# Promoting Computer Science with Video Games:

## Teaching Object-Oriented Programming through Unity

By Dylan Graves

A Senior Project submitted

In partial fulfillment of the requirement for the degree of

Bachelor of Science in Computer Engineering

Computer Engineering Department

California Polytechnic State University

San Luis Obispo, CA

June 2015

## Abstract:

In recent years, the demand for people educated in computer science has continued to grow as the possible applications of software expand. For this reason, it is important to not only offer young adults the ability to learn computer science, but also to intrigue them with interesting and engaging applications of computer science. Organizations, such as the CollegeBoard and Exploring Computer Science, have been working to make computer science education more available and desirable to younger audiences. This project contributes to these goals by creating and implementing a curriculum for teaching object-oriented programming to high school students through video game programming. Students were taught how to use a third-party game engine called Unity to create complex 3D games. The curriculum was designed to provide students with a basic understanding of the tools and concepts they needed to make a game, and provided students with several hours of learning time to create their own games. This course was taught at San Luis Obispo High School to a group of 20 students. A survey was taken by 14 of the 20 students before and after the course. This survey revealed that half of the students who did not understand objects in programming before the course did understand objects after the course. The survey also showed that many students were able to learn C#, an object oriented programming language, during the course. Lastly, most of the students ended the course with a completed project that could be added to a portfolio.

## Introduction:

The professional field of computer science has been growing, and it seems evident that it will continue to grow. This makes it increasingly important to offer an education in Computer Science to young adults, so that they can continue advancing technology as they enter the work force. In response, many high schools across America have started to offer technology-focused elective classes and the Advance Placement program has offered AP Computer Science since 1984 ("AP* Computer Science: A Brief History"). As a subset of the engineering field, computer science can be very challenging for students. Engineering skills are different from the typical grade school courses because they rely heavily on a set of problem solving skills that many grade school students haven't developed yet, which leads some students to avoid computer science altogether. For this reason educators need to not only offer courses in computer science, but also intrigue and inspire students to pursue computer science. In this way students will develop a passion for computer science.

It is fairly common today that young adults have a passion for video games. Also video game programming showcases many of the concepts essential to computer science. For example most game engines use object oriented programming to handle objects in the video game world. This project intended to use students' passion for video games to help them engage in learning about computer science by creating games of their own. By having the focus of the course be game creation, students have a result that shows them how their programming skills come together to create a comprehensive program. In addition, it was hoped that the desire to create an interesting game to show to their friends would inspire students to learn outside of the classroom.

The amount of time and experience required for creating a video game makes implementing the class described a challenge. For the most part these problems can be solved by having the students learn by using a third-party game engine. A game engine is the core software of a video game containing all generalizable aspects of a game. A typical game engine will have systems for managing resources, rendering, input, resources, physics, and script interpretation (Thorn). Since the game engine is such a vital part of a game, it is necessary to have the engine in place before much of a game's development really begins. This is why having students build off of an already established game engine allowed students to start game development quickly, making it more feasible to accomplish the course's goals in the ten weeks provided.

There are several game engines available to the public, with the most widely known being Unity3d and Unreal Engine4. Of these two engines, Unity3d is currently holding 45% of the market share in game industry software ("The Leading Global Game Industry Software"). In addition to having a large market share, Unity3d has a plethora of information on the web that students can access to advance their knowledge further than any ten-week course would ever be able to. Therefore in this course high school students were also exposed to the important skill of self-driven learning.

San Luis Obispo High School (SLOHS) recognizes the importance of offering courses to their students that will prepare them for college and life. For this reason, SLOHS offers a variety of courses in computer science and information technologies. The computer science teacher at SLOHS, offered her programming III class as a platform for teaching high school students about Unity.

## Previous/Related Work:

### Computer Science Curriculums:

Since the influence of computer science has grown in the technology industry, many educational organizations have been trying to make effective curriculum to get students involved. For example, Exploring Computer Science (ECS) is an organization that looks to expand the computer science education opportunities in the Los Angeles School District (Exploring Computer Science). ESC works with teachers who are already established in the school system to enhance their curriculum to better suit the students.

The Advanced Placement (AP) program, by the CollegeBoard, has also created a computer science curriculum that is well respected in the public education system. This curriculum is designed to teach students the fundamental concepts of computer science using the Java programming language. It covers important topics like object-oriented program design, data sturctures, algorithms, etc (CollegeBoard). In 2011, about 2000 high schools across the nation offered AP computer science to their students, and with the success of the curriculum that number has been growing ("CS Education Statistics"). In fact, the AP computer science curriculum is widely accepted by colleges as a way to earn college credits for successful students (CollegeBoard).

### Unity Resources:

Unity was first released on June 6$^{th}$, 2005 and over the past ten years has not only expanded and improved their engine, but also improved the resources available to learn about their engine. Unity has become a very popular choice for independent developers for many reasons, and at least among new developers one of the top reasons is the sheer amount of

learning material published for Unity. Unity technologies has a large collection of tutorials and lessons, including live lessons, that have been created by their team and posted on their website to educate people on the tools that Unity provides ("Learn with Unity"). In addition, Unity Technologies hosts a forum on their site where any of the 4.5 million developers that use Unity can posts tutorials or help other members with problems ("The Leading Global Game Industry Software"). Unity was chosen for this project because of the large amount of learning content that is easily accessible to students.

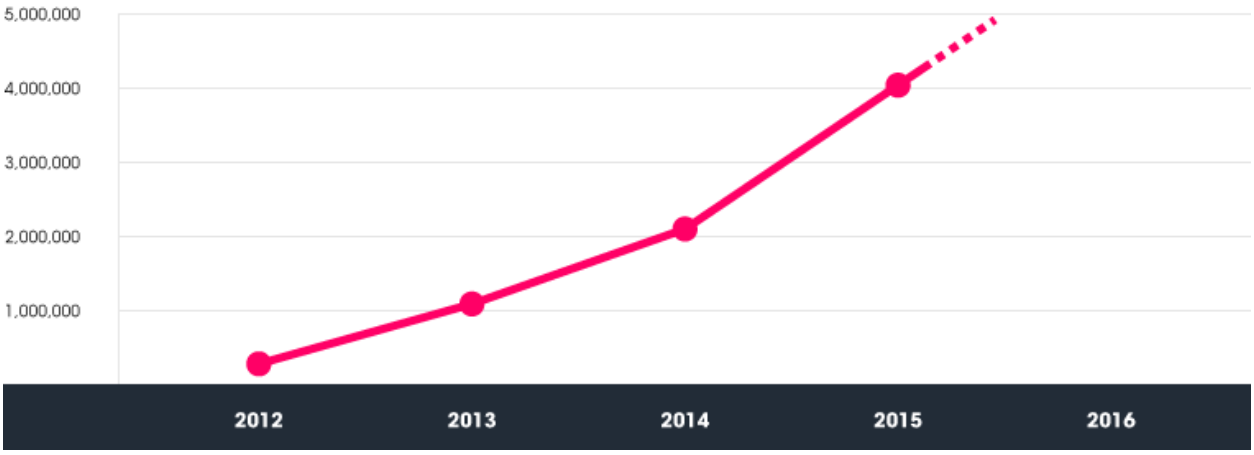



Figure 1: Registered Unity Developers from Unity3d.com

## Class Structure:

The goal of this project was to create a class curriculum that used Unity3d to teach object orientation through the medium of video game programming. This curriculum was taught at San Luis Obispo High School in the ten-week long Programming III class consisting of 20 students. The curriculum was broken into four phases that gradually transitioned from lecture-based instruction time to individual and teamwork time. At the start of the course, the instructor taught students about the tools available to them, and by the end students were using the tools they had learned to build a finished game.

## Phase 1: Object Orientation and C#:

Phase one took about two weeks and was about the programming language C#, which is used to write scripts in Unity. Since the students already had experience writing programs in Visual Basic, these lectures covered the syntax differences and an introduction to objects. Additionally it was important to start showing the students how to use Unity's documentation as soon as possible. For this reason parts of the lessons would include "documentation races", where the students would be asked to find a function in the documentation that could solve a problem presented in the lesson.

This phase also introduces students to the concept of objects in C#. Objects are very important for programming in unity because almost everything in the game is a GameObject. Students needed to understand how to create a class and how the properties and the functions of the class related to objects of that class. Since there would likely be several instances of the same GameObjects in their games, it was important for students to grasp the idea of instantiation where there can be several instance of one class where they are all distinct from each other.

## Phase 2: Beginner Tutorials and Short Lectures

Phase two of the class was where the majority of the direct instruction was given through tutorials and lesson plans. The first lessons came directly from the Unity Roll-a-Ball tutorial (see figure 2), which covers many of the very basics of using the Unity editor. After completing the Roll-a-Ball tutorial students should have understood how to navigate through the Unity interface and a few scripting concepts, including collisions and rigid bodies. Most students were very happy with the simple game they had created, so they were given another few class periods to add their own touch to the game. They were given about three hours of class time and were only

restricted by what they were able to implement, with some help from the instructor. By the end of the class time students should have had a unique Roll-a-Ball (See Appendix A), with at least one new feature that was not a part of the tutorial.

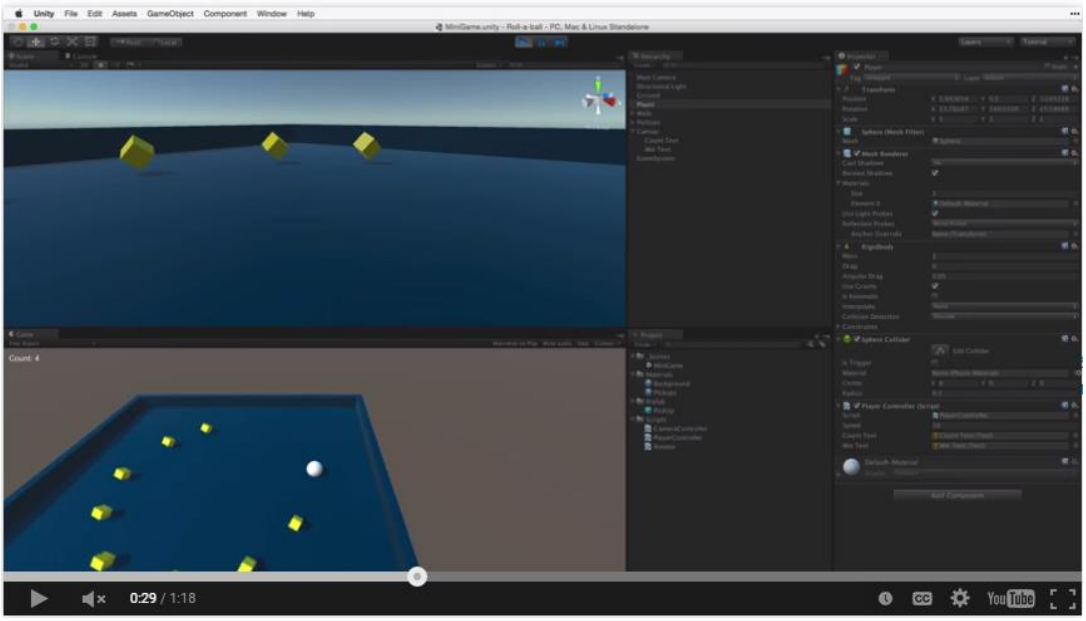



Figure 2: Screen Capture of Roll-A-Ball tutorial

Lectures were intended to equip the students with the basic tools provided by Unity and to provide an example of how to use the tools. The class was designed to be very hands-on for the students, allowing them to learn through their own experience. For this reason, lectures would be short and cover one or two topics so students had more time to actually use the tools. The lesson plans were adapted from a lesson plan posted on the unity forums by the user that goes by the user name AnomalusUndrdog.

The lectures topics were coved in the following order:

- Creating objects – Taught students about the primitive GameObjects in Unity and how to keep track of them in the Unity editor. Students to learn how to change properties in the inspector window, to organize objects in the hierarchical window,

and how objects can relate to each other. Parenting objects was an important concept for the students to understand early in the class.

- Prefabs – Explained how prefabs allow students to create many of the same object and save that object as a Prefab. Showed the similarity between classes and objects to prefabs and objects.
- Scripting – Introduced students to writing small scripts in C# that gave some more logic to their GameObjects. Made sure students understand the Start(), Update(), FixedUpdate(), and LateUpdate() methods.
- Colliders and Tags– Taught students how to use the OnColliderEnter(), OnColliderStay(), and OnColliderExit() methods. Showed them how to determine what type of object was hit by comparing the object's tag to a given value. Students to have a strong understanding of what events will cause this code to execute.
- Triggers – Similar to colliders with the important methods of OnTriggerEnter, Stay, and Exit. Showed students how a collider can be used as a trigger.
- Graphical User Interface (GUI) – Taught students to create a simple GUI that consists of text information such as score or health. Students to understand the key feature of Unity's new user interface system, the use of canvases and rect transforms.
- Materials – Created new materials in the project tab to make games more colorful. This lecture also taught students about textures and bump maps. Created a brick wall that is a cube with a texture and bump map.
- Character Controllers – Showed students how a character controller is similar to, but also different from, a collider with an attached controller script. Important methods

include: OnCharacterControllerHit and IsGrounded(). Created a script to control character jumping.

- Sound – Taught students how to import sound files and add them to their games. Talked about the difference between 3D sounds and 2D. Adjusted the settings on a 3D sound to see how the settings affect the sound.
- Animation, and Animator Controllers – Used the animation tool to create a simple animation that changes the scale of a GameObject primitive. Applied an animator controller to the GameObject to run the animation. This was also a great opportunity to introduce students to the concept of a finite state machine. Talked about the transitions between states and the conditions that cause those transitions to occur.
- Nav Mesh Agents – Showed students how a nav mesh agent can be used to create a simple artificial intelligence. Creatde a simple scene out of primitive objects and made sure all obstacles were labeled as static, and then baked a nav mesh using the navigation tool. Applied the nav mesh to a nav mesh agent on an object and set the agents destination to the position of the player.

## Phase Three: Starting Projects

Phase three overlapped a little bit with phase two in that lectures continued to be given, but at this point students formed groups of three and designed a game as teams. The first important thing that students needed to learn here was to keep their game within scope, not trying to build a masterpiece, but rather trying to build off of what they had already learned. It was also important to keep in mind that with only ten weeks for the entire course, students would only have seven weeks to build a complete game. The video *How to start your Game Development* by Extra

Credits was very helpful in communicating the idea of scope and demonstrated a good attitude for students to have when approaching this game-making stage of the course.

At this point in the course the majority of the class time was given to the students to work with their groups on their games. Lectures were then focused on more advanced topics such as bump mapped materials, animations, and navigation meshes.

### Phase 4: Group Projects

The last phase of the class was four to five weeks to give the students freedom to work with their teams in order to complete their games. The instructor became a helping hand to anyone who was stuck or needed some direction. Since there were likely several students that needed help at any given time it was beneficial to encourage students to help each other. It was also helpful to point students to resources online, such as the Unity tutorials or documentation, so that they could learn from a resource that would almost always be available. One problem with this phase was that some students were just not motivated and would try to use the time for other classes, making their teammates do all the work. This was really just a problem with group dynamics in general for future classes adding more incentive for all members to participate in the group would help solve this problem. One method to solve this problem would be to allow each team to evaluate the work of each member and have those evaluations affect the individuals' grades.

## Results:

### The Survey:

This class was taught to the Programming III class at SLOHS with a total of 20 students ranging from sophomores to seniors. These students also had a range of programming experience with some coming from AP Computer Science where they had learned about java and object orientation, while others coming from the Programming I & II classes where they had learned about visual basic and very little about objects. Before the course started, students were asked to define the relationship between classes and objects. Of the 13 participants, four students were unsure of the answer and two skipped the question. The same question given after the course ended found that only two students did not know the answer and one other still had an unclear answer (see Table 1). From this we can conclude that the course was effective in teaching students about objects and object orientation. If more time was spent showing students the Instantiate method, which creates a new object of a given type, then more students would effectively learn about objects.

**Table 1 Responses to Object Question (number of students)**

| | Good Answers | Weak Answers | Skipped |
|---|---|---|---|
| Before Class | 6 | 3 | 5 |
| After Class | 10 | 2 | 2 |

Another learning objective of the class was to introduce students to the programming language C#, which is an Object Orientated language used in unity for scripting. The success of the students learning C# is the most evident result of the survey. Coming into the class, only two students said they knew C#, and by the end ten students said they knew C# (see tables 2). Even the students who did not learn the language had at least been exposed to it and now have experience working with C# code. From this we can conclude that there is validity in using this

course to motivate to learn students a new programming language. In addition, four students decided to use the time in class to learn python. One of these students used python to create a text-based adventure game. It is important to note that none of the students were brand new to programming; most of them knew how to program in Visual Basic.

**Table 2 Known programming languages before class**

| Language | # Before | # After |
|---|---|---|
| Java | 8 | 6 |
| Javascript | 1 | 2 |
| C | 1 | 1 |
| C# | 2 | 10 |
| C++ | 2 | 2 |
| Python | 2 | 6 |
| Visual Basic | 13 | 13 |

The survey was also designed to gauge students' interest in video game development and whether they would continue to pursue game development. The survey showed that even before the class had started, only two of the 14 students would not be interested in game development as a hobby or a career, and of the 12 other students, seven said they would like to create games as both a hobby and a career. This shows that there is definitely already an interest in the video game industry among high school students. This gives reason to believe that students would be intrinsically motivated to learn the material in a course that teaches them how to continue in a career path where are interested. This question had the same results after the class with one less person saying they were interested in neither a hobby or career in video games and one more

saying they could see it as a hobby (see Table 3). Clearly the course was in line with students' interests and potential future careers. They were given an opportunity to get started early in a subject matter that they could see being applicable to life.

**Table 2 Interest in Video Game paths**

| | Hobby | Career | Both | Neither |
|---|---|---|---|---|
| Before | 5 | 0 | 7 | 2 |
| After | 6 | 0 | 7 | 1 |

One thing the survey did not test for was a correlation between the students' interests in video game development and their interest in other computer science topics. It would have been useful to include a question asking about interest in computer science in general. This would allow us to see if interest in computer science would increase after seeing computer science applied to video game development. The results of the survey can show us that students will likely continue to learn computer science topics in order to continue their game development, since game programming requires computer science. There was also one student who decided to start learning about network programming in C# because he wanted to apply it in a game context. Another student learned about namespaces and hierarchical structures which he used to implement his real time strategy game. These two are examples of how an interest in game programming can motivate students to learn complicated subjects in computer science.

## The Games:

The games that the students created varied greatly in theme and technical challenge. This difference in the quality of games was very much due to the laissez faire structure of the class.

Students that were dedicated to creating a complicated and fun game would use their time more effectively because they realized their goal required their time. Other students who didn't have as much motivation did minimal amounts of work. Overall however every student did end the course with a game that they could present to the class.

Some of the more successful team projects included a cart racing game, a real time strategy (RTS), and a role playing game (RPG). Interestingly each of these games focused on excelling at different aspects of game development while all demonstrating the basic functions of a game. The cart racing game, called Shmario Cart, did a lot of research on creating realistic automobile physics, modelling, and spline curves. This resulted in them having a realistic cart controller and an enemy cart that followed a set path. On the other hand, the RTS team focused on creating an object orientated system to make their game scalable in the future. They ended up with a simple map, but a very large user interaction that allowed users to select different units and tell them actions to perform. In the near future, one member of the team hopes to add a more sophisticated artificial intelligence to make the game more entertaining. Lastly, the RPG team focused on creating visually engaging effects, including terrain and particle systems. One member of the team became a master at using Unity's tools to create an interesting map with multiple-textured terrain and different kinds of obstacles. At the same time, another member became an expert on using Unity's particle systems to create interesting spell effects, such as a very cool fireball. Also note that particle systems and terrain were not even a part of the lectures discussed in the curriculum section.

## Personal Reflections:

Overall this project was successful and worthwhile for me, the students in the class, and anyone else who may read about it later. This project gave me more experience teaching about subjects that I enjoy, which is something I hope to continue doing my whole life. One of the observations I made about my own teaching is that I need to slow down when giving lectures. I often rushed through the material mostly because of nerves or just lack of public speaking practice. It wasn't just my lack of experience in speaking that could have been improved, the curriculum could also use some revisions.

If I were to teach this class again, I would try to have more in-class assignments prepared for the students. Lectures were often too stand alone and didn't allow students to try implementing the tool I talked about until they saw they needed it in their own game, by which point they had forgotten how the tool worked. It also became apparent toward the latter half of the course that students would have benefitted from a more in depth study of C# in the beginning of the course.

Most students expressed that they had enjoyed taking the class, a few even giving a "10/10 would take again" review. Not only was the course enjoyable, but it gave these students a unique experience that they can talk about in future in potential future interviews. The more successful games can easily be part of the student's portfolio for years to come. In addition to the game that resulted from their hard work, the group project experience was very similar to that of the work world. They were in a group with colleagues who had a similar understanding to themselves, and worked together to learn the necessary information to accomplish their goals while being responsible for different parts of the project. Of course, these benefits vary in their applicability based on how much effort the individuals themselves put in, but it seems that most students put in the effort required.

One of my favorite observations during this class was how the freedom affected the different students. The vast majority of students thrived when given the freedom to make whatever they wanted. This first became apparent after all the students completed the Roll-a-Ball tutorial and were asked to create their own version of the Roll-a-Ball game. The creativity of the students was amazing; no student had the same variation on their game as another. Also nearly all of the students worked diligently on their games, actually getting ahead of the class by learning more tools on their own. After seeing the results of this assignment, I was tempted to just have the students just continue to build on the Roll-a-Ball game for the duration of the course to see how they would end up. Thankfully the enthusiasm driven by their creative minds continued even when starting from scratch on their own games. The success of this class was highly dependent on the enthusiasm that came from the students being given the freedom to attempt to build whatever they could imagine.

## Bibliography:


AnomalusUndrdog. “Unity Lesson 1Draft.” *Unity Lesson 1 Draft.* N.p., 3 July 2009. Web. 30 May 2015. <http://forum.unity3d.com/threads/unity-lesson-1-draft.103421/>.

“AP* Computer Science: A Brief History.” Web. 12 June 2015. PDF file. <http://www.apsi.thecubscientist.com/05_DailySchedule/historyAPCS.pdf>.

CollegeBoard. “AP Computer Science A: Course Overview.” Web. 12 June 2015. PDF file.

“Course Overview.” *AP Computer Science A.* CollegeBoard, n.d. Web. 20 May 2015.

“CS Education Statistics.” *Exploring Computer Science CS Education Statistics Comments.* Exploring Computer Science, n.d. Web 12 June 2015.

“How to Start Your Game Development.” YouTube. Extra Credits, 14 Jan. 2015. Web. 20 Feb 2015.

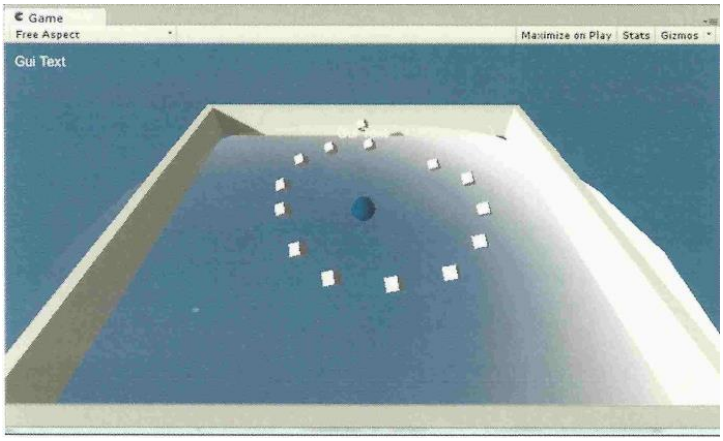
“Learn with Unity.” Unity. Unity Technologies, n.d. Web. 30 May 2015. <http://unity3d.com/learn>.

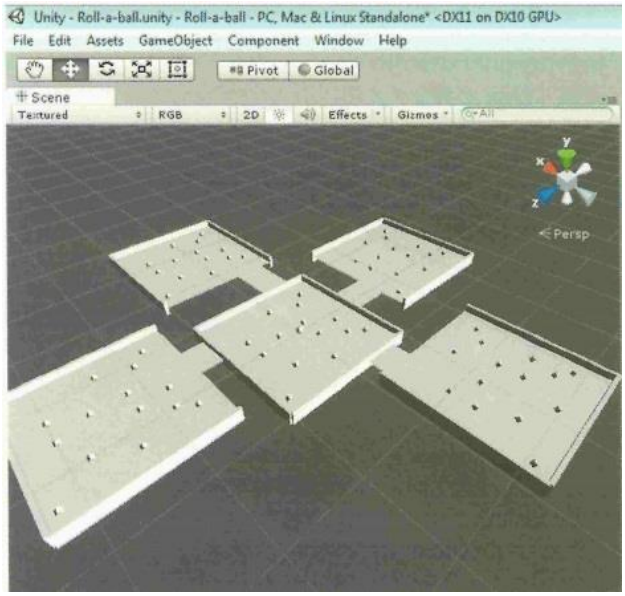
“The Leading Global Game Industry Software.” Unity. Unity Technologies, n.d. Web. 30 May 2015. <http://unity3d.com/public-relations>.

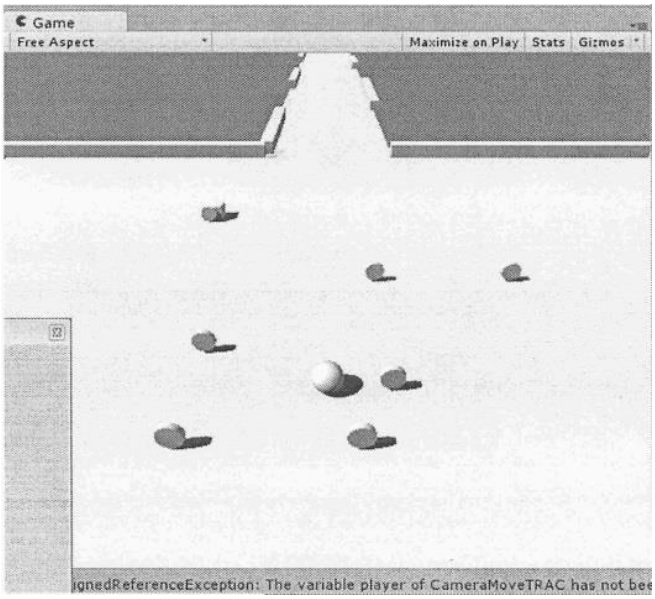
Thorn, Alan. *Game Engine Design and Implementation,* Sudbury, MA: Jones & Bartlett Learning, 2011. Books. Google. Web. 12 June 15.

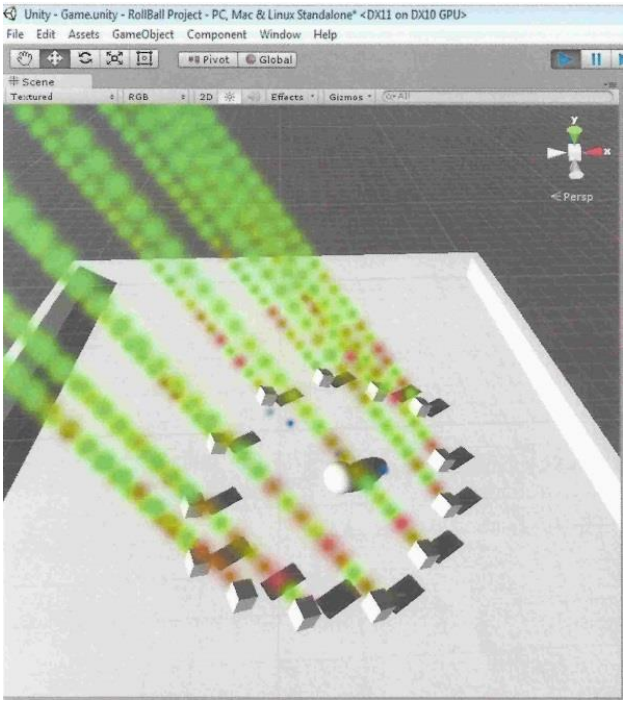
“Our Mission.” *Exploring Computer Science.* Exploring Computer Science, n.d. Web. 20 May 2015.

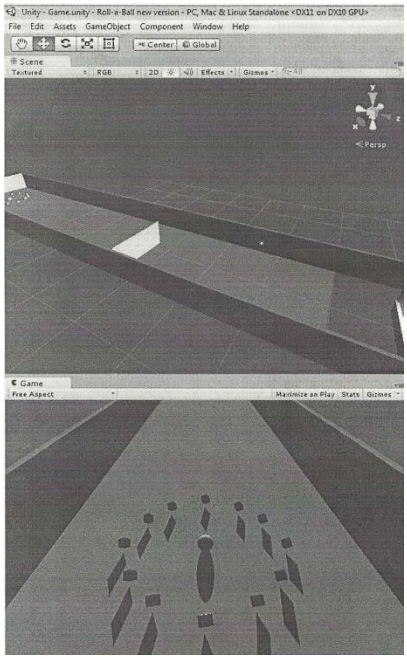
# APPENDIX A: Student Roll-a-Ball

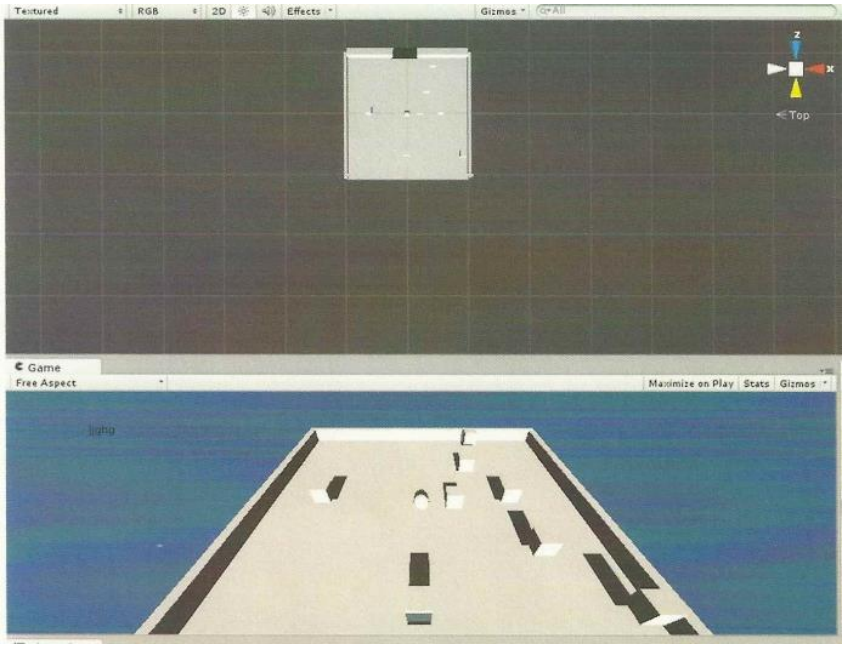



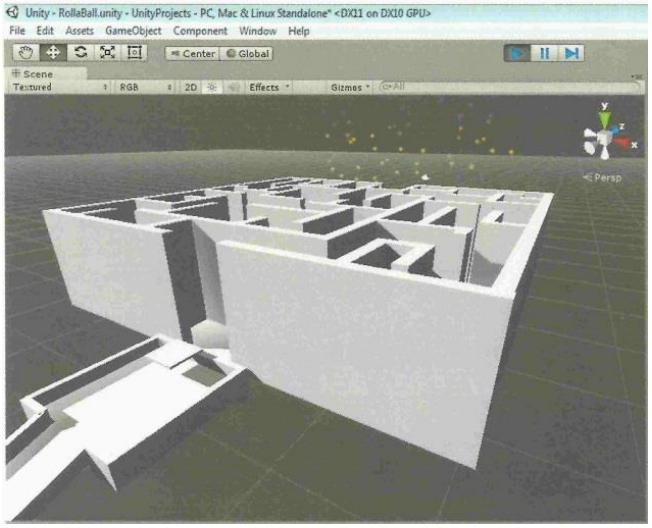



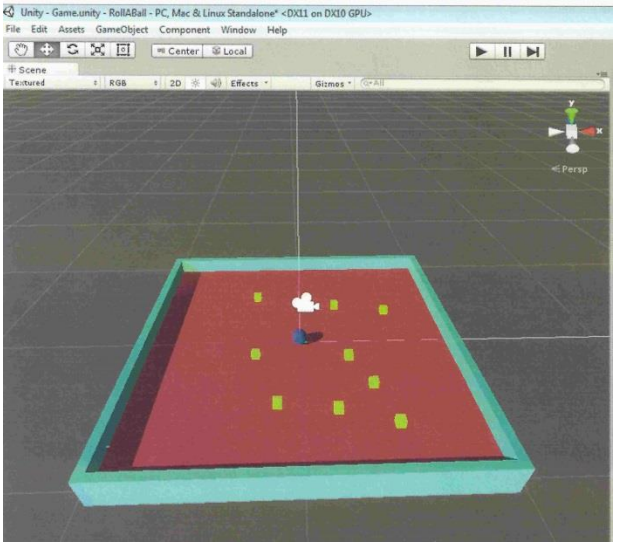





Unity - Game.unity - Roll-a-Ball new version - PC, Mac & Linux Standalone <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Center Global
Scene
Textured RGB 2D Effects Gizmos
Persp
Game
Free Aspect
Maximize on Play Stats Gizmos


Textured RGB 2D Effects Gizmos
Top
Game
Free Aspect
Maximize on Play Stats Gizmos


Unity - RollaBall.unity - UnityProjects - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Center Global
Scene
Textured RGB 2D Effects Gizmos
Persp


Unity - Game.unity - RollABall - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Center Local
Scene
Textured RGB 2D Effects Gizmos
Persp