Visual Design Problem-based Learning in a Virtual Environment Improves Computational Thinking and Programming Knowledge

Amy Banic* and Ruben Gamboa**
University of Wyoming

ABSTRACT

In this paper, we present our design of a high school summer course which uses our Visual Design Problem-based Learning Pedagogy using Virtual Environments as a strategy to teach computer science. Students solved visual design problems by creating 3D sculptures in an online virtual environment. These creations were further explored and refined in immersive display systems fostering embodied learning and remote peer presence and support. To achieve the desired design, students use programming and computing concepts, such as loops, to solve those visual design centered problems, i.e. solving for composition, positive/negative space, balance, as opposed to computational problems first, i.e. create a loop, a fractal, randomized lines, etc. We present results from a study conducted on three high school summer courses. We compared the use of our Visual Design Problem-based teaching strategy (students wrote code to solve challenges based on art and design principles) to a traditional strategy (students wrote code to demonstrate comprehension of computer science concepts). Our results showed that test scores were higher for students in our Visual Design Problem-based courses. This work may have a positive impact on computer science education by increasing engagement, knowledge acquisition, and self-directed learning.

Keywords: Visual Design Problem-based Pedagogy, problem-based learning, Online Virtual Environments, Embodied Learning, Virtual Sculpting, Increase Programming Knowledge, Spatial Art.

Index Terms: CCS [Human-centered Computing]: Human-computer interaction (HCI)- Interaction paradigms- Virtual Reality; [Human-centered Computing]: CCS Human-computer interaction (HCI)- Empirical studies in HCI;

1 INTRODUCTION

While others have been successful at combining fine arts and computer science to promote learning [1-3], to our knowledge little work has been done in using art design principles as the primary focus for problem-based teaching methods to foster self-directed learning in computer science. To demonstrate our idea, we developed a summer course, "Generating Art in a Virtual World", as a part of the larger outreach program for the University of Wyoming High School Summer Institute (UWYO HSI). The primary objective of the course was to teach high school students introductory computer science concepts and skills. This includes topics such as primitive data types (e.g., numbers, words, or images), data records (e.g., a ball composed of a position, radius, and color), recursive data types (e.g., lists and arrays), functions, selections (e.g., if this happens then do that), loops, and iteration (e.g., do this to each element of a list). Furthermore, to achieve our goal, we used an Online Virtual Environment (OVE) since it provided students the capabilities to create and manipulate 3-dimmensional visual creations using computer programming code that provided additional spatial information. OVEs are a type of VE that allow participants to access a virtual environment and connect with others all remotely through the internet [4]. They used this OVE to develop and program their 3D sculptures and designs through a combination of desktop, head-mounted displays, and a CAVE system [5]. Further, since UWYO HSI high school students are spread out across the state, it makes for having a supportive cohort to learn programming difficult. We sought to provide an accessible community that connected students with each other and foster small group peer learning, which has been shown to improve learning in STEM fields [6]. By using an OVE, students can have access from their homes, especially in rural areas like Wyoming, to continue embodied learning in the virtual environment and with other peers. These OVEs are 3-dimmensional (3D) environments that can be made more immersive through a stereoscopic display technology, i.e. a head-mounted display or a virtual reality viewer.

Figure 1: Example of creative 3D Spatial Art in an OVE.

The objective of this work was to determine if problem-based instruction using art and visual design problems, as opposed to computing problems, would engage high school students in self-directed learning and increase programming knowledge.

We hypothesized based on the self-directed learning literature [7] that students would be more engaged by the art challenges (as opposed to using computer programming problems) and increase self-directed learning of the computer programming concepts. In this paper, we present results from a study and data collected from three courses about students' knowledge gained. We present the materials, lessons, and activities of the course. We utilized problem-based instruction methods for all courses. In one course, we provided problems that focused on the computer programming concepts (a traditional teaching strategy). In the other two courses, we used a Visual Design Problem-based teaching strategy where we provided problems that focused on art and visual design principles, such as composition, positive/negative space, color, space, form, lines, balance, etc., and related ideas from generative art (Figure 1), such as randomization and repetition. For courses that used Visual Design problem-based learning, we observed that more students exhibited self-directed learning to find programming concepts that fulfilled their creative vision, than students in the course where we used computing problems. Our results showed that students learned more computer science knowledge and debugging skills with Visual Design Problem-based Pedagogy than using computing concepts as a strategy for CS instruction.

{*abanic,**ruben}@uwyo.edu
,

2019 IEEE Conference on Virtual Reality and
3D User Interfaces
23-27 March, Osaka, Japan
978-1-7281-1377-7/19/$31.00 ©2019 IEEE

2 Related Work

2.1 Problem-based vs. Inquiry-based Pedagogy

There have been many solutions that have been shown to improve attitudes towards computer science, engage students, and improve computational thinking and programming skills [8,9]. Self-directed learning has been shown to improve knowledge acquisition in STEM fields [6]. Problem-based and Inquiry-based teaching methods are of a set of teaching methods that have been shown to foster self-directed learning [10-14]. Our idea combined these types of problem-based teaching methods with art. To do this we present design problems to be solved by students using art and design principles, but while doing so fosters self-directed learning in computer science (Figures 1 and 2). In figure 2, the programming functions llRand (outputs a random number, and in this case random position and color values) and llRez (generates a 3D object in space, and in this case the triangular objects shown) are used to produce the desired effect. Through this process of experiential learning [15], students learn and connect potential results with different programming concepts.

2.2 Virtual Environments for Education

Research has shown that online virtual environments have been used successfully for educational purposes [16-19] but not specifically for problem-based instruction, using art and design principles, to foster self-directed learning. A previous study used different gaming technology, as a medium for learning art curriculum, game theory, and complexity thinking [20]. Students used their personal experiences to create games in the form of visual expression not programming. This study did not focus on pedagogy that incorporated design principles to learn computer programming skills, as presented in our work.

2.3 Visual Arts and Design Curriculum

The course lessons are outlined in such a way that the students receive a combined curriculum of basic art concepts and basic programming concepts, and during the course they learn to combine these skills together through computer-based art projects. We follow a model from visual arts where students go through an iterative process of create-experience-reflect [21, 22]. In this process for each lesson, concepts from visual arts and design curriculum were selected for the problems.

3 Course Design Using Visual Design Problem-based Pedagogy for a Virtual Environment

3.1 Platform and Virtual Environment

We used an online virtual environment to fill three important roles. First, OVEs provide a platform where the students can program. For example, students must program to create objects, which can range from static images to full-blown, dynamic works of art. Second, they created an environment where the students can interact with instructors and each other, even after the UWYO HIS program was complete. Third, they were able to archive the students' artworks and make them available to future generations of students. Students' final art pieces were showcased in a persistent virtual gallery to be experienced in a Head-Mounted Display, CAVE, Mobile+ VR Viewer, or standard desktop display. Online virtual environments create a community that can connect students with each other. We used only open-source tools that students can install on any computer, including PCs running Microsoft Windows and Linux, as well as Apple Macintosh computers. We have used both Curiosity Grid and Kitely for our OVE, where each have their own strengths and weaknesses but for purposes of our methods both worked sufficiently. Both platforms, and others available, are built upon the Linden Scripting language. What worked well, offered by both platforms, is that we could set up an individual island for each student as a workspace and then a larger island as a shared virtual gallery. This model worked much better for our course because it allowed for students to experiment with primitives and not have the system slow down. We could set permissions to allow students to visit each other's workspaces but not be able to change anything in the environment unless the owner student permitted them to. The gallery analogy fostered iterative design permitting multiple versions in a workspace and a formal gallery space for finished pieces. A virtual world viewer can be used to interface with these platforms, such as but not limited to, Imprudence and Firestorm. This viewer permits the students to interact with object with the graphical user interface and scripting. Users gain instant visual feedback about what they change in the code (assuming there are no programming errors). In each of these environments, it is easy to create objects through a drag-and-drop interface. A graphics user interface enables each manipulation of properties of 3-D objects. However, the interface for writing programming code is not drag-and-drop but not as difficult as using a standalone compiler. Students can write scripts that are attached to each object. Depending on an event with that object, such as a "mouse-click" on that object, functions defined in the script will execute. Some debugging features are available and provide error messages if the script does not compile or there is some other error. Since this platform was more widely used, we could connect with other students across the nation in our virtual environment.

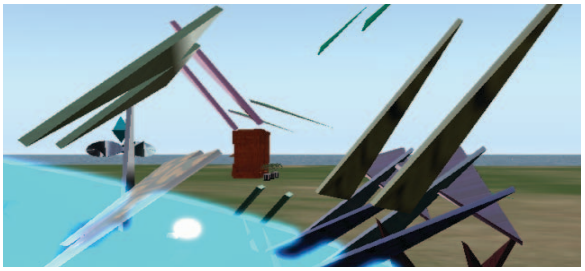

Figure 2: Example of creative 3D Spatial art in an OVE.

3.2 Art and Visual Design Problem Set

Five main problems focused on art and visual design principles. Each problem took about 1-2 days to complete. At the beginning of class each day, students were presented with information about the programming functions and other code that would be useful to them to be able to solve the problem. Students were provided the online API and documentation as reference. The problems were focused on 1) basic art elements: form, lines, shapes, color, value, texture and size, 2) design principles part 1: scale, repetition, rhythm, movement, hierarchy, and dominance 3) design principles part 2: composition, harmony, unity, variety, and balance 4) design principles part 3: positive and negative space, contrast, and gradation, 5) final art piece. In lesson 1, students were given instruction on the basic art elements and asked to create objects and use both the user interface and the scripting code to modify these elements of the objects. In lessons 2, 3, and 4, instruction and examples were provided on each respective set of visual design principles. Students were then asked to create a virtual art piece or 3D Sculpture that reflected one or more of those principles using the programming scripts as much as they could. An example art piece is shown in figure 2, solving for shapes, repetition, and unity. In lesson 5, students were asked to focus on the design principles we learned to create one final creative piece. They could build upon previous pieces by reusing their code or learn new concepts. To inform and provide inspiration for lesson 5, during our field trip to

the art museum, students completed a deep looking exercise that fostered taking notes on each of the art pieces that mapped to each of the visual design principles. They could use these notes to inspire their own use of the principles. An example that resulted from this inspiration is shown in Figure 3, where the art work displayed in the museum consisted of paper boxes with these intricate stairs that allowed for interesting lighting and showing effects.

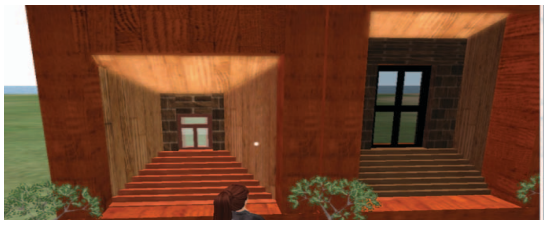

Figure 3: Inspiration from local artists driving concept of this piece.

```
touch_start(integer num_touches)
{

    for(i=1; i<=5; i=i+1)
{
    if(level<5)
{
    vector direction=<llFrand(1), llFrand(2), 2>;
    rotation rot=get_rotation(<1,0,0>, direction);
    direction=direction/llSqrt(direction*direction);

    llRezObject("Cynlanderrs1", llGetPos()+direction*2.5, ZERO_VECTOR, rot, level);
    llRezObject("New KG", llGetPos()+direction*5, ZERO_VECTOR, rot, level);
}
```

Figure 4: Example script to produce and rotate a virtual object.

3.2.1 Example Lesson in an OVE

In this example lesson, students are taught about positive and negative space and asked to explore visual solutions to that design principle. Students create 3-D virtual objects and write programming code in the computer lab classroom. Students are writing code to solve a visual design problem (Figure 4), as described in section 3.2. An example visual design problem is described in the next paragraph. Students will do this using the interface of the OVE and writing programming scripts within the OVE (Figure 4). The interface provides a window into the 3-D virtual world displaying real-time updates and interaction with the virtual world in first-person and third-person views. Side/bottom windows are provided for viewing and editing scripts. Students could directly edit their 3-D sculptures using OVE interface tools and add scripts to them. Students worked during the week through desktop displays but, once a week through embodied learning, they experience and refine their creative works in the OVE displayed in an immersive environment, such as a head-mounted display or CAVE. For example, several students commented about not realizing the size of different objects until they experienced them in an immersive environment. Students refined their designs and it changed how they understood the spatial components to improve the design of their subsequent projects. We followed this process of desktop development and immersive refinement through embodied learning due to the limited number of immersive displays. In the future, we could investigate if full-time access has an even stronger influence on learning.

One sample of a student's Virtual Environment art piece/ 3-D virtual sculpture is shown in Figure 5, where a student used complex programming code to produce the specific desired creative goals in positive and negative space. The composition of this piece is complex as well as the color decisions and animation sequences. In this example, a student used the OVE interface to model one virtual character. Then, the student used scripting to replicate and animate/rotate the other black and white virtual objects. The program uses a loop, if/else statements, and OVE script functions to accomplish the goal. The following describes the sequence of programming code to produce the image in Figure 5. All black and white character objects are facing the same direction before any object is clicked. The full color character in the scene is the primary/control virtual object and contains the primary script. When the color character is clicked, the "touch_start" function executes and loops the llRez function (Figure 4) to create each additional black and white character. Then it passes a message with a variable to each of the objects using a llSay function to rotate and change color. Based on the value passed, it rotates around the y-axis (up vector) by that amount and changes color using llSetColor.

Figure 5: Example of OVE Spatial Art piece.

3.3 Computing Concepts Problem Set

To compare to the Art and Visual Design problem set, we created a problem set that would focus more primarily on computing concepts first, rather than the creative work in a virtual environment. There were five main problems given in the 'traditional' course, specifically related to generative art. The problems used were focused on solving the programming challenges first, instead of visual design challenges. Each problem took about 1-2 days to complete. Students were presented with information about the functions and other code that would be useful to solve the problem. Students were encouraged to look up programming code in the provided documentation and online API. The problems focused on 1) random lines, 2) random circles and spirals, 3) message sending between objects, 4) fractals of trees, 5) generative art final project. For lesson 1: random lines, students were asked to create an object that would generate a line in 3D space when clicked. Students created two objects, such as two spheres, one serving as a starting point and the other as a target. When the start object was clicked, that object executed a script which would produce a cylinder or multiple cylinders to create the 3D line. Lesson 2 was a variation on lesson 1, where they would formulate random circles and spirals with the lines. Functions such as llRezObject, llSetPosition, llSetScale, and llRand (to create random directions) were used in lessons 1 and 2, as well as vectors, variables, etc. In lesson 3, students were presented with the problem using clicks to control the change in position, size, color, and other parameters of the objects. In lesson 3, llListen, llSay, and llSetParams functions were presented. We introduced if/else statements. For example, to test for a set number of clicks and then make a change. In lesson 4, students were given the problem to use an object to generate a pattern of objects. For example, when a sphere is clicked, two cylinders are produced in opposite directions. When clicked again, two more cylinders are produced and so on. The action could be repeated over and over, resembling a tree. A variation of this was to have it duplicate copies of itself. Loops were introduced in this problem. Finally, the fifth lesson, the students were tasked with creating their own generative art piece. Students could reuse parts of previous solutions or create a new piece. At the end, generative art pieces were displayed in an OVE virtual gallery.

4 Experimental Study of Knowledge Acquisition

4.1 Procedure

We taught a course, "Generating Art in a Virtual World", for the University of Wyoming High School Summer Institute (UWYO HSI) using an OVE for three consecutive summer semesters, each

with a different sample set of high school students. As part of the UWYO HSI program, high school students were selected from Wyoming high schools based on their applications to the program Students choose to participate in one science class and one humanities class. Their daily schedule of the UWYO HSI program is as follows: in the morning students attend one of those classes (either humanities or science), have lunch, attend the other class (science or humanities), and then have social activities and free work time in the evenings. Approximately 14-18 students on average have filled our course. We used an OVE (described in section 3.1 Platform) in a classroom setting during students' enrollment in the UWYO HSI program and then students continued working remotely in the OVE from home for the remainder of the summer. Each of our classes were scheduled for four out of five days in a week, for a duration of three weeks on campus. The duration of the class was about 2.5 hours each class. During most class days, students worked in the OVEs on standard desktop computers in the classroom. On one of those four days, each week we took them on a field trip. Field trips that we included in our course were the UWYO 3D Shell Visualization Center, 3-D Interaction and Agents Research Lab, and the UW Art Museum. Students were exposed to their creations in an immersive display during three separate occasions, once each week. More details on what students did during their time in class are provided in section 4.1.1 Classroom Activities. On the first day, students were given the pre-questionnaire as the very first task. Then we spent that day learning the basics of the OVE platform. During the last week, students completed the last assignment (described in sections 3.2 and 3.3). Virtual sculptures were exhibited in the OVE virtual gallery and visitors are invited to the special event through immersive displays. The last task of the last class day, students completed the post-questionnaire and exam on the programming concepts. Students had the option to opt out of any or all of the pre-questionnaire, post- questionnaire, and exam. After the three-week period at UWYO HSI, students return home but are provided online access to the space in the online virtual environment designated for this project outside of class so that they will be encouraged to explore their creativity and programming skills outside of class in collaboration with other students.

4.1.1 Classroom Activities

We structured the scheduled class time of 2.5 hours with a presentation of the material orally and visually by one of the instructors in the first 30 minutes and for the remaining of the time students design and program their projects. During each design and program work periods, the students spent time working in the virtual environment on the classroom computers, not in an immersive display. To solve the problems presented, students created 3-dimmensional (3D) graphical representations (objects) and wrote computer programs that activate/control each objects' behaviors within the OVE. During the design and program work periods, two instructors and two undergraduate student helpers walked around the room providing one-on-one support. Both instructors were skilled in computer science. One of the instructors was also skilled in fine arts. At least a graduate or undergraduate student from both fields should be involved if an instructor from both fields is not possible. The duration of a project lasts for about one to two class days and we move on to the next problem. If we noticed students needed more time, an additional half-to-full class day would be used as general work time so students could spend more time on their art projects, individually or in small groups of students. These days gave us more opportunities to work with the students one on one, so we can help them succeed at their projects. In a typical day, the students were taught a new programming technique or basic art concept, taught about a new aspect of art, worked on a project to apply the new techniques to an art project, and shared their artwork with the other students in the class. Students also offered solutions to assist their peers and worked in pairs or small groups of students (this was optional and not forced through the classroom activities). One week, students experienced their individual creations immersed in the OVE through a head-mounted display with wide-area tracking offering the ability to naturally walk around a 20'x20' space. Another week, students experienced their individual creations immersed in a CAVE facility also with wide-area tracking in the open space of the CAVE. In the last week, we brought two head-mounted displays to the classroom where they could explore the OVE art gallery containing each other's work using the classroom computers and taking turns using those immersive displays with using a keyboard for navigation through the environment. Through these immersive experiences, students could further their embodied learning by experiencing their 3D sculptures relative to their own bodies. While it was not feasible at the time, for the students work the entire time in head-mounted displays, in future courses, we can investigate more immersive exposure in the classroom setting using this teaching and learning approach. When students returned home from the program, they used desktops and low-cost portable virtual reality viewers, such as a Google Cardboard to continue designing and interacting with an instructor and students. Another example of how OVEs can be used to enhance peer support is to connect with other networked classes. During one year, we connected to Clemson's Palmetto Island group, who was using the platform to teach computer science concepts through single object lessons with note cards in the environment providing tutorial instructions. One of their objectives is similar in that the purpose of their program is to engage students and raise awareness in Computer Science [23]. Our course is different because we use visual art and design pedagogy to drive the lesson-plans. Despite the differences in curriculum and assignments, students from our group could engage with each other with students from the Palmetto Island group, participate in a fun scavenger hunt together within the virtual environment, and share programming tips fostering peer support.

5 Results

The UWYO Institutional Review Board approved data collection activities and only used data which we had student consent or appropriate parental consent and student assent, depending on age. We used an analysis of variance (ANOVA) to test difference of means among the data. For all ANOVAs, we used a p=0.05 for significance. Each ANOVA passed the Mauchly's Sphericity test, where variances of differences between possible pairs were equal.

5.1 Participants

Each course enrolled a unique set of students about 14 to 18 students, however data collection for the pre- and post-questionnaires, and tests were voluntary. We have collected data from 57 students over three courses. Course A is the traditional course while course B we started experimenting a little with Art Pedagogy Driven model. In course C, we had fully implemented a curriculum of Art Pedagogy Driven model. For the course where we used generative art computational problems, we had data for 14 students (Males=7, Females=7). For the course which we used generative art computational problems, we had data for 14 students (Males=7, Females=7). During the other three courses we collected data from 16 students (Males=6, Females= 7, Not Reported= 3), 12 students (Males=3, Females=6, Not Reported=3), and 14 students (Males= 5, Females=7, Not Reported=2). All students reported that they were Juniors in High School. The mean number of computing courses reported was 1.81 (SD=1.35).

5.2 Actual Knowledge Gained

To assess actual knowledge gained, students completed a quiz relating to scripting and computational topics on the last day of the course for all three courses. There were two parts. In part one, students described what each code snippet did. Some questions featured code specific to Linden Scripting Language while others focused on general computing concepts, such as if/else statements and loops. In part two, we collected qualitative data on how well students knew the Linden Scripting Language where snippets of code were written incorrectly, then students had to identify what was missing or incorrect. These tests were optional and one class was not administered the test. The results of a one-way ANOVA revealed a significant difference for the performance on test scores $F(1,37)= 11.83$, $p=0.002$. The courses where we used our Visual Design Problem-based Pedagogy courses yielded significantly higher test scores (M=76.89%, SD=14.48) than the course that used generative art computational problems (M= 55.87%, SD=22.23). A multivariate ANOVA was used to compare test scores among the courses. The ANOVA revealed a main effect of course type, $F(2,30)= 5.91$, $p=0.007$, $n^2= 0.28$, power= 0.841 (Figure 5). There was no main effect of gender found and no interaction effect of course type by gender found. A LSD post-hoc test revealed that the two Art Pedagogy-Driven courses produced higher knowledge-based test scores than the traditional course. Specifically, course C was found to be significantly higher (M=81.73%, SD=18.21) than course A (M=55.80%, SD=22.23), p=0.003. While some improvement in course B (M=66.14%, SD= 14.32), there was not a significant difference found. *In summary, students' actual programming knowledge and comprehension (specifically of state_entry, beginning a script, positional/rotational commands, property manipulators such as color, random number generator, and vectors), understanding (i.e. differentiating when an end bracket was missing, if a parameter name was missing, or for vectors), significantly increased from pre- to post-course activities during the Visual-Design Problem-based Pedagogy courses B and C than course A, using traditional CS problems.*

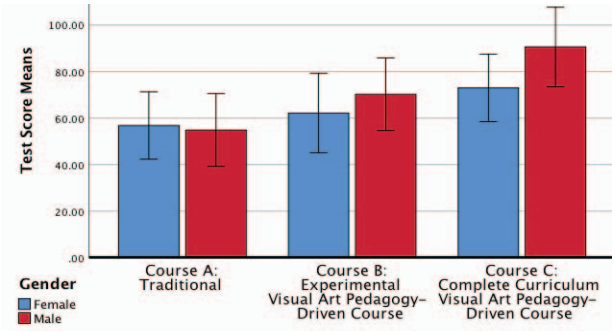

Figure 5: Test Score Means by Course and Gender Type.

5.3 Perceived Knowledge Gained

To assess perceived knowledge gained, students rated their confidence on knowledge of terminology and were quizzed on specific content. We collected this data using a 5-point Likert scale from 1=not confident at all and 5=very confident. There was also an option to select 0 to indicate not applicable or never heard of the term. The topics included: variables, loops, conditional statements, compilers, recursion, random number generators, object oriented programming, functions, parameters, software, hardware, input device, output device, variable declaration, initialization, booleans, execution, iteration, syntax, comments, debugging, arrays, and code errors. A Repeated Measures ANOVA revealed a main effect of perceived knowledge by course type for each topic, as follows:

- *Debugging*: $F(2,38)=3.15$, $p=0.05$, $n^2=0.142$, power=0.570, where perceived knowledge of Debugging was significantly higher for Courses B from pre (M=0.55, SD=0.82) to post (M=1.45, SD=1.22) and Course C from pre (M=1.64, SD=1.69) to post (M=2.71, SD=1.60) than Course A from pre (M=1.25, SD= 1.13) to post (M=1.87, SD=1.41).
- *Arrays*: $F(2,38)=3.40$, $p=0.044$, $n^2=0.152$, power=0.604, where perceived knowledge of Arrays was significantly higher for Courses C from pre (M=0.43, SD=1.45) to post (M=2.50, SD=1.79) and Course B from pre (M=0.45, SD=0.69) to post (M=1.36, SD=1.12) than Course A from pre (M=0.69, SD= 1.01) to post (M=1.38, SD=1.46).
- *Loops*: $F(2,38)=4.82$, $p=0.014$, $n^2=0.216$, power=0.763, where course B had the most significant increase from pre (M=1.38, SD=0.83) to post (M=2.82, SD=1.08) and course C also had a significant increase from pre (M=1.50, SD=1.65) to post (M=2.21, SD=1.48).
- *Conditions (programming)*: $F(2,38)=4.23$, $p=0.022$, $n^2=0.182$, power=0.707, where course B had the most significant increase from pre (M=0.64, SD=1.03) to post (M=2.91, SD=1.04) and course C also had a significant increase from pre (M=1.50, SD=1.74) to post (M=2.57, SD=1.45). Course A (traditional) only increased very slightly from pre (M=2.00, SD=2.03) to post (M=2.56, SD=1.50).
- *Hardware*: $F(2,38)=3.75$, $p=0.033$, $n^2=0.165$, power=0.650, where course B had the most significant increase from pre (M=0.82, SD=1.08) to post (M=2.45, SD=1.30) and course C also had a significant increase from pre (M=1.57, SD=1.09) to post (M=2.29, SD=1.20). Course A (traditional) only increased very slightly from pre (M=2.06, SD=1.34) to post (M=2.37, SD=1.20).

A Repeated Measures ANOVA revealed a main effect of perceived knowledge by gender type for each topic as follows: *Booleans*: $F(1,35)=5.82$, $p=0.021$, $n^2=0.143$, power=0.650, where females perceived knowledge of Booleans significantly increased from pre(M=0.40, SD=0.68) to post(M=1.20, SD=1.32) more than males from pre (M=1.33, SD=1.65) to post (M=2.00, SD=1.38).

In summary, perceived knowledge (of debugging, arrays, loops, conditions, and hardware) significantly increased from pre- to post-course activities in Visual-Design Problem-based Pedagogy courses B and C than in course A, using traditional CS problems.

5.4 Student Comments and Feedback

The following are student comments on engagement and learning:

- "I didn't know a lot [about computer science and programming] and this class helped me learn a lot more"
- "I actually know how to make a simple code now. I never thought I'd learn how to do that"
- "I would like to keep the combination of 3D art and coding"
- "learn all the cool things that could be done in a virtual world"
- "we got to use our imaginations", "I am better at scripts"
- "I have become a little more comfortable with computers in general and was able to have a little fun with my creativity"
- "experimenting to find out just what you are capable of doing"

5.5 Empirical Results

In the Visual Design Problem-based Pedagogy courses, students explored various programming concepts to achieve their desired visual designs. Students planned their creations, then identified the programming concepts to achieve their desired affects. Students would investigate other concepts to try to achieve the desired effect. We observed three major observations: 1) were more engaged, 2) were more creative in their solutions, and ultimately 3) who independently sought to learn more additional computer programming concepts on their own. Students were more engaged and enjoyed their tasks more. Students were tending to skip their break sessions, come back earlier or stay later after class to complete their tasks. When teaching lessons directly related to

programming concepts, students were less creative in their resulting visual elements and code. For each assignment, students' code was very close to a single solution, with only slight variation from the examples. For Visual Design Problem-based Pedagogy lessons, students exhibited more creativity in their resulting visual elements. Most importantly, the code produced by students was more sophisticated and complex (added programming elements that were not covered in the lesson which they learned on their own through the wiki pages, online tutorials, and forums). There was an increase in variation across variable naming conventions, code ordering structure, and algorithmic solutions.

6 Discussion and Conclusion

Results on measured knowledge gained show that the curriculum that incorporates the use of problem solving for visual design first, where programming is used to solve those problems, helps to improve knowledge on programming concepts for students of both genders. The results of measured computer programming knowledge gained is a direct result of the presence of self-directed learning of computer programming concepts. It has also been detected by observations of students' actions and behaviors. Self-directed learning of programming has been encouraged purposefully through the pursuit of trying to achieve desired visual effects. The desired visual effects are determined as a solution to address a problem relating to visual design principles. Therefore, we can deduct that by using a Visual Design Problem-based Pedagogy as a means for encouraging self-directed learning, can lead to more advancement of knowledge in computer programming concepts. This may extend further to other STEM fields but as a limitation to this study and the procedures used, would need to be investigated in the future. Results on perceived knowledge gained, as well as the positive comments deriving from students' opinions about the method, tell us that through an experience such as this for high school students, regardless of gender, increases the confidence of students on computer programming concepts.

This paper presents an interesting use of Visual Design Problem-based Pedagogy to drive experiential learning and embodied learning for computer science knowledge acquisition. Using this method, students learned more about introductory computer programming concepts. Additionally, we found through observations that students were more engaged, enjoyed completing their tasks, and took charge of their own learning of new concepts. When students' goals are driven by creativity and experiential learning, students learned more about computing even if the tasks were challenging. The results of this work may assist with other educators wanting to conduct similar activities. Using online virtual environments provides students with the capability to connect with other students online for collaboration and peer support to further assist with learning computer science concepts. Our hope is that this paper will serve to share our experiences and encourage other educators to develop and host these types of courses. While our course was delivered to high school students, it may be appropriate for undergraduate introductory level courses and potentially (with modification) to middle school students. In that respect, as the community grows, we can blur our geospatial borders and learning through peer collaboration and support will increase.

References

[1] Barker, Lecia J., Kathy Garvin-Doxas, and Eric Roberts. "What can computer science learn from a fine arts approach to teaching?." ACM SIGCSE Bulletin. Vol. 37. No. 1. ACM, 2005.

[2] Brunvand, Erik. "Arts/tech collaboration with embedded systems and kinetic art." ACM SIGGRAPH 2013 Talks. ACM, 2013.

[3] Schmidhuber, Jürgen. "Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts." Connection Science 18.2 (2006): 173-187.

[4] Bell, Mark W. "Toward a Definition of "Virtual Worlds"". Journal of Virtual Worlds Research (2008): 1 (1).

[5] Cruz-Neira, Carolina, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. "The CAVE: audio visual experience automatic virtual environment." Communications of the ACM 35, no. 6 (1992): 64-73.

[6] Springer, Leonard, Mary Elizabeth Stanne, and Samuel S. Donovan. "Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis." Review of educational research 69.1 (1999): 21-51.

[7] Hendricson, William D., et al. "Educational strategies associated with development of problem-solving, critical thinking, and self-directed learning." Journal of dental education 70.9 (2006):925-36.

[8] Deek, Fadi P., Howard Kimmel, and James A. McHugh. "Pedagogical changes in the delivery of the first-course in computer science: Problem solving, then programming." Journal of Engineering Education 87.3 (1998): 313.

[9] Williams, Laurie, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller. "In Support of Pair Programming in the Introductory Computer Science Course." Computer Science Education Vol. 12 , Iss. 3, (2002).

[10] Lieberman, Debra A., and Marcia C. Linn. "Learning to learn revisited: Computers and the development of self-directed learning skills." Journal of research on computing in education 23.3 (1991).

[11] LeJeune, Noel F. Problem-based learning instruction versus traditional instruction on self-directed learning, motivation, and grades of undergraduate computer science students. 2002.

[12] Magnussen, Lois, Dianne Ishida, and Joanne Itano. "The impact of the use of inquiry-based learning as a teaching methodology on the development of critical thinking." Journal of Nursing Education 39.8 (2000): 360-364.

[13] Gormally, Cara, et al. "Effects of inquiry-based learning on students' science literacy skills and confidence." International journal for the scholarship of teaching and learning 3.2 (2009): 16.

[14] Savery, John R. "Overview of problem-based learning: Definitions and distinctions." Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows (2015).

[15] Kolb, David A., Richard E. Boyatzis, and Charalampos Mainemelis. "Experiential learning theory: Previous research and new directions." Perspectives on thinking,learning, and cognitive styles (2001):227-47.

[16] Dickey, Michele D. "Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education." British journal of educational technology 36, no. 3 (2005).

[17] Baker, Suzanne C., Ryan K. Wentz, and Madison M. Woods. "Using virtual worlds in education: Second Life® as an educational tool." Teaching of Psychology 36, no. 1 (2009): 59-64.

[18] Warburton, Steven. "Second Life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching." British Journal of Educational Technology 40, no. 3 (2009): 414-426.

[19] Wiecha, John, Robin Heyden, Elliot Sternthal, and Mario Merialdi. "Learning in a virtual world: experience with using second life for medical education." Journal of medical Internet research 12, 1 (2010).

[20] Patton, Ryan M. "Games as an artistic medium: Investigating complexity thinking in game-based art pedagogy." Studies in Art Education 55, no. 1 (2013): 35-50.

[21] Ellmers, Grant. "Reflection and graphic design pedagogy: Developing a reflective framework to enhance learning in a graphic design tertiary environment." (2006).

[22] Ellmers, Grant, I. Brown, and S. Bennett. "Graphic design pedagogy: Employing reflection to support the articulation of knowledge and learning from the design experience." In Proc of the Conference on Experiential Method, Knowledge and Methodology. 2009.

[23] Green, Brittany, Charles Jones, Larry F. Hodges, and Kaylee Nichols. "Palmetto Island: Developing Computer Science Awareness in Middle and High School Students."