ZORQ: A Gamification Framework for Computer Science Education

John Hastings
Cyber Systems Department
University of Nebraska at Kearney
Kearney, NE USA
hastingsjd@unk.edu

Sherri Weitl-Harms
Cyber Systems Department
University of Nebraska at Kearney
Kearney, USA
harmssk@unk.edu

Adam Spanier
Information Systems and Technology
University of Nebraska at Omaha
Omaha, NE USA
aspanier@unomaha.edu

Matthew Rokusek
School of Computing
University of Nebraska - Lincoln
Lincoln, NE USA
mrokusek4@huskers.unl.edu

Ryan Henszey
Self Funded
Kearney, NE USA
henszey@gmail.com

***Abstract*—This research paper introduces a unique system called ZORQ that is a combination of a game development framework and a gamification framework (GDGF). The ZORQ GDGF acts as a catalyst to help motivate students by increasing student engagement and success within undergraduate Computer Science (CS) education, regardless of student experience and background. The dynamic gamification elements utilized within the GDGF make it an attractive learning method for students. After collaborative game space customization, ZORQ gameplay sees each student tasked with designing a ship movement philosophy and then implementing their own code to autonomously control the ship in an interstellar game space filled with supplies, obstacles, and enemy ships. The particulars of engagements between ships can vary greatly by semester, along with the resources/objects present in the game, depending on the collaborative customization and the independent ship strategies implemented.**

A preliminary ZORQ trial was conducted over five years in an undergraduate Data Structures and Algorithms (DSA) course. The ZORQ trial is designed to fulfill the following objectives: 1) implement DSA concepts discussed within the course, 2) identify appropriate problem-solving approaches, 3) apply one or more solutions, 4) build depth with a coding language, 5) bridge the gap between limited concept assignments and large, multi-developer software systems by allowing students to build code within a larger architecture, 6) introduce students to version control, 7) illustrate the use of prior mathematics coursework in practical applications, and 8) introduce unit testing in software systems. In exit surveys, students expressed overwhelming satisfaction with this approach. More than 84% of the students surveyed found the system useful in their educational experience and saw benefit to inspecting a completed software project. 82% of the students found that ZORQ increased software development comprehension. 80% enjoyed using their own personal creativity in designing a ship controller, 76% found ZORQ helped them learn how to implement and use DSAs. 71% found the system engaging and found the system interaction to be clear and understandable. Observations of student performance in later courses suggest better student maturity and comprehension in preparation for proposing and implementing their own independent projects.

***Index Terms*—Computer Science Education, Dynamic Gamification, Active Learning, Game Development Based Learning, Game Development Gamification Framework**

I. Introduction

Gamification is "the use of design elements characteristic for games in non-game contexts" [1]. While the use of games in education is almost as old as education itself [2], the advent of digital systems revolutionized the landscape of possible games that could be used in the classroom. As early as the CD-ROM era, educators have been attempting to use digital technologies to augment traditional learning practices [3]. Though these early implementations didn't yet truly encapsulate the idea of modern gamification [1], they proved the concept enough for researchers to begin working with and understanding gamification application interventions and the resulting student behavior outcomes.

As a modern discipline, gamification remains relatively young, only formally existing since 2011 [4]. Luo [5] notes that studies that achieved meaningful gamification in the educational domain are limited, while also indicating that engagement is a key measure of gamification's effectiveness. The relative novelty within the discipline provides a great impetus for researchers to understand which gamification interventions attain the best outcomes in specific areas of education and to seek a better understanding of the structures and classifications that may exist in gamification [6]. By understanding the applications and structures within gamification, researchers can hone gamification interventions to better suit the needs of different educational requirements.

While traditional methods still remain the bulwark of most modern courses, the use of gamification as a means to augment modern education is becoming increasingly popular [7]. The use of gamification in education is generally motivated by a desire to improve both motivation and engagement in students [8]. As such, existing empirical research in psychological fields dealing directly with motivation and engagement such as Self Determination Theory (SDT) [9], [10] and Flow Theory (FT) [11] can provide greater fidelity and effectiveness in

terms of *how* gamification can be best implemented. Gamification fits well with SDT, a macro-psychological theory of human motivation, which contends that to effectively sustain motivation and persistence in a discipline (such as CS), students need to feel competent, autonomous, and related to others [9], [10], [12].

FT seeks to understand how individuals achieve the full, immersive participation in the present moment [11]. Rather than design and implement gamification interventions within the strict confines of experiential or anecdotal contexts, the use of well-known motivational techniques can provide a more stable foundation for student success. The 'gameful' experience [13] is tied to flow experience, which is composed of nine dimensions and is directly related to student motivation and engagement [11], [14]. In this way, flow experience is highly related to student performance within an educational context [11]. While the effects of gamification on the flow state have been researched to a limited extent [14], [15], there exists a dearth of research pertaining to *how* individual differences in gamification applications moderate the influence of gamification on flow state in learners [14].

Gamification provides the component of fun that helps in transforming student attitudes towards learning [16]. Further, the use of game mechanics in learning interventions can improve the ability to learn new skills by up to 40% [8]. These benefits coupled with the ever-increasing complexity of modern technological systems create a strong motive to incorporate specialized gamification applications in education.

Additionally, the use of gamification can be used to great effect to bolster engagement, motivation, and student success in traditionally under-served schools [17] where exposure to CS concepts is generally lacking [18]. When entering college, this lack of experience generally hinders interest in CS majors and often leads to feelings of "impostor syndrome" [19]. The use of gamification applications can not only bridge the gap in knowledge, but also provide an impetus to engage with the material being presented.

With these benefits in mind, a more concise and effective gamification intervention can be designed through analysis and understanding of various gamification applications that currently exist in education. Within the scope of this research, gamification applications in Data Structures and Algorithms (DSA) courses can inform areas where more effort is needed. In the classification system in [6], a series of five characteristics-based classifications for DSA gamification applications is proposed and explored. While three of the proposed categories find existing candidates in DSA courses, two classifications appear unexplored. Most notably, the classification titled *Dynamic Gamification* [6] presents several beneficial characteristics that present significant potential benefits to student learning in DSA courses.

The rest of this paper is organized as follows. Section II details game development-based learning and gamification in CS Education. Section III presents the ZORQ Gamification Framework. Section IV outlines the ZORQ user study. Section V presents the discussion of the findings of this research. Section VI outlines potential future research, and section VII presents the conclusion reached through this study.

II. Game Development-Based Learning and Gamification

The use of game development as a means to facilitate education is growing alongside gamification. Game Development-based learning (GDBL) as defined by Wu *et al.* [20] is any process by which "students are required to modify or develop a game as a part of a course using a game development framework (GDF) to learn skills within CS and software engineering (SE)." A GDF as applied to GDBL is any toolkit or codebase by which games can be modified or developed [20]. GDF's include game engines, platforms and simulations as well as integrated development environments [20]. By nature, GDF's are technical artifacts where GDBL is a method by which game development is used to instruct students.

While the creation of a game as a means to instruct also boosts student motivation through topical associations with games and game environments [20], gamification and GDBL are not concurrent domains. Where gamification is the explicit use of games mechanics and elements in traditionally non-game environments [1], GDBL is the use of game development to meet specific learning objectives [20]. In some cases, the use of game development simply supports the learning of some code base, CS concept, or GDF. Since the game being developed is not the emphasis of the learning activity, this variety of GDBL falls into a more traditional method of teaching. In contrast, gamified GDBL is the overlapping domain where GDBL methods implement game mechanics in non-game environments as a means to instruct students [1] as shown in Figure 1.

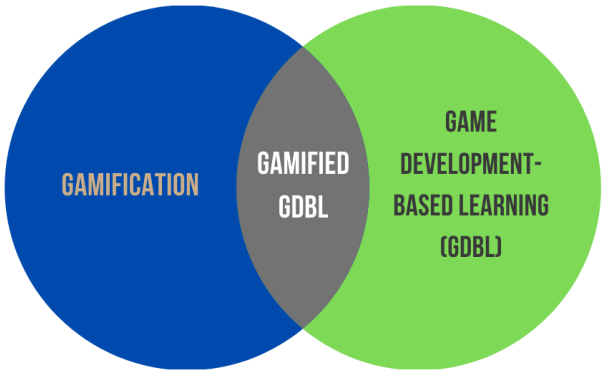

Fig. 1. Gamification and Game Development-Based Learning

While gamified interventions and GDBL applications offer unique and interesting insights when used individually, the occurrence of simultaneous GDBL and gamified methodologies presents novel student intervention characteristics. Currently, the use of static gamified applications [6] appears to dominate the CS gamification landscape. Yet, as proposed by Gibson and Jakl [21], allowing the learner to make dynamic choices in the gamified experience will facilitate the occurrence of more meaningful gamification, and learners see increased intrinsic

motivation. While static game play can allow some student choice, the game framework must keep student input within a specific domain [6]. By combining the use of highly student-centric decision-making methods in a GDBL methodology and gamifying the process, motivational benefits of both gamification and GDBL can be further augmented by allowing the learner to choose not only how they play the game, but also how to design a gamified application.

Dynamic Gamification (DG) [6] is a novel combination of GDBL and gamification which exhibits a dynamic phase-based, iterative process using a Game Development/Gamification Framework (GDGF). DG, while not directly related to student psychological imperatives, provides a framework by which SDT and FT can be fully achieved in gamification interventions. The game application development phase is followed by a gamified testing phase where each student operates within the static confines of the gamified environment. After testing, playing, and experiencing the environment, the students can cycle back into a collaborative GDBL phase. The iterative process allows the dynamic integration of student input, GDBL, and Gamification into a single streamlined process.

Through this iterative, team-based process, collaboration is fostered among the student learners. Collaboration helps learners better interact with the content. According to Kim *et al.* [22], as players collaborate, they "engage in a shared, relevant, goal-oriented activity". This sharing of effort provides not only student support and camaraderie, but also helps develop positive student learning behaviors as learners work together and interact with the content [23].

As an added benefit, the iterative model presented in [6] allows for student correction and adjustment in both the development of the game and the playing of the instantiated gamified application. This conceptual acceptance and correction of "mistakes" fosters concepts espoused by Alsawaier [16], [24]. The Alsawaier study found that successful game design allows the players to try multiple times to achieve success. In the gamified environment, mistakes generate "opportunities to learn and correct" [25]. Players who received constructive feedback following failure in the gamified environment expressed positive emotions about their experience [26].

With respect to DSA coursework, specifically oriented toward the DG classification, this research presents: 1) a novel GDGF called ZORQ, 2) student survey results evaluating the effectiveness of ZORQ, and 3) discussion concerning the benefits and detriments of the intervention.

III. ZORQ: A Game Development and Gamification Framework (GDGF)

ZORQ (Zero Operator ReQuired)[1] is a GDGF in which spaceships navigate a 2D game universe filled with objects which affect a ship either positively or negatively. Examples of game objects that have been used include fuel, shields, mines, black holes, ship-jump portals, electromagnetic pulses, bullets, and lasers. Students adjust the GDGF architecture as needed each semester, and then implement code to automatically control their own ship in a gamified process.

In the current version of ZORQ, ships earn points for each frame in which they remain active/alive, and also by gathering resources which award bonus points. Remaining active is the primary means by which ships maximize points. Negative encounters cause a ship to be deactivated for five seconds, after which they respawn in a different location. Examples of negative encounters include being successfully targeted by another ship, running into an obstacle, or getting sucked into a black hole. Ships can target other ships, and if successful, steal a percentage of the score from the attacked ship. Engagements between ships generally favor the ship with more resources (e.g., fuel, bullets, shield energy, etc.) or a superior strategy. The particulars of these engagements can vary greatly by semester, along with the resources/objects present in the game, depending on how students want the game to behave.

A. ZORQ's Evolution as an Educational Tool

Over the span of several years, a third semester DSA course curriculum included the development and use of a series of GDGFs as class projects. ZORQ was one of the earliest GDGFs created (in 2006). Using an agile approach, the instructor and students worked interactively as a team to brainstorm game ideas. Based on the game idea, the instructor would lay out an initial architecture including the game engine and initial graphics, and the class would incrementally add game components to the GDGF. Within these projects, students would design and implement controllers that would act autonomously within the game. As a semester progressed and new components were added to the game, or changes occurred, students would update their controllers to handle the changes.

Over time the delivery of this gamified GDBL approach was refined based on some lessons learned. These included:

1) **Competitions:** In early versions, the game was pitched as a competition and students would design controllers that aimed to perform better than other student controllers. As the semester progressed, repeat competitions were held. While some students loved the competitive aspect, other students were turned off by this and seemed to disengage as the semester progressed. Competitions in gamification applications have been found to fail to engage and even demotivate some students [27], [28].
2) **Shared project repository:** Just prior to each competition, students were required to commit their controllers to a shared project repository visible to the entire class. When students saw a controller that handled some aspect of the game well, they often would incorporate that code into their own controllers rather than analyzing the problem and designing a solution themselves. This was neither fair to the students who had dedicated time to formulating a winning approach, nor the students who missed the

[1]The ZORQ project public repository is available at https://gitlab.com/ZORQTEAM/zorq.

opportunity to themselves learn how to implement the particular strategy or further explore alternate approaches.

3) **Use of class time:** Developing a GDGF over the course of the semester was a meaningful exercise for the students. However, developing a unique GDGF each semester was not only time consuming, but also required investigating content that didn't necessarily fit nicely within the course (e.g., computer graphics) and also consumed time needed for core DSA or CS concepts (e.g., asymptotic efficiency) which didn't always fit appropriately into the development of the GDGF.

The DSA course delivery was adjusted to address these concerns. After multiple years of creating unique one-off GDGFs, one year using the outside gamification application Robocode [29], and a few years with mobile application development [30], in 2017 the use of ZORQ as the GDGF was revisited. ZORQ has been used in the DSA course since then, and was selected for several reasons:

- ZORQ is fully implemented, and students could focus on the shorter term goal of implementing a controller rather than developing an entire system. In addition, because the GDGF had seen use, most bugs were resolved;
- ZORQ gives students an early (third semester) experience seeing a much larger system, and learning skills to begin understanding and working with a larger systems;
- Student engagement with ZORQ was observed to be as high or higher than any of the other GDGFs used;
- ZORQ is a system under the instructor's control and can be refactored (modified/extended) as necessary;
- As a system under our control, students can write test code for their controllers to gain an understanding of the important concept of unit testing early in their careers;
- ZORQ is complex enough that students can creatively implement controllers in an almost endless number of ways; and
- ZORQ can be used in the later courses; such as the SE course when talking about cohesion/coupling.

B. ZORQ Architecture

The following text describes the most recent version of ZORQ. At initial startup, the user is met with a settings screen that provides various game simulation options such as multi-threading, item spawn rate, universe size, graphical scrolling background toggle, and headless mode. Next, the user selects the ship controllers to be used in the simulation. The user may select as many controllers as desired. Once selected, the controllers are internally connected to ships, one controller per ship, and the game begins. If headless mode was not selected, then graphics are set up before the game begins. During game time, the game space and elements are depicted with moving images as shown in Figure 2.

In the upper left of the screen, stats are displayed including controls, frame rate, and a listing of the active ship controllers and their scores. The frame rate can be adjusted via the keyboard, allowing for rapid simulation, with one engine cycle (i.e., game state) per displayed frame. The game runs indefinitely until the program is closed.

Fig. 2. ZORQ Screenshot

If headless mode is selected, the graphical display is replaced by console output which periodically shows stats including the game frame/cycle rate. In headless mode, the frame rate is uncapped and the simulation runs as fast as the hardware permits.

An overview of the ZORQ classes most relevant to this discussion appears in Figure 3. The classes are separated and color coded based on their level of abstraction, where level 0 is the most abstract and level 3 is the least abstract. Objects at higher levels know about objects at lower levels. The Engine manages the game and contains the main game loop which advances the game state while maintaining a steady frame rate. The Universe holds the less abstract game elements, and steps forward to the next game state when triggered by the Engine. Elements in the Universe are special types of FlyingObject.

Each ship is connected to a custom ShipController, developed by the students, which dictate how the ship behaves. Each controller has access to the Universe and thus all the elements in the game space. Students decide and implement what they want their ship to do (via the controller) based on the state of the game elements in the Universe.

There are a few main components to the scoring system. As ships move around the game space, their scores will gradually increase. If a ship is destroyed, it then loses some of its score. If it was destroyed by an opposing ship, then the opposing ship gains a portion of the destroyed ship's score. Because a ship's resources are limited, students often develop very complex strategies.

There are several game elements described as bonuses, as shown graphically in Figure 2 and listed as classes in Figure 3. Fuel bonus objects resupply ships with a limited amount of fuel which is otherwise consumed as a ship accelerates. Bullet bonuses refill and upgrade a ship's bullet supply as ships can shoot bullets. Each ship has a metered shield that is decremented when powered on to deflect dangerous objects.

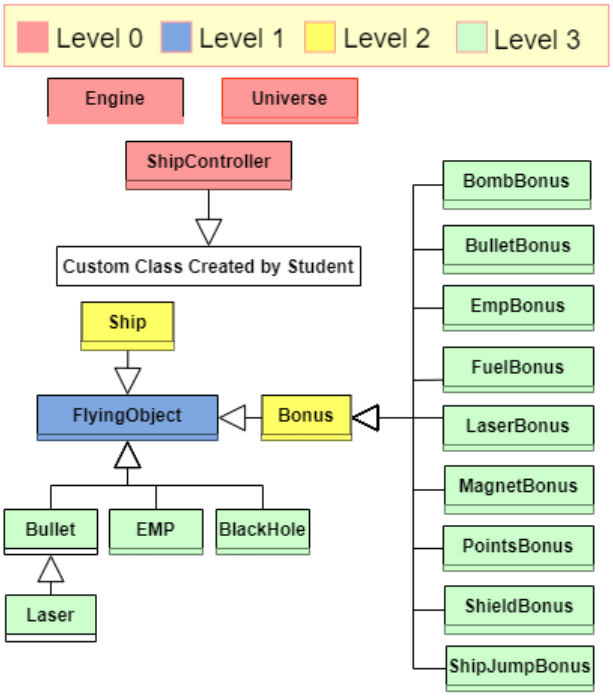

Fig. 3. ZORQ UML Class Diagram

Shield bonuses refill this meter. Ships are allowed to teleport to another location on demand using ship jump bonus objects (shown as blue circles with white centers). Ships can also shoot lasers and laser bonus objects (shown as flaming wands) refill a ship's laser supply. Electromagnetic pulse (EMP) bonus objects (shown as three green circles) give ships EMP blast capabilities. Points bonus objects (shown as dollar signs) give ships that pick them up points to improve their score. Magnet bonus objects cause a ship to pull in nearby bonus objects for a short amount of time.

Several objects have negative outcomes for ships. Bombs and bullets either damage a ship's shield or destroy a ship if touched. The distinguishing factor between bombs and bullets is that bullets originate from opposing ships. Lasers are a special type of bullet that move very fast, making it harder for opposing ships to react. A ship hit by an EMP loses its ability to shield and shoot objects for a short amount of time during which it becomes vulnerable to attack. Black holes are hazards that pull in ships and destroy them. Black holes also attract bullets and other objects that get close to them, as shown in Figure 2.

Because ZORQ was initially created as a one semester, one-off project, the inaugural focus was primarily on quickly getting ZORQ running and engaging students, without dedicating much time to making the best design and architectural choices. After coming back to ZORQ, work resumed to improve, refactor, and adjust ZORQ. Most of these changes occurred in the classroom, in a GDBL way that is engaging and educational for the students, and provides opportunities for students to have a stake in collaboratively providing input on its development. Examples of improvements/additions include:

- ZORQ was refactored to better separate the model and view. In addition to showing best practices, another benefit of improving the design was the opportunity to speed up the frame rate. Rather than simply displaying the game at a steady frame rate, ZORQ was modified so that it can be adjusted to run as fast as the computer can support. Over 1000 simulated frames per second (fps) in graphical mode are possible depending on the hardware, the number of controllers, and the CPU demands of the controller activities/strategies. This allows students to quickly see how well their controllers will behave over time and supports stress testing.
- The architecture was stress tested at high frame rates using a profiler to identify and resolve issues such as logical errors and memory leaks caused by failing to remove items from collections. Profilers can conduct a variety of dynamic software analyses and are available for most popular environments, dating back to 1982 [31].
- The headless mode was also a recent addition. Speeds in excess of 100k cycles per second are possible, allowing for further stress testing and for the fast generation of training data needed to develop future controllers using machine-learning/deep-learning approaches.
- A multi-threaded mode was added, in which controllers running in separate threads. Typically, simulations are run with around 15-20 controllers. The possible benefits of the multi-threaded mode need to be investigated further.
- The architecture was reviewed from a coupling and cohesion[2] [32], [33] perspective using CodeMR [34]. Because of the quick inaugural development and iterative updates, ZORQ's architecture was found to be somewhat weaker in these areas. Several instances of low cohesion and high coupling in ZORQ were subsequently discussed and repaired in the SE course. For example, the graphics were separated from game objects allowing game objects to maintain only the logic of how they should function rather than how they should look. These types of improvements make way for new features to be more easily added to the architecture. The current cohesion and coupling results can be seen in Figure 4, and according to the results, further improvements are possible.
- Additional perfective changes were made such as profiling ZORQ to identify code that could be sped up, e.g., replacing the precise distance formula which uses the slow "sqrt" math function with distance comparisons using the distance squared calculation. According to Swanson [35], perfective maintenance is maintenance performed to eliminate processing inefficiencies, enhance performance,

[2]Coupling is a measure of the interconnectedness of modules within a system, while cohesion is a measure of the degree with which components within a module are related to each other. High coupling complicates system maintenance as updates to one class are more likely to impact other classes with a cascading effect possible. In object-oriented software, for the sake of maintainability and scalability, it is generally preferred to have low coupling and high cohesion.

or improve maintainability.

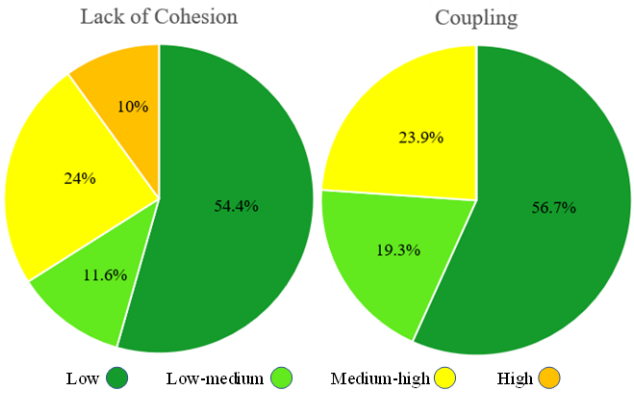

Fig. 4. ZORQ Coupling Cohesion Analysis

C. Approach to delivering ZORQ as a Gamification Tool

The gamified GDBL approach using ZORQ in a DSA course provides an opportunity for students to meet the following learning objectives: 1) implement DSA concepts discussed within the course, such as priority queues and hash maps, in fun and engaging ways, 2) identify appropriate problem-solving approaches, 3) apply one or more solutions, 4) build depth with a coding language, 5) bridge the gap between limited concept assignments and large, multi-developer software systems by allowing students to build code within a larger architecture, 6) introduce students to version control, 7) illustrate the use of prior mathematics coursework in practical applications, and 8) introduce unit testing in software systems.

The approach to delivering ZORQ has been refined over time, based on observations and students' suggestions, but follows the implementation life cycle shown in Figure 5. Briefly, ZORQ is currently utilized in the DSA course as follows:

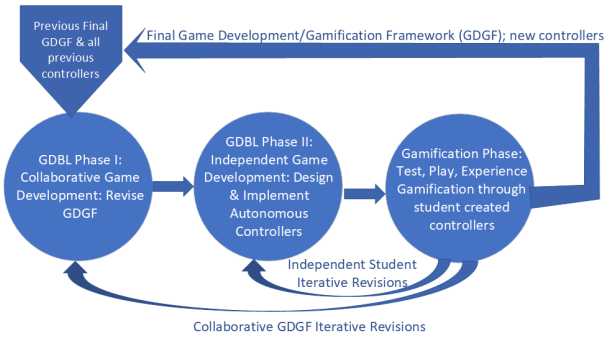

Fig. 5. ZORQ Implementation Life Cycle

1) GDBL Phase I:

- ZORQ is introduced a month before the end of the semester to give sufficient time for students to interact with the GDGF. This addresses the previously mentioned issue of consuming too much time by fully implementing a unique GDGF each semester. With perfective changes, each new semester uses the most recent version of ZORQ.
- After an initial introduction, the class collaboratively discusses what changes or additions to make to the GDGF before working on ship controllers. These changes can include adding or removing game objects or changing the behavior of the game or existing objects. For example, over time, new game bonuses such as shields have been added.
- The changes in the configuration for a given semester are made collaboratively as a class. Depending on class time availability, these changes are made either inside or outside of the class by the instructor or a student. Either way, the changes are shared and explained to the class as part of the GDBL experience.

2) GDBL Phase II:

- Once the configuration of the architecture is decided, the students are provided some small sample skeleton controllers that demonstrate in a basic way that a ship controller should analyze the state of the game (e.g., the game objects, their locations, and what they are doing) and then select an action to take. The example code demonstrates how to utilize some of the built-in utilities, e.g., the function to compute angles to other game objects. and demonstrates that capable controllers can be built without using Artificial Intelligence (AI) concepts, which students will see in a later course.
- Students are instructed to design and implement a controller that follows their own unique, creative philosophy, using DSA concepts. Grading on the assignment is not tied to how their ships score when the game runs, but rather how well students: put effort into the assignment, implement their philosophy, describe their philosophy to the class, document their code, and test their code. When designing controllers, students generally focus on maximizing their scores, although some students might pursue other goals such as path finding or resource gathering. This approach has lead to some creative solutions. For example, one student implemented a 'copycat' controller that would mirror the actions of the closest opposing controller. The shift away from "competition" to implementing a philosophy addressed the issue of some students not enjoying an atmosphere of competition [36].

3) Gamification Phase:

- Students are provided a library of precompiled, obfuscated controllers created by past students, for testing purposes. Students can test their ship controllers against these controllers and adjust their algorithms prior to the in-class game play session, which includes controllers from other current student developers. With changes to ZORQ each year, the software engineering open-closed principle [37] has been followed, in which the code is open for extension, but closed for modification. The idea is to not "break" previously created controllers so that new students have something to test against.

- One class day at the end of the semester is dedicated to running the students' controllers. This is intended partly so students don't get caught up in the need to repeatedly compete with each other. Students who wish to view the game as a competition can do so, but they are instructed that they are not graded on that. During the "simulation", each student is required to discuss their creation. This activity serves the additional benefit of helping students get comfortable talking in front of peers, as well as talking about their creation, the DSAs used and the software development process.
- Students submit their code to their own private repository which they share with the instructor for grading.

IV. ZORQ User Study

The primary aim for the use of ZORQ is to increase student engagement and success within undergraduate CS education, regardless of experience and background. To understand the teaching effectiveness of using ZORQ as an gamification tool in undergraduate CS education, its usage in a DSA course over multiple semesters was evaluated.

A. Study Design

An anonymous online exit survey was distributed to students who had used ZORQ in a DSA course at the University of Nebraska at Kearney (a regional university serving primarily rural students) from 2017-2021. Due to the rural nature of the region, few students encounter CS and SE concepts before attending university [18], [38], [39]. The study received ethical approval from the university, and students were given informed consent to opt-in to completing the survey, but no incentive was provided. Since the DSA course is a required course that all students must pass to continue in the program, only students who passed the course were surveyed, to ensure each student was only surveyed one time.

The students were asked to evaluate their agreement to several prompts using a 5-point Likert scale, with strongly disagree given a value of one, with the other rankings (disagree, neither, agree, and strongly agree) given increasing values, concluding with strongly agree having a value of five. The following prompts were posed:

Q1. I saw benefit to inspecting a completed software project.
Q2. I found the system useful in my educational experience.
Q3. Using the system increased my understanding of software development.
Q4. I was able to use my personal creativity in designing a ship controller.
Q5. Using the system enabled me to learn how to implement and use data structures and algorithms.
Q6. I found the system engaging.
Q7. Learning to interact with the system was clear and understandable.
Q8. Using the system increased my understanding of data structures and algorithms.
Q9. It was easy for me to become skillful at creating my own ship controller within the system.

B. User Study Results

Of the 98 students in the population of students who have completed the course in the semesters surveyed, there were 49 completed responses. While low (50%), this was encouraging given response rates for such optional, anonymous university surveys is often 20-30% (pre-COVID-19) [40]. Nine responses were from female students, (18.3%) which matches the male/female distribution in the overall population surveyed. One student did not indicate gender, and 39 respondents identified as male.

Table I shows the Likert scale results for the prompts asked, where SA=Strongly Agree; A=Agree; N=Neither Agree or Disagree; D=Disagree; and SD=Strongly Disagree. The average on the 5-point scale is reported as is the percentage of students who selected either SA or A, for each prompt.

TABLE I
Student Evaluation 2017-2021 (N=49, 9 female;39 male)

						Overall		Male		Female	
	SA	A	N	D	SD	AVG	SA&A	AVG	SA&A	AVG	SA&A
Q1	34	7	3	2	3	4.37	84%	4.44	85%	4.10	80%
Q2	25	16	3	3	2	4.20	84%	4.26	85%	4.00	80%
Q3	22	18	4	3	2	4.12	82%	4.15	82%	4.00	80%
Q4	27	12	3	3	4	4.12	80%	4.23	85%	3.70	60%
Q5	17	20	6	5	1	3.96	76%	4.15	82%	3.50	60%
Q6	28	7	10	2	2	4.16	71%	4.21	74%	4.00	60%
Q7	18	17	8	4	2	3.92	71%	4.00	74%	3.60	60%
Q8	16	14	12	5	2	3.76	61%	3.92	67%	3.30	40%
Q9	15	15	11	7	1	3.73	61%	3.85	67%	3.20	40%

The study found that overall, 84% of the respondents saw the benefit of being able to inspect ZORQ as a completed software project and found the system useful in their educational experience. 82% of the students reported that using the system increased their understanding of software development. 80% of the students reported that they were able to use their personal creativity in designing a ship controller. 76% found that using the system enabled them to learn how to implement data structures and algorithms. The lowest agreement reported by the students was in the ease of becoming skillful at creating their own ship controller within the system (61%), which is likely due to the complexity of the ZORQ system.

While there were only nine female respondents, their agreement in response to the prompts was lower than their male counterparts, on all survey prompts. Overall, while the range of scores was lower for the female students, a similar ranking of agreement for the research prompts was found, with "seeing the benefit of being able to inspect ZORQ as a completed software project" having the highest agreement and "the ease for them to become skillful at creating their own ship controller within the system" having the lowest agreement.

V. Discussion

The results found that students overwhelmingly responded positively to their experience with ZORQ. This aligns well with existing research about the value of gamification when applied to CS education [6], [41].

As described in the introduction, SDT and FT can provide greater fidelity and effectiveness in terms of how gamification can be best implemented. ZORQ functions to fulfill both SDT and FT through the implementation of Dynamic Gamification (DG). ZORQ provides full autonomy of action within each students' ship controller that allows students gain competence in relation to others. In this way the ZORQ intervention fully applies SDT concepts. FT is achieved through the fluid, iterative development process that sees students adjusting code on the fly while simultaneously working together within the framework. By applying DG concepts, full immersive flow can be attained by both the variance and diversity of outcomes presented by the ZORQ GDGF.

The results demonstrate that the ZORQ GDGF acts as a catalyst to help solve the challenges of increasing student engagement and success within undergraduate CS education, even among underserved populations, as studied. ZORQ provides each student with a personal motive to engage and excel in the learning process. The gamified GDBL approach is a valuable tool in the arsenal of CS educators.

The lead researchers have taught multiple undergraduate CS courses at the same university for more than 20 years and see the students in multiple courses as they progress through the program. Observations of student performance in later courses suggest better student maturity and comprehension in preparation for proposing and implementing their own independent projects, with the use of ZORQ.

Additionally, improvements in introducing and explaining ZORQ to the students lead to stronger student agreement with the benefit of ZORQ in the most recent implementation. While 71% of students overall found that interacting with the system was clear and understandable (Q7), 100% of the students in the most recent course, fall 2021, (n=7) agreed or strongly agreed. Similarly, 100% of these students reported that they could use their personal creativity in designing a ship controller (Q4), saw the benefit of being able to inspect ZORQ as a completed software project (Q1) and found the system useful in their educational experience (Q2).

While ZORQ demonstrates a boost to motivation, engagement, and comprehension, some areas exist where the use of ZORQ can be improved. One such area lies in helping students become skillful in creating functional ship controllers. Further, students also need to better understand the connection between fundamental DSA concepts like Queues, Stacks, Hash Tables, and Trees and the ZORQ GDGF.

The results of this study can help instructors better implement ZORQ in a way that enables future students to build confidence in their knowledge of computing; lay a strong foundation for more advanced classes; generate enthusiasm for creative projects; and encourage continuing with their studies (major retention).

VI. Future Work

In the utilization of ZORQ in DSA courses, an exploration into ways for students to become skillful at creating ship controllers within the system and methods to help students understand ZORQ's connection to DSA concepts is needed. One option to be explored is providing an overview of ZORQ earlier in the semester, to give students time to develop their ideas about changes that they want to make and to help them see the connections to the DSA concepts. The use of ZORQ in other courses, such as SE and AI, also needs explored.

In terms of the ZORQ system implementation, students have expressed a desire to have an option for their controller to view the world through its front window. The specifications of existing components could also be made configurable. Adding a subsystem to allow users to easily add objects is being considered. More unit testing is needed, and issues identified by the coupling/cohesion analysis need addressed. Design patterns are planned to be implemented where appropriate. The benefits of the multi-threaded mode need to be investigated. Testing the impact on efficiency when using parallel CPU processing needs conducted. There are additional updates being considered, such as making the universe size based on number of controllers and allowing the system to randomly select controllers. Setting up controllers to run as micro-services is being considered. There are future plans to make ZORQ available from the code repository, with a formalized initialization and installation process for other schools to use. There are future plans to use deep learning to create controllers by learning from the state of the system and how well existing controllers performed.

A deeper exploration of ZORQ as a *Dynamic Gamification* [6] framework is needed. Additionally, studying ZORQ by adopting a mixed-method analysis, with quantitative, qualitative and sentiment analysis would add to a deeper understanding of ZORQ and to how gamification could improve learning, critical thinking and retention [24]. Finally, a systematic study of the impact of using a GDGF on student performance in later courses should be examined.

VII. Conclusion

In summary, a GDGF named ZORQ was introduced, and a preliminary study of student experiences was presented. ZORQ is a novel combination of Game Development-Based Learning and Dynamic Gamification that enables a streamlined phase-based, iterative process where students collaborate to create a unique instance of a GDGF, then independently design and implement a ship controller, followed by a testing phase where each student's autonomous controller moves in the confines of the gamified environment. After experiencing the environment, students can cycle back to the previous GDBL phases. ZORQ provides students with a high level of engagement with the system and a high level of social engagement in its collaborative customization.

The preliminary results provide evidence that the primary aims for the ZORQ GDGF (to increase student engagement and success within undergraduate CS education) have been met. This study shows that meaningful gamification in undergraduate CS education can be achieved, when student engagement is through coding within the gamification system itself, as demonstrated with ZORQ.

References

[1] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining 'gamification'," in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*. New York, NY, USA: Association for Computing Machinery, 2011, pp. 9–15.

[2] A. Hellerstedt and P. Mozelius, "Game-based learning: A long history," in *Irish Conference on Game-based Learning 2019, Cork, Ireland, June 26-28, 2019*, vol. 1. Cork, Ireland: Irish Conference on Game-Based Learning, 2019, pp. 1–4.

[3] K. Becker and S. Nicholson, "Gamification in the classroom: Old wine in new badges," *Learning, education and games*, vol. 61, pp. 61–85, 2016.

[4] B. Kim, "Gamification," *ALA TechSource*, vol. 51, pp. 10–18, 2 2015.

[5] Z. Luo, "Educational gamification from 1995 to 2020: A bibliometric analysis," in *2021 the 6th International Conference on Distance Education and Learning*, ser. ICDEL 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 140–145. [Online]. Available: https://doi.org/10.1145/3474995.3475740

[6] A. M. Spanier, S. K. Weitl-Harms, and J. D. Hastings, "A classification scheme for gamification in computer science education: Discovery of foundational gamification genres in data structures courses," in *2021 IEEE Frontiers in Education Conference (FIE)*. Lincoln, NE, USA: FIE, 2021, pp. 1–9.

[7] A. Manzano-León, P. Camacho-Lazarraga, M. A. Guerrero, L. Guerrero-Puerta, J. M. Aguilar-Parra, R. Trigueros, and A. Alias, "Between level up and game over: A systematic literature review of gamification in education," *Sustainability*, vol. 13, no. 4, p. 2247, 2021.

[8] G. Kiryakova, N. Angelova, and L. Yordanova, "Gamification in education," in *Proceedings of 9th International Balkan Education and Science Conference*, 2014.

[9] E. L. Deci and R. M. Ryan, "The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior," *Psychological Inquiry*, vol. 11, no. 4, pp. 227–268, 2000.

[10] A. Mishkin, "Applying self-determination theory towards motivating young women in computer science," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1025–1031. [Online]. Available: https://doi.org/10.1145/3287324.3287389

[11] J. Nakamura and M. Csikszentmihalyi, "Flow theory and research," *Handbook of positive psychology*, vol. 195, p. 206, 2009.

[12] R. M. Ryan and E. L. Deci, "Self-determination theory," *Basic psychological needs in motivation, development, and wellness*, 2017.

[13] J. Högberg, M. O. Ramberg, A. Gustafsson, and E. Wästlund, "Creating brand engagement through in-store gamified customer experiences," *Journal of Retailing and Consumer Services*, vol. 50, pp. 122–130, 2019.

[14] W. Oliveira, J. Hamari, S. Joaquim, A. M. Toda, P. T. Palomino, J. Vassileva, and S. Isotani, "The effects of personalized gamification on students' flow experience, motivation, and enjoyment," *Smart Learning Environments*, vol. 9, no. 16, pp. 14–26, 2022.

[15] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? - a literature review of empirical studies on gamification," in *47th Hawaii International Conference on System Sciences*, Hawaii, USA, 2014, pp. 3025–3034.

[16] R. S. Alsawaier, "The effect of gamification on motivation and engagement," *The International Journal of Information and Learning Technology*, vol. 35, no. 1, pp. 56–79, 2018.

[17] J. R. Warner, J. Childs, C. L. Fletcher, N. D. Martin, and M. Kennedy, "Quantifying disparities in computing education: Access, participation, and intersectionality," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 619–625. [Online]. Available: https://doi.org/10.1145/3408877.3432392

[18] C. Broneak and J. Rosato, "Experiences of rural CS principles educators," in *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, 2021, pp. 1–2.

[19] E. F. Churchill, "Is there a fix for impostor syndrome?" *Interactions*, vol. 25, no. 3, p. 22–24, apr 2018. [Online]. Available: https://doi.org/10.1145/3197577

[20] W. Bian and W. A. Inge, "A guideline for game development-based learning: A literature review," *Int. J. Comput. Games Technol.*, vol. 2012, jan 2012. [Online]. Available: https://doi.org/10.1155/2012/103710

[21] D. Gibson and P. Jakl, *Theoretical Considerations for Game-Based e-Learning Analytics*. Springer International Publishing, 2015, pp. 403–416.

[22] Y. Kim, M. Glassman, and M. S. Williams, "Connecting agents: Engagement and motivation in online collaboration," *Computers in Human Behavior*, vol. 49, pp. 333–342, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563215002010

[23] D. Zhang and T. Clear, *Shaping Behaviours Through Space and Place in Gamified Virtual Learning Environments*. Springer International Publishing, 2015, pp. 331–354.

[24] R. S. Alsawaier, "Research trends in the study of gamification," *The International Journal of Information and Learning Technology*, vol. 36, no. 5, pp. 373–380, 2019.

[25] M. D. Hanus and J. Fox, "Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance," *Computers & Education*, vol. 80, pp. 152–161, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131514002000

[26] P. Herzig, M. Ameling, B. Wolf, and A. Schill, *Implementing Gamification: Requirements and Gamification Platforms*. Springer International Publishing, 2015, pp. 431–450.

[27] S. Deterding, "Gamification: Designing for motivation," *Interactions*, vol. 19, no. 4, p. 14–17, jul 2012. [Online]. Available: https://doi.org/10.1145/2212877.2212883

[28] C. Pilkington, "A playful approach to fostering motivation in a distance education computer programming course: Behaviour change and student perceptions," *International Review of Research in Open and Distributed Learning*, vol. 19, no. 3, 2018.

[29] F. N. Larsen, "Robocode," https://robocode.sourceforge.io, 2022, accessed: April 10, 2022.

[30] S. Harms and J. Hastings, "A cross-curricular approach to fostering innovation such as virtual reality development through student-led projects," in *2016 IEEE Frontiers in Education Conference (FIE)*. Erie, USA: FIE, Oct 2016, pp. 1–9.

[31] S. L. Graham, P. B. Kessler, and M. K. McKusick, "Gprof: A call graph execution profiler," in *SIGPLAN '82*, 1982.

[32] W. P. Stevens, G. J. Myers, and L. L. Constantine, "Structured design," *IBM Systems Journal*, vol. 13, no. 2, pp. 115–139, 1974.

[33] G. Booch, *Object-Oriented Analysis and Design with Applications (3rd Edition)*. USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[34] Q. Ltd, "CodeMR," https://www.codemr.co.uk/, 2022, accessed: March 10, 2022.

[35] E. B. Swanson, "The dimensions of maintenance," in *Proceedings of the 2nd International Conference on Software Engineering*, ser. ICSE '76. Washington, DC, USA: IEEE Computer Society Press, 1976, p. 492–497.

[36] C. Almeida, M. Kalinowski, and B. Feijó, "A systematic mapping of negative effects of gamification in education/learning systems," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2021, pp. 17–24.

[37] B. Meyer, *Object Oriented Software Construction*. Prentice-Hall, 1988.

[38] code.org, "2021 state of computer science education," https://advocacy.code.org/2021_state_of_cs.pdf, 2021, accessed: June 16, 2022.

[39] B. Dorn, D. Babb, D. M. Nizzi, and C. M. Epler, "Computing on the silicon prairie: The state of CS in Nebraska public schools," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 296–301. [Online]. Available: https://doi.org/10.1145/2676723.2677261

[40] A. Jaffray, C. Finn, and J. Nurse, "Sherlocked: A detective-themed serious game for cyber security education," in *Human Aspects of Information Security and Assurance, HAISA 2021*, vol. 613, 2021, pp. 34–45.

[41] A. Ahmad, F. Zeshan, M. S. Khan, R. Marriam, A. Ali, and A. Samreen, "The impact of gamification on learning outcomes of computer science majors," *ACM Trans. Comput. Educ.*, vol. 20, no. 2, apr 2020. [Online]. Available: https://doi.org/10.1145/3383456