

**International University of Business Agriculture and Technology**

**Assignment**

**Project Report**

**Course Code:** CSC 455

**Submitted To:**

Abdullah Mohammad Sakib

Lecturer,

Department of CSE, IUBAT

**Submitted By:**

**Group Name: ShaDow\_MasK**

**Section: I**

**Team members:**

Md. Rashedul Islam (22103222)

Sakib (22103241)

Hasnain Tafsir Chowdhury (22103035)

**Mosquito Behavior Simulation Highlighting Dengue Risks in OpenGL and C++**

**1. Introduction**

OpenGL is cross language programing interface used to create 2D and 3D Graphics. It’s used in computer graphics applications like video games, simulation, CAD, scientific visualization. We can draw or create simple to complex shapes and objects and provide tools where we can apply various transformation, coloring etc. Its supported by many programming languages like C++, Python, Java etc. OpenGL helps us to create real time and dynamic 2D and 3D graphics.

**Interactive Graphics** are visual representations that allow us to interact with the content. We can move or control any element of the content. **Visualization** means using images or animations to represent something to simplify complex topics/data. **Infographics** is the combination of images, charts that is used to present any information or data. Digital Awareness Campaign uses online tools to spread awareness to educate people with the digital platform.

**2. Contextual Issues**

In Bangladesh, there are many village people who is not educated and unaware of some common disease. Due to less knowledge of these disease most people died without any treatment as they haven’t taken the issue seriously. But visual representation or animation can teach them about these diseases. Not only them but also children can learn from the animation.

**3. Details of the Solution**

**Problem and Solution**

Dengue has become a common disease nowadays but if treatment can be done in time, then it can be cured completely. So, the problem is to make them aware about dengue so that no one dies due to lack of knowledge.

The solution is we have to clean where water is preserved for many days, clean the dust regularly and use mosquito net. We have created a visual graphics how a man is getting bite.

**Selection of the Scenario**:

The scenario—a man is cleaning the dust and suddenly a mosquito come and bite the man and he is getting affected by the bite that is causing dengue.

**Design and Technical Explanation**:

Key design elements include:

* **Geometrics**: Polygons, different shapes and circles are used to create objects like mosquito, hospital, buildings, helicopter, car and man.
* **Functions**: Modular functions ensure reusability and maintainable code.

The program's technical structure reflects careful planning:

* **Functions**: Simplified rendering logic through functions that is controlling the elements.
  + initGL(): Initializes OpenGL settings, background color, and mosquito positions.
  + drawRectangle(): Utility function to draw a colored rectangle.
  + drawSchool(): Renders a school building with windows and a flag.
  + drawHospital(): Draws a hospital building with windows and a red cross symbol.
  + drawDustHill(): Creates a complex dust hill shape using a polygon.
  + drawSun(): Renders a yellow circular sun in the sky.
  + drawCloud(): Draws a cloud shape using overlapping circles.
  + drawRoad(): Creates a horizontal road with yellow borders and a dashed centerline.
  + drawVerticalRoad(): Renders a vertical road segment.
  + drawMosquito(): Draws a regular mosquito with body, wings, legs, and antennae.
  + drawRedMosquito(): Renders a red mosquito, similar to the regular one but with a red body.
  + handleKeyboardInput(): Manages arrow key inputs to move the user-controlled mosquito.
  + checkCollisionWithCleaner(): Detects collision between the controlled mosquito and the cleaner's head.
  + drawControlledMosquito(): Renders the user-controlled red mosquito.
  + drawCleaner(): Draws the cleaner character with body, arms, and an animated broom.
  + drawCircle(): Utility function to draw a filled circle.
  + drawHelicopter(): Renders a detailed helicopter with body, rotors, and windows.
  + drawCircleOutline(): Utility function to draw a circle outline (unused in the provided code).
  + drawCar(): Creates an ambulance car with details like lights and a red cross.
  + animateMosquitoes(): Updates positions of mosquitoes and handles boundary collisions.
  + animateBroom(): Animates the cleaner's broom swinging motion.
  + animateHelicopter(): Moves the helicopter across the screen.
  + display(): Main display function that draws all elements of the scene.
  + timer(): Manages the animation timing, calling for screen updates.
  + main(): Sets up GLUT, creates the window, and starts the main event loop.
  + glColor3f(): Sets the current drawing color using RGB values.
  + glColor4f(): Sets the current drawing color using RGBA values (including alpha for transparency).
  + glTranslatef(): Applies a translation transformation to the current matrix.
  + glRotatef(): Applies a rotation transformation to the current matrix.
  + glScalef(): Applies a scaling transformation to the current matrix.
  + glPushMatrix(): Pushes the current matrix stack down by one, duplicating the current matrix.
  + glPopMatrix(): Pops the current matrix stack, replacing the current matrix with the one below it on the stack.
* **Animation**: Controlled using loops and delays, simulating movement.
* **Interactivity**: User input (keys) determines navigation and animations.
* **Menus**: Hierarchical menu design for easy scene selection.

**Comparison**

Unlike static visualizations, this solution offers interactivity, engaging users with animations. This sets it apart from conventional slides or videos used in similar contexts.

**4. Implementation in Bangladeshi Digital Awareness Campaigns**

The interactive nature of this simulation makes it a strong candidate for inclusion in digital awareness campaigns. For instance:

* **Educational Tools**: Simulations can train users on how to use computer labs effectively.
* **Language Localization**: We can implement some awareness message in Bangla.
* **Lightweight and Versatile**: As an OpenGL-based tool, it can run on minimal hardware, a crucial factor in rural and underdeveloped areas.

By adapting this framework, campaigns can provide hands-on learning experiences, fostering greater engagement and understanding.

**5. Source Code**

Some functions from the source code for the project has been included below for reference. It demonstrates the modular approach and logical flow used to create the interactive graphics:

**Initialize OpenGL settings**

void initGL()

{

glClearColor(0.5f, 0.8f, 1.0f, 1.0f); // Background color: sky blue

glEnable(GL\_BLEND);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

srand(static\_cast<unsigned>(time(0))); //Seed for randomization

// Initialize mosquito positions and speeds

for (int i = 0; i < numMosquitoes; i++)

{

mosquitoX[i] = 0.6f + (rand() % 100) / 100.0f \* 0.4f - 0.2f; // Around dust hill

mosquitoY[i] = -0.2f + (rand() % 100) / 100.0f \* 0.4f - 0.2f;

mosquitoSpeedX[i] = ((rand() % 100) / 100.0f \* 0.02f - 0.01f);

mosquitoSpeedY[i] = ((rand() % 100) / 100.0f \* 0.02f - 0.01f);

}

for (int i = 0; i < numMosquitoes; i++)

{

mosquitoRedX[i] = 0.6f + (rand() % 100) / 100.0f \* 0.4f - 0.2f; // Around dust hill

mosquitoRedY[i] = -0.2f + (rand() % 100) / 100.0f \* 0.4f - 0.2f;

}

}

**Draw a rectangle**

void drawRectangle(float x, float y, float width, float height, float r, float g, float b)

{

glColor3f(r, g, b);

glBegin(GL\_QUADS);

glVertex2f(x, y);

glVertex2f(x + width, y);

glVertex2f(x + width, y + height);

glVertex2f(x, y + height);

glEnd();

}

// Draw a complex dust hill

void drawDustHill()

{

glColor3f(0.6f, 0.4f, 0.2f); // Brown color for dust

glBegin(GL\_POLYGON);

glVertex2f(0.4f, -0.5f);

glVertex2f(0.5f, -0.1f);

glVertex2f(0.6f, -0.2f);

glVertex2f(0.7f, 0.1f);

glVertex2f(0.8f, -0.2f);

glVertex2f(0.9f, -0.1f);

glVertex2f(1.0f, -0.5f);

glEnd();

}

// Draw the sun

void drawSun()

{

glColor3f(1.0f, 1.0f, 0.0f); // Yellow color for sun

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float angle = i \* M\_PI / 180.0f;

glVertex2f(0.8f + 0.1f \* cos(angle), 0.8f + 0.1f \* sin(angle));

}

glEnd();

}

// Draw clouds

void drawCloud(float x, float y)

{

glColor4f(1.0f, 1.0f, 1.0f, 0.8f); // White, semi-transparent clouds

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float angle = i \* M\_PI / 180.0f;

glVertex2f(x + 0.05f \* cos(angle), y + 0.05f \* sin(angle));

}

glEnd();

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float angle = i \* M\_PI / 180.0f;

glVertex2f(x + 0.1f + 0.05f \* cos(angle), y + 0.05f \* sin(angle));

}

glEnd();

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

float angle = i \* M\_PI / 180.0f;

glVertex2f(x + 0.025f + 0.1f \* cos(angle), y + 0.1f + 0.05f \* sin(angle));

}

glEnd();

}

**Handle keyboard input for mosquito movement**

void handleKeyboardInput(int key, int x, int y)

{

float step = 0.05f; // Movement step size

switch (key)

{

case GLUT\_KEY\_LEFT:

controlledMosquitoX -= step;

break;

case GLUT\_KEY\_RIGHT:

controlledMosquitoX += step;

break;

case GLUT\_KEY\_UP:

controlledMosquitoY += step;

break;

case GLUT\_KEY\_DOWN:

controlledMosquitoY -= step;

break;

}

glutPostRedisplay(); // Redraw the screen after movement

}

**Collision detection between mosquito and cleaner's head**

bool checkCollisionWithCleaner()

{

float dx = controlledMosquitoX - cleanerX;

float dy = controlledMosquitoY - (-0.2f); // Cleaner head Y-position

float distance = sqrt(dx \* dx + dy \* dy);

return distance <= cleanerHeadRadius; // Collision if within the radius

}

**Animate the mosquitoes**

void animateMosquitoes()

{

for (int i = 0; i < numMosquitoes; i++)

{

mosquitoX[i] += mosquitoSpeedX[i];

mosquitoY[i] += mosquitoSpeedY[i];

// Reflect off boundaries

if (mosquitoX[i] > 1.0f || mosquitoX[i] < -1.0f) mosquitoSpeedX[i] = -mosquitoSpeedX[i];

if (mosquitoY[i] > 0.1f || mosquitoY[i] <= -1.0f) mosquitoSpeedY[i] = -mosquitoSpeedY[i];

}

}

**Timer function for animation**

void timer(int value)

{

glutPostRedisplay();

glutTimerFunc(16, timer, 0); // Call the timer every 16 milliseconds (60 FPS)

}

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Dust Hill Scene");

initGL();

glutDisplayFunc(display);

glutSpecialFunc(handleKeyboardInput);

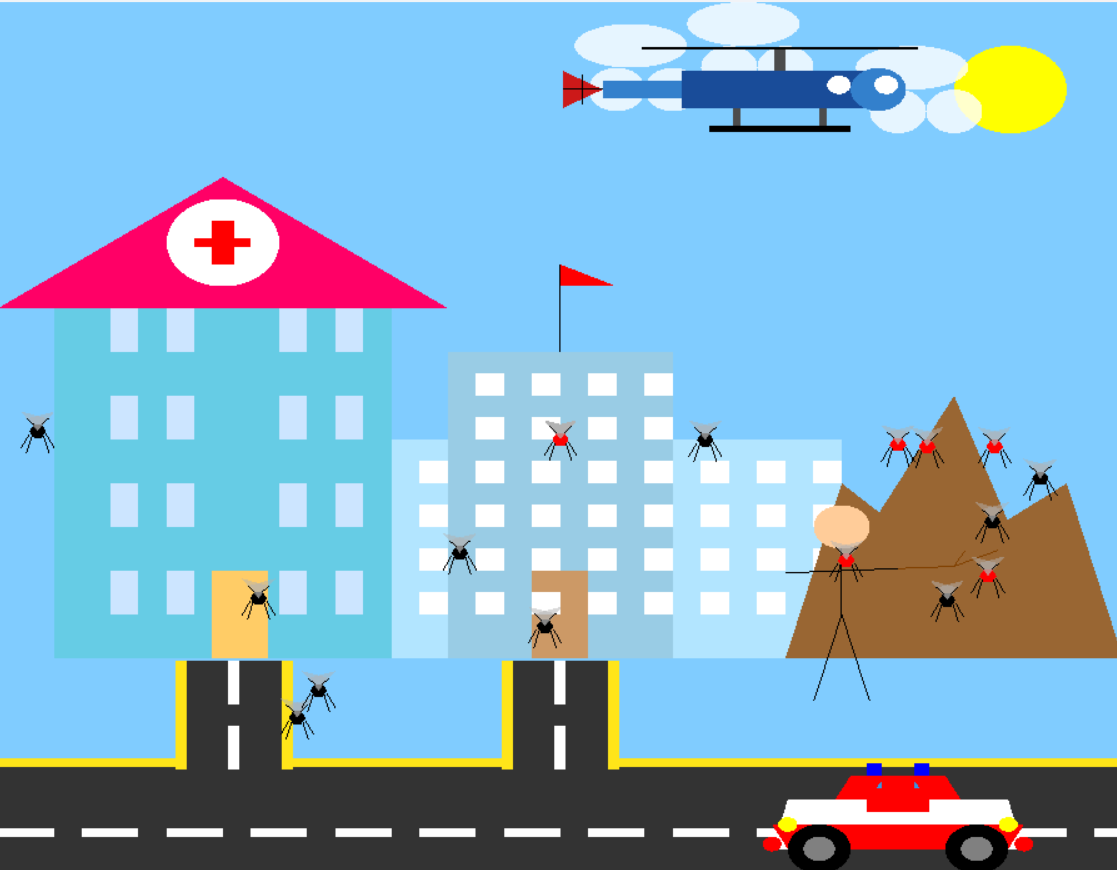
glutTimerFunc(25, timer, 0);

glutMainLoop();

return 0;

}

**6. Output and Explanation**



Here Mosquito will keep moving and we can control one mosquito by pressing 4 arrows (up, down, top, left). When mosquito touch the head it marked as red and that means he got bite by the mosquito. But he was cleaning the dust that means he is little bit aware of the dengue.

**7. Conclusion**

This project shows the effective use of OpenGL and C++ for creating interactive dengue awareness and simulation tool. By addressing real-world challenges, it demonstrates the potential for enhancing digital awareness in contexts like Bangladesh. We have to spread this awareness through the digital medias like social media and make people aware about dengue.