COMP 376: Introduction to Game Development

Assignment 3 (Fall 2012)

Second XNA Programming Assignment

Instructor: T. Fevens, fevens@cs.concordia.ca
Lab Instructor: Kaustubha Mendhurwar, k_mendhu@encs.concordia.ca
Due date: November 16, 2012

Overview

Your second XNA programming assignment is to develop a 3D version of the Puzzle Booble clone you developed in 2D in your previous assignment with a few differences as described below.

◊ *The game* begins with a series of 3D spherical bubbles textured with certain colors placed in a prearranged pattern at the top of the rectangular playing area in the form of a gently sloping ramp of a fixed width (with edging/curbs on either side of the ramp) and constant slope. The player/avatar is at the center at the foot of the ramp looking up the ramp at the bubbles in first person view. The slope and position of the camera should be such that the player can see all the bubbles. Imagine that the bubbles are held in place by static generated at the top of the ramp.

◊ The player aims a 3D firing mechanism that fires another colored bubble (whose color is indicated graphically in the firing mechanism prior to shooting) such that it rolls in a straight line up the ramp (the bubbles will reflect perfectly off the sides of the ramp) stopping when coming in contact with another bubble or the top of the ramp. The rolling up the ramp should be physically correct (see below). If the new bubble creates a group of three or more connected bubbles of the same color, these bubbles will pop. After the three or more connected bubbles of the same color pop, is there are any bubbles that are solely in contact with the popped bubbles, these bubbles will then roll down the ramp in a physically correct way (see below).

Basic Game Play

Basic game play is the following:

◊ Use the left and right arrow keys to aim the new bubble. The up arrow key should immediately "fire" the bubble. If the bubble is not fired within a certain small amount of time, it will fire automatically in the direction of the aiming mechanism.

◊ When a bubble stops after being fired, it must come to rest at the nearest grid position as in assignment 2 (each row offset by half the width of the bubble since a minimum planar packing of the bubbles is used).

◊ As the level progresses, after a few shots, the top of the ramp will move a fraction of the distance closer to the avatar, along with all the bubbles attached to it. The number of shots until the next encroachment of the top of the ramp depends on how many colored bubbles remain in the playing area.

- ◊ For scoring, the following should be used: the three or more “popped” bubbles of the same color are worth 10 points each; and, if any, the n removed “rolling” bubbles are worth $2^n * 10$ points (e.g., 3 hanging bubbles are worth $2^3 * 10 = 80$ points).
- ◊ If any (non-rolling) bubble touches the bottom of the playing area, the player loses.
- ◊ Physics:
  - o The slope of the ramp is constant, so the acceleration of the rolling bubble due to gravity is constant (say, parameter p which you will choose) straight down the ramp meaning that per time step the increase in the velocity of the bubble is a fixed value.
  - o When rolling down the ramp, the initial velocity will be zero and the rolling bubble will accelerate as it rolls straight down.
  - o When “shooting” a bubble up the ramp using the firing mechanism, you must compute an initial velocity such that as it rolls up the ramp, it slows down due to the de-acceleration due to gravity. You must pick an initial velocity such that the final velocity of the bubble at its stopping position is a fixed small positive velocity, not zero velocity, so the bubble can be seen to abruptly stop from this small velocity at its final position. The de-acceleration itself will depend on the angle the bubble is rolling up: if straight up the ramp, the de-acceleration is $-p$; if at an angle θ, the de-acceleration will be the portion of $-p$ in the direction straight up the ramp, $(-p)\cos\theta$.

User interface and gameplay parameters:

- ◊ For the colors of the bubbles, use 5 to 7 colors chosen so that it is easy to distinguish between the colors.
- ◊ The color of the new bubble should be randomly generated (you can change the likelihood of certain colors as the game progresses if you feel it enhances gameplay). The colors of the next two bubbles (including the one in the firing mechanism) should be visible to the player.
- ◊ The camera view at the position of the avatar in a first person view.
- ◊ Add some simple texturing to demonstrate that the bubble is actually rolling.
- ◊ The current score should be displayed on the screen.
- ◊ Include animations of the following:
  - o When the bubbles are popped, animate this effect (base it on the original 2D game, or create your own animation).
  - o Animate the three dimensional “loading” of the firing and aiming mechanism.
- ◊ You only need to have one level and a start-up screen is optional.
- ◊ You will have to suitably adjust your gameplay parameters, such as the colors used, size of the ramp, relative size of the bubbles to that of the ramp, relationship between the time between ceiling drops and the number of colored bubbles, etc., so as to yield a playable game that is neither too easy nor too hard.

Deliverables:

- ◊ Complete game project source code along data files and a read me file explaining how to read the code, compile and run the game on the PC. Please zip all your files into one archive and submit a single file. Include your student number in the name of your file. If zipped game files are larger than Moodle`s 20MB submission size limit, make arrangements with the Lab Instructor for submission.

◊ Demo your game to the TA during the lab period during the week of November 19th, and answer related questions on your XNA programming skills/experience.

Evaluation: (out of 100 points)

1. Working 3D implementation with all game play: (35%)
2. Physics: (15%)
3. Setting and playability (User Interface, Game world, …): (15%)
4. Aesthetics and overall impression: (20%)
5. Q&A with Lab Instructor (to demonstrate understanding of XNA programming): (15%)

Late penalty – 20% per day of delay.

Ground Rules:

You are welcome to discuss high-level implementation issues with your classmates and others, but you should avoid actually looking at one another student's code as whole, and under no circumstances should you be copying any portion of another student's code or copying complete code from the Internet. However, seeking help from other students for debugging some portion of your code is reasonable. Basically, these "ground rules" are intended to prevent a student from "freeloading" off another student or from the Internet, even accidentally.