Modern Computer Games

COMP 521, Winter 2015

Assignment 2

Due date: Monday, March 9, 2015, by 6:00pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. You must **explain** all answers and **show all work** to get full marks! Please make sure your code is in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not!

Description

Note that this assignment requires you build a scenario with 2D game mechanics. You should continue to use Unity, but will need to enforce the 2D-ness yourself.

1. First, you need to produce a game terrain. This will be a 2D profile of a canyon. The canyon top and bottom heights can be fixed and drawn as straight lines. Canyon walls should have a relatively steep slope to them (not just be straight). Unlike the sketch below, though, there should be variation and detail in the canyon walls, randomized on game start. For this you should use a midpoint-bisection strategy to give it a more natural look and fine-grain surface (edge) detail. **10**

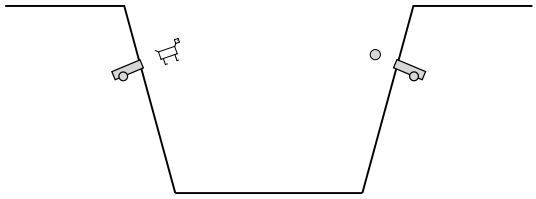

Figure 1: Idea for basic shape without noise/detail. You may modify proportions. Actual projectiles are described below.

Position and draw a cannon on each of the canyon sides, about 2/3 up the slope facing the other wall, with muzzle sticking slightly out of the wall. Your cannon does not have to be nicely drawn, but should have a recognizable barrel (rectangle) to indicate the current angle of fire.

2. The cannon on the rightmost canyon wall fires cannonballs. The cannonballs emitted as a result should be drawn as recognizable circles, with motion modelled using projectile physics, incorporating initial velocity, barrel angle, gravity, and air-resistance and wind. **10**

Determine appropriate gravity, initial velocity, and wind-resistance (assume cannonballs have unit mass). The goal is that your cannonballs should be able to reach the other canyon wall at approximately the same vertical height, given a 45° barrel angle, default/medium muzzle speed, and no wind. Your cannonball flight time should be apparent to the user, taking at least $0.5s$ of real-time for a cannonball to reach the other side.

Determine a reasonable range of wind force $[-w \ldots w]$ representing left→right movement if $w > 0$ and right→left if $w < 0$, such that when at w it can prevent the cannonball from reaching the other side but not blow it backward, given the above default configuration of other parameters.

Determine a reasonable range of initial velocities, such that in the absence of wind, the cannonballs fall into the middle of the valley with the lowest velocity and a 45° barrel angle, and can still reach the other side with the highest velocity and a 0° barrel angle.

Pressing the spacebar should result in the cannon(s) firing. Initial velocity and angle (within the 15–60° angle facing up and toward the left wall) are randomly selected at each fire.

Wind force should change randomly within your range every 0.5s. Show the current wind force as an arrow above the landscape, pointing left or right appropriately, and with length proportional to the wind force.

3. Cannonballs that go offscreen, land in the canyon valley or get outside of the canyon's left/right boundaries disappear. Cannonballs that encounter either side of the canyon, however, should collide with the canyon wall. For this you must implement your own collision detection and handling (do not use any built-in physics or colliders—you may, however, use basic geometric primitives for primitive shape intersection, distance, etc, as well as raycasting). **15**

   For simplicity, cannonballs do not collide with other cannonballs or the cannons. A cannonball that loses all motion for any reason should be destroyed after a few seconds.

   The exact parameters of your collision resolution are to you, but there should be some reasonable "bounce" to a cannonball/wall collision (ie, a coefficient of restitution of between 50% and 95%).

   Note that you do not need to model any rotational effects on the cannonball collisions.

4. The other cannon does not shoot cannonballs. Rather, it fires small, stick-figure animals, modelled using Verlets. Thus you will need a point/line model of an animal such as shown below, moving each point separately according to Verlet integration, while ensuring there is some effort to retain the overall shape and avoid gross distortions. **15**

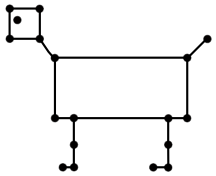

Figure 2: A dog. Or maybe it's a cat. The point is that has at least 2 legs (with knees), a body, a head separated from the body by a neck, a tail, and an eye.

   Let's call it a dog from now on. Your dogs should be small, but scaled so all features are recognizable. Once fired from the cannon the dogs have projectile physics and behaviours similar to the cannonballs described in question 2, but are fired by pressing the tab-key.

5. Like cannonballs, dogs that go offscreen, land in the canyon valley or get outside of the canyon left/right boundaries disappear, but must collide with canyon walls. However, there is no collision response (bounce) required, other than resolving interpenetration according to the Verlet model. **10**

   For simplicity, dogs do not collide with other dogs or the cannons. A dog that loses all motion (within some threshold) for any reason should be destroyed after a few seconds.

6. Cannonballs are not affected by dogs, but dogs are affected by cannonballs. Resolve any interpenetration accordingly following the Verlet model. **10**

What to hand in

Assignments must be submitted on the due date **before 6pm**. Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**.

For the Unity questions, hand in an exported project containing all files needed in order to reconstruct and run your simulations. Note that Unity exports can be extremely large, and take non-trivial time to upload (another reason last-minute submission may not work out well).

For non-Unity questions, submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

This assignment is worth 15% of your final grade. **70**