Michael Choquette

12/1/2010

My project is a physics-based puzzle game called Gravishift. In the game you have one or more balls, several goal zones, and pieces called tiles (with direction up, down, right, or left) that push any ball touching them in the tile’s specified direction. The player’s goal is to place the tiles so that when they press play (key “s”), the tiles will push the balls through every goal zone on the screen. I created the core of the game – the bouncing ball -- first, then added tiles, goals, start/stop and restart buttons, and victory recognition to bring my project to a b level. My next major step was to implement collision detection for obstacles other than the canvas walls, including accurate corner collisions. Then I set out to make a good user interface with better buttons and a main menu/level select. After the main menu was looking as nice as I wanted, I created 12 levels, with the first three being tutorials and the difficulty increasing from there. After all the levels were completed to my satisfaction, I made it save the player’s high scores for each level, and only let them select a level in the menu if they had beaten all the previous levels. In addition, I added my best scores for each level into the game: if the player matches or beats my high score for a level the button to go to that level turns gold.

Overall my game is split into four modules, with the main one being Gravishift.py. The other three are:

GSclasses: contains the classes for balls, tiles, buttons, goals, and obstacles

GSlevels: contains the data for initializing each level, as well as the main menu

GSscores: contains the score-saving/loading functions

The general flow of my program is as follows: run in Gravishift loads the scores from GSscores, then calls init which calls GSlevels to find out what to do. GSlevels, in making all the objects on the screen, repeatedly calls GSclasses to make balls, obstacles, tiles, goals, and buttons. The program then waits for input in the form or key and button presses, which are handled in the main body of Gravishift with repeated forays into GSclasses to modify objects. If the user starts the simulation, timerFired starts up and repeatedly calls moveBall, which calls PostCollisionPosition, which calls itself as needed, as well as using GSclasses to find points of intersection. The code that runs for the vast majority of the time is in the main file and GSclasses; GSlevels is only used for screen changes and GSscores is only used to save scores when a level is beaten. For more specific information about the functions in each of the files, see the other files in this folder.

User Interface: my goal was for the user interface to be clear but not the main point of the program; for that reason I kept the buttons simple and few in number and avoided flashy effects outside of the gameplay. I thought carefully about the help text in the menu and the first three levels; I didn’t want the instructions to be overwhelming or hand-holding, and so had to rewrite them several times to make sure they were sill useful.