Leo Zhang, Thomas Liautuad, Naomi Nguyen
Prof. Asish Law
CS 174A
12 June 2024

Fruiter: Final Report

COM SCI 174A Project Report

In the game Fruiter, you are to drop fruits into the playing field. When identical fruits touch they will merge into a larger and a more valuable fruit. Merging 10 fruits (Grape, Blueberry, Kiwi, Pear, Apple, Mango, Orange, Melon, Coconut, Watermelon) in a row to create the end goal of achieving a watermelon, the ultimate fruit. Users will gain a set amount of points for dropping a fruit but will receive another set amount of points for which types of fruits they are able to identically match. Dropping a fruit would be 5 points, matching grapes would be 10 points, matching blueberries would be 15 points, matching kiwis would be 20 points, matching pears would be 25 matching apples would be 30 points, matching mangos would be 35, matching oranges would be 40 points, matching melons would be 45, matching coconuts would be 50 and matching watermelon would be 75 points. But matching watermelons would make them disappear from the box, indicating it as an endless mode type game. Which fruits you drop will be randomized to the first 3 smallest fruits, but grapes will always be dropped first. Once fruit drops to the bottom of the box, it can combine with another fruit through collision to become a larger type of fruit. The objective of the game is to combine a pair of larger and larger fruits until reaching the largest tier at which point the user wins the game. To add more variety to our game, with every 25-50 fruit drops, there will be 2 types of cube (small and large) that are dropped to hinder the user's gameplay. These cubes can be merged to then also disappear so the user can get back to matching their fruits. Your score will be displayed at the top left corner of the screen as well as your high score.

Users can decide where in the x-y plane they want to drop the fruit into the box with arrow keys or a/d and pressing the s or down arrow would drop the fruit where they have decided. Users can also decide if they want to reset the game and start anew by clicking the key q. Users can even play the game on fast mode by clicking the key t where they will not be restricted on how fast they want to drop the fruits.

We encountered significant challenges with the physics and collision behavior of the fruit objects, represented as spheres. Initially, achieving a realistic rolling motion for these spheres required substantial trial and error, along with a thorough review of physics concepts within lectures. First off, we wanted to implement the basics of both elastic and in-elastic collisions for our game, where fruits that were the same type that collided would do so inelastically and thus conserve linear momentum, and fruits that were not the same would collide elastically, albeit to an extent due to the necessity of introducing a restitution coefficient to stop the fruits from bouncing too far from one another. Additionally, once the textures were added, it became obvious that when balls rolled on the floor, they would not actually roll but rather translate from one part of the floor to another. We had to introduce a rotation transform that would only be used in the drawing method if the fruit's horizontal velocity magnitude was greater than zero, as it is the fruit's horizontal velocity as well as its position on the floor that dictates whether it should roll or not. Furthermore, the introduction of textures themselves to make the spheres visually resemble fruit presented additional difficulties. Although the spheres rolled and bounced as intended, the stationary textures created a visual inconsistency, making the fruits appear unrealistic. To address this, we applied a technique similar to the scroll-and-spin mechanism used in Assignment 4 for cubes, ensuring that the textures moved congruently with the rolling

motion of the fruits. The integration of various shapes, based on peer review feedback, further complicated the physics. We added cubes as game obstacles, initially treating them as a separate variable from the fruits. This approach resulted in unrealistic movement for the cubes. By including the cubes within the fruit variable declaration and assigning them appropriate materials, we were able to achieve more realistic interactions.

Incorporating shadows to enhance visual realism also posed challenges. Initially, we attempted to use the floor's material for the shadows, which led to significant bugs as the program struggled to distinguish between the shadow and the floor, causing crashes. We resolved this by introducing a new element for the shadow, matching the fruit's shape and coloring it black. This solution minimized errors, although it occasionally produced a ring around the fruits when they were in contact with the floor. Implementing an if-statement to make the shadow disappear upon floor contact resolved this issue.

Adding a scoreboard was straightforward in terms of updating score increments with each merge and drop event. However, displaying the score without cluttering the already crowded game screen required integrating it into the website. This necessitated modifications to the index.html file from previous assignments. By learning basic HTML and understanding JavaScript manipulation functions, we successfully displayed both the current score and the high score in the top left corner of the website. For the Loss Condition indicator, we also initially had trouble with texturing the bitmap to select the correct letters. Following the example files provided by the TA’s in the past homeworks, we were able to understand the behavior of the unique texture-mapping of each letter given the material properties. Once we had solved this

challenge, we were able to select an online image bitmap to display the loss condition in a unique font, adding to the stylistic creativity of the scene as a whole.

While calculating timers for various functions (i.e loss timer, timer between drops), we initially had trouble due to the time being recorded at irregular intervals due to the frequency of the function not precisely lining up with exact time values; an example to illustrate this was that you could not directly compare (t = 1.0) since the render may happen at t = 0.99 and 1.01. However, this was easily solved using delta time (dt), which could increment time variables to easily implement different timers for any type of function we needed in the project.

While collisions were solved in the later half of the project, we initially had trouble with energy systems since without conservation of energy we encountered vibrating balls, clipping through walls, and a variety of different complications. However, with the addition of damping and position re-assignments (in addition to velocity re-assignments), all of the major bugs were able to be addressed effectively. Additionally, we used a minimum velocity threshold to halt the movement of the fruits after it was already moving slow enough, to simulate the effects of dynamic friction stopping the fruits.

With more time, we considered several different extensions of the game in terms of both game mechanics and graphical/coding advancements. In terms of the game mechanic side, we considered adding different types of obstacles and power ups to make the game more entertaining and add variety. We thought about adding a combo system for the scoreboard, adding score multipliers not only for fruit types (which we already implemented) but for example specific orders of fruit combinations and/or “chaining” combinations together using the physics. On the technical side, we thought about advancing the physics so that it would have diagonal

collisions as well, adding mass-consideration to the collisions (although this is mostly dealt with through radius considerations), different camera angles and enhanced textures/mappings of the fruits so they look more realistic or cartoon-like depending on the intended stylistic choice. However, the highest priority items we would add if we were given more time would be to supplement the physics with greater flexibility, allowing fruits a greater range of angles/movement within the box.

The peer review process was helpful with determining areas that needed improvement and to hear suggestions from fellow students. Given that this is the first time they've seen our project, the feedback that provided possible suggestions for each criteria (for example, suggestions to increase creativity) was helpful for brainstorming extensions to the project that we could add. Additionally, peers pointed out areas for improvement that we had not initially considered significant, and we were able to focus on bug-fixing and other priorities more efficiently given the feedback.

Suggestions for improvement of the project process would be to have more practice with the Javascript graphics library to prepare for the project, maybe increasing the amount of homeworks while decreasing the length of each homework. This is because the project felt more like a "figure it out as you go" where the basics of the homeworks were not enough to extend to the more advanced objectives of the project. Therefore, more practice with just using the library would be extremely helpful as you could spend more time implementing what you want for your project, using ideas from the course like applying different mappings/shadings, without spending too much time figuring out technical syntax and structures.