A Bouncing Ball Game for First-Year Computer Graphics

Neil A. Dodgson
Department of Engineering and Computer Science
Victoria University of Wellington
neil.dodgson@vuw.ac.nz

ABSTRACT

A bouncing ball is one of the simplest physics simulations yet provides a novice graphics programmer with a host of useful experience. The student creates a ball that bounces off the walls of a box, adds a bat, then gamifies the whole experience into a simple block-out game. The assignment is designed as the first significant piece of programming on a 2D computer graphics course. It is designed to be accessible to students who have taken an introductory programming course and who have physics and algebra to the level of a high-school graduate.

CCS CONCEPTS

• **Computing methodologies** → *Physical simulation*; *Collision detection*;

KEYWORDS

education,animation,graphics,simulation

ACM Reference Format:
Neil A. Dodgson. 2018. A Bouncing Ball Game for First-Year Computer Graphics. In *Proceedings of SIGGRAPH '18 Educator's Forum.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3215641.3215653

1 INTRODUCTION

A bouncing ball is straightforward to implement yet enormously fulfilling for the student because something that requires just a few lines of code suddenly comes to life when the code is run. The student develops a better understanding of simple vector mathematics, discrete physical simulation, and the sorts of things that can go wrong with even such a simple scenario. This is the second assignment on our first-year undergraduate course "Introduction to Computer Graphics" [Dodgson and Chalmers 2017].

2 MATERIALS

Students are given a lecture on the basics of physical simulation of a bouncing ball including the lecturer spending 20 minutes demonstrating in class how to write code to make a ball bounce off the sides of the graphics window. Students are provided with a worksheet that takes them through some basics of physical simulation; this is designed that students can work through it in one hour in a lab class. The combination of these two means that a student who has paid attention will be trivially able to complete about a third of the credit for the whole assignment.

Table 1: CGEMS metadata

Summary	A block-out game in *Processing*. Topics: simulation, algebra, collision detection, instancing
Learning outcomes	Core: understand basic physical simulation, simple vector mathematics. Completion: instancing. Challenge: more challenging vector mathematics.
Classification	(1) Animation, (3) Fundamentals
Audience	CS1
Dependencies	introductory programming, high-school algebra and physics
Strengths	students enjoy seeing their code cause a physical animation
Weaknesses	students with weak mathematics find it difficult to get beyond the basics
Variants	adding gravity makes a more challenging assignment; the "challenge" part of the assignment gives two ways in which the assignment could be upgraded for a higher-level student cohort
Assessment	students are assessed on a number of binary achieved/unachieved goals (e.g., does the ball move, does it bounce off the sides) and on a graded measure of code quality

This assignment was designed to be implemented in the language *Processing* [Reas and Fry 2015]. Students are expected be familiar with *Processing* or to be familiar with *Java* and have had an introduction to *Processing*. We provide lectures, a laboratory session, and an assignment on introductory *Processing* for students who have *Java* experience.

We choose to provide no code framework for this assignment: students must program it from scratch. This is reasonable because of the way *Processing* is designed to be accessible to a novice programmer. Using another language for this assignment may require the instructor to provide a framework so that students can concentrate on the learning objectives rather than on building the scaffolding.

3 STRUCTURE

This assignment has three sections: core, completion and challenge. A C-grade student would be expected to complete the core to a good standard. An A-student will complete core and completion to a good standard. The challenge part of this assignment is to stretch

SIGGRAPH '18 Educator's Forum, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5884-2/18/08.
https://doi.org/10.1145/3215641.3215653

the best students. We allocated just 10% of the credit for this final part to allow A-grade students the ability to choose whether or not to devote time to the challenge.

3.1 Core assignment

The core is to implement a ball and a bat.

Outline specification. Implement a moving ball in 2D that bounces off the sides of the window. Add a bat, an axis-aligned rectangle that is controlled by the mouse. The ball must bounce off the bat in a plausible manner, but it does not need to be physically accurate.

Comments. Bouncing off the sides of the window requires subtlety because the ball has a finite radius that must be taken into account in the intersection calculations. Bouncing off the bat is more challenging. We added the condition that the bouncing be "plausible" rather than "physically accurate" after our experience in the first year of running this course, where we found that many of the better students got stuck on how to handle the case when the ball hits a corner of the bat. The weaker students did not realise that there was a problem and so we had the situation of the better students spending far longer on the assignment than was necessary. For bouncing off the window's sides, the student needs to consider four cases: one for each side. For bouncing off the bat, we ask them to consider only the same four cases. This can lead to odd behaviour when the ball bounces off a corner of the bat, but that is acceptable for the core assignment; we ask students to fix this behaviour in the challenge part of the assignment.

In the first two years of running the course, we gave the students the option of adding gravity to the simulation. Gravity adds the challenge that the simulation becomes unstable: a naïve implementation will have a ball that noticeably loses or gains energy. It was beyond most of the students to recognise that this instability is a problem and, if they did, fixing it is non-trivial. Therefore, we only recommend adding gravity to the assignment, with supporting lecture material, if repurposing this assignment for higher-level graphics students.

3.2 Completion assignment

The student "completes" the assignment by gamifying what they have produced into a version of the block-out game.

Outline specification. Add a number of rectangular obstacles off which the ball bounces. Turn this into a game. Each rectangle will have a colour that indicates its status: green = never hit, yellow = hit once, red = hit twice. When a rectangle has been hit three times it vanishes. The aim of the game is to remove all the rectangles.

Comments. This teaches students about instancing graphical objects and maintaining status for each instance. There is a nice touch that the static obstacles and the moving bat can reuse the same intersection code. The students need to ensure that their additions to handle the static obstacles do not also apply to the bat: it cannot change colour or vanish. The vanishing is a useful point for marking. Some students will make an obstacle "vanish" by painting it the same colour as the background. You need to determine whether this is a clever hack or a point of failure. Some students will fail to turn off intersections with vanished obstacles: this is clearly wrong.

In the traditional block-out game, the obstacles are identical rectangles in a rectangular grid at the top of the window. Most students implement this. Some students, however, take the approach of putting rectangular blocks of varying sizes at random locations. This is not an error and such students are worth noting to see how such independence of thought plays out in later assignments.

3.3 Challenge assignment

We provided students with a choice of two challenges, A and B, either of which would give them full marks.

Outline specification. Option A. Go back to just the ball simulation (the core) and make a simulation that allows multiple balls, all interacting with one another. Once you have it working, set up some sort of demonstration to show to your assessor, for example, implement drag and gravity then simulate a bunch of balls being poured into a container.

Option B. Building on top of the completion assignment, make the bat/ball interaction robust. Ensure that the ball bounces correctly off the bat if it hits a corner. Ensure that, if the ball and bat are both moving, the ball still bounces off the bat rather than them appearing to pass through one another. You should also aim that the bouncing is physically accurate.

Comments. Both of these are interesting assignments in their own right. The first requires some careful calculations to make the balls bounce off one another correctly. Students can find tutorials on the web in how to do this. We like the idea of making students think up their own demonstration that their algorithm works, as this helps them to consider how you would test a physical simulation for correctness and allows them to scope how they want to be assessed. The second option is based on our observations of the subtle inaccuracies in most students' implementations. In particular, almost no student considers that both the bat and ball can be moving; they assume that the bat is stationary, forgetting that the user is able to move the bat at high speed.

4 TAKING THIS FURTHER

The two challenge assignments described above are both viable options for *all* students to attempt on more advanced courses. It is also possible to specify that the game be made considerably more polished. As an example of this, look at the first game in our course's 2016 showreel: https://vimeo.com/196225486 (students on our first-year course are required to write a 2D game as the capstone assignment; a few choose to use the bouncing ball as the foundation for their game; one of the best is showcased in the video).

ACKNOWLEDGMENTS

Thanks to Dr Andrew Chalmers for leading the tutorial team that provided feedback on the first iteration of the assignment.

REFERENCES

Neil A. Dodgson and Andrew Chalmers. 2017. Designing a Computer Graphics Course for First Year Undergraduates. In *EG 2017 — Education Papers*. The Eurographics Association, 9–15. https://doi.org/10.2312/eged.20171020

Casey Reas and Ben Fry. 2015. *Getting Started with Processing* (2^{nd} ed.). Maker Media.