DoodleJump Design Questions

Due: Sunday, October 25 - Wednesday, October 28
Help Session: Thursday, October 29, 4:00pm - 6:00pm Barus & Holley 166

Please read the DoodleJump handout carefully before answering these questions. Also run the demo by typing `cs015_runDemo DoodleJump` to play around with the standard demo, or `cs015_runSnazzyDemo DoodleJump` to catch a glimpse of some of the cool bells and whistles you can add to your project. The demo will help you answer the DQs listed below. Also, if you have trouble with these questions, we highly encourage you to attend the Help Session and bring any questions you may have. You may also come to TA hours, as always.

IMPORTANT NOTE: LIVE DESIGN CHECKS!

This design check will be graded interactively! We still expect good written work. You should prepare them as if the design questions will be turned in and graded. But this time you will be explaining your thought process to a TA during a 15-minute meeting. This way you can get live feedback on your design and start coding DoodleJump sooner!

Be sure to sign up for a design check; you should have received an email to sign-up through a Doodle (haha get it?) Poll. REMEMBER: Write down the time you sign up for. We will post the time that you signed up for at a later date. You will lose 20% of your DQ grade if you fail to sign up for a slot by Saturday, October 24 at midnight. You will get a 0% for missing your design check. If you are late to your design check, you will not be given extra time.

During the meeting, we expect you to have answers prepared for all of the questions listed on the next sheet. The design check will focus on the material that you have prepared in advance. We will ask you to explain in detail (some or all of) the questions below.

We are requiring you to physically bring in your answers to all of the design check questions. You should also be able to answer any other questions a TA may have about your planned design. You can take notes away from the design check.

Your grade will be emailed to you after the design check. It will be worth approximately 18% of your total grade for DoodleJump.

Remember, all the work you put in now with design will pay off later when coding the project!

For these DQs, you will need to write pseudocode! Here’s a quick-and-dirty rundown of writing pseudocode, which will be covered in much more detail in Lab 7.
Pseudocode Crash-Course

Pseudocode is “code” written for humans to understand, rather than a compiler. You can think of pseudocode as code written in English that can be understood by anyone, even with no knowledge of computer science.

Pseudocode is **not** language-specific: given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

For example, let’s say we want to write a method that makes a latte, and we have an instance of EspressoMachine, `_machine`. In Java, we might write it like this:

```java
public void makeLatte(int size, Milk milk){
    Cup cup = new Cup(size);
    _machine.turnOn();
    while (!cup.isFull()){
        _machine.makeEspresso();
    }
    Milk frothedMilk = _machine.frothMilk(milk);
    cup.add(frothedMilk);
}
```

In pseudocode, this might look something like:

```java
method makeLatte(cupSize, milk):
    create new cup with size cupSize
    turn _machine on
    until the cup is full:
        the _machine continues making espresso
    the _machine froths the milk
    add frothed milk to cup
```

The pseudocode should be specific, with roughly a line of pseudocode for every line of real code (though this is not exact). Note that punctuation (semicolons, brackets) are not included in pseudocode because those are language-specific.

---

The Design Questions:

1. Use UMLet to create a containment diagram for DoodleJump. If your design requires it, you should also create an inheritance diagram. (You don't need to create an inheritance
diagram just for your App extending from Application). Remember, your containment and inheritance diagrams should not be connected to each other! You must create your diagrams in UMLet as always.

2. In the Math and Making Decisions lecture, Andy introduced the static function Math.random() which returns a random double between 0.0 inclusive and 1.0 exclusive. As mentioned in the Doodle Jump handout, you’ll want to generate x and y coordinates for your platforms that are random within a certain range.

   a. Write pseudocode for a method that takes in two integer parameters, high and low, that returns a random integer between low and high inclusive. (For example, if the function is passed a min of 1 and a max of 4, the output will either be 1, 2, 3, or 4).

   Hint: look at the Making Decisions lecture to see how you can manipulate the output of Math.random() to achieve different results.

   b. The function you just wrote will be very useful for generating x and y coordinates for platforms. What constraints should you place on the coordinates you generate for each new platform?

3. Assume you have a variable _doodle that is your instance of a Doodle, and a variable _platforms which is your instance of an ArrayList of Platforms.

As detailed in the handout, at every timestep, you’ll need to:

● Update the Doodle’s velocity
● Depending on the Doodle’s position on the screen, either:
   ○ Update the Doodle’s position, or:
   ○ Move all of your Platforms down.
● Check for collisions between the Doodle and your Platforms.

The following questions are designed to help you figure out how to implement these steps.

   a. Write pseudocode for how you will update your _doodle’s position and velocity at each timestep. Use the variables _doodle and/or _platforms if necessary. Remember that your Doodle should always fall under the effect of gravity. (You don’t need to take into account keyboard input, collisions or vertically scrolling the platforms yet.)

   b. When your Doodle reaches the vertical midpoint of the screen, you will need to halt its upward movement and instead move the platforms down. Describe in a
few short sentences how you will check the Doodle’s location, and, if necessary, move every platform in your ArrayList down by the appropriate amount.

Then, write pseudocode for scrollPlatforms().

Some things to think about: When should you check for this condition? How much should your platforms actually move when this happens?

Tip: Play around with the demo to get a feel for how this vertical scrolling works!

c. Write pseudocode for a bounce() function that checks if the Doodle is colliding with any of the platforms in _platforms, and if it is, sets the rebound velocity. Keep in mind that the direction the Doodle is moving influences whether or not your Doodle should bounce off of a platform, so your function should also account for that. Please account for as many details as possible.

Hint: Make use of javafx.scene.shape.Shape’s intersects(...) method, described in the handout.