PROJECT 2: R&A

For my prototype, I am creating a game that involves path-drawing in order to guide a ball into a target. A ball drops from a certain place in a box, and you have to draw lines inside this box to create a bridge or path on which the ball can roll into the target. I wanted to create a puzzle game in which the user could construct something in order to get to the answer, and where there may be more than one right answer. I liked the idea of drawing one's own solution, so the path idea came up. There are multiple path-drawing games out there that I have played, but the two that I've drawn the most inspiration from are *Sugar, Sugar* and *Trace*.

Sugar, Sugar is a Bart Bonte game that can be found in many different free gaming sites on the Internet. The mechanics of the game are as follows. "Sugar," in the form of tiny little solid squares, flows out of a spout at the top of the screen, and our job is to guide a certain number of them into cups placed around the screen by drawing paths along which the sugar particles are guided. Because the sugar is comprised of small, discrete particles, they are very "floaty" and they drift serenely to the bottom of the screen, landing more or less directly below from where they fell. Because of the simulated air drag and the somewhat thin trickle of sugar, the particles do not fall all at the exact same place, but drift apart a bit as they fall down, which makes the sugar look very light and in some ways hard to control. Players are allowed to draw as many lines as they want, and at whatever times they want, to guide the sugar into the cups (most often by creating a "slide" into each cup that the sugar can follow).

The puzzle aspect of it is interesting, since we sometimes have to be creative with where and when we draw the slides in order to fill all the cups. For example, because we

cannot erase any line we've drawn at each level, we have to decide which cup we should draw a slide to first, because sometimes one slide can block the sugar's path to another potential slide. Similarly, we have to place the slides well so that we don't restrict the paths of the sugar too much. There are also several gimmicks that I've encountered in the few levels I've played. They make effective use of holes, so that one hole at the bottom of the screen corresponds to another at the top, and sugar that falls through the hole at the bottom thus comes out of the hole at the top. Also, sometimes there are differently-colored cups, and we have to pass the sugar through small "filters" on the screen to turn their color before we can guide them to the correct cup. Another is switching the direction of gravity. In some levels, there is a button that, once it's pressed, it changes gravity from pulling the sugar down to pulling it up. These become added complications to overall gameplay, as we have to figure out how to direct the sugar into the cups with these new restrictions. The gravity shift is the most difficult to work around since the motion of the sugar becomes non-intuitive. We have to make sense of how to best work with it after having worked with gravity going in the other direction for so long.

However, the one thing about this game is that there isn't really any way to fail. At one point, I waited too long and the sugar ran out, so I stuck without any way to fill the cups, but otherwise the sugar flowed for a long while and in large quantities, and the cups' capacities were not very high so that isn't a limiting factor either. Additionally, all the gimmicks were not too difficult to figure out—even with the weirdness of the gravity shift, it was easy to get used to it after a couple tries. The other two barely changed gameplay; they just added an extra target that we have to account for before we reach a cup. The hardest aspect was honestly the fact that we couldn't erase lines, so we had to

plan our lines well. So, in my game, I'm trying to make it a little harder by limiting the number of lines one is allowed to draw in the game to get the ball to the target. Additionally, in order to really stress good planning, I want to design it so that the player has to draw all the lines first, and then when the ball is rolling no changes can be made. Finally, my ball will be one heavy object to guide, rather than a thousand light objects like the sugar, so this should change the feel of the game physics quite a bit. The ball will have much more inertia than the sugar pieces, so it might be harder to control, but it will also be one consolidated object, so in that sense it might be easier.

The other game I've played, *Trace*, is an iOS game that also involves path drawing. In it, a person needs to be able to walk from one point on the screen to another (the target), and often there are obstacles in the way that we need to fix (for example, there might be a big hole between the person and the target, so we have to draw a bridge that he can walk on). The biggest difference between this and my game idea is that you can control the person walking. You can tell it to go left or right or to jump at any point in each level. Additionally, in this game we can erase parts of a path whenever we want. This, added with our control of the person's movement, makes the interaction between our paths and the person much higher. For example, it is now possible to walk partway onto a path, erase some part of the path that is now a hindrance to the person in their new position, and draw another piece of the path to advance the person more. This is especially important in this game since they also have included a much higher element of failure. There are "enemies" or dangerous objects in this game, so that if the person touches them, they are automatically replaced at the start of the level again. Thus sometimes drawn lines need to not only be paths to reach a goal, but also avoidance

mechanisms for moving, harmful objects in a level. I liked the idea of getting hurt (having a way to fail) in this game, so I thought I might create either stationary harmful objects (that a wildly bouncing ball would have to avoid), or moving ones as well. I thought maybe I could expand on the idea of avoidance more by making lines actually protect the ball, so one could trap an enemy with a circle drawn around it. However, that would also use up one of the allotted number of lines allowed to be drawn in each level, so there would be some give-and-take in that decision.

The actual physical mechanics of *Sugar, Sugar* and *Trace* are very similar. One presses down on the mouse (or on the touchscreen in the case of *Trace*, a mobile game) in order to start creating a line, and drags it across the screen. The line ends when the mouse is unpressed. The line follows the mouse exactly, so if the player's hand shakes, the line will be shaky as well. A shaky line is translated into how the objects move on top of it—for example, sugar can build up in small valleys, or the person would have to jump in order to navigate a zig-zagged line. This can be used either against or for the player, depending on their goal. I feel that in my game, since the ball will move of its own accord and is probably bigger than any rough patches in a line, this will be less of an issue, so the mechanics may be smoother. Otherwise, though, I am planning on making my line mechanics about the same—lines are created with the press of the mouse and ended when the mouse is unpressed, and all shakes in the line are kept in the final product. It may be annoying if someone were going for a smooth line, but hopefully it'll make gameplay for my game just as juicy and interesting as it was for me when I first played *Sugar, Sugar* and *Trace*.