Pacman Debug Report

Summary

Bugs Fixed / Features added

1. **Identifying and Resolving Memory leaks.**
2. **Added animation to the Packman character.**
3. **Randomized the movement of characters.**
4. **Added more Ghost characters.**
5. **Added fix for the problem of ghosts getting stuck inside the den.**

Problem: The game crashes after running for 20-30 seconds.
Solution: Identifying and Resolving Memory Leak

The main challenge involved identifying and resolving a recurring game crash that occurred approximately 20 to 30 seconds after launch. Through rigorous debugging, I successfully identified an unhandled exception within the Drawtext method of the Drawer.cpp file. However, the issue persisted despite attempts to resolve it by implementing a try-catch block.

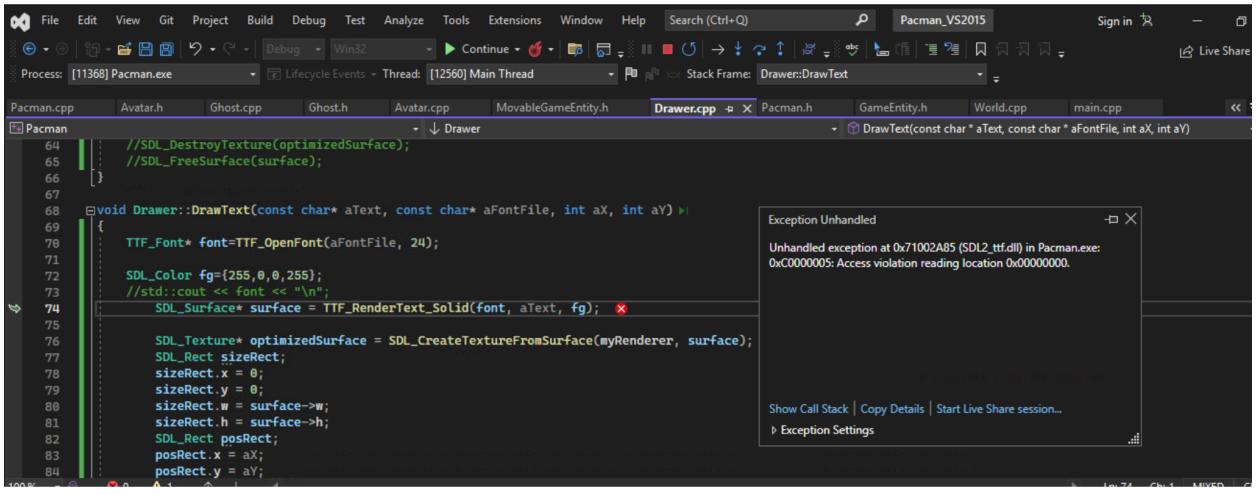


Image 1

During thorough research on StackOverflow, I discovered that the issue stemmed from a null value assignment to a variable, leading to unexpected behavior. To validate this theory, I implemented code to print the value of the "font" parameter and determine whether it indeed contained a null value.

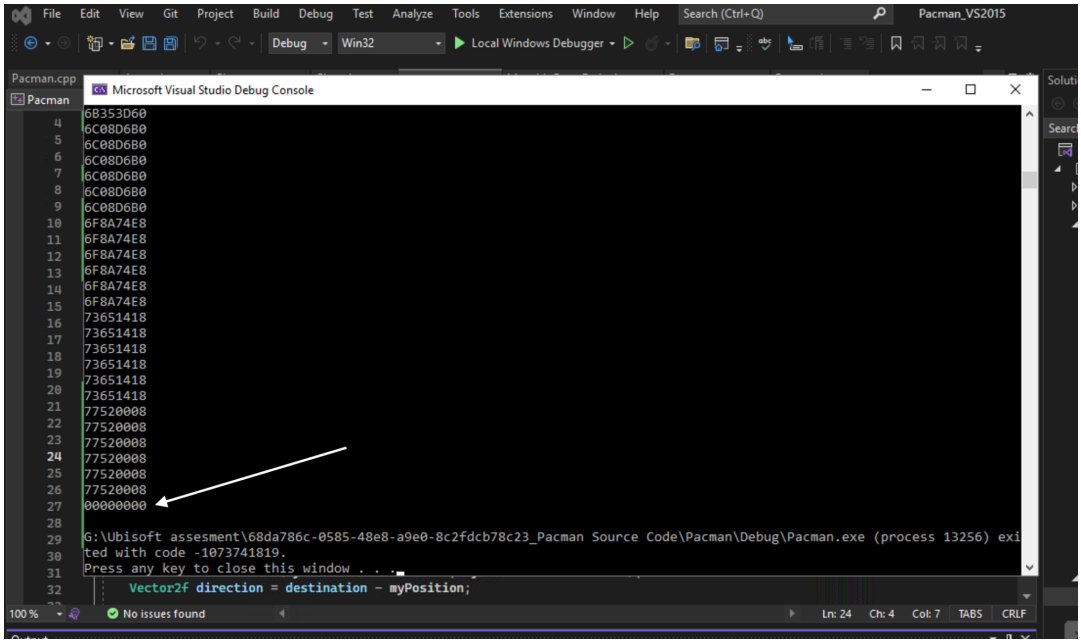


Image 2

Upon examining the final value of the "font" variable, it became evident that the program was crashing due to a null pointer exception. However, since the Drawtext method functioned properly during the initial 20 seconds, it indicated a potential memory leak within the code. A closer inspection revealed that the Draw method lacked the necessary DestroyTexture and destroyFreesurface calls, resulting in the memory leak. Once these lines of code were added (as referenced in the image below), the game stopped crashing, resolving the issue successfully.

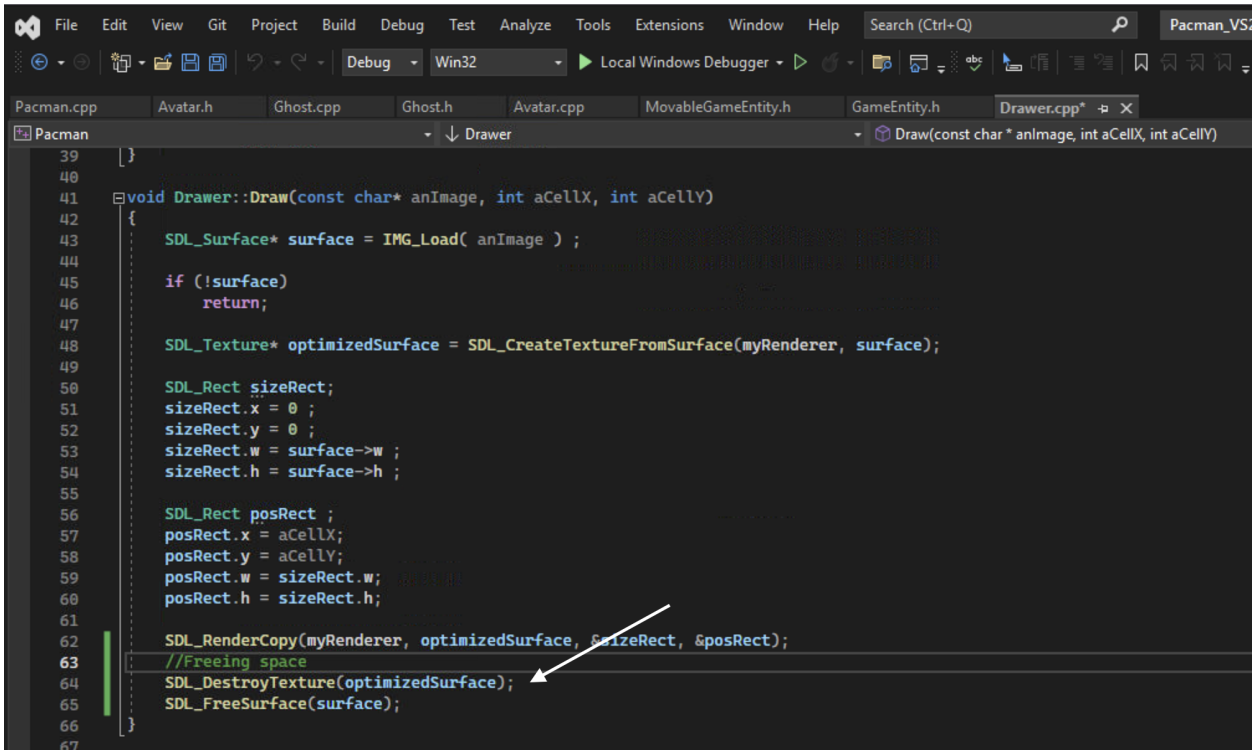


Image 3

Problem: No animation for the Packman character
Solved: Added animation to the Packman character

To enable animation for the Pacman character, it was necessary to track its orientation. To accomplish this, I added a currentOrientation variable of type string to the Avatar.h file. This variable stored values such as "left", "right", "up", and "down" to indicate the Pacman's direction. Additionally, I implemented an Animate method in the Avatar.h file. This method accepted a string parameter named position and was responsible for animating the Pacman character based on the provided position.

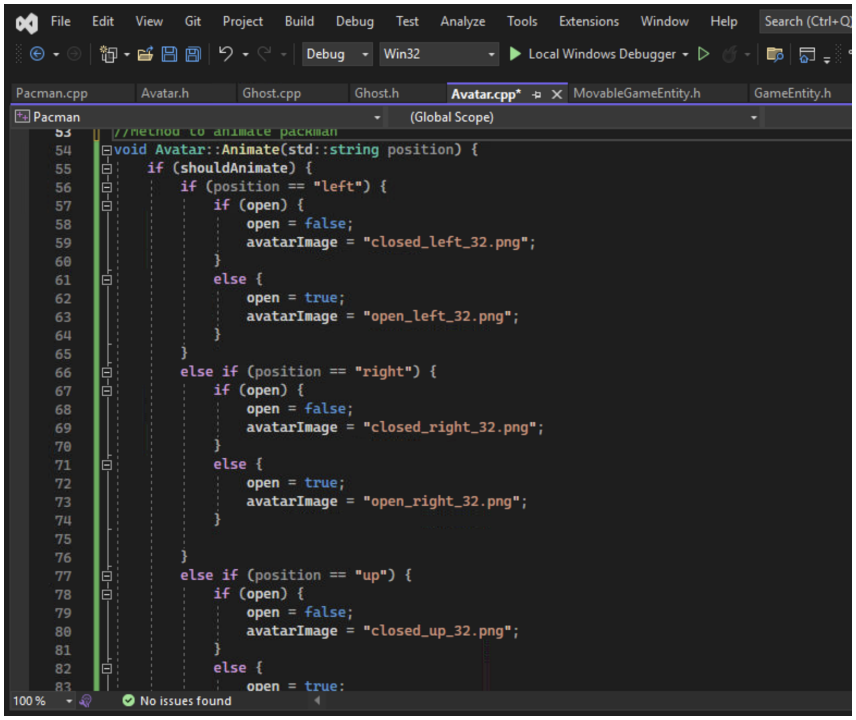


Image 4

This method would animate the Packman using a simple boolean logic as shown above.

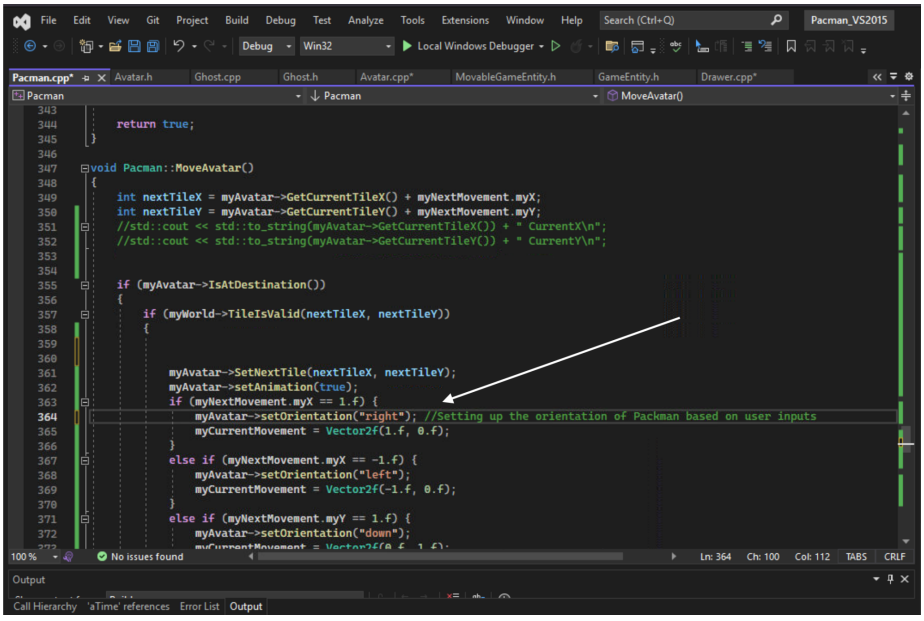


Image 5

The orientation of the Packman is being changed based on the user input also it is important to note that the value of orientation is not changed until the next tile is valid.

Problem: Pacman Stops at Invalid NextTile Value
Solved: Added a new variable to keep check

If we press the arrow up button and the path above is invalid the Packman stops moving and gets stuck.

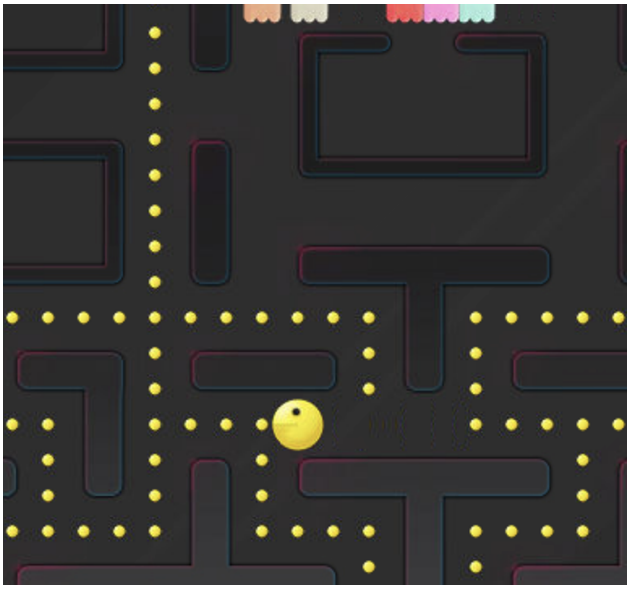


Image 6

To solve this problem I created a new **Vector2 variable** **myCurrentMovement** and its value gets updated only when the **nextTile** is valid. And if the tile is invalid then I assign this value to **myNextMovement** variable so that the Packman keeps on moving in the previous direction that it had.

Problem: Animation Continues Even After Pacman Reaches a Dead End
Solved: Added a boolean check

To achieve this feature I simply put a boolean check **shouldAnimate** inside the **Avatar.cpp** file. The value of this variable is changed from **MoveAvator()** function if the next tile is valid.

Problem: The game had only one Ghost character
Solved: Added 5 Ghost characters for a better gameplay

To add the Ghost characters I declared the characters in the **Packman.h** file. Since all the ghost characters had an almost similar function I just copied the ghost action code for all the other ghosts.

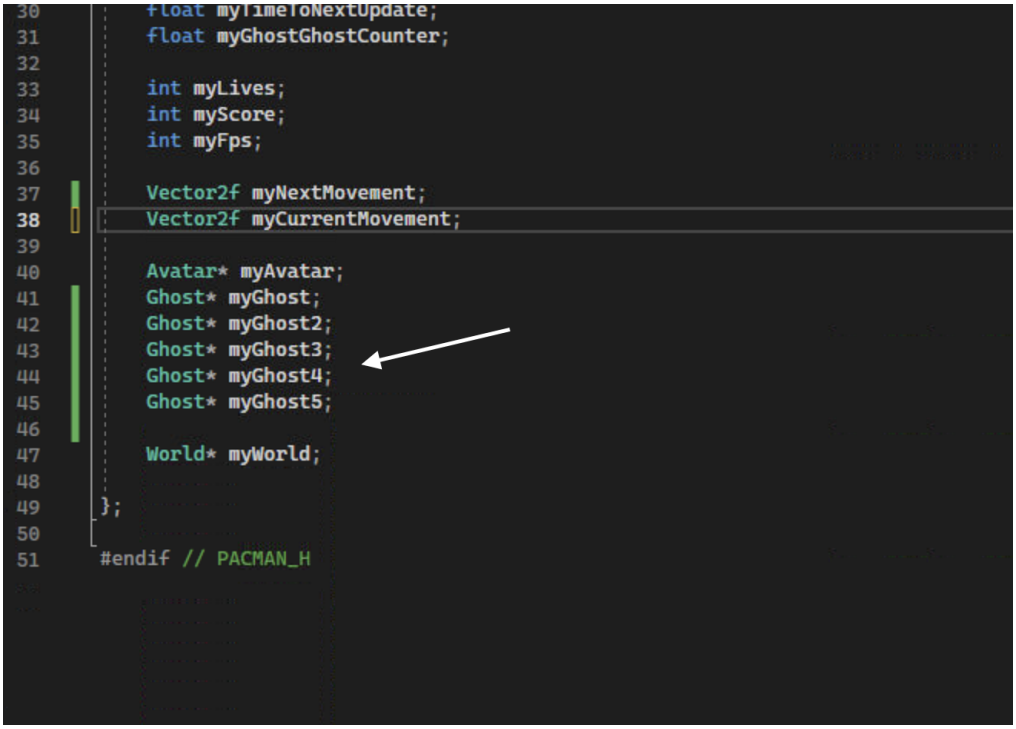
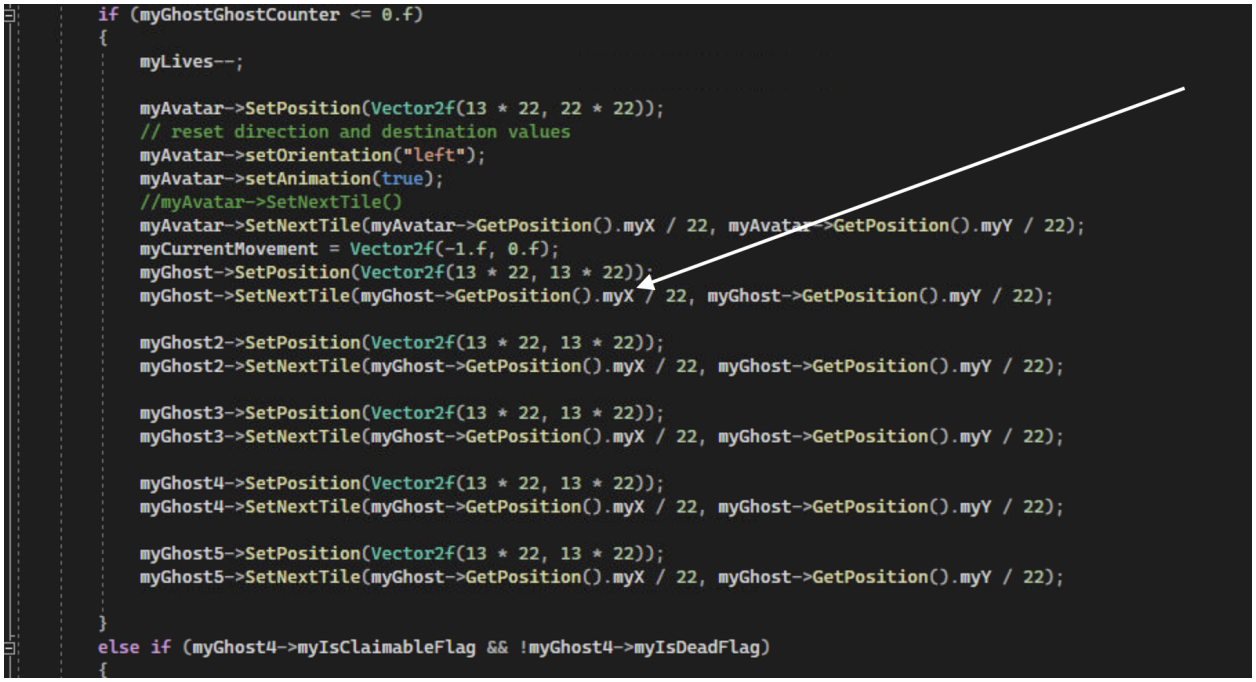


Image 7

Problem: Abnormal Ghost Behavior After the Game Reset

The ghost character, as well as the Packman character's next tile, was not being reset so as soon as Pacman dies the ghost directly moves to the next tile which it had set before death.



```
if (myGhostGhostCounter <= 0.f)
{
    myLives--;

    myAvatar->SetPosition(Vector2f(13 * 22, 22 * 22));
    // reset direction and destination values
    myAvatar->setOrientation("left");
    myAvatar->setAnimation(true);
    //myAvatar->SetNextTile()
    myAvatar->SetNextTile(myAvatar->GetPosition().myX / 22, myAvatar->GetPosition().myY / 22);
    myCurrentMovement = Vector2f(-1.f, 0.f);
    myGhost->SetPosition(Vector2f(13 * 22, 13 * 22));
    myGhost->SetNextTile(myGhost->GetPosition().myX / 22, myGhost->GetPosition().myY / 22);

    myGhost2->SetPosition(Vector2f(13 * 22, 13 * 22));
    myGhost2->SetNextTile(myGhost->GetPosition().myX / 22, myGhost->GetPosition().myY / 22);

    myGhost3->SetPosition(Vector2f(13 * 22, 13 * 22));
    myGhost3->SetNextTile(myGhost->GetPosition().myX / 22, myGhost->GetPosition().myY / 22);

    myGhost4->SetPosition(Vector2f(13 * 22, 13 * 22));
    myGhost4->SetNextTile(myGhost->GetPosition().myX / 22, myGhost->GetPosition().myY / 22);

    myGhost5->SetPosition(Vector2f(13 * 22, 13 * 22));
    myGhost5->SetNextTile(myGhost->GetPosition().myX / 22, myGhost->GetPosition().myY / 22);

}
else if (myGhost4->myIsClaimableFlag && !myGhost4->myIsDeadFlag)
{
```

Image 8

To counter this problem, I changed the value of the next tile for all the elements once the Packman dies.

Problem: The ghosts in the game were getting stuck inside the den.
Solved: Added 2 points inside the den area. When the ghosts intercept these points their movement direction changes in an upward direction.

When the game starts some of the ghosts fail to move outside the den, to counter this problem I copied the tile number of the circled tiles and whenever a ghost's next tile is this tile I change the direction of the ghost to move upwards, this way they are able to move out of the den.

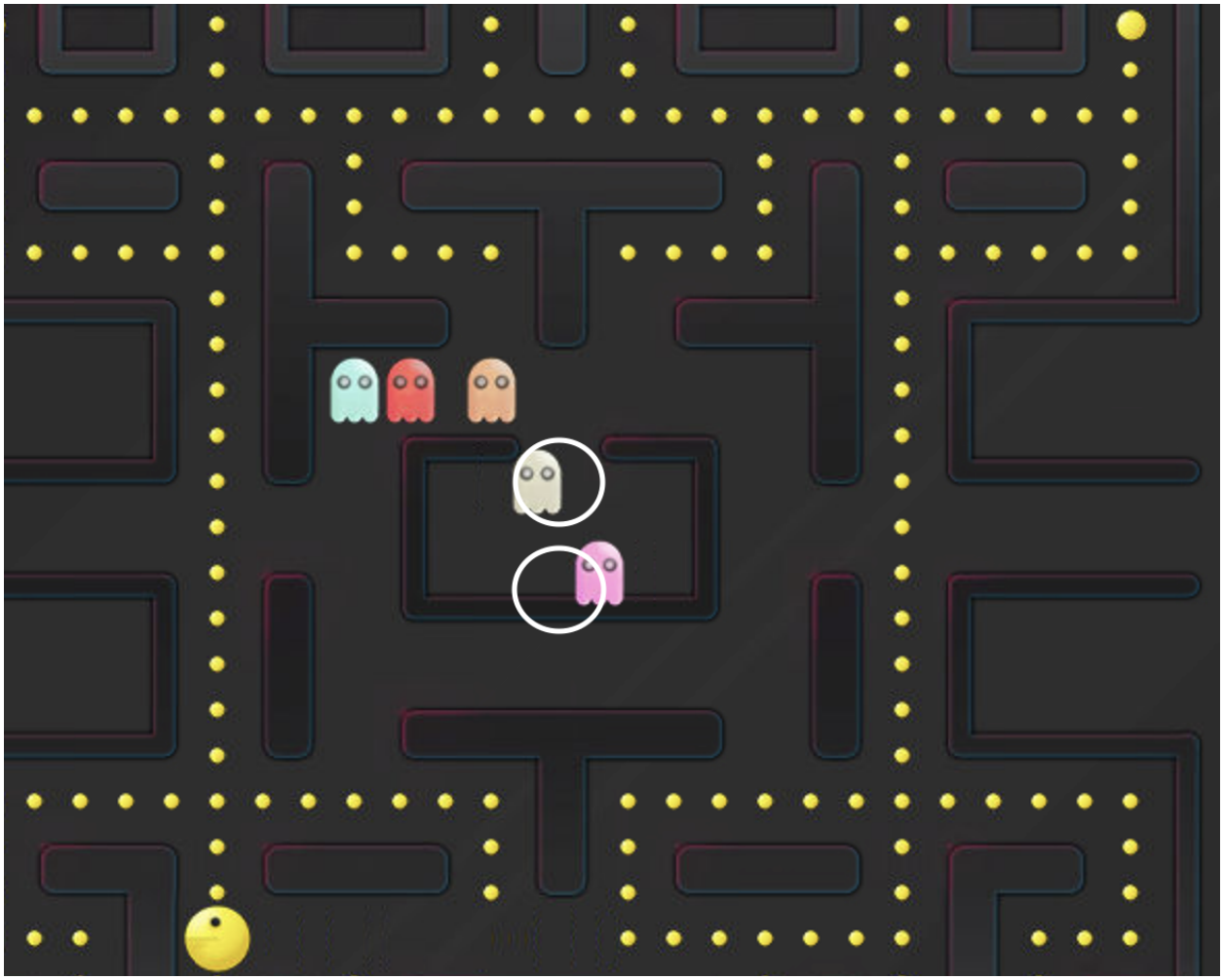


Image 9

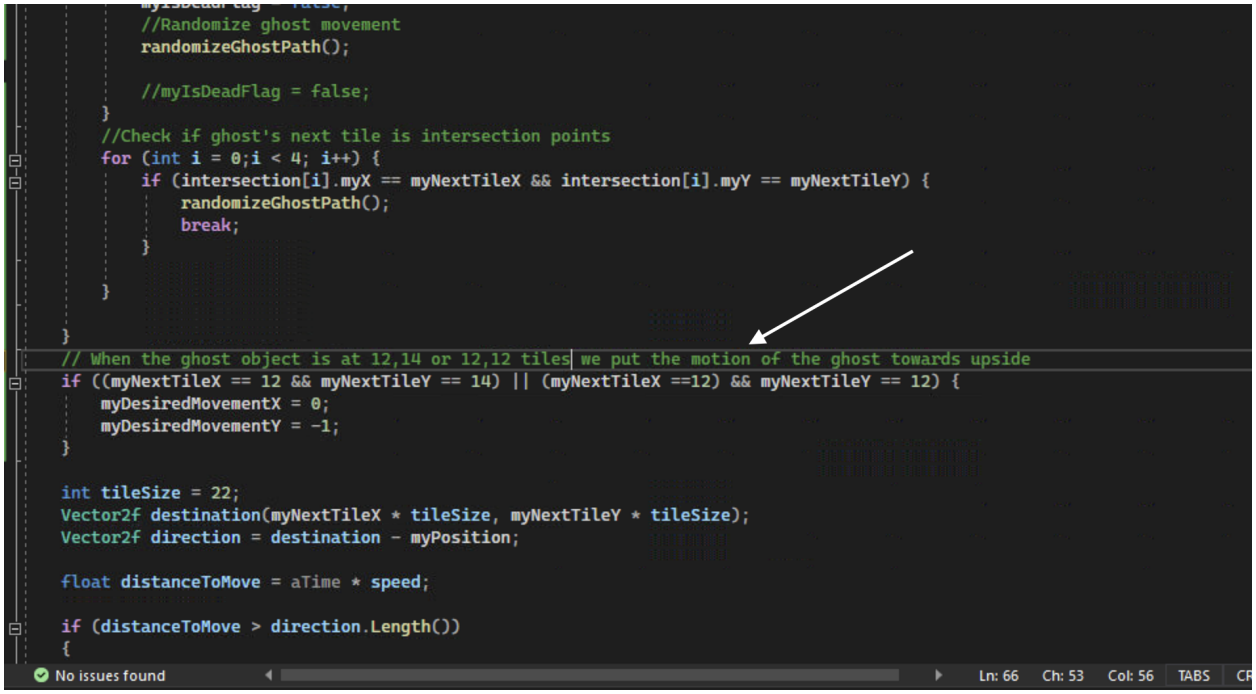


Image 10

These changes are made inside the update function of Ghost.cpp file.

Problem: The movement pattern of the ghosts in the game became predictable and repetitive, leading to a less engaging gameplay experience.

Solution: To enhance the gameplay and add more variety to the ghost behavior, the ghost movement was randomized. By introducing randomization in their movement decisions, the ghosts exhibited less predictable patterns, making it more challenging for players to anticipate their movements.

Earlier the game had a similar pattern of movement for all the ghost's characters. I changed the code to make it feel more randomized and make the gameplay fun. So I implemented a way to make sure that whenever any ghosts are encountered with a blocking invalid tile, I randomize the outcome of this event.

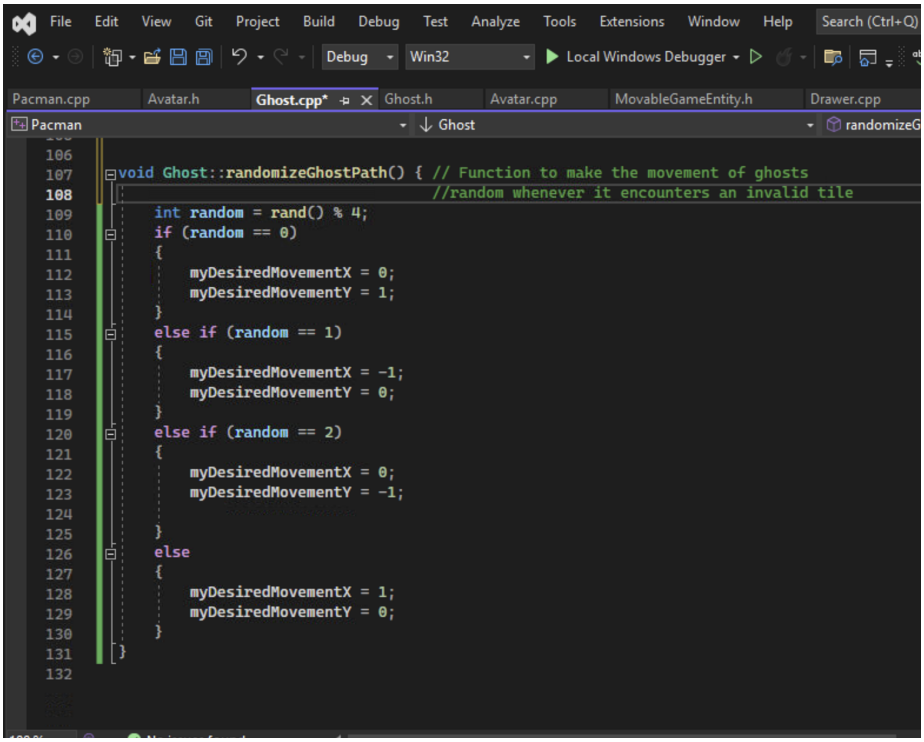


Image 11

This method is called from the Update function of Ghost.cpp whenever the next tile is invalid.

Problem: Ghost movement doesn't change at the intersection points.
Solved: Making the Ghost movement more randomized by making them change direction at intersection points

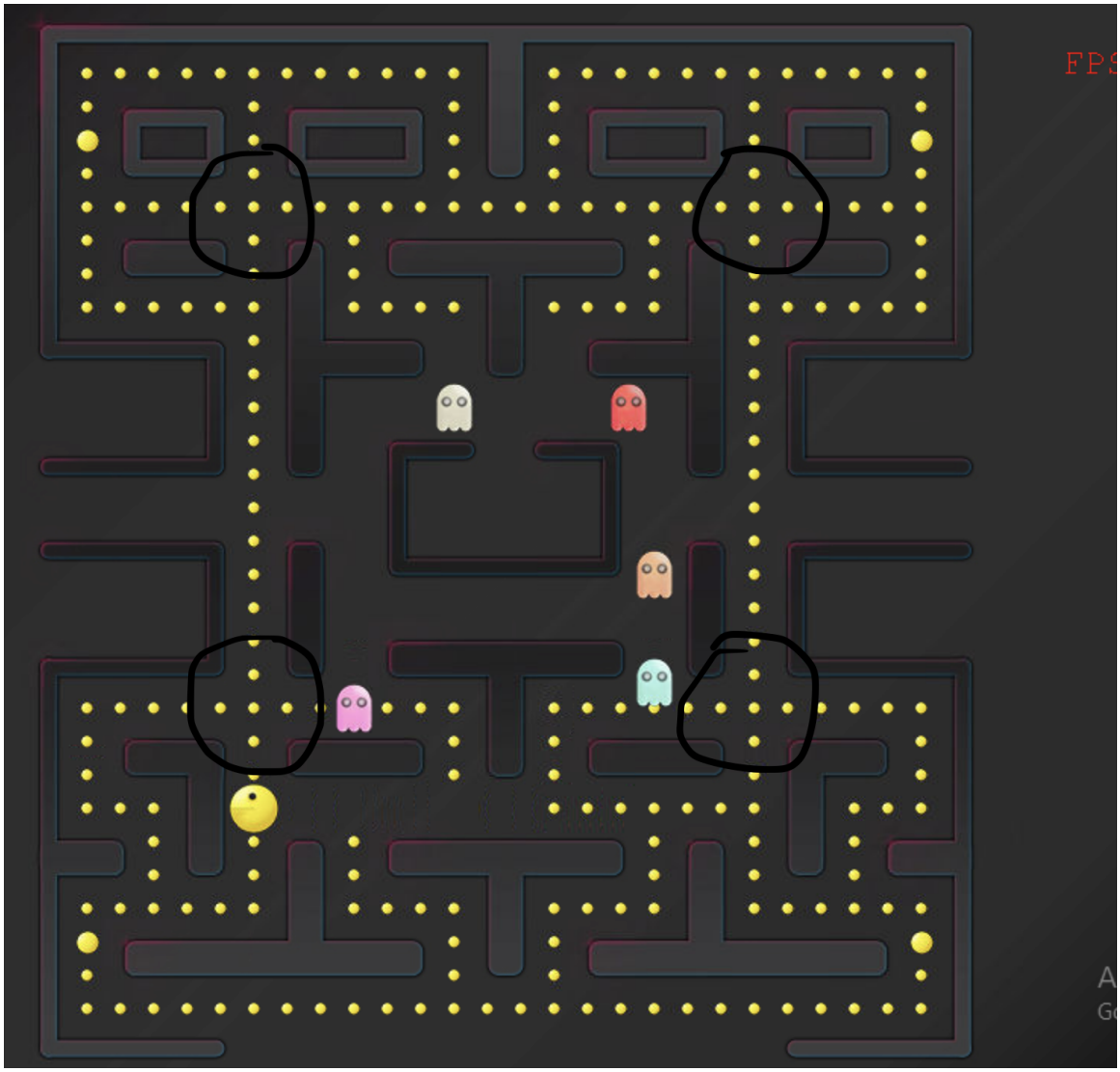


Image 12

Earlier the Ghost character kept their movement the same in the above-mentioned 4 points**.** Firstly I tried to find the tile value of these four intersection points by printing the value of **Pacman's** currentTileX and currentTileY value as I could move the Pacman easily to extract the tile values of these points. Once I found out the tile value of these four intersection points, I created a constant Vector2 array in Ghost.h file and assigned these four values.

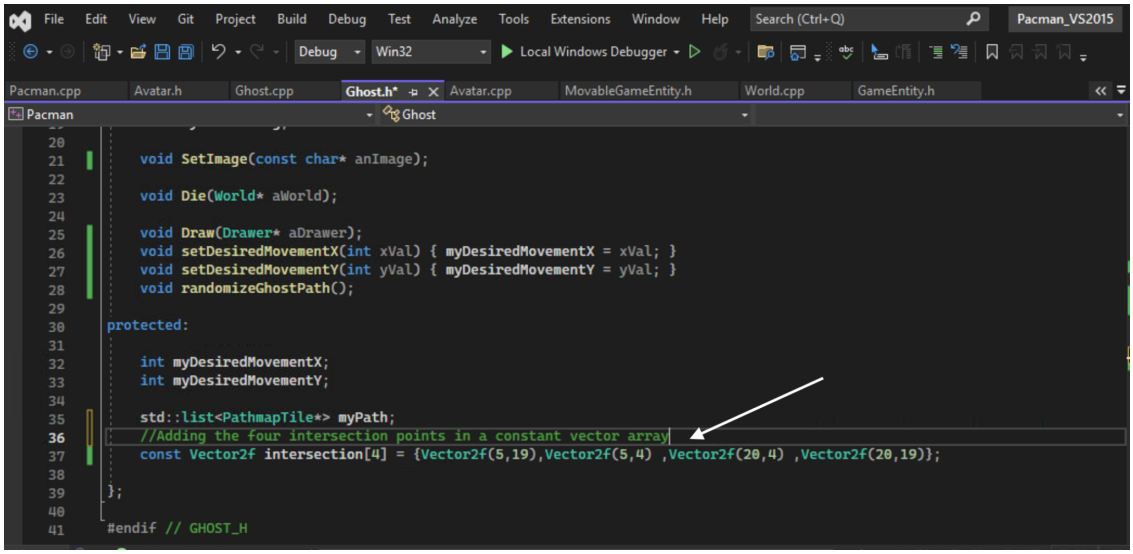


Image 13

Inside the update function of Ghost.cpp I initially checked if currentTile was equal to the values on the array or not but once I ran the code it seemed like putting a check on the currentTile would make the code not work because by the time the code would run the ghost element would have already crossed the intersection, so instead I put a check on myNextTile and the code started working properly.

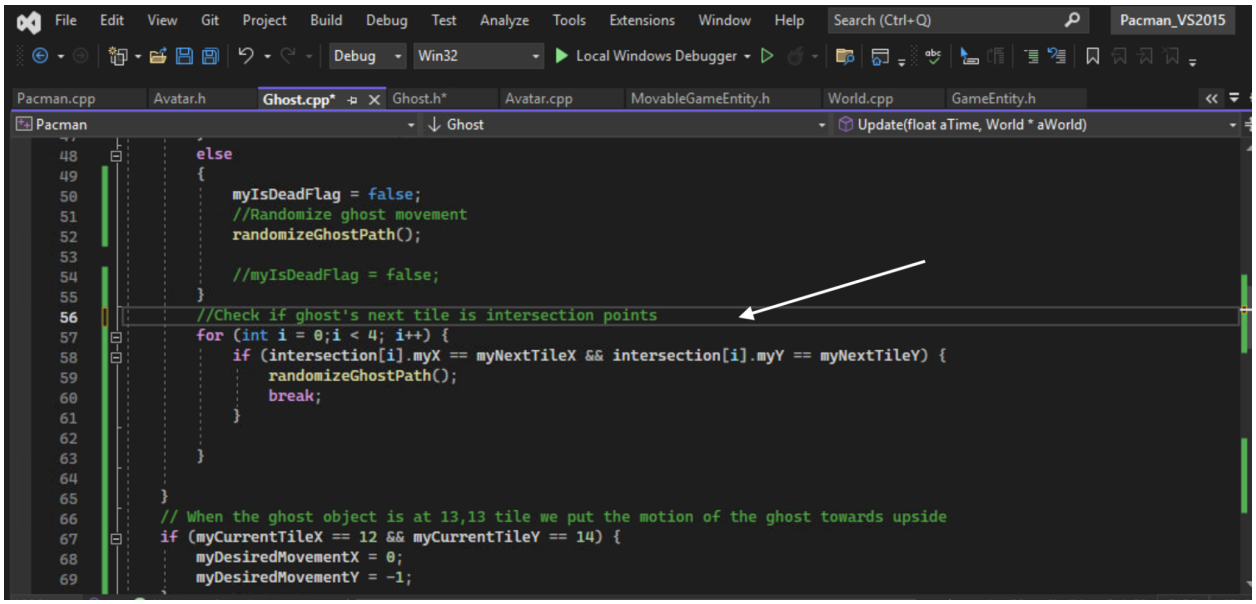


Image 14