regulation perspective. In *Proceedings of the 23rd Annual Psychology of Programming Interest Group Conference.*

[49] Neil Andrew Gordon. 2016. *Issues in retention and attainment in Computer Science.* Higher Education Academy.

[50] Cecilia M Gorriz and Claudia Medina. 2000. Engaging girls with computers through software games. *Commun. ACM* 43, 1 (2000), 42–49.

[51] Sigrun Gudmundsdottir and Lee Shulman. 1987. Pedagogical content knowledge in social studies. *Scandinavian Journal of Educationl Research* 31, 2 (1987), 59–70.

[52] Glenda A Gunter, Robert F Kenny, and Erik H Vick. 2008. Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games. *Educational technology research and Development* 56, 5 (2008), 511–537.

[53] MP Jacob Habgood, SE Ainsworth, and Steve Benford. 2005. Endogenous fantasy and learning in digital games. *Simulation & Gaming* 36, 4 (2005), 483–498.

[54] Nancy B Hastings and Monica W Tracey. 2004. Does media affect learning: where are we now? *TechTrends* 49, 2 (2004), 28–30.

[55] John Hattie. 2008. *Visible learning: A synthesis of over 800 meta-analyses relating to achievement.* Routledge.

[56] Orit Hazzan, Yael Dubinsky, Larisa Eidelman, Victoria Sakhnini, and Mariana Teif. 2006. Qualitative research in computer science education. In *Acm Sigcse Bulletin*, Vol. 38. ACM, 408–412.

[57] David Hestenes, Malcolm Wells, and Gregg Swackhamer. 1992. Force concept inventory. *The physics teacher* 30, 3 (1992), 141–158.

[58] Wenhao Huang, Wenyeh Huang, Heidi Diefes-Dux, and Peter K Imbrie. 2006. A preliminary validation of Attention, Relevance, Confidence and Satisfaction model-based Instructional Material Motivational Survey in a computer-based tutorial setting. *British Journal of Educational Technology* 37, 2 (2006), 243–259.

[59] Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, Vol. 4. 1722.

[60] Chris Johnson, Monica McGill, Durell Bouchard, Michael K Bradshaw, Víctor A Bucheli, Laurence D Merkle, Michael James Scott, Z Sweedyk, J Ángel, Zhiping Xiao, and others. 2016. Game Development for Computer Science Education. In *Proceedings of the 2016 ITiCSE Working Group Reports.* ACM, 23–44.

[61] Slava Kalyuga and Jan L Plass. 2009. Evaluating and managing cognitive load in games. In *Handbook of research on effective electronic gaming in education.* IGI Global, 719–737.

[62] Robert B Kozma. 1994. Will media influence learning? Reframing the debate. *Educational technology research and development* 42, 2 (1994), 7–19.

[63] Thomas S Kuhn. 1970. *The Structure of Scientific Revolutions, 2nd enl. ed.* University of Chicago Press.

[64] MR Lepper, TW Malone, RE Snow, and MJ Farr. 1987. Aptitude, learning, and instruction: III. Conative and affective process analyses. (1987).

[65] Raymond Lister and Ilona Box. 2008. A citation analysis of the SIGCSE 2007 proceedings. *ACM SIGCSE Bulletin* 40, 1 (2008), 476–480.

[66] Hugh MacPherson. 2004. Pragmatic clinical trials. *Complementary therapies in medicine* 12, 2 (2004), 136–140.

[67] Michael J Mahoney. 1978. Experimental methods and outcome evaluation. *Journal of Consulting and Clinical Psychology* 46, 4 (1978), 660.

[68] Christos Malliarakis, Maya Satratzemi, and Stelios Xinogalos. 2014. Designing Educational Games for Computer Programming: A Holistic Framework. *Electronic Journal of e-Learning* 12, 3 (2014), 281–298.

[69] Lauri Malmi, Judy Sheard, Roman Bednarik, Juha Helminen, Päivi Kinnunen, Ari Korhonen, Niko Myller, Juha Sorva, Ahmad Taherkhani, and others. 2014. Theoretical underpinnings of computing education research: what is the evidence?. In *Proceedings of the tenth annual conference on International computing education research.* ACM, 27–34.

[70] Richard E Mayer. 2004. Should there be a three-strikes rule against pure discovery learning? *American psychologist* 59, 1 (2004), 14.

[71] Richard E Mayer. 2005. *The Cambridge handbook of multimedia learning.* Cambridge university press.

[72] Richard E Mayer. 2009. *Multimedia learning (2nd).* Cambridge University Press New York.

[73] Richard E Mayer and Cheryl I Johnson. 2010. Adding instructional features that promote learning in a game-like environment. *Journal of Educational Computing Research* 42, 3 (2010), 241–265.

[74] Richard E Mayer and Roxana Moreno. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist* 38, 1 (2003), 43–52.

[75] Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49, 8 (2006), 90–95.

[76] Jan HF Meyer and Ray Land. 2005. Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher education* 49, 3 (2005), 373–388.

[77] Punya Mishra and Matthew J Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record* 108, 6 (2006), 1017.

[78] Roxana Moreno. 2004. Decreasing cognitive load for novice students: Effects of explanatory versus corrective feedback in discovery-based multimedia. *Instructional science* 32, 1 (2004), 99–113.

[79] Roxana Moreno and Richard Mayer. 2007. Interactive multimodal learning environments. *Educational psychology review* 19, 3 (2007), 309–326.

[80] Paul Newton and Stuart Shaw. 2014. *Validity in educational and psychological assessment.* Sage.

[81] David J Nicol and Debra Macfarlane-Dick. 2006. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education* 31, 2 (2006), 199–218.

[82] Janni Nielsen and Nina Christiansen. 2000. Mindtape: a tool for reflection in participatory design. In *PDC.* 309–313.

[83] Janni Nielsen, Torkil Clemmensen, and Carsten Yssing. 2002. Getting access to what goes on in people's heads?: reflections on the think-aloud technique. In *Proceedings of the second Nordic conference on Human-computer interaction.* ACM, 101–110.

[84] Harry O Neil and R. S Perez. 2008. *Computer games and team and individual learning.* Elsevier.

[85] Marina Papastergiou. 2009. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education* 52, 1 (2009), 1–12.

[86] Arnold Pears, Stephen Seidman, Crystal Eney, Päivi Kinnunen, and Lauri Malmi. 2005. Constructing a core literature for computing education research. *ACM SIGCSE Bulletin* 37, 4 (2005), 152–161.

[87] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin* 39, 4 (2007), 204–223.

[88] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A design science research methodology for information systems research. *Journal of management information systems* 24, 3 (2007), 45–77.

[89] Giani Petri and Christiane Gresse von Wangenheim. 2016. How to Evaluate Educational Games: a Systematic Literature Review. *Journal of Universal Computer Science* 22, 7 (2016), 992–1021.

[90] Giani Petri and Christiane Gresse von Wangenheim. 2017. How games for computing education are evaluated? A systematic literature review. *Computers & Education* 107 (2017), 68–90.

[91] Giani Petri, C Gresse von Wangenheim, and Adriano Ferretti Borgatto. 2016. MEEGA+: an evolution of a model for the evaluation of educational games. *INCoD/GQS* 3 (2016).

[92] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. 2017. A large-scale evaluation of a model for the evaluation of games for teaching software engineering. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track.* IEEE Press, 180–189.

[93] David Pinelle, Nelson Wong, and Tadeusz Stach. 2008. Heuristic evaluation for games: usability principles for video game design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 1453–1462.

[94] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. 2013. Success in introductory programming: What works? *Commun. ACM* 56, 8 (2013), 34–36.

[95] Justus Randolph, George Julnes, Erkki Sutinen, and Steve Lehman. 2008. A Methodological Review of Computer Science Education Research. *Journal of Information Technology Education* 7 (2008).

[96] Justus J Randolph. 2008. *Multidisciplinary methods in educational technology research and development.* HAMK Press/Justus Randolph.

[97] Mark R Rank, Hong-Sik Yoon, and Thomas A Hirschl. 2003. American poverty as a structural failing: Evidence and arguments. *J. Soc. & Soc. Welfare* 30 (2003), 3.

[98] Carriann E Richey Smith, Priscilla Ryder, Ann Bilodeau, and Michele Schultz. 2016. Use of an Online Game to Evaluate Health Professions Studentsfi Attitudes toward People in Poverty. *American journal of pharmaceutical education* 80, 8 (2016), 139.

[99] Lloyd P Rieber. 1996. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational technology research and development* 44, 2 (1996), 43–58.

[100] Gina Roussos and John F Dovidio. 2016. Playing below the poverty line: Investigating an online game as a way to reduce prejudice toward the poor. *Cyberpsychology: Journal of Psychosocial Research on Cyberspace* 10, 2 (2016).

[101] Rigby C.S. Ryan, R.M. and A. Przybylski. 2006. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion* 30, 4

(2006), 344–360.

[102] Rafael Savi, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. 2011. A model for the evaluation of educational games for teaching software engineering. In *Software Engineering (SBES), 2011 25th Brazilian Symposium on*. IEEE, 194–203.

[103] R Keith Sawyer. 2005. *The Cambridge handbook of the learning sciences*. Cambridge University Press.

[104] Michael James Scott and Gheorghita Ghinea. 2013. Integrating fantasy role-play into the programming lab: exploring the'projective identity'hypothesis. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 119–122.

[105] Michael James Scott and Gheorghita Ghinea. 2014. Measuring enrichment: the assembly and validation of an instrument to assess student self-beliefs in CS1. In *Proceedings of the tenth annual conference on International computing education research*. ACM, 123–130.

[106] Michael James Scott and Gheorghita Ghinea. 2015. Reliability in the assessment of program quality by teaching assistants during code reviews. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 346–346.

[107] Sharon A Shrock. 1994. The media influence debate: Read the fine print, but don't lose sight of the big picture. *Educational Technology Research and Development* 42, 2 (1994), 49–53.

[108] HA Spires, KA Turner, J Rowe, B Mott, and J Lester. 2010. Game-based literacies and learning: Towards a transactional theoretical perspective. *American Educational Research Association (AERA), Denver, CO* (2010).

[109] Martijn Stegeman, Erik Barendsen, and Sjaak Smetsers. 2014. Towards an empirically validated model for assessment of code quality. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. ACM, 99–108.

[110] Martijn Stegeman, Erik Barendsen, and Sjaak Smetsers. 2016. Designing a rubric for feedback on code quality in programming courses. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. ACM, 160–164.

[111] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312.

[112] Cynthia Taylor, Daniel Zingaro, Leo Porter, Kevin C Webb, Cynthia Bailey Lee, and M Clancy. 2014. Computer science concept inventories: past and future. *Computer Science Education* 24, 4 (2014), 253–276.

[113] Allison Elliott Tew and Brian Dorn. 2013. The case for validated tools in computer science education research. *Computer* 46, 9 (2013), 60–66.

[114] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 111–116. DOI: http://dx.doi.org/10.1145/1953163.1953200

[115] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research*. ACM, 19–26.

[116] Christiane Gresse Von Wangenheim and Forrest Shull. 2009. To game or not to game? *IEEE software* 26, 2 (2009), 92–94.

[117] Lev S Vygotsky. 1978. *Mind in society: The development of higher mental process*. Cambridge, MA: Harvard University Press.

[118] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 39–44.

[119] David Watson, Lee A Clark, and Auke Tellegen. 1988. Development and validation of brief measures of positive and negative affect: the PANAS scales. *Journal of personality and social psychology* 54, 6 (1988), 1063.

[120] Nicola Whitton. 2009. *Learning with digital games: A practical guide to engaging students in higher education*. Routledge.

[121] Lisa Woolfson. 2011. *Educational psychology: The impact of psychological research on education*. Pearson Education.

[122] Amri Yusoff, Richard Crowder, and Lester Gilbert. 2010. Validation of serious games attributes using the technology acceptance model. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*. IEEE, 45–51.

A POINTER CONCEPT INVENTORY

This concept inventory has been designed to align with the learning objectives presented in each level of the game. Items below are presented according to learning outcomes for outcomes A-E. Items have yet not been developed for Outcomes F, G, and H. We have also skipped Outcomes C, since it doesn't seem to be fundamental to pointers. The legend for the type of question posed is C for Concepts/Definition Questions, T for Tracing, and W for Writing.

For code tracing questions, we note that all of these questions assume knowledge of iostream, using namespace std, and code inside a main function. Students may think that the entire quiz is nothing but "gotchas"and answer none of the above on every question. We could make all of the questions more verbose by including the elided code, or just assume none one would make that assumption.

We also note that Learning Outcome C does not seem to be fundamental to pointers, and therefore no items are included in this version of the assessment. Since this is a preliminary version of this concept inventory, it has not yet been piloted or validated.

A.1 Learning Outcome A: Value Basics, No Pointers

These questions are fundamentally basic. We include them here, since we have a learning outcome for values and they might be useful by providing a mechanism to remove a participant from the study. (If they can't answer these questions correctly then they can't be expected to learn pointers.)

(1) (C) For the following snippet of code, select all statements that are true about line 3.

```
1 int a = 1;
2 int b = 2;
3 a = b + 1;
4 cout << a << " " << b;
```

(a) line 3 will change the value of the variable a when executed (*)
(b) line 3 will change the value of the variable b when executed
(c) line 3 will evaluate to false because 1 does not equal 3
(d) line 3 will prevent the program from compiling because 1 does not equal 3

(2) (C) For the following snippet of code, which lines will produce compile-time errors?

```
1 int a = 1;
2 int b;
3 char c = 'A';
4 char d;
5 a = b;
6 a = c;
7 b = d;
8 d = c;
```

(3) (T) What would the following snippet of code print when run?

```
1 char a;
2 char b;
3 a = 'B';
4 b = a;
5 a = 'A';
6 cout << a << " " << b;
```

(a) A A
(b) A B (*)
(c) B A
(d) B B
(e) None of the above

(4) (T) What would the following snippet of code print when run?

```
char a[2];
char b;
a[0] = 'A';
a[1] = 'B';
a[0]= a[1];
a[1] = a[0];
cout << a[0] << " " << a[1];
```

(a) A A
(b) A B (*)
(c) B A
(d) B B
(e) None of the above

(5) (W) For the following snippet of code,

```
int a;
cin >> a;
cout << a;
```

which of the following lines, if placed in line 3, would create a program that prints the number 1 greater than what the user enters?
(a) `a + 1;`
(b) `a = a + 1;` (*)
(c) `a++;` (*)

A.2 Learning Outcome B: Pointer Basics

(1) (C) For the following snippet of code, match the line number to the description of the operation performed on that line.

```
1 char a;
2 char *p;
3 a = 'A';
4 p = &a;
5 cout << *p;
```

(a) pointer declaration (2)
(b) pointer dereference (5)
(c) pointer reference (4)

(2) (T) What would the following snippet of code print when run?

```
char *p;
char *q;
p = new char('B');
q = p;
p = new char('A');
cout << *p << " " << *q;
```

(a) A A
(b) A B (*)
(c) B A
(d) B B
(e) None of the above

(3) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char('A');
a = *p;
p = new char('B');
cout << *p << " " << a;
```

(a) A A
(b) A B
(c) B A (*)
(d) B B
(e) None of the above

(4) (T) What would the following snippet of code print when run?

```
char a;
char *p;
a = 'A';
p = &a;
a = 'B';
cout << a << " " << *p;
```

(a) A A
(b) A B
(c) B A
(d) B B (*)
(e) None of the above

(5) (T) What would the following snippet of code print when run?

```
char a;
char *p;
a = 'A';
p = new char('B');
*p = a;
cout << a << " " << *p;
```

(a) A A (*)
(b) A B
(c) B A
(d) B B
(e) None of the above

(6) (T) What would the following snippet of code print when run?

```
char *p;
char *q;
p = new char('A');
q = new char('B');
*q = *p;
cout << *p << " " << *q;
```

(a) A A (*)
(b) A B
(c) B A
(d) B B
(e) None of the above

(7) (W) The following snippet of code is incomplete:

```
char *p;
char *q;
p = new char('A');
q = XXX;
cout << *q;
```

Which of the following in place of XXX will complete the program so that it prints A when run.
(a) `p` (*)
(b) `*p`
(c) `&p`
(d) None of the above

(8) (W) The following snippet of code is incomplete:

```
char *p;
char a;
p = new char('A');
a = XXX;
cout << a;
```

Which of the following in place of XXX will complete the program so that it prints A when run.
(a) `p`
(b) `*p` (*)
(c) `&p`
(d) None of the above

(9) (W) The following snippet of code is incomplete:

```
char *p;
char a;
a = 'A';
p = XXX;
cout << *p;
```

Which of the following in place of XXX will complete the program so that it prints A when run.
(a) `a`
(b) `*a`
(c) `&a` (*)
(d) None of the above

(10) (W) The following snippet of code is incomplete:

```
char *p;
char a;
p = new char('B');
a = 'A';
*p = XXX;
cout << *p;
```

Which of the following in place of XXX will complete the program so that it prints A when run.
(a) `a` (*)
(b) `&a`
(c) `*a`
(d) None of the above

(11) (W) The following snippet of code is incomplete:

```
char *p;
char *q;
p = new char('A');
q = new char('B');
*q = XXX;
cout << *q;
```

Which of the following in place of XXX will complete the program so that it prints A when run.
(a) `p`
(b) `&p`
(c) `*p` (*)
(d) None of the above

A.3 Learning Outcome D (Values Vs Pointers AKA Common Pointer Errors)

(1) (C) For each variable in the following snippet of code, specify whether the memory for the variable contains a char value or an address that refers to a location in memory that contains a char value.

```
char a = 'A';
char *p = &a;
char b = *p;
char *q = p;
```

(a) `a` (value)
(b) `p` (address)
(c) `b` (value)
(d) `q` (address)

(2) (C) For the following snippet of code, which lines will produce compile-time errors?

```
char a = 'A';
char b = 'B';
char *p = new char('C');
char *q = new char('D');
a = b;
p = &a;
b = *q;
p = q;
a = p;  (*)
```

```
q = b;  (*)
p = *a; (*)
b = &q  (*)
```

(3) (T) What would the following snippet of code print when run?

```
char *p;
char *q;
p = new char('A');
q = p;
p = new char('B');
cout << p << " " << q;
```

(a) A A
(b) A B
(c) B A
(d) B B
(e) None of the above (*)

(4) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char('A');
a = p;
p = new char('B');
cout << *p << " " << a;
```

(a) A A
(b) A B
(c) B A
(d) B B
(e) None of the above (*)

(5) (T) What would the following snippet of code print when run?

```
char a;
char *p;
a = 'A';
p = a;
a = 'B';
cout << a << " " << *p;
```

(a) A A
(b) A B
(c) B A
(d) B B
(e) None of the above (*) (compile-time error for incompatible types)

A.4 Learning Outcome E (Pointer Arithmetic)

(1) (C) For the following snippet of code, what does line 2 do?

```
int *p = new int[2] {1, 2};
p++;
cout << *p;
```

(a) Changes the address of the pointer p.
(b) Changes the value that the pointer p refers to.
(c) Neither of the above, it is an error.

(2) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char[2] {'A', 'B'};
a = *p;
p++;
*p = a;
cout << *p << " " << a;
```

(a) A A (*)
(b) A B
(c) B A
(d) B B
(e) None of the above

(3) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char[2] {'A', 'B'};
a = *(p + 1)
*p = a;
cout << *p << " " << a;
```

(a) A A (*)
(b) A B
(c) B A
(d) B B
(e) None of the above

(4) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char[2] {'A', 'B'};
a = *(p + 2)
*p = a;
cout << *p << " " << a;
```

(a) A A
(b) A B
(c) B A
(d) B B
(e) None of the above (*)

(5) (T) What would the following snippet of code print when run?

```
char *p;
char a;
p = new char[2] {'A', 'B'};
a = *p + 2;
*p = a;
cout << *p << " " << a;
```

(a) A A
(b) A B
(c) B A
(d) B B
(e) None of the above (*) (It prints 'C'.)

(6) (W) The following snippet of code is incomplete:

```
char *p;
p = new char[2] {'A', 'Z'};
p = XXX;
cout << *p;
```

Which of the following in place of XXX will complete the program so that it prints Z when run.

(a) `p + 1` (*)
(b) `*p + 1`
(c) `*(p + 1)`
(d) None of the above

(7) (W) The following snippet of code is incomplete:

```
char *p;
char a
p = new char[2] {'A', 'Z'};
a = XXX;
cout << a;
```

Which of the following in place of XXX will complete the program so that it prints Z when run.

(a) `p + 1`
(b) `*p + 1`
(c) `*(p + 1)`
(d) None of the above

Received 9 September 2017; revised 20 November 2017; accepted 1 December 2017