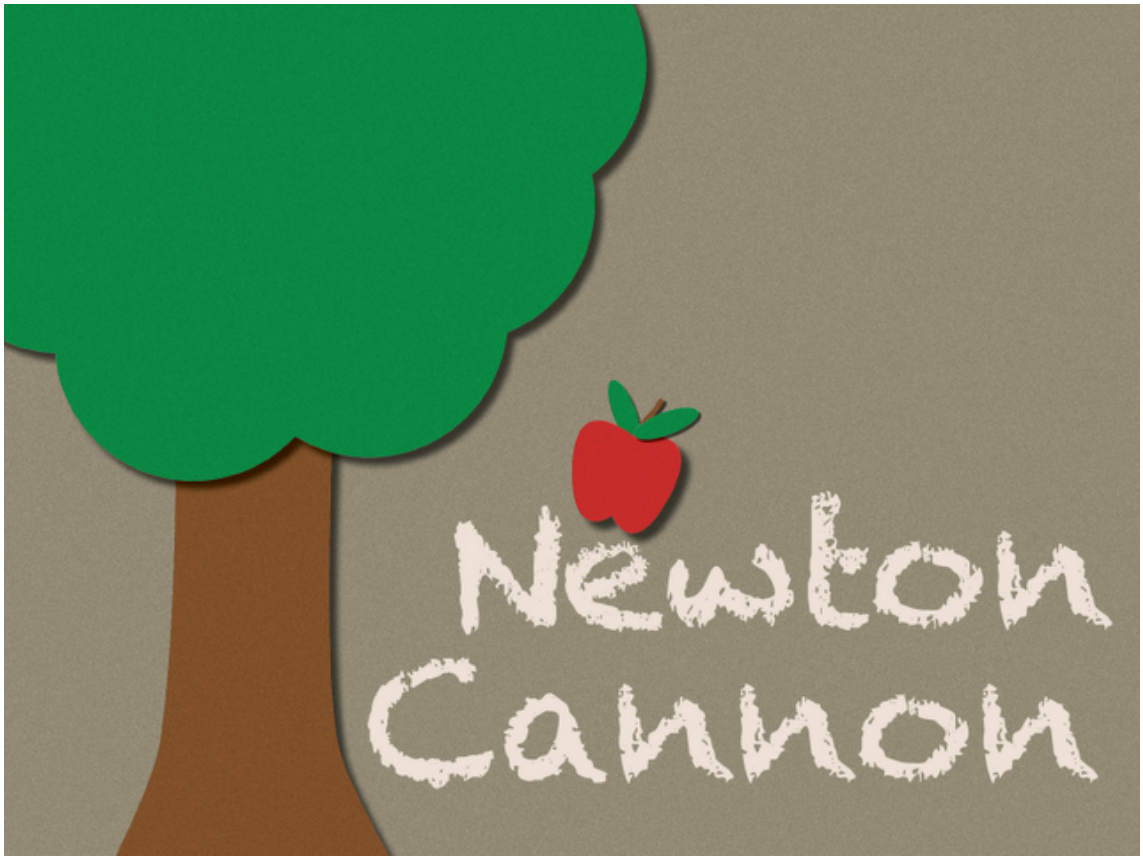


Developer Documentation: Final

The Ohio State University
Department of Computer Science & Engineering
CSE694: Professor Rajiv Ramnath, PhD.

Team 6
Samual Bantner (bantner.2@osu.edu)
Jarrod Freeman (freeman.320@osu.edu)
Mike Zazon (zazon.1@osu.edu)

THE OHIO STATE UNIVERSITY
DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Table of Contents

Section 0: About Newton Cannon

This App was created as part of the CSE694 Pilot Course in Mobile Application Development (Dr. Rajiv Ramnath). During the term, small teams formed and branched out to develop an application for Android and iOS platforms as part of the term project. Each team was able to choose their own application idea and developed the idea into code using design methodologies learned during the course. The course offered OSU Faculty support and guidance if teams were interested in submitting to the STEM competition, and we took advantage of this opportunity. Dr. Dean Cristol from the School of Education and Dr. Andrew Heckler from Physics advised us on building this application with regards to the STEM competition.

Newton Cannon is designed to teach students the effect of gravity and other physical world properties on a projectile. Students may be familiar with the gameplay as many popular games in the mobile application markets and stores have a similar design. The goal was to create a fun way to teach the effect of different

physical properties on a ball that is launched from a Cannon. Our target audience is K-12 students, but anyone will enjoy the 3D visuals and gameplay that this application provides.

Because of the age level of our targeted audience, some assumptions had to be made about the physical world in which the Newton Cannon operates. To keep things simple, we assume that the initial force is applied to the ball instantaneously, and that no deformation of the ball occurs. Also, we omit any form of friction in the world. This includes friction between the ball and cannon, as well as air resistance. These assumptions about the world are important in keeping the focus of the application on the effect of gravity on a physical object.

Within the application, gravity can be manipulated by the user according to which planet the user selects to fire the cannon on. For simplicity, gravity is presented as a percentage relative to earth's gravity. The planets available to the user and their associated gravity levels are:

Earth (1.0)
Mercury (0.3)
Venus (0.9)
Undiscovered (0.7)
Mars (0.3)
Jupiter (2.3)
Saturn (1.0)
Uranus (0.8)
Neptune (1.1)
Undiscovered (0.6)

Section 1: Use Cases

User Launches Game

When a user launches the game, they are presented with a splash screen which displays the name of the game "Newton Cannon". Two buttons are displayed, one called "Start Game" and one called "How To Play".

User Presses Start Game Button

The user will press the start game button to load the first objective and the cannon view.

User Presses How To Play Button

The user will press the How to Play button to load some game rules and FAQs about gameplay. They will have the option to return to the splash screen from this point.

User Fires Cannon

The cannon will oscillate up and down waiting for the user to fire the cannon. The cannon is be fired by tapping the screen in any location (except for the pause button in the upper right corner). The ball will travel along a path, highlighted by a tail which follows the ball to help track progress. The distance and height of the ball will be displayed in real time in the upper left corner of the screen. When the ball collides with and comes to a stop on the ground, the game will indicate if the objective has been reached.

User Completes Objective

When a user completes an objective, they will be presented with the next objective in the game, and the game will load the cannon scene again with the new objective displayed.

User Fails Objective

If a user fails an objective, the game will prompt the user to change the gravity properties of the world to help them reach the objective desired.

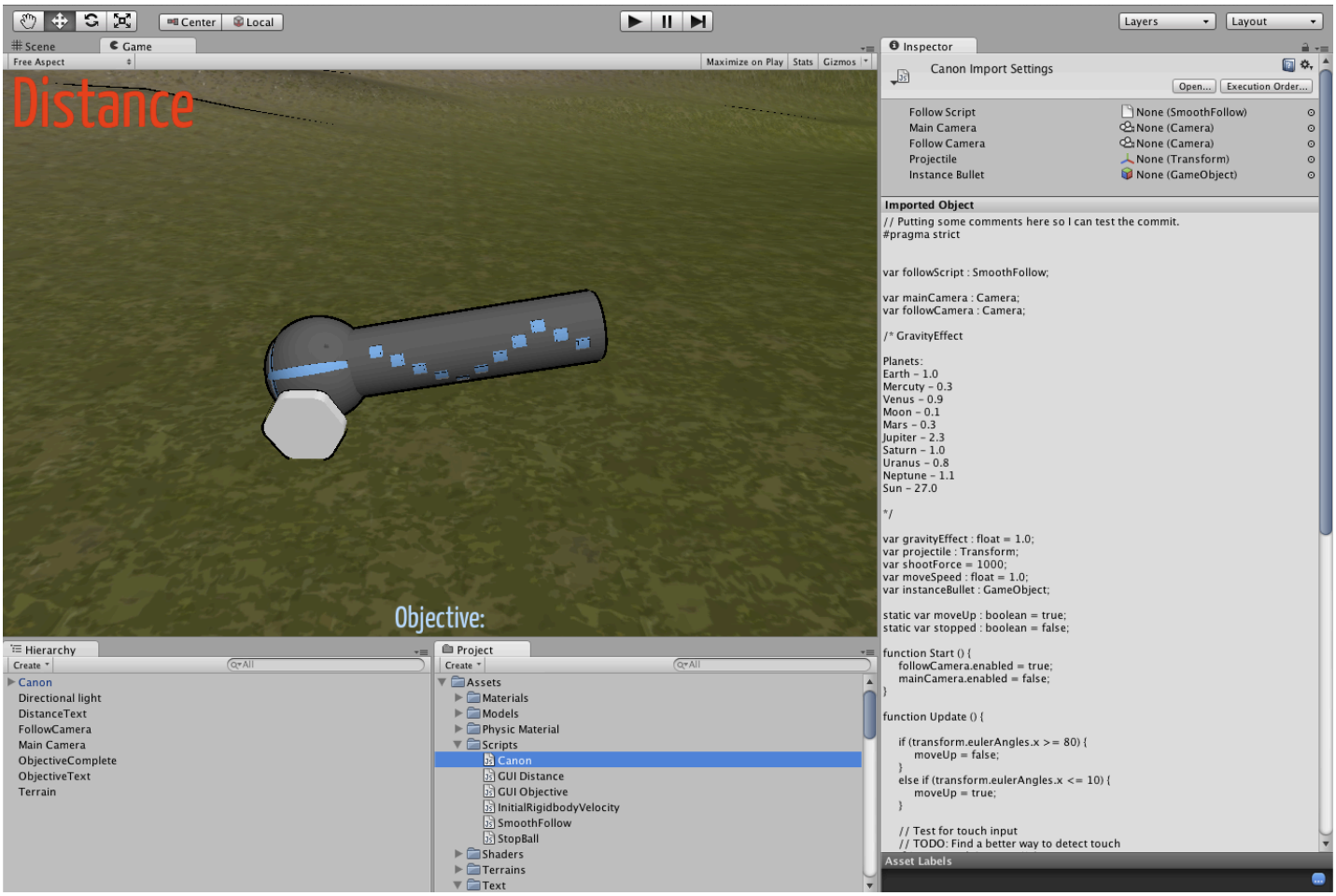
User Pauses Game

A user can pause the game by tapping the “Pause” icon in the upper right corner of the screen during gameplay. They can resume the game by pressing the “Resume” button.

Section 2: Development and Architecture Choices

Unity

The team chose the Unity 3D engine (http://www.unity3d.com) to develop this application. The Unity Engine provides an IDE that displays a project hierarchy, a three-dimensional perspective for design, asset views, and a code preview pane.



The Unity Engine provides a powerful and cross-platform solution for developing games and applications on an expansive list of architectures. We are developing this application specifically for the iOS architecture, however because of the flexibility the Unity Engine provides, we can cross-compile for PC, Mac, and Android.

iOS

The team has chosen to target iOS as the platform for development because of the availability of physical devices available to the team (iPhone 4S and 3rd generation iPad). These devices have excellent graphics processors on board which are needed to run some of the 3D effects smoothly.

Section 3: Class Diagrams and Hierarchy

Design

When designing the classes, we decided that it was too complicated to check what “real” Java and Objective-C classes the Unity Engine produces when it compiles the code. The Unity Engine, as a cross-compiling solution, uses JavaScript and C# for its internal scripting language, mainly because of cross-platform and also being widely used in industry. It compiles this javascript and C# along with the graphics engine and physics engine to produce the Java or Objective-C code that will compile and run on Android or iOS, respectively. The Unity Engine produces the thousands of lines of code that would be difficult to extract the classes for documentation purposes. Because of the vast amount of code produced by this engine, we decided to simplify the class diagram exercise by abstracting the various components of the game into “virtual classes”. A description of these “virtual classes” and their purpose is as follows:

Classes

Cannon

Fires the ball which the user will launching. Changes angle to alter the flight path of the ball

Ball

Launched via the cannon by the user. Affected by various forces. Collides with ground.

World

Encompasses all the other object of the game. Applies forces (gravity, etc) to the ball.

Camera

Displays the game world. Follows path of the ball.

Ground

Collides with ball.

Section 4: Application Requirements

Must have a UI (i.e it can't *only* be a background service)

Newton Cannon makes use of a fully featured UI that requires user interaction to progress through the stages of gameplay. For example, the user will be presented with a “Start Game” button that they must click to begin the game. Screen touches are another way to interact with the application. From within the game state, a screen touch will fire a ball from the cannon.

Must have a rich enough set of domain objects

The application includes a wide variety of domain objects that are detailed in other sections of this guide.

Must have data that is persistent across sessions

Application data is persistent. Game state will be saved after closing the application so that the user may restart the application at the same objective that they ended on.

Must use one or more external services - such as maps

Due to the nature of the Newton Cannon application, we decided that meeting this requirement would detract from the quality of our STEM submission. We had considered implementing the ability for the user to play music in the background, but ultimately decided that this would be detrimental to the learning experience that our application is trying to create.

Must use one or more sensors - GPS, accelerometer, light etc.

For similar reasons to the above requirement, our application does not make use of any sensors. Because our application teaches the basics of gravity by demonstrating the flight of a ball, there is really no effective way to use sensor data. Any attempt to use the sensors would just make the application feel clumsy and less user friendly.

Must incorporate techniques for resiliency against crashes, screen orientation changes, application being killed by the OS etc.

This requirement is handled by the Unity framework with which the application is built upon.

Must have suitable logging incorporated for debugging purposes

Debug logging is built in and has been used during the development of the application.

Must be profiled and analyzed for performance

The performance of the application was analyzed at various points throughout its development, and in some cases changes were made to improve performance. For example, at one point, the application was rendering roughly 180,000 vertices which was severely limiting the FPS. After some analysis, we determined that the iPhone can render around 40,000 vertices at an acceptable framerate. The iPad can handle slightly more,

maybe around 60,000 vertices, but it was still very clear that we needed to make some changes to lower the amount of vertices being rendered.

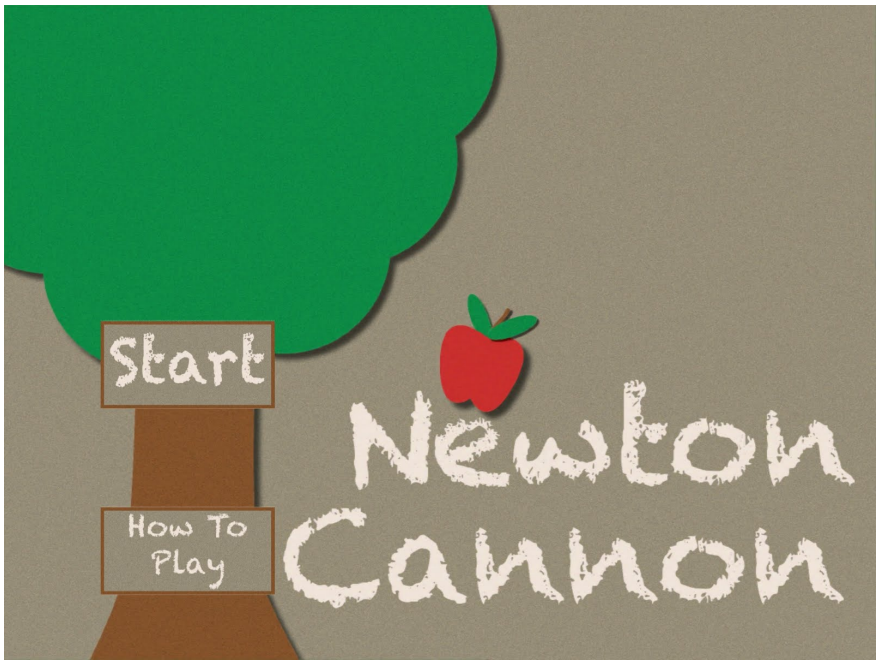
Must be well-designed and documented

The design of the application has been well thought out and implemented accordingly. It has also been well documented with this developer's guide as well as a user's guide.

Section 5: Gameplay

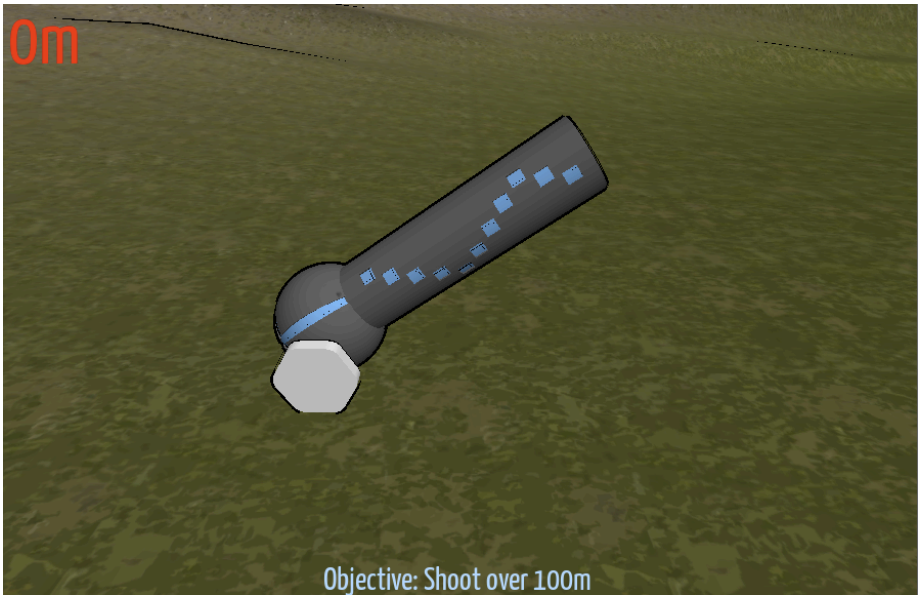
Splash Screen and Main Screen

The gameplay of the Ball Game is very simple. Once the application loads, the user is presented with a splash screen and two options: "Start Game" and "How To Play".

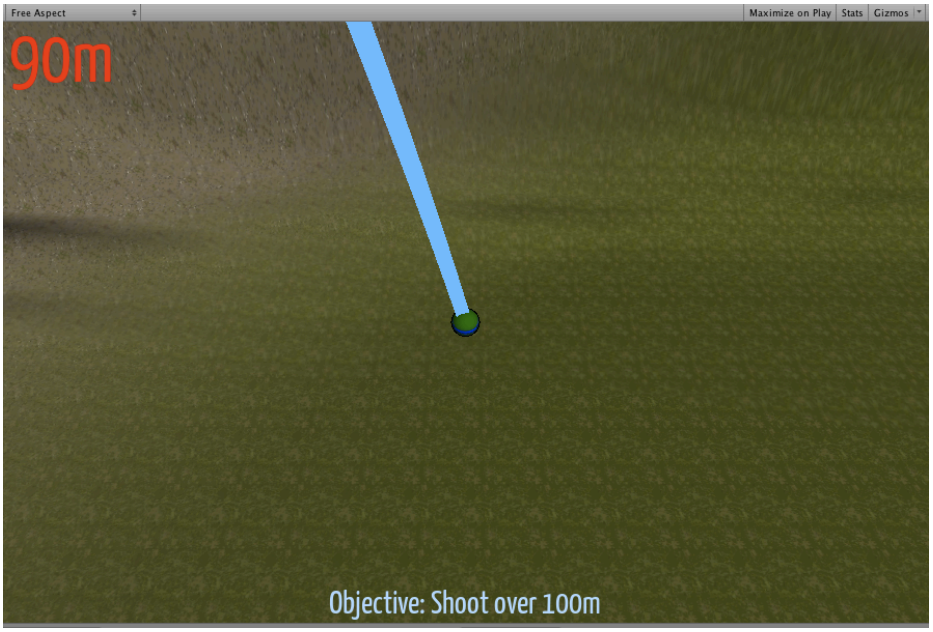


Cannon View

If the user presses Play Game, they are presented with the first objective along the bottom of the screen and the cannon view.



Once the screen is tapped (when the desired height of the cannon is reached), the ball will launch and the distance will be constantly updated in real time (in meters) at the top left corner of the screen.



Ball Flight

During the flight of the Ball, the camera tracks progress and the user can watch the ball until it slows down and comes to a stop on the ground. The game will then report if the objective has been met.

Objectives

After a completed objective, the next objective (if available) will load and the user will be presented with the cannon view. If the user fails an objective, they will have the option to adjust the gravity of the world to make the objective more attainable. The objectives implemented in the application are:

Shoot at least 50 meters

Shoot between 50 and 100 meters

Shoot between 150 and 350 meters

Shoot at least 100 meters

Shoot between 200 and 300 meters

Shoot between 200 and 400 meters

Shoot between 200 and 450 meters

Shoot at least 150 meters

Shoot between 150 and 200 meters

Shoot between 400 and 600 meters

Shoot at least 200 meters

Shoot between 50 and 200 meters

Shoot at least 250 meters

Shoot between 400 and 550 meters

Shoot between 300 and 400 meters

Shoot at least 300 meters

Shoot between 50 and 150 meters

Shoot between 350 and 500 meters

Shoot at least 350 meters

Shoot between 100 and 200 meters

Shoot between 350 and 550 meters

Shoot at least 400 meters

Shoot between 100 and 150 meters

Shoot between 250 and 400 meters

Shoot at least 450 meters

Shoot between 50 and 250 meters

Shoot between 100 and 250 meters

Shoot between 200 and 350 meters

Shoot at least 500 meters

Shoot between 150 and 300 meters

Shoot between 400 and 500 meters

Shoot between 350 and 400 meters

Shoot at least 550 meters

Shoot between 100 and 300 meters

Shoot between 150 and 250 meters

Shoot at least 600 meters

Shoot between 250 and 450 meters

Shoot between 200 and 250 meters

Shoot at least 650 meters

Shoot between 300 and 450 meters

Shoot between 350 and 450 meters

Shoot between 250 and 300 meters

Shoot between 250 and 350 meters

Shoot at least 700 meters

Shoot between 300 and 500 meters

Shoot between 300 and 350 meters

Section 6: Roadmap Into the Future

Release 1.0 (Current)

• App Complete and Submitted

Release 1.1 (1 Week)

• Bug Fixes
• Add sound effects
• Add loading screen when Start is pressed

Release 1.2 (2 Weeks)

• Bug Fixes
• Add Leaderboard
• Connect to GameCenter

Release 1.3 (3 Weeks)

• Bug Fixes
• Overhaul UI

Release 1.4 (4 Weeks)

• Bug Fixes
• New Game Mode: Free Play