TRIBHUVAN UNIVERSITY

Institute Of Engineering

Pulchowk Campus

Department of Electronics and Computer Engineering

Final Report
of **Graphics Project**
on
“iRobot Learning to Walk ”

Submitted to

Department of Electronics and Computer Engineering

Pulchowk Campus

Jan. 08, 2008

by

Abhishek Dutta
061BCT501
adutta.np@gmail.com

Suraj Sapkota
061BCT543
ssapkota@gmail.com

Acknowledgment

We would like to offer our sincere gratitude to our mentor Mr. Bikash Shrestha for guiding us throughout the project. We also want to thank Cindy Grimm and Ken Goldman of Washington University for creating a wonderful presentation slide titled "Perspective Projection[1]" (prepared for Washington University CSE131) that helped us understand the basic concepts and implementation details of perspective projection.

1 http://www.cse.wustl.edu/~kjg/cse131/modules/arrays/PerspectiveProjection.pdf - see references

Table of Contents

Abstract

iRobot is a project that displays an animation of robot that is learning to walk. We implemented the basic perspective projection technique, lighting effects in a graphical environment and back face detection algorithm for visible surface detection in Java programming language to build this project. A very simple 20 frame animation (using the concepts of key frame animation) of the walking robot was created in this project.

Objectives

The main objective behind doing this project is to have a hands on programming experience of applying the skills learned during the study of course “Computer Graphics – EG678EX”.

Programming Environment

We used Java 1.6 (jdk1.6.0_02) programming language to create the animation of a robot leaning to walk. Swing (which comes bundled with JDK) was used for User Interface (UI) portion of the project. We used Net beans 5.5.1 IDE for the project development. Code collaboration between the two developers was done using the subversion (svn) repository provided by google-code (http://code.google.com).

Methodology

The structure of robot, defined in world space coordinate (3D), was passed through the graphics pipeline to obtains 2D screen coordinates. The complete project task can be divided into these three sections:

- Implementation of Graphics Pipeline
- Creating the body of the robot
- Animation of the robot

Implementation of Graphics Pipeline

The robot's initial location was defined at the origin of the world space coordinate (x,y,z). Up vector was chosen along the z-axis as the robot stands along z-axis. The look point was fixed at the center of the robot's body. This gives us a look vector as shown in Figure 1.

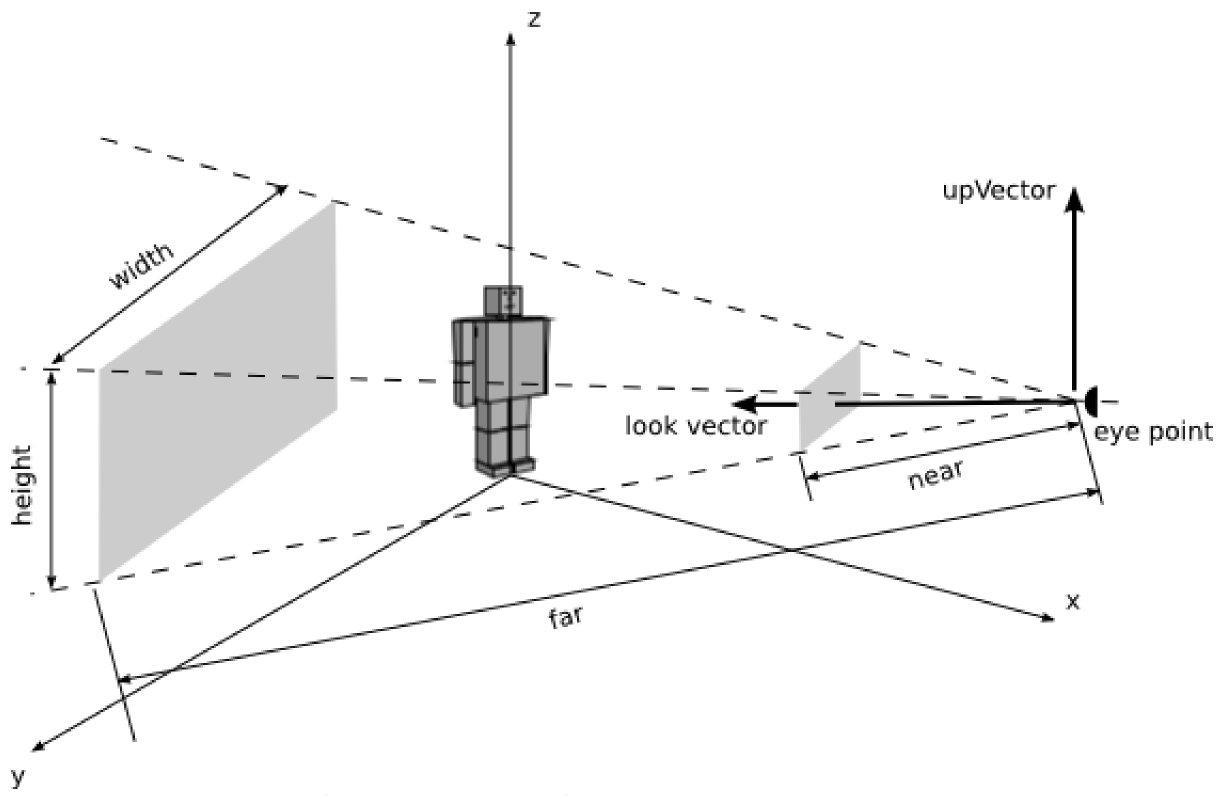

Figure 1.: Illustration of implementation of graphics pipeline

Near and Far specifies the percentage of volume along the look vector. To capture the complete image of the robot, the following values were chosen for the graphics pipeline:

near = 0.1 far = 1000

width = 800 height = 460 (as the screen display area is 800x460 pixels)

Up Vector = 0i + 0j + 1k (along z-axis)

eye was placed at a position such that initial the view of robot is similar to that shown in Figure 1.

Now the following series of matrix transformations were applied to obtain the perspective projection of the objects defined in world space coordinate.

1. Translate (T): translates the coordinate system defined at the eye point to the world space coordinate system's origin.

$$T = \begin{bmatrix} 1 & 0 & 0 & -eyeX \\ 0 & 1 & 0 & -eyeY \\ 0 & 0 & 1 & -eyeZ \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2.: Translation matrix T

where, eye coordinate = (eyeX, eyeY, eyeZ)

2. Rotate (R): rotates the translated eye space coordinate system to align with the world space coordinate system.

Given, eye coordinate (eyePoint) = (eyeX, eyeY, eyeZ) , upVector = (0,0,1) and lookPoint = (0,0,0) [at origin]

hence, the orthogonal unit vectors r, u, n which defines the axes of the coordinate system located at eye point are given by

n = (lookPoint – eyePoint) / (| lookPoint – eyePoint |) = (nx, ny, nz)

r = (upVector x n) / (| upVector x n |) = (rx, ry, rz)

u = n x r = (ux, uy, uz)

$$R = \begin{bmatrix} rx & ry & rz & 0 \\ ux & uy & uz & 0 \\ nx & ny & nz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.: Rotation matrix R

3. Scale (S): it is used to as pre-adjust for aspect ratio (of the final 2D image representation of the world space objects).

W = width of final image = 800px (in our case)

H = height of final image = 460px

far = far plane's distance from the eye point = 1000

$$S = \begin{bmatrix} (H/W)/far & 0 & 0 & 0 \\ 0 & 1/far & 0 & 0 \\ 0 & 0 & 1/far & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.: Scale matrix S

4. Perspective (P) – It scales, x and y coordinates by value of z. It is responsible for fore-shortening.

where, k = near / far

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/(k-1) & k/(k-1) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Figure 5.: Perspective matrix P

The combination of all these four matrix transformations produces the camera space coordinates.

$$\begin{bmatrix} u \\ v \\ d \\ w \end{bmatrix} = PSRT \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figure 6.: combined transformation for world coordinate to camera coordinate

Each world space coordinate (3D) is converted to camera coordinate (2D) using these transformations. Finally, the coordinates are converted to image space using the following method:

$$(x', y') = \left(W\frac{(u/w+1)}{2} \; , \; H\frac{(v/w+1)}{2} \right)$$

where (x' , y') are image space coordinates.

Creating the body of the robot

As in human, here the whole body of the Robot is divided into different parts. To be specific the whole body is divided into 15 parts. Each, such part is termed as Component and it can be clearly seen that each component is cuboid in shape. Each component here is defined in terms of height, width and breadth. During the execution of the program all the edge-points and center-points of each component are determined on the basis of its width, height and the breadth. These edge points are used to create each of the six surfaces of every components of the robot (head, hands, legs, etc).

The back-face detection of each component was done by a general method of normal calculation. And to maintain the realistic look of the robot, the components were painted on the basis of its distance from eye, the one which is farthest being painted first and and the nearest being the last one.

The technique of “Flat Shading” has been used as the illumination model for the robot's body. The assumption that a light source of some fixed intensity is located just above the eye point. The value of ambient light intensity was chosen so that the robot's body surfaces look realistic.

Animation of the robot

The animation of a walking robot was created by performing different types of transformation to hand and legs of the robot. There are 20 frames of robot's position has been defined which forms the basis of the animation. These 20 frames are displayed, with some delay inserted between each frame, to create the illusion of a walking robot.

Here the walking action of the robot is a bit different from normal human nature of walking, which is typically random and very complex to animate. Hence, we considered simple walking action which resembles to a real robot. Here we have considered (for simplicity) that only hands and legs undergo rotation after translation and all other parts are simply translated. During walking, all the motion of hand and legs are defined in terms of angles and each of these moving components have their own reference point (or pivot) . All the rotations of hand and legs are based on practical-walk and moreover on the basis of hit and trial method. Finally, Each set of rotation angle and translation forms a frame.

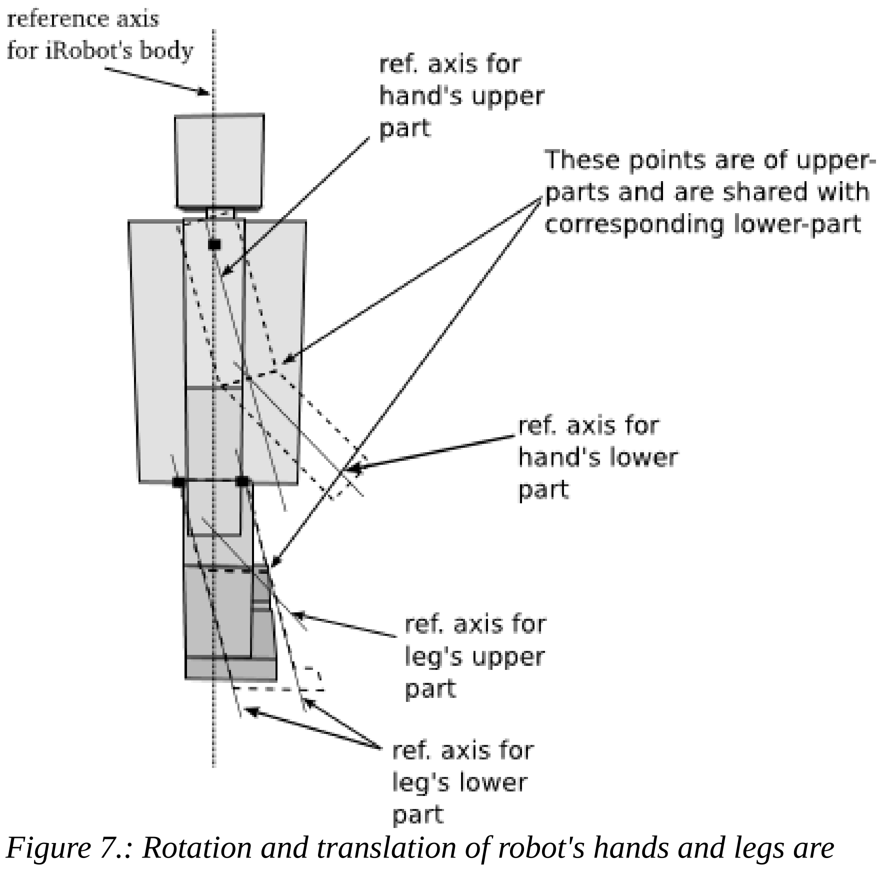
reference axis for iRobot's body
ref. axis for hand's upper part
These points are of upper-parts and are shared with corresponding lower-part
ref. axis for hand's lower part
ref. axis for leg's upper part
ref. axis for leg's lower part

Figure 7.: Rotation and translation of robot's hands and legs are performed according to predefined angles and distance for each frame (using previous frame as a reference)

Program Flow

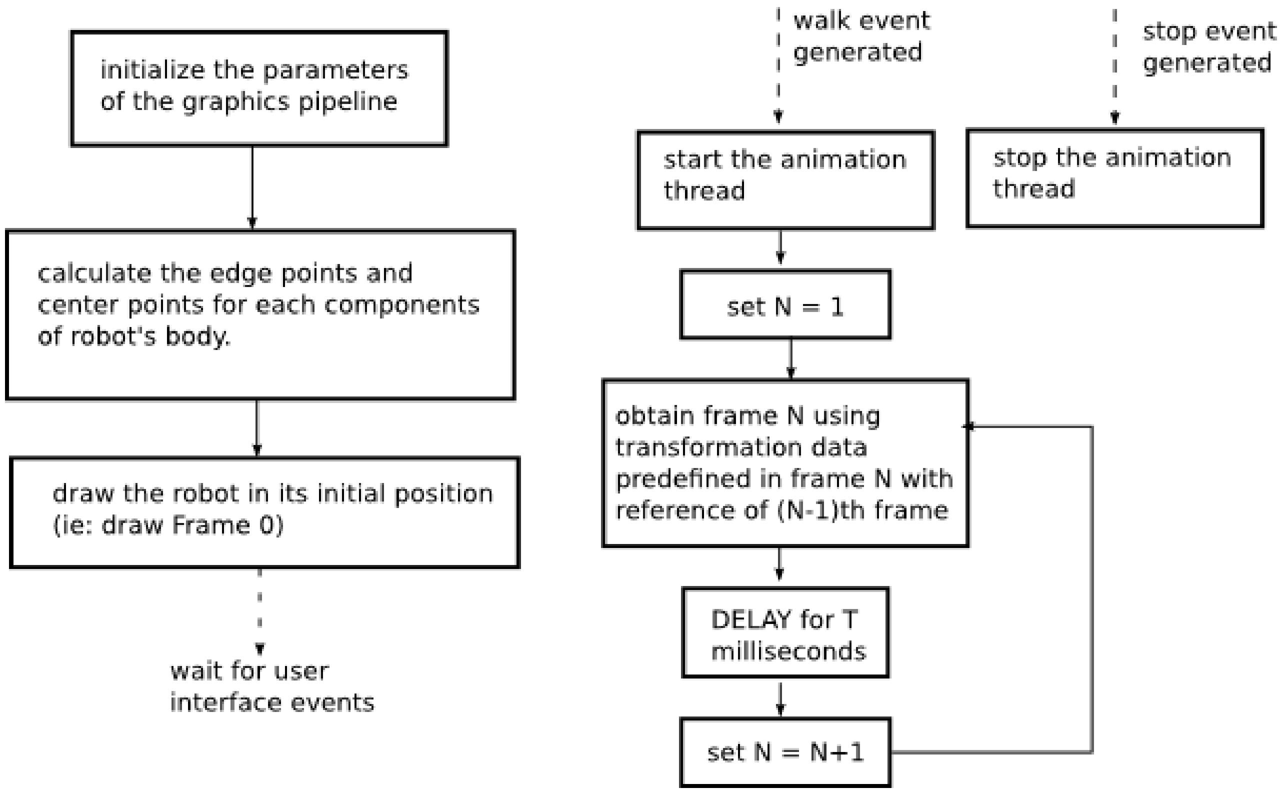
initialize the parameters of the graphics pipeline
calculate the edge points and center points for each components of robot's body.
draw the robot in its initial position (ie: draw Frame 0)
wait for user interface events
walk event generated
start the animation thread
set N = 1
obtain frame N using transformation data predefined in frame N with reference of (N-1)th frame
DELAY for T milliseconds
set N = N+1
stop event generated
stop the animation thread

Figure 8.: Flowchart for iRobot

Sample Output

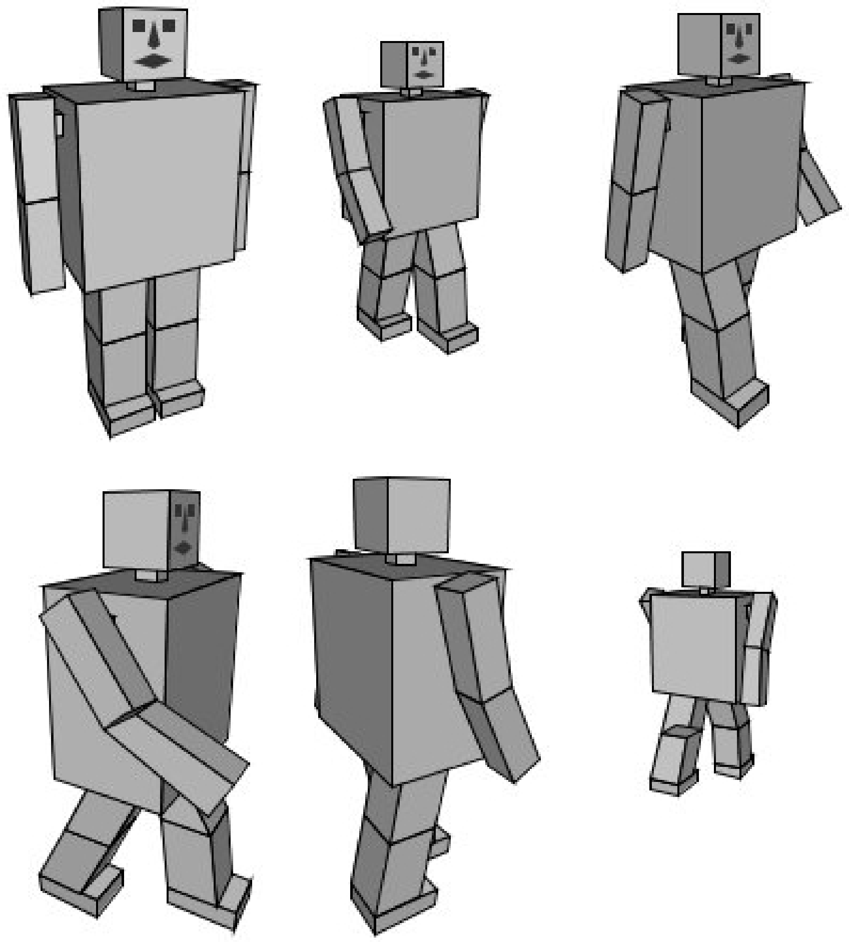

Figure 9.: Screen shots taken from iRobot project

More screen shots (and other documents) are available at iRobot's project website [http://irobot-graphics.googlecode.com]

Limitations

The limitations of iRobot project are:

1. The robot cannot mimic the natural walking action of a human being.
2. The order of painting the component is based on the distance of its center from the eye point. It sometimes causes error as all the component does not lie in a same axis while moving. This defect of painting order is visible in some frames.
3. The body parts of the robot are Cuboid. The look would look more natural if the body parts are formed our of cylinders.

Conclusion

Although this project does not perform any useful task, it proved a very useful project for learning the basic programming techniques used in computer graphics. No optimizations were performed on the implementation of some algorithms to preserve clarity of the code. Most of the part of the code are well documented with some simple text based illustrations. We also realized that human movements are very complex to perform in a animation.

References

1. 6.837 Computer Graphics - course at MIT Open course ware

   [http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-837Fall2003/CourseHome/index.htm]

2. 3D WireMesh Generator - ECE576 Final Project

   [http://instruct1.cit.cornell.edu/courses/ece576/FinalProjects/f2007/ms883_sa453/ms883_sa453/index.html]

3. “Perspective Projection” - slides prepared by Prepared by Cindy Grimm and Ken Goldman for Washington University CSE131

   [http://www.cse.wustl.edu/~kjg/cse131/modules/arrays/PerspectiveProjection.pdf]

4. "Computer Graphics - C version" - 2nd edition , Donald Hearn & M. Pauline Baker, Pearson Education

5. “Style Translation for Human Motion” - Homepage of Jovan Popović, MIT Computer Science and Artificial Intelligence Laboratory

   [http://people.csail.mit.edu/jovan/]