When I first approached this assignment, I knew I wanted to create a more polished and interactive 2D animation. My primary focus was improving the brick layout, implementing realistic physics, and ensuring smooth player controls—all while maintaining clean, well-structured code.

I began by reorganizing the bricks into a neat grid at the top of the screen, replacing the original scattered arrangement. This not only made the design more visually appealing but also set the foundation for proper collision detection. I introduced different types of brick, some indestructible and reflective, others destructible with varying durability. Implementing this feature proved more challenging than I anticipated. At first, some bricks would disappear after a single hit, while others failed to register collisions at all. After careful debugging, I realized the issue stemmed from improperly initialized hit points and inconsistent collision checks. By refining the Brick class to track “hitPoints” accurately and updating each brick’s color as it took damage, I finally achieved the intended behavior. It was incredibly satisfying to see the destructible bricks break apart after multiple hits while the indestructible ones remained solid.

Next, I turned my attention to user interaction. I added a paddle at the bottom of the screen, controlled using the left and right arrow keys. Ensuring smooth movement required precise event handling, but the real challenge was making the ball interact realistically with the paddle and bricks. I implemented “speedX” and “speedY” attributes to control the ball’s movement, applying proper reflection angles upon collisions. Physics took some trial and error, getting the bounce just right when the ball hit the paddle’s edges was particularly tricky. To make the animation more dynamic, I added a mechanic where colliding balls would merge into a larger one, adding an unexpected twist to the gameplay. They don’t always collide properly with the blocks, so hit detection needs to be improved.

Throughout the process, I prioritized clean, maintainable code. I refactored the entire project, using descriptive variable names like “isIndestructible” and “ballRadius” to improve readability. By including detailed comments, especially around collision logic, I made sure the code would be easy to modify later. By the end, not only did the animation function as intended, but the underlying structure was robust enough to support future enhancements.

This project taught me the importance of persistence in debugging and the value of well-organized code. What started as a simple animation evolved into a challenging & dynamic project.