UTM

UNIVERSITI TEKNOLOGI MALAYSIA

Faculty of Computing

SECV2213 - 2: Fundamental Of Computer Graphic

Group Assignment 3

Date: 16/6/2023

Steve from Minecraft

Prepared by: Group 4

NO.	NAME	MATRIC NUMBER
1	AWFAN ALFA VINANDRY	A21EC4008
2	ANATASYA HUMAIRA	A20EC0261
3	AL FARABI BAITAR MOHANI	A21EC0248

Table Of Contents

Introduction

This code is an implementation of a 3D model of a character named Steve from the game Minecraft using OpenGL. The code sets up a graphical environment where the character can be rendered and interacted with. It provides an interactive environment where the user can control the movement of Steve, the iconic character from Minecraft, using various key functions. The model consists of different body parts, including the head, body, arms, and legs, which are meticulously drawn using OpenGL primitives.

The code begins by including necessary libraries and declaring various variables to control the character's movements and animations. It initializes the OpenGL environment and sets the initial object position.

Figure 1. Steve and Alex from Minecraft.

2D Model of Steve from Minecraft

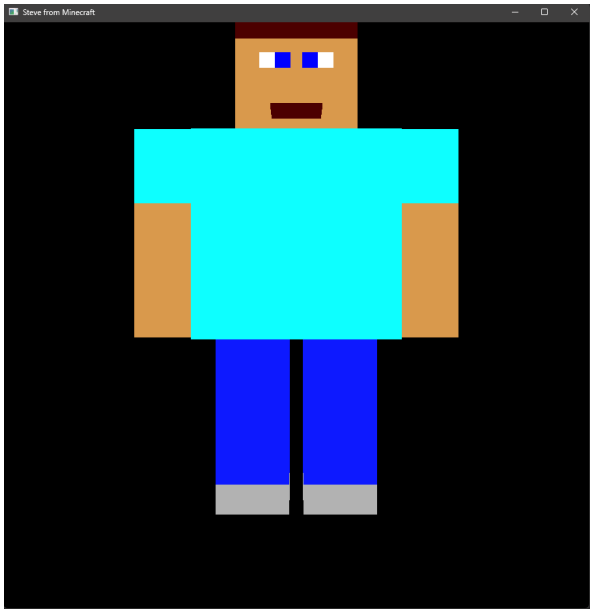


Figure 2: Model of Steve First Output

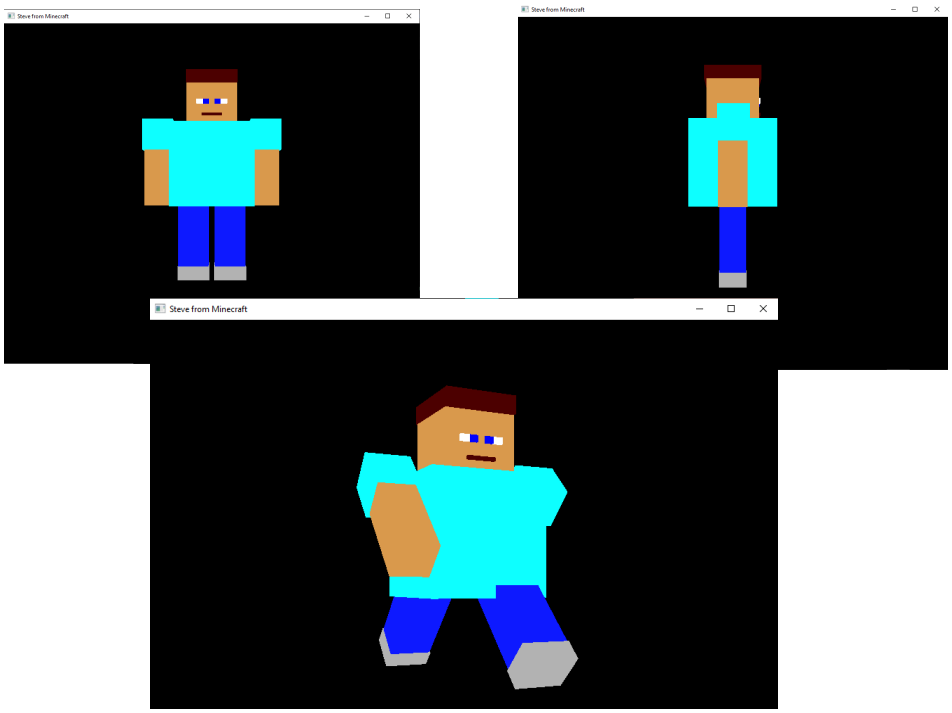


Figure 3, 4, 5: Model of Steve FinalOutput

Diagram of the Hierarchy

Within the suggested hierarchical framework, a primary node occupies the highest position, serving as the foundation of the hierarchy. This parent node assumes a pivotal role, acting as the central point for the entire model and facilitating the transformation of all subordinate nodes. Each child node possesses the capacity to harbor its own distinct set of child nodes, thereby constructing a structure akin to a tree. Figure 1 illustrates the hierarchy diagram and outlines the intended 3D transformation.

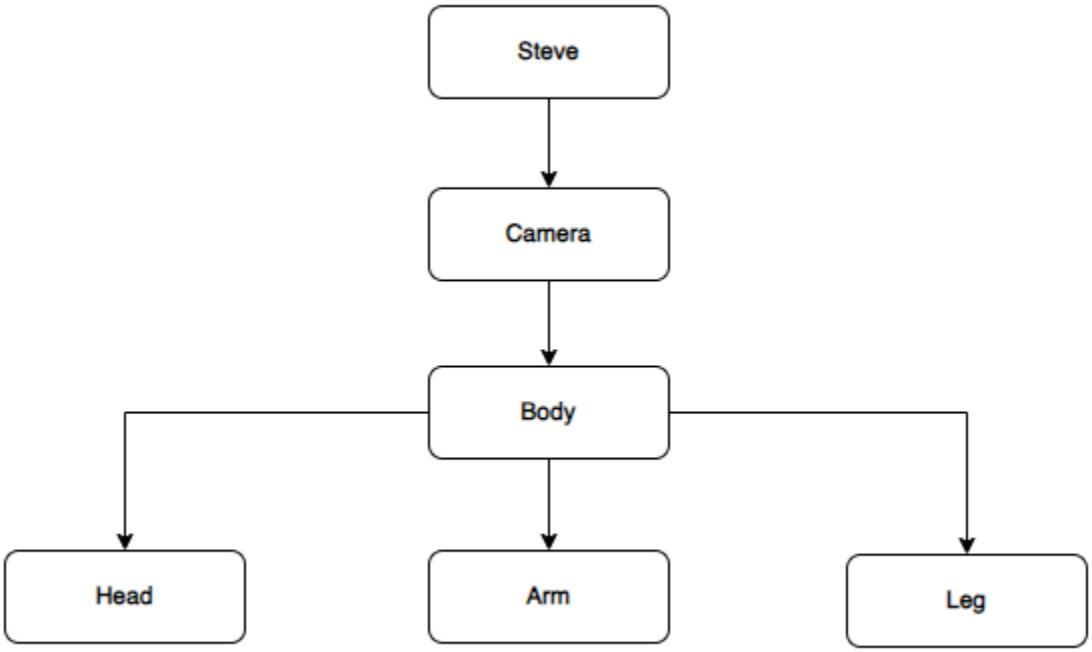


Figure 1: Diagram of the hierarchy and planned 3D transformation

Functions

This code is an implementation of a program that renders a 3D character model of Steve from the game Minecraft using OpenGL. Here's a summary of the functions within the code:

1. **init(void)**: Initializes the OpenGL settings and clears the color buffer.
2. **Camera():** Consists several variables to give the appearance of moving the camera on z-axis which implements zoom functionality to the objects.
3. **projection_setup():** To adjust the coordinates of drawn objects based on the width and height of the GLSurfaceView where they are displayed. This function requires filed_of_view variable input which is declared inside the camera function stated above.
4. **model_view():** Includes both modeling and viewing transformations that position the viewer at the origin, with the view direction aligned with the negative Z axis, which in this code is initialized as -10.
5. **mouse_click(int button, int state, int x, int y):** A function that is called once after a mouse button has been pressed and then released. Each press and each release generates a mouse callback to the function where it generates zooming effects.
6. **mouse_drag(int x, int y):** The mouse coordinates (x,y) are converted to a Points struct while being turned into window coordinates within this function.
7. **drawBackground():** This function sets the texture coordinates as well as enables the sky texture for the background. It consists of different objects which include the sky, grass, trees, and leaves.
8. **display()**: Handles the rendering of the character model by drawing various body parts using glutSolidCube to create cubes of different sizes.
9. **reshape(int width, int height)**: Resizes the viewport and sets up the projection matrix for the 3D scene.
10. **keyboard(unsigned char key, int x, int y)**: Handles keyboard input and performs actions based on the pressed keys. It controls zooming, rotation, bow movement, and walking animation of the character.
11. **main(int argc, char** argv)**: The main function that initializes GLUT, sets up the window, enables depth testing, and registers callback functions for display, reshape, and keyboard events. It enters the GLUT main loop to handle events and render the scene.

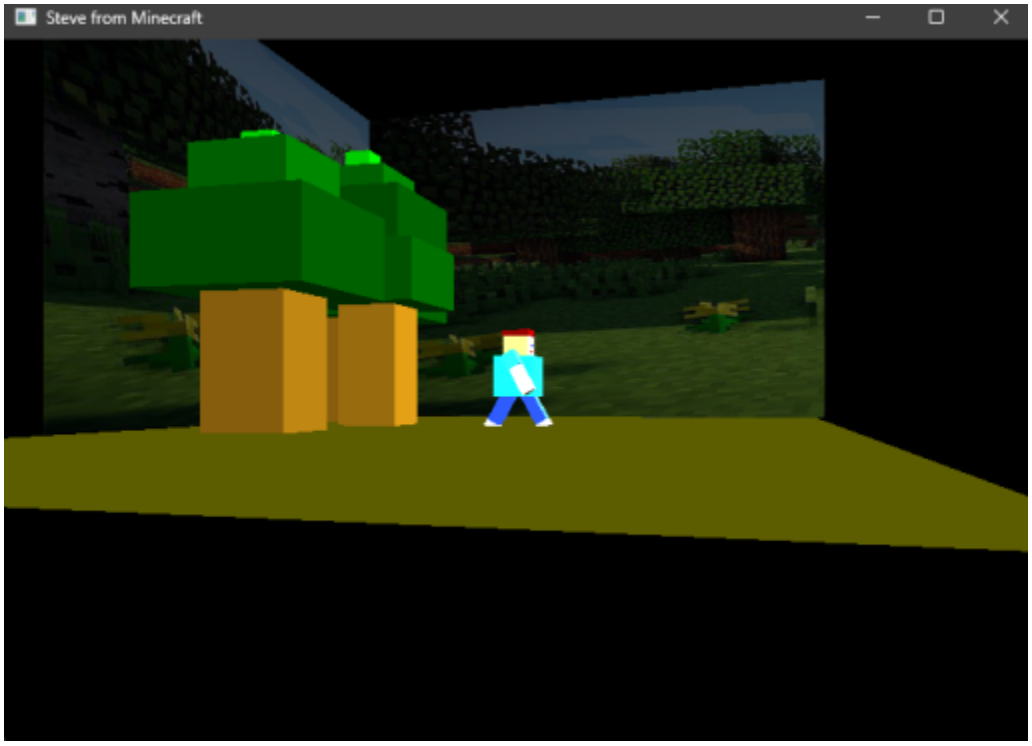
Overall, the code combines several OpenGL functions and utilities to create an interactive 3D rendering of the character model, allowing for user interaction and animation.

Implementation

GLUT enables us to create objects that recognise keyboard input using either "normal" keys or special keys such as F1 and Up. This section shows which key is functioning and what are the purposes or utilities for each control.

Controls	Function
E or e	Rotate Left
Q or q	Rotate Camera Right
8	Zoom In
9	Zoom Out
C or c	Bow front
X or x	Bow back
W or w / Mouse Click	Walk
O or o	Play sound ‘walking.wav’

Output



Link Video + Code Source: GOOGLE DRIVE

Conclusion

In conclusion, the provided code represents a simple but interesting implementation of an OpenGL program. It showcases a character, which appears to be Steve from the popular game Minecraft, and provides various interactive features for the user. The code starts by initializing the necessary libraries and setting up the initial variables that determine the position and movements of the character. It sets the background color and shading model, preparing the rendering environment.

The `display()` function is responsible for rendering the character on the screen. It uses a combination of translations, rotations, and scaling to create the different body parts of the character, such as the shoes, legs, torso, arms, and head. Each body part is defined using solid cubes, and different colors are applied to achieve the desired appearance.

The character's movements are implemented through the `walk1` and `walk2` variables, which control the rotation of the shoes and legs. These variables are updated in each frame, creating a walking animation. Additionally, the character can be rotated and tilted using the 'q' and 'e' keys, respectively. The 'z' and 'Z' keys allow the user to zoom in and out, respectively, by adjusting the `zoomFactor`. This feature provides a closer look at the character or a wider view of the scene. Moreover, the 'c' and 'x' keys enable the character to perform a bowing motion. The `bow` variable controls the angle of the bow, and it is incremented or decremented within certain limits. This feature adds a dynamic aspect to the character's appearance.

Furthermore, pressing the 'w' key triggers a walking animation. The `walkState` variable cycles through four states, causing the character's legs to move in a continuous walking motion. The program also handles window reshaping through the `reshape()` function. It adjusts the viewport and projection matrix to maintain the aspect ratio and provide a proper perspective for the scene. Lastly, the `keyboard()` function handles user keyboard inputs. It captures key presses, such as the arrow keys, and updates the character's behavior accordingly. For example, the 'esc' key terminates the program, allowing the user to exit gracefully.

In summary, the code demonstrates a character rendering in OpenGL with interactive functionalities. It provides an opportunity for further development, such as adding more complex animations, incorporating user controls for character interactions, and expanding the scene with additional objects or environments.