

Laser Ball: An Interactive Construction Kit

Jonathan Crider
Karim Elatov
Matthew Fetig
Bobby House

TABLE OF CONTENTS

Introduction:.....	3
Overview:.....	3
Implementation:	4
<u>Image Capture</u>	4
<u>Bouncing Balls</u>	5
<u>Object Collision Detection</u>	6
<u>Combinatorial Behavior</u>	7
<u>Software and Equipment</u>	8
Possible Applications:.....	8
Looking Toward the Future:	9
Conclusions:.....	9

TABLE OF FIGURES

Figure 1: Conceptual Diagram of Laser Ball.....	3
Figure 2 Glare Produced by Projector	4
Figure 3: Keystone Effect.....	5
Figure 4: Collisions Formula	6
Figure 5: Multi-Colored Balls.....	6
Figure 6: Normal Vector Calculation Illustration.....	7
Figure 7: Combined Blocks	8

Introduction:

Laser Ball is a construction kit that joins physical and virtual objects to create an interactive user experience. Users are able to move physical blocks around a screen, specify an area to draw a virtual ball, and then watch the ball bounce off the physical blocks. As part of our implementation we ran into several design challenges and to address them we made some interesting design decisions. Unfortunately, we were unable to implement some features that we would have liked to see be incorporated into Laser Ball. Given these extra features we believe they would make for some interesting applications of Laser Ball.

Overview:

When running Laser Ball, you would first take the physical magnetic wooden blocks and place them on a magnetic board. The blocks can be combined to create any number of new shapes. You then would shine a laser at the screen. As of the current release, one would then need to click the mouse on the computer to take a snapshot. Now with the snapshot we run image recognition software against the image. We first detect all black shapes and outline them in red. We then search for the green laser and draw a ball at the laser's position. The created ball contains information to make it behave like a real ball might, falling due to the force of gravity, and bouncing off physical objects in its way. This falling and bouncing behavior is simulated in a graphical application that is then projected onto the board. Users can then continue to create any number of balls, taking a picture each time, and watching them interact with the physical blocks, falling due to gravity, and bouncing off one another.

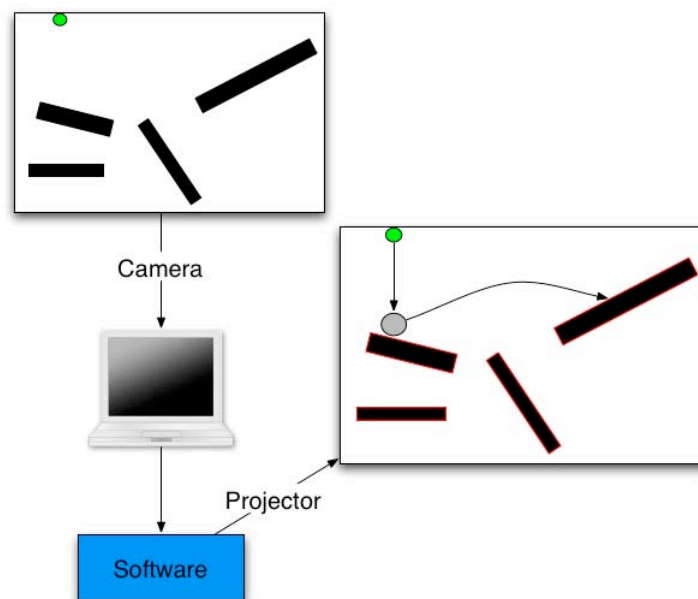


Figure 1: Conceptual Diagram of Laser Ball

The types of possible interaction in Laser Ball are physically moving the magnetic blocks and physically pointing a laser at the screen and then having a virtual ball being drawn in its place. Also, virtual balls bouncing off of physical blocks. Lastly, virtual balls bouncing off of virtual balls.

Implementation:

For our implementation of Laser Ball we split into two teams to try and develop the application in parallel along a logical separation. One team was given the task of developing the image capture portion of the project. The second team was given the task of writing the software to get the balls to bounce in a way that models reality. Both teams ended up doing some work on collision detection and responding to collisions because this is where the two halves of our project meet.

Image Capture

For the image capture portion of the project we used an image recognition library known as JMyron. With JMyron we were able to specify a color to track and then scan for all pixels that match the specified color. We completed this in two steps. First we would track black for the boxes and then draw bounding boxes, connecting the points. The second step would then scan for the laser. Unfortunately, due to the limitation of JMyron we were unable to track two colors at the same time.

During the process of doing the image capture we ran into several problems. The first was that the green laser showed up as white do to the surface we were working with. Instead of being able to track green we had to track white. White also happened to be the same color as the surface itself. Due to the way the recognition library works we were given an outer bounding box that binds the entire screen. This ended up making sense for our application because it acts as a border. However, the glare caused by the projector projecting onto the screen also shows up as white. Again, due to the way the image recognition library works only the most intense white is tracked which happens to be the laser. As long as the laser is being shown on the screen, only the laser will be recognized instead of the glare caused by the projector.



Figure 2 Glare Produced by Projector

Because of the glare problem we were unable to do continuous image recognition. If we did balls would keep being drawn where the glare from the projector was being seen as the most intense white light. As a result, we implemented a snapshot style approach. While the laser is pointed at the screen, you click the mouse, which takes a single image of the board. For that instant in time the brightest white light will be the laser and thus avoiding the problem caused by the glare.

The last major difficulty that we encountered was a keystone effect caused by projecting onto a surface and then recording the captured image.

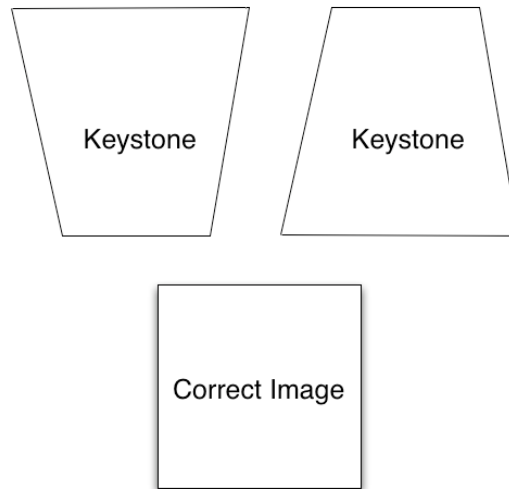


Figure 3: Keystone Effect

This caused the projected bounding boxes to be slightly skewed and not directly on top of the physical blocks. Obviously, we are only simulating the balls bouncing off physical objects and they instead bounce off the bounding boxes we draw on the screen. This means that it is clear at times that the balls are not actually bouncing off the physical boxes. Due to usability concerns we left the colored bounding boxes on the screen so that a user can clearly see what is happening. While, this approach is imperfect it hopefully does not detract from how well someone could use it. We did try and combat this problem by specifying a scale constant in our application. During initial setup “tweaking” this number can help reduce the keystone effect.

Bouncing Balls

For the bouncing balls it was important for the balls to bounce in a way that modeled reality. For example we wanted the balls to detect a slope and bounce in a direction appropriate for the gradient of the slope. To make the balls bounce realistically from the magnetic blocks placed on the board we used the equation for Conservation of Momentum. Combining the conservation of Momentum and the conservation of Kinetic Energy equations we got the following equations to enable us to detect head on collisions.

$$v_{1,f} = \left(\frac{m_1 - m_2}{m_1 + m_2} \right) v_{1,i} + \left(\frac{2m_2}{m_1 + m_2} \right) v_{2,i}$$

$$v_{2,f} = \left(\frac{2m_1}{m_1 + m_2} \right) v_{1,i} + \left(\frac{m_2 - m_1}{m_1 + m_2} \right) v_{2,i}$$

Figure 4: Collisions Formula

Since we were working with multiple dimensions; in our case two dimensions the momenta can be resolved into x and y components. We then calculated each component separately, and combined them to produce a vector result. The magnitude of this vector is the final momentum of the isolated system.

While the original bouncing ball code used multi-colored balls, once we began the integration process it was clear that this would not work. The reason is because we needed the balls to be able to bounce across the screen without being detected. However, color variations between the white and black would immediately be recognized as one or the other (laser – white or block - black). A more neutral grey seemed to sneak past this limitation and go undetected.

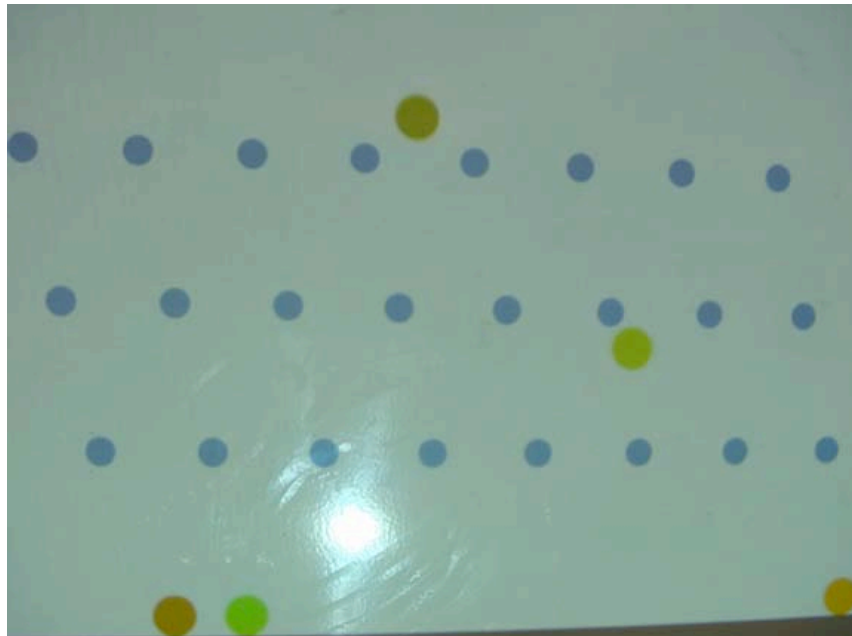


Figure 5: Multi-Colored Balls

Object Collision Detection

Because JMyron returns arrays of points that it determines to define an object, we chose to use this raw data format for collision detection for objects. Unfortunately, this is a small area where parallel tasking may have ended up hurting the performance of the software in the end. With integration, the data points were all copied to another data format which contained the same data, but the collision functions were already written to handle. In the end the function takes an array of data points and checks to see if the ball

will hit encircle one in the next frame. If it will, a normal vector is calculated depending on the points on either side of the point closest to the center of the ball. An illustration of how this is done can be seen in Figure 6.

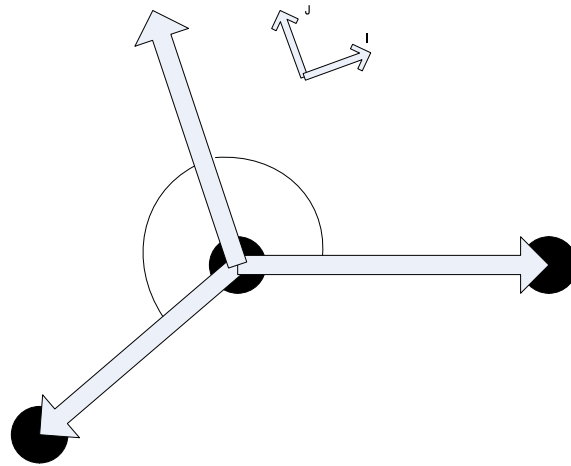


Figure 6: Normal Vector Calculation Illustration

What is done is the total angle between the two vectors is found and a vector is drawn at exactly half of that angle. One issue that has been seen with this approach is that the image recognition library sometimes returns a somewhat jagged line. Because of this, it is thought that perhaps sensitivity could be adjusted to find the balance between processing time and accuracy. This sensitivity would directly affect the amount of points used on either point of the point used to compute the normal vector. Once this normal vector has been calculated, this is the direction that the force is applied. The velocity of the ball in the I direction remains unchanged, while the velocity of the ball in the J direction is defined as $V_J = -e * V_J$. In this equation e is the coefficient of restitution, which was picked to be an arbitrary value and could be changed depending on how the user/developer wants the ball to interact with the objects.

Combinatorial Behavior

During our parallel development process we kept in close contact to make sure that both teams were trying to accomplish non-conflicting goals. One common goal was to develop our application so that it makes use of some of the principles that are common to most construction kits. One of these is the use of combinatorial behavior. Users can combine objects in the world of Laser Ball to produce predictable behaviors from the objects. As part of the experience users can combine objects to create new and interesting behaviors. While, the behaviors might be new, we felt it was important that the behaviors were predictable. As a user interacts with Laser Ball user expectations are set on how the objects interact with one another.

When trying new combinations users should have a sense of what might happen. This most notably takes place as one combines the magnetic blocks to create new shapes.



Figure 7: Combined Blocks

Software and Equipment

We used several pieces of software and equipment. We developed the entire application in Processing with the addition of the JMyron image recognition library. Processing is an open source programming language and environment for people who want to program images, animation, and interactions. JMyron is the cross-platform, cross-language, open source, video capture and computer vision plugin. Using both of these we were able to take a picture and parse it for pixels. These tools made it easy to interact with any kind of camera and provided easy tools to manipulate moving and still images.

For the physical equipment we used a Logitech QuickCam 9000 2.0. It has an effective image capture resolution of 1600x1200 at 30 fps. For the projector we used an Epson 77c with a resolution of 1024x768 with 2200 lumens. The laser pointer was a slight step up from ordinary laser pointers you might use. The strength is approximately 30 mW.

Possible Applications:

We think with some additional work our application could be extended to other applications. We think it would be interesting to build something similar to the various marble construction sets that are widely available today. Another idea (for the distant future) might be to create an interactive dollhouse like the Cube World brand of toys. With this idea, the projector might animate a character on the screen and the user could place different objects for the character to interact with. One of the original inspiring ideas for this application was the game “Incredible Machine” While the game was meant to be a game and was not necessarily grounded in reality we believe you could make interactions more life-like and create a kit that was educational. It could be used to demonstrate some basic principles taught in introductory physics.

Looking Toward the Future:

While we are pleased with the amount of work we were able to accomplish there are still some areas that we would like to improve and extend. If given three more weeks we could fairly easily create a calibration mode. This would alleviate someone from the currently tedious process of getting the application initially configured. The configuration is mainly due to the keystone problem discussed earlier. The calibration mode would also take care of the scale factor and offsets dynamically for the user, so that the distance from the screen and the position of the camera relative to the projector is no longer relevant. We would also like to add support for multiple colors. As it is right now our application isn't very visually appealing. What is preventing this as of right now is that the sensing of the black blocks must have a very low sensitivity. If the sensitivity is increased, the area outside the projected area would return a lot of data points, which don't really matter, but we are iterating through them anyway. All that is needed is some sort of filter that will check each object found and ignore it if it is off of the screen.

If given three more months there are several ways that we would like to extend our application. Primarily, we would like to be able to have virtual objects interact with physical objects in a way that causes the physical objects respond. Proposed ideas have included having bells ring when a ball comes in contact with them or make a lever move when a ball bounces off of it. It turns out that Processing has a serial library that could be used to easily command a microcontroller to deflect a servo. Other obvious optimizations would be to further refine our image recognition algorithms and ball collision algorithms. One could spend days trying different JMyron configuration values to see which ones work best.

Conclusions:

We believe that image recognition can be used in some interesting ways to create interactive construction kits. We hope that Laser Ball demonstrates some possibilities that could be explored to make even more fun and engaging construction kits. Despite the challenges we faced we were all very pleased to find how simple image recognition actually is. We would encourage anyone thinking of doing a similar project to give it a try.