Modern Computer Games
COMP 521, Winter 2013

Assignment 2: Physics

Due date: Monday, Feb 25, 2012, 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

In this assignment you will investigate basic game physics as well as simple terrain generation. For this you will need to develop a 2D graphical application to display the result. You may use a language and graphical environment of your choice, but we highly recommend C++, or failing that, Java. You may not use any built-in features of your language for collision detection (other than basic primitives, such as line and/or box intersections if you find that useful).

In this assignment, you will write your own 2d side scroller. Your avatar will be a simple stick figure capable of running left and right, and capable of jumping. The game world will have jagged terrain with watery holes - if the avatar falls in the water, the game ends.

Demonstration: Make sure that each step can be tested independently. The TA should be able, either through menu options, key presses, or compiler settings, to view part 1, parts 1-2, parts 1-3, and parts 1-5.

1. First, you need to produce basic game functionality. To begin, model the terrain as flat lines on the screen, with breaks filled in by water, producing a screen like the following: **8**

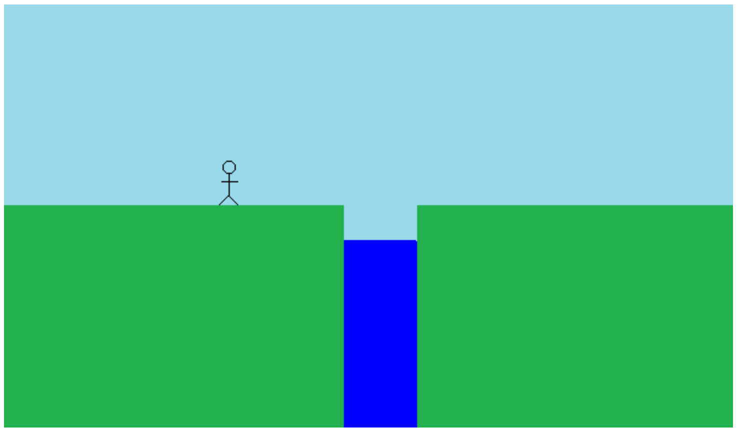

Figure 1: An example terrain.

Pressing the left and right arrow keys should move the avatar left and right on the screen, and pressing space should cause the avatar to jump. However, the avatar should have momentum, and thus pressing the keys should apply an acceleration in the appropriate direction, rather than simply changing the velocity. Of course, you will need to cap the avatar's speed to keep the simulation realistic.

The avatar should be affected by gravity, and falling into the water should cause the game to end with a loss. This means that your avatar will need some basic collision detection to prevent it from falling through the terrain.

Rather than being a true side scroller, you are to employ a screen swapping approach. When the avatar moves off the right hand side of the screen, a new terrain should be generated, with the avatar appearing on the left hand edge of the new screen.

2. Now lets make the terrain more interesting. Rather than a simple flat terrain, use 2d Perlin noise to make your levels bumpy and interesting. Use a diagonal line across the screen as your base, then add perlin noise. (e.g., a line from a random spot on the left edge of the screen to a random spot on the right edge.) Your collision detection may need to be improved to prevent the avatar from falling through the more complicated terrain. **8**

   Water should be now added to the level by putting water over terrain below some value of y. Each level must always have at least one water trap. However, you must prevent water from being so wide that the player cannot jump across.

3. Create a time-varying wind parameter, and indicate it in some fashion on the screen. Wind strength and direction (left or right) should change over (real-)time and should affect the motion of the avatar by adjusting the avatar's velocity. (e.g., a wind velocity of v_{wind} would modify the player's current speed by $min(v, v_{max}) + v_{wind}$). **4**

4. With the game in place, you need to introduce a higher challenge to the game. We will do this by adding random cannonballs that are generated at the top of the screen with an initial velocity in the x-direction only. They should fall until striking the terrain, then bounce and roll down the terrain before coming to a rest in a valley, or falling into the water or off the screen and disappearing. They should be affected by wind. **12**

   Cannonballs require significant collision detection and resolution. Cannonballs must never fall through the terrain, and if two cannonballs collide, they should bounce off each other. If the avatar touches a cannonball - even one at rest in a valley, the game should end with a loss.

5. Winning the game occurs when a player successfully crosses 10 screens. Balance the game as you see fit, with the goal of providing a challenge that you find appropriate while still preserving some amount of realistic looking behaviour. You will want to look at the rate of cannonball generation, wind speed and variation, and player speed, as well as terrain generation parameters such as amplitude, intensity, and terrain and water height. **3**

Note: As discussed in class, it is important that your code aim at the goals of efficiency and believability. This is especially important for your motion, collision detection and response calculations—assume that while you are developing this on a PC, you may need to deploy it on a machine with less computational power. Up to 25% of the total marks may be deducted for inefficiency.

What to hand in

For assignment submission use moodle: `http://moodle.cs.mcgill.ca/moodle`
For each of the above questions you should provide a separate application or clearly described options to a single application that demonstrates appropriate behaviour.

For programming questions hand in your source code code only, and include directions (readme.txt) if compilation and usage is not trivial and obvious. For any graphics packages that you use, include complete instructions on how to obtain and install any needed libraries. If the TA cannot follow your instructions and get your program working, you will receive a %10 deduction from your final grade.
For non-programming questions you can provide an ASCII text file with your answers, 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.

Note: by submitting an assignment you are declaring that it represents your own, exclusive efforts, and that all ext and code have been written by you.

This assignment is worth 15% of your final grade. $\overline{\mathbf{35}}$