Artifact Description

The artifact is the same as in the previous enhancement. It's a 2D-graphical program from cs330 computer visualization course, then further enhanced in my former enhancement. After enhancement, it went from a simple 2d graphical program into an interactive brick-breaking game implemented using classes like Brick, Circle (representing the ball), and Paddle

Justification for Inclusion

I selected this enhanced artifact for my ePortfolio because it demonstrates my ability to significantly improve an existing project algorithm design and data structure implementation, and because it is related to graphics programming and game development, which are two passions of mine.

Specific components that show my skills include:

- Procedural Generation: Overhauling the brick placement system to create dynamic, grid-based levels with randomness, and a color-picking algorithm for the boxes.
- Refining the collision detection algorithm: It now accurately calculates the closest points between the ball and the bricks to solve issues like ball sticking and improve realism.
- Data Structure Usage: Incorporating std::tuple for RGB color values and std::pair for grid positions. I also had to implement a system for managing state in the game.
- Game Design: Multiple levels, difficulty adjuster, and user record keeping.

Course Outcomes

- Designing and implementing complex algorithms: I believe that my new procedural generation algorithm and refined collision detection meet this outcome.
- Utilizing new data structures: The use of tuples for color management and pairs for grid positions shows my skills in choosing suitable data structures.
- Problem-solving in software development: Refining the game, improving gameplay mechanics, optimizing performance and realism, demonstrate this outcome.

Reflection on the Enhancement Process

Lessons learned:

- The complexity of collision detection: Refining the collision system required research which I found to be fascinating. It made me understand the importance of precise calculations as well as how to optimize performance, as this can be a very expensive calculation to make. Writing a good algorithm and using an appropriate data structure is the difference between an enjoyable experience and a stutter-filled experience.

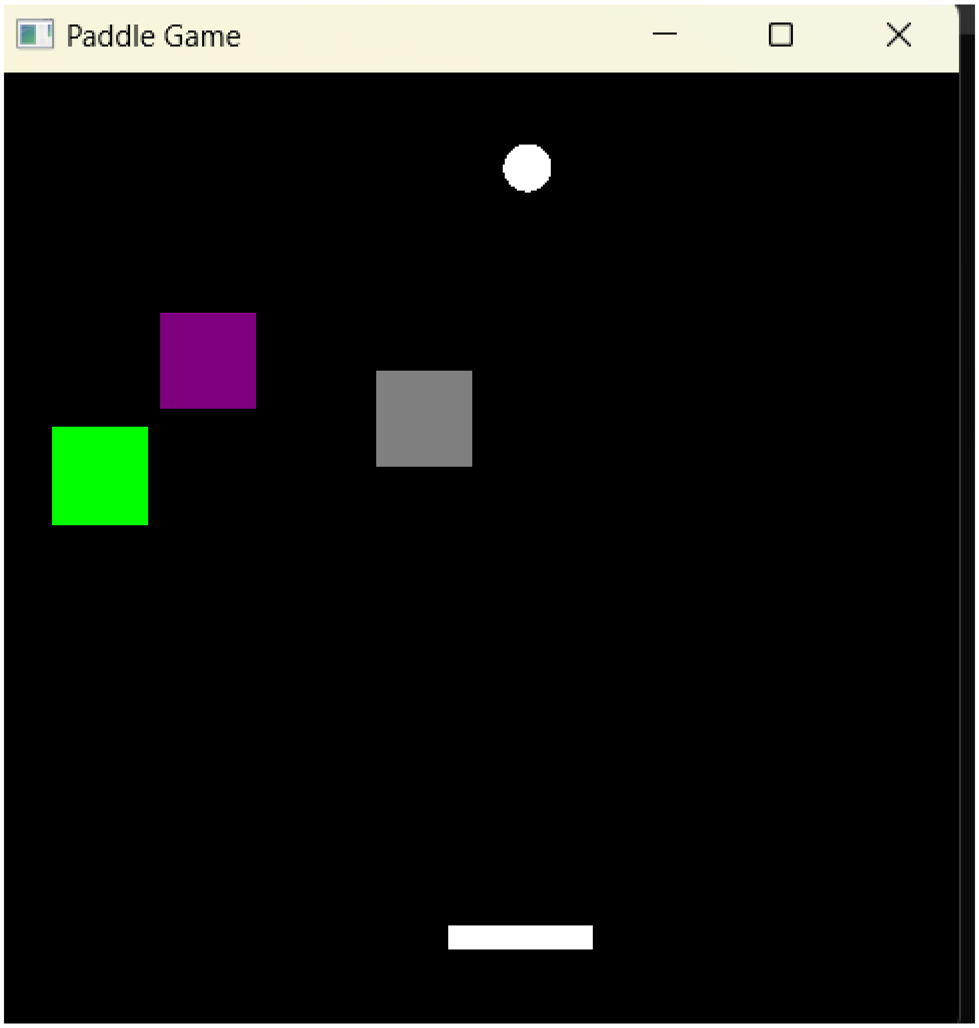
- Since this enhancement resulted in a much more significant re-work of the program, I learned the importance of software architecture and how the systems interact and affect each other.

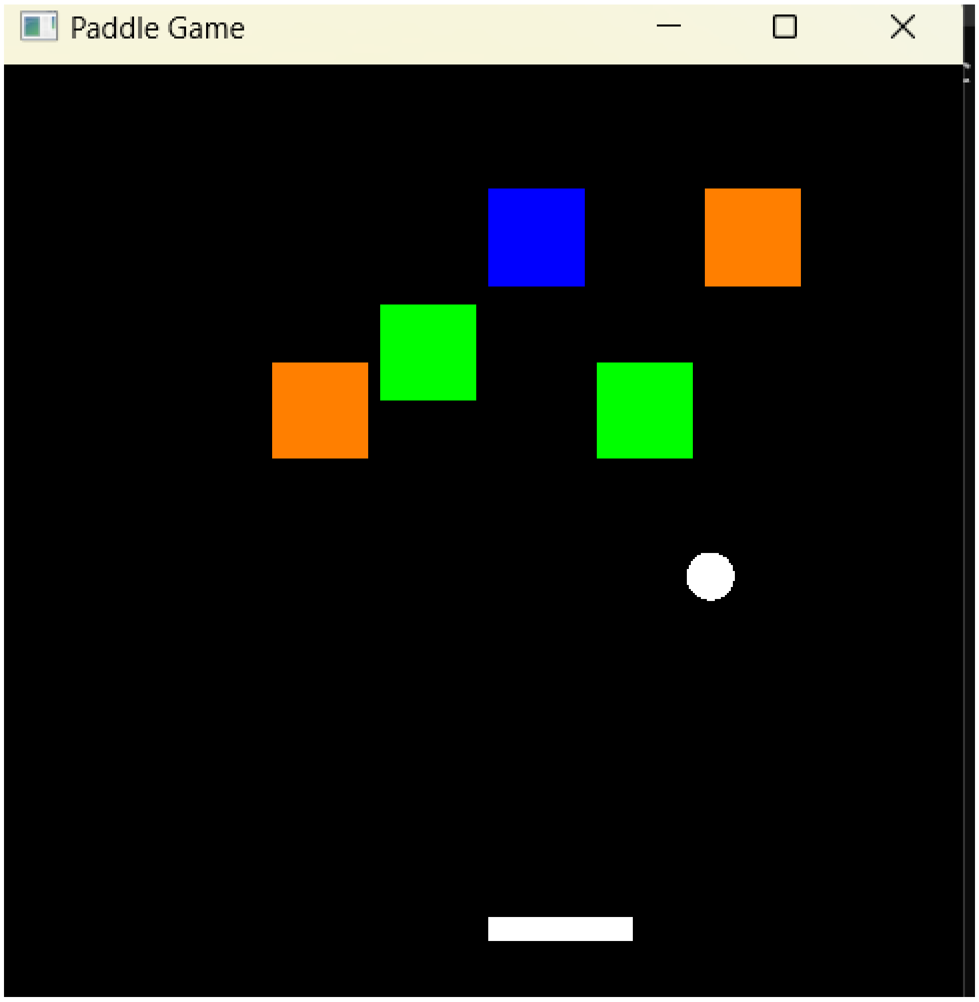
Challenges faced:

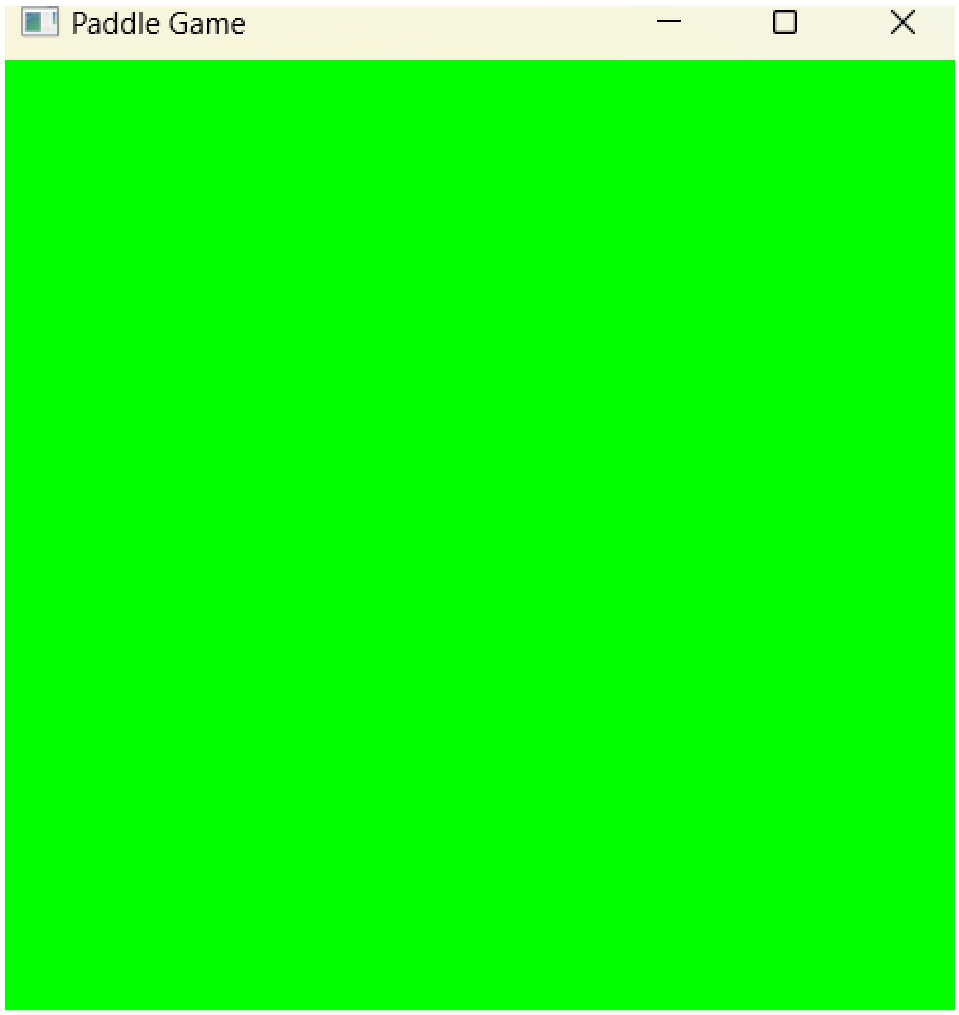
- Balancing randomness: I noticed that too much randomness resulted in the game reaching a “deadlock” state, where the player could not reach the desired box to hit due to another box (that can't be destroyed) blocking it. This required improving the randomness algorithm to not be too random as well as not to clip outside of the screen.
- Optimizing the collision detection algorithm to remain realistic and fun while maintaining accuracy and performance
- Introducing new features like different colors, game states, and others required extensive rework of the code base, so the challenge was to do it while keeping the codebase stable.

I found this project to be a great learning opportunity to apply theoretical knowledge to a practical application. It improved my understanding of Algorithms and data structures, graphics rendering, game development, optimization, and a further more advanced look into real-life software development.

Screenshots:


Paddle Game


Paddle Game


Paddle Game