AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Project

Project Title:	VIRUS INVADERS		
Course Title:	Computer Graphics	Date of Submission:	18/09/2025
Course Code:	CSC4118		
Section:	L		
Semester:	Summer 2024-25	Course Teacher:	NOBORANJAN DEY

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the faculty for review and comparison, including review by external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of their arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 2

No	Name	ID	Program	Signature
1	**GOLAM IFFAT RAHMAN FARAH**	23-50230-1	BSc [CSE]	
2	**AYON KUMAR BHOWMICK OVI**	23-51050-1	BSc [CSE]	
3	**KAZI WARISA TABASSUM**	23-52623-2	BSc [CSE]	
4			Choose an item.	
5			Choose an item.	
6			Choose an item.	
7			Choose an item.	

Faculty use only		
FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

INTRODUCTION:

We implemented a 2D interactive computer game using OpenGL and GLUT where the player is represented by a cartoon human in a park environment. The player moves along the park's cross-shaped pathways while viruses continuously spawn and move towards them. The player can spray disinfectant to destroy viruses and collect vaccines to restore health. The game has multiple levels, a scoring system, high score tracking, and transitions between Playing, Game Over, High Scores, and Level Complete states. Essentially, this is a COVID-19 themed educational action game that combines graphical design, collision detection, and simple AI-driven enemy behavior.

This project was implemented to practice and demonstrate fundamental computer graphics concepts, game mechanics, and interactive programming. It makes use of important OpenGL techniques such as drawing geometric shapes, handling real-time inputs, detecting collisions, and managing different game states. The theme of COVID-19 was chosen both to make the game relatable and to apply these technical concepts in a meaningful, real-world inspired scenario. In short, the game was built to show how technical programming skills can be applied to create engaging, educational, and visually interactive software.

The significance lies in both technical learning and social awareness. From a technical side, the project provides strong hands-on experience in OpenGL graphics, event-driven programming, and object-oriented design. It helps students understand how to structure a game engine, manage animations, and optimize performance in real-time applications. From a social perspective, the COVID-19 theme highlights the importance of hygiene, vaccination, and safety, using a fun and interactive approach. Games like this can make serious topics more accessible, especially for younger audiences, by turning preventive measures into a gameplay metaphor.

The project is mainly targeted toward students, beginners in computer graphics, and young audiences who are interested in both learning and playing. For computer science or engineering students, this acts as an educational project that improves their coding and visualization skills. For children and casual players, the game works as an awareness tool that teaches basic concepts like virus avoidance and the value of vaccination in a simplified way. Hence, the target population includes learners in computer graphics/game development as well as general players who can benefit from playful learning about health practices.

TOOLS USED:

1. **glBegin() and glEnd():**

   These functions are used to start and end the definition of geometric primitives. For example, in your code, glBegin(GL_TRIANGLE_FAN) draws circles, and glBegin(GL_QUADS) draws rectangular shapes like the player's body or the syringe. Between glBegin() and glEnd(), vertices are defined with glVertex2f().Used extensively for drawing various game elements like paths, player character, virus spikes, and vaccine syringes.

2. **glVertex2f(x, y):**

   This function specifies the coordinates of a vertex in 2D space. It is used inside glBegin()/glEnd() blocks to define shapes like circles, rectangles, and triangles that make up the characters, viruses, and environment.Used to define points for all geometric shapes in the game, including the player character, viruses, and background elements.

3. **glColor3f(r, g, b):**

   This sets the current drawing color using red, green, and blue values (each between 0 and 1). In your project, it is used to color different game elements: green for bushes, pink/yellow for flowers, blue for the player's shirt, red for viruses, etc.Used to set colors for all game elements, including the player, viruses, vaccines, spray, and background.

4. **glClear():**
   Clears buffers to preset values. GL_COLOR_BUFFER_BIT clears the color buffers. Usage in project: Used in the display function to clear the screen before rendering each frame.

5. **glutSwapBuffers():**

Swaps the front and back buffers in double-buffered OpenGL applications, making the newly drawn frame visible.Usage in project: Called at the end of the display function to present the rendered frame. swaps the front and back buffers when using double buffering, ensuring smooth rendering without flicker.

6. **glLoadIdentity ()**

Replaces the current matrix with the identity matrix, effectively resetting any transformations.Called at the beginning of the display function to reset the transformation matrix.

7. **gluOrtho2D()**

Sets up a 2D orthographic projection matrix. It defines a clipping region with left, right, bottom, and top coordinates. Used in the reshape function to set up the coordinate system for 2D rendering. In my code, it ensures that all objects fit within the range (-1, 1) for both axes, making the drawing consistent regardless of window size**.**

8. **glViewport()**

   Sets the viewport, which is the rectangular region of the window where the image is drawn.Called in the reshape function to adjust the drawing area when the window is resized

9. **glPushMatrix() / glPopMatrix()**

   These functions push and pop the current matrix stack, allowing you to save and restore transformation states.Used when drawing symmetrical elements like flowers to apply transformations without affecting subsequent drawings.

10. **glScalef()**

    Multiplies the current matrix by a scaling matrix. It scales objects along the x, y, and z axes.Used to create symmetrical flower patterns by scaling and mirroring the petal drawing.

11. **glutBitmapCharacter()**

    Renders a character using a bitmap font at the current raster position.Used in the drawText() function to render text for the HUD, game over screen, and high scores.

12. **glRasterPos2f()**

    Sets the current raster position for bitmap text rendering.Used to position text before rendering characters with glutBitmapCharacter().

13. **glLineWidth()**

Specifies the width of rasterized lines (though not explicitly called in the code, line drawing is used). Used implicitly when drawing lines for virus spikes and other elements.

14. **GLUT callback functions**

    GLUT provides various callback functions for handling events:

    - glutDisplayFunc(): Sets the display callback function
    - glutReshapeFunc(): Sets the window resize callback function
    - glutSpecialFunc(): Sets the special key callback function
    - glutKeyboardFunc(): Sets the keyboard callback function

- glutTimerFunc(): Sets a timer callback function

Used to set up the game's event handling system.

15. **glutKeyboardFunc(), glutSpecialFunc(), and glutTimerFunc():**
    These are GLUT callback functions that link user inputs and timing to the game logic.

    glutKeyboardFunc() and glutSpecialFunc() handle input from normal keys (like space/ESC) and special keys (like arrow keys).

    glutTimerFunc() repeatedly calls the update function every few milliseconds to keep the game running smoothly.

16. **glRasterPos2f(x, y) and glutBitmapCharacter():**

    These two are used together for rendering text. glRasterPos2f() sets the starting position where text should appear, while glutBitmapCharacter() draws individual characters of a string in a chosen font (e.g., GLUT_BITMAP_HELVETICA_18) for HUD elements like score, health, and instructions.

17. **PlaySound() (Windows-specific)**

    A Windows API function that plays a sound file.Used to play background music in the game.

    Additional Libraries Used

    Standard C++ Libraries

    Various C++ standard libraries were used:

    - <cmath>: For mathematical functions like sin(), cos(), and sqrt()
    - <cstdlib>: For general utilities like rand() and srand()
    - <string>: For string manipulation
    - <sstream> and <iomanip>: For string formatting
    - <ctime>: For time functions to seed the random number generator

    This combination of OpenGL functions, GLUT utilities, and standard C++ libraries created a complete 2D game with graphics, input handling, collision detection, and game state management.

KNOWLEDGE APPLIED FIELD:

This project applies practical knowledge of computer graphics, interactive programming, and game development that is highly valuable in the job market. Skills such as handling real-time rendering with OpenGL, managing user input through event-driven programming (GLUT callbacks), and implementing collision detection and object interaction are directly relevant to careers in game design, simulation software, UI/UX visualization, and multimedia development. Moreover, experience with graphics APIs, coordinate systems, and rendering pipelines demonstrate strong problem-solving and software engineering skills, which are also transferable to roles in software development, embedded systems, and AR/VR applications.

For higher academic pursuits, this project demonstrates a solid foundation in computer graphics, algorithms, and human–computer interaction (HCI). The implementation shows an understanding of graphics primitives, geometric transformations, and real-time systems, which are essential for advanced studies in fields like Computer Graphics, Artificial Intelligence for Games, Computer Vision, or Human–Computer Interaction research. By integrating health-awareness themes such as COVID-19, the project also reflects the ability to combine technical programming with social relevance, a quality that strengthens applications for higher studies where interdisciplinary research and innovation are highly valued.

REFERENCE MATERIALS:

I. GitHub Link: https://github.com/AyonBhowmick/VIRUS_INVADERS

II. Graph:

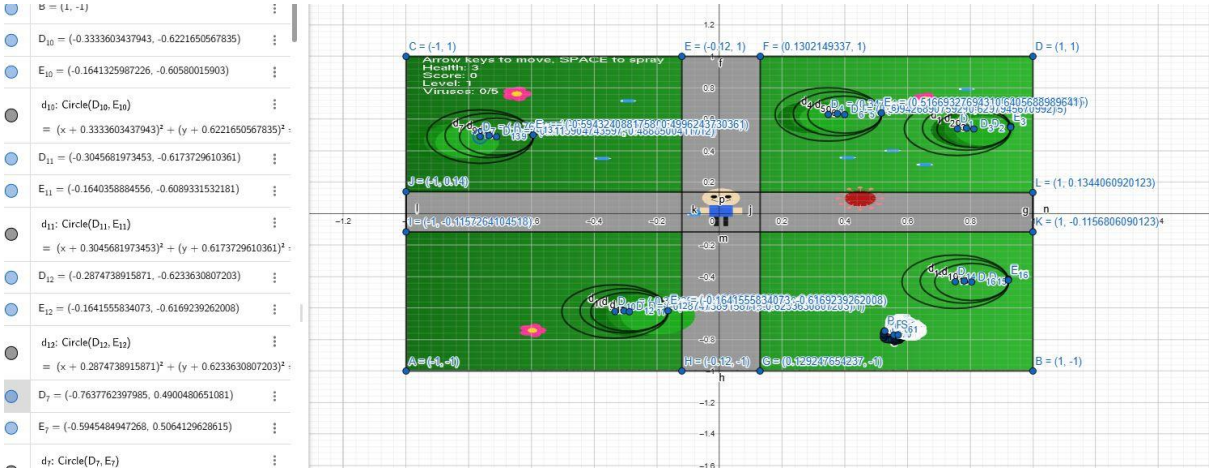

Figure 01: Graphical points for Horizontal, Vertical Road, Flower and Brushes

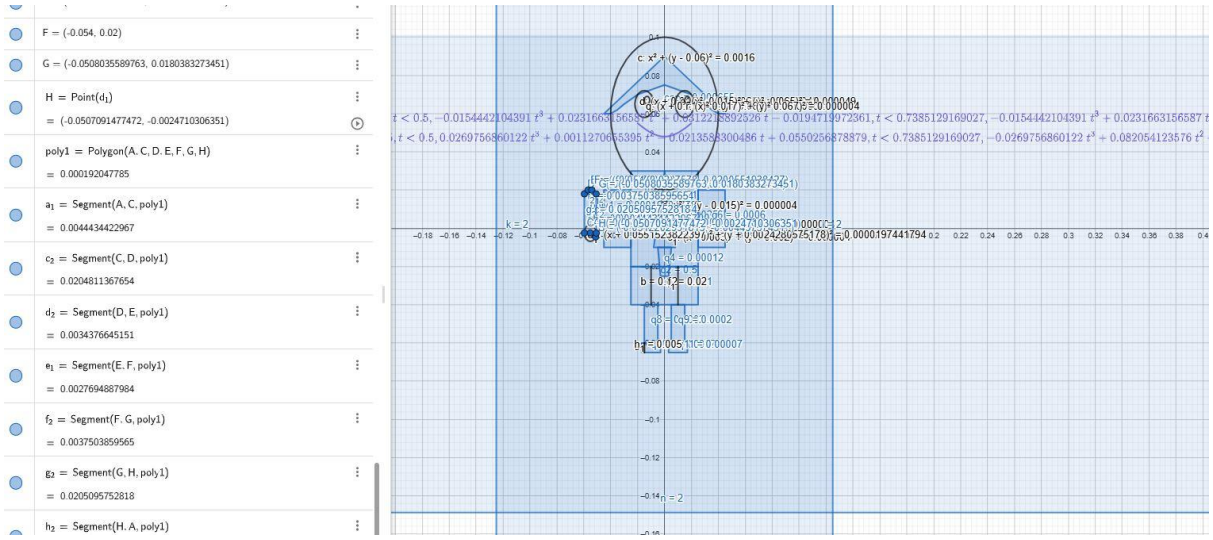

Figure 02: Graphical points for Player drawing

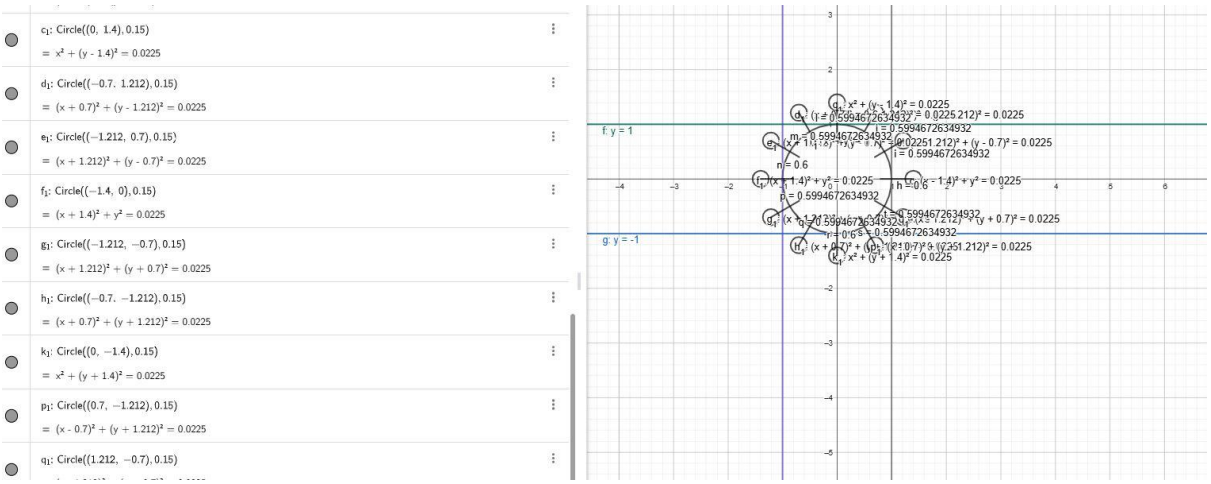

Figure 03: Graphical points for Virus drawing

Screenshots of Project:

Scenario 1:

Figure01: Level 1 ,1 virus is coming and road path width default mode

Scenario 2:

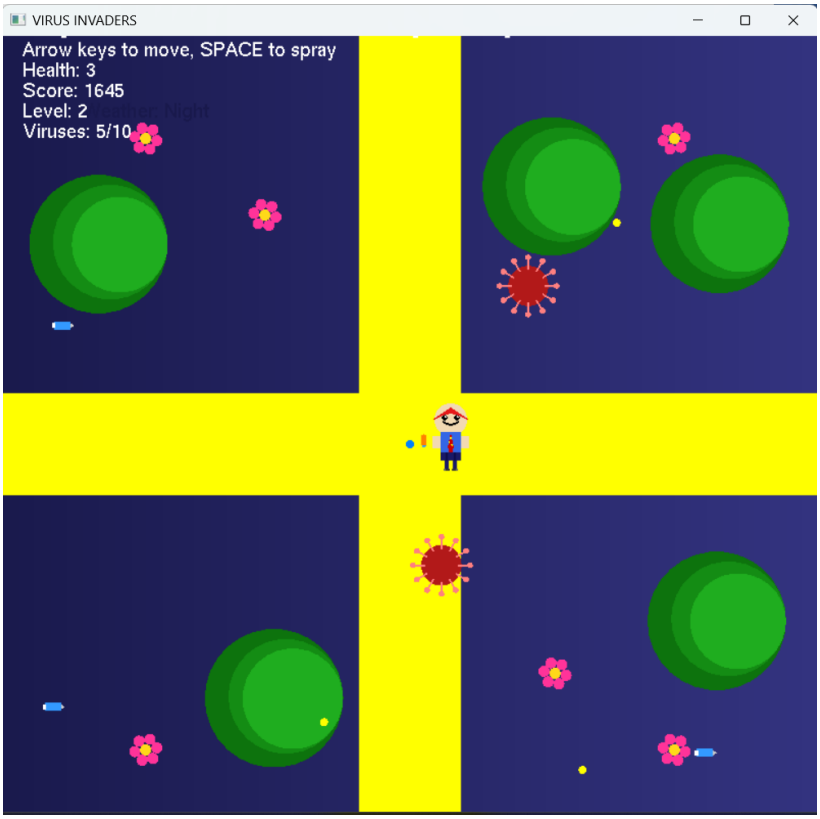

Figure02: Level 2 ,2 virus is coming and road path width level 1

Scenario 3:

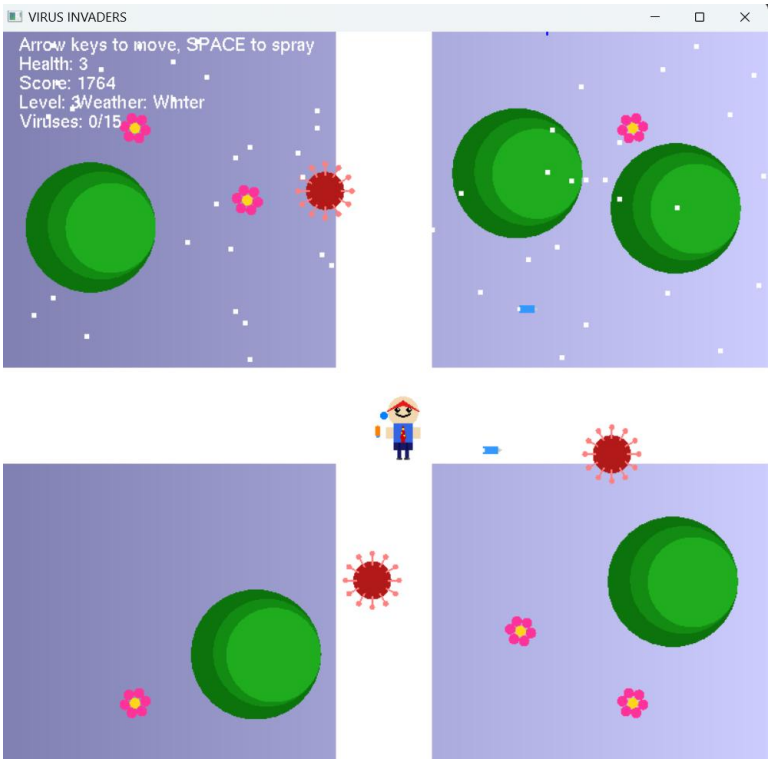

Figure03: Level 3, 3virus is coming and road path width level 2

Scenario 4:

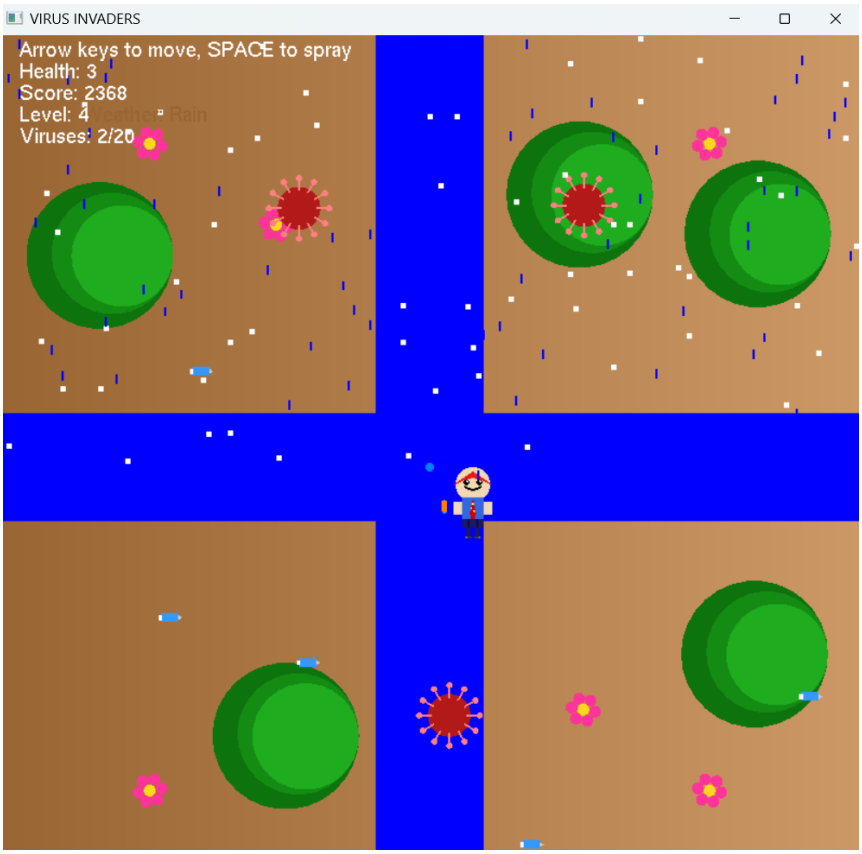

Figure4: Level 4, 4 virus is coming and road path width level 3

CONCLUSION:

Virus Invaders is a successful blend of computer graphics concepts and fun play, realized with the OpenGL platform. This exercise succeeds in demonstrating how mathematical functions—coordinate geometry, vector calculations, and transformation matrices—are realized as a real-time visual program. Player movement, collision detection, and dynamic enemy repositioning in the game all rely on geometric computations, showing the mathematical foundation for game programming. The progressive level design, where the paths to navigate are progressively reduced, not only challenges, but also demonstrates the use of spatial reasoning and boundary control in virtual space.

Aside from its technical success, Virus Invaders provides full user experience with its health system, score, and high-score display, demonstrating significant elements of game architecture and state management. The graphics appearance, generated through the primitives and transformations of OpenGL, is a coherent and pretty appearance without the necessity for added assets. This exercise identifies the potential of programming as an art of

creative expression and interactive storytelling, and it provides a good foundation in ideas of graphics programming that would be pursued to more sophisticated applications or even prospective game development exercises.