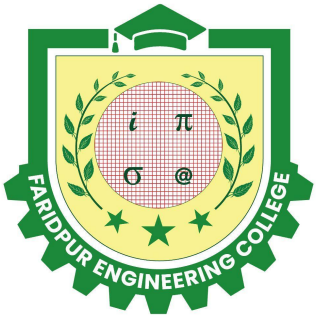
Faridpur Engineering College, Faridpur



Department of Computer Science & Engineering
Course Name: Computer Graphics (Sessional)
Course Code: CSE 802

A project report on:

Moving Person using OpenGL

Submitted to
Md. Rony Ahmed
Lecturer
Dept. of Computer Science & Engineering
Faridpur Engineering College, Faridpur.

Submitted by
Md. Rakibul Islam Nahid
ID: 2537 || Reg: 1797
4th Year 2nd Semester

May 2023

Abstract: Here is a C++ script that utilizes the graphics.h library to animate a guy walking. The while loop constantly draws and updates the screen to reflect the man's current location. The guy is sketched out using a variety of circles, rectangles, and lines, and his location is dynamically modified according to the direction flag variable. When the left or right arrow keys are pressed, the direction flag is changed accordingly. The programme is equipped with code that will cause the man's location to wrap around to the other side of the screen if he were to reach the screen's edge. The animation loops endlessly until the user manually stops it.

1. Introduction

1.1 Computer Graphics: The term "computer graphics" refers to the study and practise of utilizing computers to generate, manipulate, and display visual content.The field of computer graphics has grown significantly over the last several decades, and it is now used extensively in many fields such as the arts, marketing, teaching, medical, and construction.Computer graphics is a broad discipline that includes several specialized areas, such as computer vision, CAD, VR, and AR. Rasterization, ray tracing, three-dimensional modeling, and animation are all tools in the computer graphics toolbox.

Software for creating computer graphics ranges from simple sketching programmes to complex 3D modeling and animation suites. Adobe Photoshop, Autodesk Maya, Blender, and Unity are just some of the well-known programmes in this category.

1.2 OpenGL: As a popular graphics API (Application Programming Interface), OpenGL (Open Graphics Library) allows programmers to create dynamic 2D and 3D graphics programmes. It's an open-source library that works on several platforms.OpenGL is great since it can be used on any platform. It's compatible with Windows, macOS, Linux, and even mobile devices. Because of this, many programmers choose it when they need to construct cross-platform, graphically heavy apps.OpenGL is a library of operations for creating and manipulating 3D scenes, including the addition of objects, textures, lighting, and transformations (including rotations and translations). It also offers a means through which these things may be shown on a flat screen.OpenGL is also the foundation for numerous other tools and libraries, including the widely used Unity graphics engine.In sum, OpenGL is a powerful and flexible platform for developing 2D and 3D graphics software. Developers that wish to construct high-quality, cross-platform graphical apps often choose it because of its flexibility and sophistication.

2. Contributions (Integration)

This is a simple animation programme in which the user controls the on-screen man's leftward and rightward movement using the left and right arrow keys. The graphics.h library is used to create graphical elements like lines and shapes on the screen, whereas the conio.h library is used to take input from the keyboard.

The initgraph() function, which accepts the graphics driver and graphics mode as input, is used to initialize the graphics system at the beginning of the programme. The programme then determines the man's starting point and uses a flag variable to keep tabs on which way he's heading.

In the while loop, the man's location on the screen is constantly revised in response to changes in the direction flag variable. The kbhit() method is also used to detect user input and change the direction flag in response to the arrow keys. Last but not least, if the guy leaves the screen, his location will be wrapped around to the other side.

The programme combines operations like circle(), rectangle(), and line() to create circles, rectangles, and lines on the screen. For coloring and formatting the shapes and lines, it employs the setcolor() and setlinestyle() built-in methods. By inserting a pause in between each frame using the delay() method, the animation's pace may be adjusted.

In order to relinquish control to the operating system when a programme has completed, the closegraph() function is used.

3. Requirement Specification

3.1 Software Requirements Specification

The software requirements for this project are as follows:

1. C compiler: Since C is the programming language used to create this project, a C compiler is needed to run the code.
2. Graphics library: The graphics.h library is used in this project to create graphical user interface elements. The majority of C compilers already have this library pre-installed.
3. Keyboard input library: The conio.h library is used to process keyboard input in this project. C compilers usually come along with this library.
4. Operating system: Any OS with the aforementioned libraries and a C compiler may execute this project.
5. Integrated Development Environment (IDE): Code::Blocks and Dev-C++ are two IDEs that may be used to modify and generate the executable.

3.2 Hardware Requirements:

The hardware requirements for this project are as follows:

1. Computer: Any machine with an OS and C compiler capable of running the required minimum requirements may run this project.
2. Monitor: The graphics output must be seen on a monitor.
3. Keyboard: In order to use the programme and direct the animation, a keyboard is necessary.
4. Mouse: A mouse is unnecessary but may be beneficial for navigating the IDE or making programme changes.

4. System Design

4.1 Program Layout: C++ is used for the project's code, and the graphics.h and conio.h header files are included. Using the left and right arrow keys, the user may steer the on-screen man's animated stroll in one of two different directions. The basic structure of the app is as follows:

- The initgraph() method is used after the DETECT constant is used to define the graphics driver and mode.
- The programme then determines the character's starting point (in x and y) and the man's first movement direction (1 for right, -1 for left).
- The code goes into an infinite while loop until the user presses a button.
- The graphics window is reset with cleardevice() within the loop, and then the man is drawn using tools like circle(), rectangle(), and line(). When the direction flag changes, the man's location is updated by either adding or removing ten.
- When a user presses a key, the programme detects it using the kbhit() function and retrieves the key with the getch() function. Each key push alters the direction flag accordingly.
- Finally, the code uses getmaxx() to see whether the guy has left the visible area of the screen, and if so, it wraps him around to the other side.
- When a key is pushed, the loop runs until closegraph() is called to close the graphics window and the programme ends.

4.2 Program Graphics: The graphics.h library is used to draw a basic animation of a guy strolling across the screen.

The code sets the variables x and y to the starting positions of the guy and the graphics mode. To indicate whether the individual is looking to his left or right, we use the variable direction as a simple flag.

The while loop keeps the animation going until the user presses a button. The cleardevice() method is used within the loop to discard the current frame and draw a new one. Several methods, including circle(), rectangle(), line(), and setlinestyle(), are used to create the man's head, torso, arms, and legs.

To create the illusion of movement between individual frames, animators utilize the delay() method. Within the loop, the x variable is changed to send the man in the direction indicated by the direction flag.

To determine if a key has been struck, use the kbhit() method. When a key is pushed, the getch() method retrieves the key, and the man's location is adjusted accordingly. If the man's x coordinates are beyond the visible window, the programme utilizes the getmaxx() method to wrap him around to the other side of the display.

When you're ready to end the programme, execute closegraph() to get rid of the graphics window and return 0.

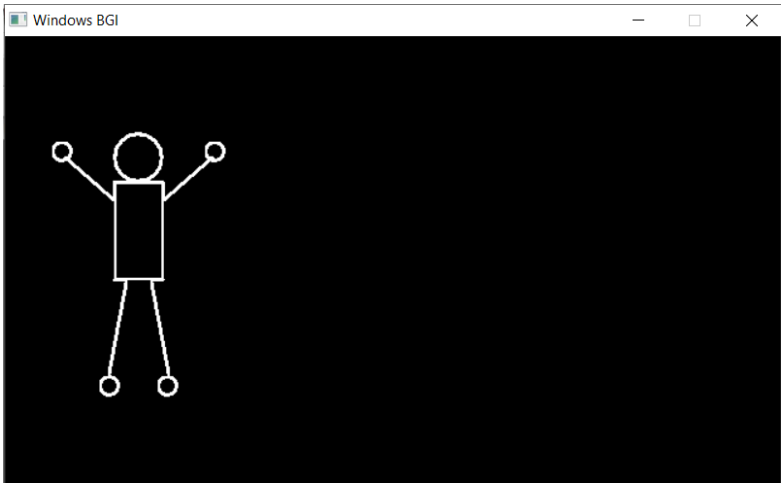
5. Implementation (Graphics Functions)

The project's code is built in C++ and makes use of the graphics.h library to animate a stick figure moving in a forward and backward fashion. A rundown of the code's graphical functionality follows:

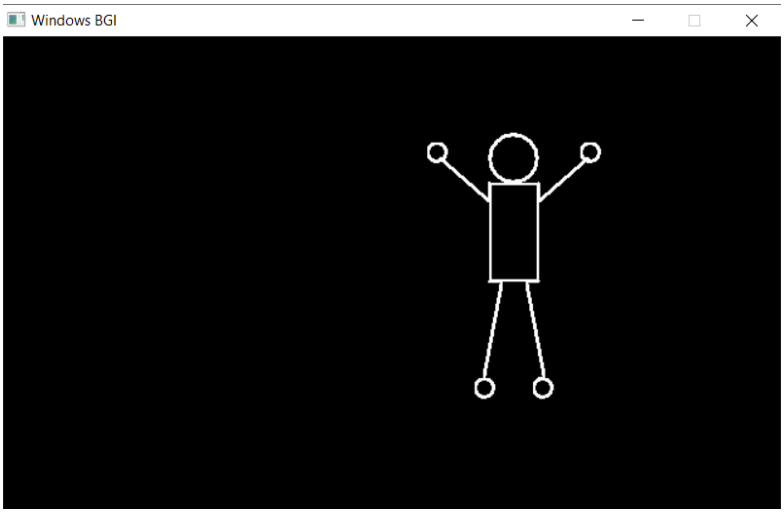
1. initgraph(&gd, &gm, ""): Sets the device name to an empty string and initialises the graphics system using the given graphics driver (gd) and graphics mode (gm).
2. cleardevice(): Clears the screen.
3. setcolor(color): Sets the drawing color to the specified color.
4. circle(x, y, radius): Draws a circle with center at (x, y) and the specified radius.
5. rectangle(left, top, right, bottom): Creates a rectangle with the given left and top corners and the given right and bottom coordinates.
6. setlinestyle(style, pattern, thickness): Controls the line's appearance and thickness in future drawing.
7. line(x1, y1, x2, y2): Draws a line from (x1, y1) to (x2, y2).
8. delay(milliseconds): Pauses the execution for the specified number of milliseconds.
9. kbhit(): Checks if a key has been pressed on the keyboard.
10. getch(): Waits for the user to hit a key on the keyboard before returning the corresponding ASCII value.
11. getmaxx(): Returns the maximum x-coordinate of the screen.
12. closegraph(): Closes the graphics system and frees the allocated memory.

Within the main() method, the graphics system is initialised, the stick figure's initial location and direction are established, and a loop is entered to constantly update the stick figure's position and direction depending on keyboard input and wrap it around the screen boundaries. The loop starts with a blank screen, draws the stick figure, and then waits 100 ms before starting again. The cycle will keep going until a key is hit on the keyboard.

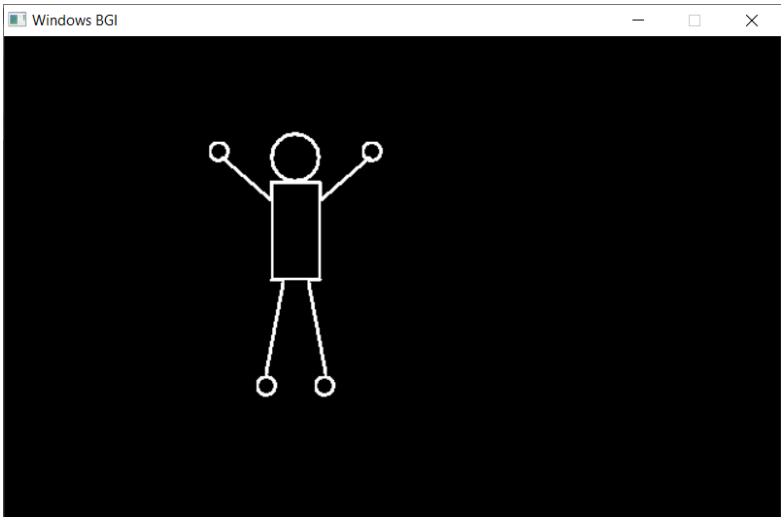
6. Features (Snapshots):



Moving Man in the left to right side.



When the Left arrow is pressed it starts to move on the right to left side.



7. Testing

7.1 Life Cycle of Testing: The life cycle of testing typically consists of several phases, each with its own objectives and activities. Here is a generalized overview of the testing life cycle:

1. Test Planning: In this phase, the testing objectives, scope, and strategies are defined. Test plans and test cases are created, and resources and timelines are allocated.
2. Test Design: Test scenarios and test cases are developed based on requirements and design specifications. Test data and expected results are identified.

3. Test Execution: Test cases are executed, and actual results are compared against expected results. Defects or issues are reported, tracked, and managed. Test environments are set up and test data is prepared.
4. Test Reporting: Test results, metrics, and defects are documented and communicated. Test reports, including test summaries and defect reports, are generated to provide stakeholders with an overview of the testing progress and outcomes.
5. Test Closure: The final phase involves evaluating the completion criteria, reviewing the test process, and preparing the test closure reports. Lessons learned and recommendations for future testing are documented.

7.2 Types of Testing: There are various types of testing that can be performed to validate different aspects of software. Here are some common types of testing:

- The purpose of functional testing is to ensure that the developed programme performs as expected. It verifies that the programme works as expected and that its planned functions are being carried out.
- The goal of unit testing is to ensure that each component of the product works as expected when tested independently. Finding and fixing problems in individual modules of code is much easier using this method.
- The purpose of integration testing is to ensure that all of the interfaces and connections between the various software parts are working properly. It guarantees the combined parts perform as intended and without hiccups.
- The purpose of system testing is to ensure that all requirements have been met by evaluating the system as a whole. It puts the programme through its paces in a variety of conditions to see how it reacts.
- When evaluating software's responsiveness, scalability, and stability under different workloads and situations, we refer to this as "performance testing." Throughput, resource consumption, and other velocities are all quantified.
- Testing the programme from the viewpoint of the end-user to make sure it lives up to their needs and expectations is what's known as User Acceptance Testing (UAT). Users or other stakeholders often do this test before approving the product.
- To make sure that new or altered features do not break or otherwise adversely affect previously implemented ones, regression testing is performed.

Any combination or adaptation of these approaches is possible, depending on the nature of the programme and the nature of the project. To guarantee quality and dependability, testing must be performed repeatedly and continuously throughout the software development life cycle.

8. Conclusion

In conclusion, the code supplied displays a basic walking man animation using the BGI graphics package in C++. A man's head, torso, arms, and legs make up the animation, and he walks horizontally across the screen in response to commands.

The code uses the graphics.h and conio.h libraries' routines to accomplish things like initialise the graphics window, render the figure in various poses, and recognise key inputs. As long as the key is not touched, the walking guy will continue to play in a loop.

The code does what it's supposed to, but there's room for expansion and improvement. Input validation and error handling, code modularization for enhanced organisation and reusability, user interface enhancement with instructions or extra buttons, and performance optimisation are all examples.

Further improving the animation and making it more entertaining for users would be to include cutting-edge elements like graphical upgrades, object-oriented design, sound effects, and cross-platform compatibility.

In conclusion, the given code provides the basis for a basic animation that may be enhanced by the addition of other contributions and integrations. These updates would improve the animation by making it more engaging on several fronts and accessible on more devices.

9. Future Enhancements

Future enhancements for this code could include:

1. Character Animation: Improve the character animation by making it more complex and engaging. This may be done by animating facial expressions, adding frames to make walking more fluid, or introducing new actions like leaping or crouching.
2. Background and Environment: Create a more immersive setting for the animation by adding a picture or scene to the backdrop. You may also give the player control over the character's environment by placing barriers, platforms, and other interactive objects.
3. Level Design: Create a series of levels or stages with progressively more challenging or varied difficulties. To keep the animation fresh and exciting, each level might have different visuals, challenges, or goals.
4. Sound and Music: Use music and sound effects that go along with the animation and the activities the characters are doing. This improves the quality of the user's audiovisual immersion.

The animation has the potential to develop into something more engaging, informative, and fun for viewers by taking advantage of these upcoming improvements. These enhancements broaden the animation's scope and appeal, opening up new avenues for participation, replayability, and amusement.