Security Camera Specification

1. Introduction

1.1 Warning

*There may be errors in this document. While every attempt is made to ensure the completeness and accuracy of this document, some errors may remain.*

1.2 Notation

There is an important typesetting convention used in this document.

Words and phrases are typeset in a bold italic font like *this* to indicate that they are defined or explained rigorously in Section 1.5, “Definitions, Acronyms, and Abbreviations”.

1.3 Overview

This document describes all external requirements for a security camera control system. It also describes the interfaces for the system.

The goal of this system is to provide a flexible, computer-controlled television display that can supply views of a warehouse containing valuable assets. The display will be used by the security staff of the warehouse.

Software referenced in this document can be found on the cs-t11 server under the csfiles share directory.

Full details of these requirements are contained below.

1.4 Scope

This document is the only document that describes the requirements of the system. It is meant for use by the developers, and it will also be the basis for validating the final, delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarification, where necessary, and will not make any alterations without the permission of the client.
1.5 Definitions, Acronyms and Abbreviations

**click on an area** ................. A user clicks on an area of the microcomputer screen by **positioning the mouse cursor** over the area, and **clicking** the left mouse button.

**position the mouse cursor** .......... A user positions the **mouse cursor** by moving the mouse.

**mouse cursor** ..................... The graphic arrow on the screen that follows the motion of the mouse on the desk.

**click** ............................... To click a mouse button, the user presses the mouse button down and then releases the mouse button without moving the mouse.

**point at** .......................... The **camera** is said to point at an area if it’s line of sight intersects with a line perpendicular to the floor at the center of the area. The point of intersection of the line of sight with this line should be **viewing height** from the floor.

**warehouse** ....................... The warehouse is Olsson 001 at the University of Virginia in Charlottesville, VA.

**microcomputers** ................ The microcomputers in the **warehouse** are Intel 80586-based Micron Millenia Pro2 PCs.

**camera** ........................... The camera is a Canon VC-C3 communications camera.

**EEEEEE** .......................... EEEEE stands for Extremely Expensive Emerald Enterprise of England.

**initial position** .................. The initial position of the **camera** should be such that it is aimed toward the center of the room with the line of sight perpendicular to the wall and parallel to the floor. This is the position that it returns to when initialized.

**full speed** ........................ Full speed is 76 degrees per second panning, and 70 degrees per second tilting.

**viewing height** .................. The viewing height is defined to be 40 inches from the floor.
1.6 Overview of Developer’s Responsibilities:

The developer is responsible for:

• developing the system,
• installing the software on the client’s hardware,
• conducting any user training that might be needed for using the system, and
• maintaining the system for a period of 2 months after installation.

2. General Description

2.1 Product Functional Overview

EEEE has recently purchased a new emerald warehouse. In order to protect against theft, they have purchased security cameras. They have also hired two shifts of security personnel to operate the security camera using a recently purchased microcomputer.

The required software system is to control one of the security cameras. There are two (2) different ways in which EEEEE management have decided that they want the camera to be controlled. In the first (programmed control), the camera will follow a predefined path scanning important areas of the warehouse. In the second (direct control), the guard will be able to determine where the camera points by clicking the mouse. In both, a graphic representing the plan view of the warehouse containing the camera is to be displayed on the computer’s monitor. The area of the warehouse that is in the camera’s field of view is to be highlighted on the plan graphic by displaying a small colored box at the place at which the camera points.

The mode of operation of the camera is to be determined by a menu that allows selection between predefined scanning sequences and the direct control mode.

2.2 User Characteristics

The users of this system are security personnel with little experience or training in using microcomputers.

2.3 General Constraints

The system should run on the microcomputers in the warehouse and should interface with the camera in the warehouse.
3. Functional Requirements

3.1 Basic User Interface

The interface should present a plan view, i.e., from above, of the warehouse on the computer monitor screen along with a six-item menu. The interface should indicate what area of the map the camera is pointed at by placing a small square at the area in a different color than the background of the map. The menu buttons are to provide for switching between three different predefined scanning sequences and direct control and to control the zoom function while in direct control.

3.2 Programmed Control

The program is to read three predefined viewing paths from three text files. The file format is as follows:

- An input file consists of a sequence of zero or more movements.
- A movement consists of four parts - a position indicator, a pan speed indication, a tilt speed indication, and a delay indication. The parts are all on one line and are separated by one or more blank characters.
- A position indication is a pair of numbers separated by one or more blanks representing the X and Y coordinates of the desired location to point at. The distances are measured in feet from the origin which is assumed to be the intersection of the walls and floor nearest to the main entry door of the warehouse.
- A speed indication is a single number representing the speed with which the head is to move as a percentage of its full speed.
- A delay indication is a number representing the number of seconds that the camera is to remain at a location.

Each movement starts on a new line. Each of the fields of a movement is separated by one or more spaces.

All the numbers are expressed as floating-point values with at most 6 digits (not including the ‘.’). See, for example, the values shown in the table below.

The starting position for the camera is to be the initial position. Thus the first movement command in a file defines the destination after the first move. If any of the positions given are outside the room, the user should be notified and the system should then truncate the position to the boundaries of the room.

The camera is to repeat the sequence until the user changes the required action through the menu. The camera will be in autofocus mode and at the median zoom range.
Here is an example path that the camera might follow:

<table>
<thead>
<tr>
<th>Starting Position</th>
<th>Ending Position</th>
<th>Speed to move at (pan/tilt)</th>
<th>Time to stay before going to the next position</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial position</td>
<td>(1.2, 13.5)</td>
<td>100%/100% full speed</td>
<td>2 seconds</td>
</tr>
<tr>
<td>(1.2, 13.5)</td>
<td>(24.6, 13.5)</td>
<td>50%/100% full speed</td>
<td>1 second</td>
</tr>
<tr>
<td>(24.6, 13.5)</td>
<td>(8.8, 20.7)</td>
<td>25%/50% full speed</td>
<td>5 seconds</td>
</tr>
<tr>
<td>(8.8, 20.7)</td>
<td>initial position</td>
<td>100%/75% full speed</td>
<td>10 seconds</td>
</tr>
</tbody>
</table>

3.3 Direct Control

If direct control is selected from the menu, scanning operation is to cease and the camera is to move at full speed to a specific view. The view is to be determined by the mouse. At any time during direct control operation of the system, the user can click on a point on the warehouse map. The system should then cause the camera to turn at full speed to point at the area represented by that area of the map which was clicked on.

While under direct control, the user can zoom in or out by selecting the appropriate buttons on the menu.

4. Camera Interface

The system must interact with the camera software already developed by EEEE. A Head class (contained in files camerahead.h and vccamerahead.lib) has been developed that provides access to the pan/tilt head on the camera and to the simulators. Below are the member functions provided (note: the pan range is -90 (left) to +90 (right) degrees, the tilt range is -25 (up) to +30 (down) degrees, and the speed range is 1 to 76 degrees/second panning and 1 to 70 degrees/second tilting):

- int Head.Home() Moves the head to it's home position: (pan of 0 degrees, tilt of 0 degrees).
- int Head.MoveAbsolute(x, y) Moves the head to the absolute coordinates (x, y). x is the number of pan degrees off of the origin, and y is the number of tilt degrees off of the origin.
- int Head.PanAbsolute(x) Pans the camera to the absolute coordinate x, i.e., pans 'x' degrees off the origin. Tilt position remains the same.
- int Head.TiltAbsolute(y) Tilts the camera to the absolute coordinate y.
The pan position remains the same.

```cpp
int Head.MoveRelative(x, y) Moves the head x-degrees relative to the current pan position, and y-degrees relative to the current tilt position.
```

```cpp
int Head.PanRelative(x) Moves the head x-degrees relative to the current pan position. The current tilt position remains the same.
```

```cpp
int Head.TiltRelative(y) Moves the head y-degrees relative to the current tilt position. The current pan position remains the same.
```

```cpp
int Head.GetPosition(x, y) Get the camera’s current position. The pan and tilt positions are returned through the reference variables x and y.
```

```cpp
int Head.SetPanSpeed(x) Sets the head’s pan speed in degrees/second.
```

```cpp
int Head.SetTiltSpeed(x) Sets the head’s tilt speed in degrees/second.
```

```cpp
int Head.GetSpeed(x, y) Get the camera’s current pan and tilt speeds. The speeds are returned through the reference variables x and y.
```

```cpp
void Head.SetAutoFocus() Places the camera into autofocus mode.
```

```cpp
void Head.SetManualFocus() Places the camera into manual focus mode.
```

```cpp
int Head.GetFocusRange(x, y) Get the minimum and maximum range for the camera’s focus ring. Values are returned through the reference values x and y.
```

```cpp
int Head.GetFocus(x) Get the current setting for the camera’s focus ring. Setting returned through x.
```

```cpp
int Head.SetFocus(x) Set the camera’s focus ring to x.
```

```cpp
int Head.GetZoomMax(x) Get the camera’s maximum zoom setting. The minimum zoom setting is 1. Setting is returned through x.
```

```cpp
int Head.SetZoom(x) Set the camera’s zoom setting to x.
```

```cpp
int Head.GetZoom(x) Get the camera’s current zoom setting. Setting is returned through x.
```

```cpp
void Head.SetAddress(x) Sets to x the network address of a lab PC that is assumed to be controlling the camera.
```

```cpp
int Head.Close() Closes the network connection with the remote PC.
```
After issuing a command to the Camera interface, the camera interface will respond with one of three possible values upon completion of the command: CommandOK, TransmissionError, or ReceptionError. CommandOK is returned when no errors are encountered. TransmissionError is returned when the interface could not make connection with either the simulator or camera server. ReceptionError is returned when the server or simulator receives an illegal command or an illegal value. Note that in some cases, the interface may return an error when the action was completed correctly. This behavior has been observed in panning operations and in operations taking long periods of time.

5. View Simulator

So that software can be developed without the use of the actual hardware, a simulator is available to simulate the movement of the pan/tilt head. This simulator is a “camera view simulator”. It displays a three dimensional image of the inside of the room representing what the camera is viewing. As the camera pans and tilts, the image of the room shifts appropriately.

The simulator is used as follows. The program CameraSimulator.exe is executed on one of the PC’s in the Olsson 001 laboratory. You will be prompted to identify which camera you are simulating (#1-4; see section 8 to identify the cameras). This program turns the PC into a simulated camera and associated television monitor. Your application program executes on a second PC and sets the network address of the simulator using the SetAddress member function of the Head class. The parameter to the member function call is the name of the PC running the simulator. Thus if the simulator is running on labpc18 and you have an object of type Head called SimHead, the member function call is merely:

```c++
  SimHead.SetAddress("labpc18");
```

The simulator may also be used on the same PC as your test program if it has networking installed. Simply set the address of the Head object to “localhost”. Please note that this will severely affect the performance of any programs running on the computer since CameraSimulator is a resource-intensive program.

6. Design Constraints

6.1 Software Constraints

The system must be written in C++, compile with Microsoft Visual C++ 6.0, and run under Microsoft Windows 2000.

6.2 Hardware Constraints

The system must be installed and must run correctly on the microcomputers in the
7. Coordinate Systems

The following is an overview of how to take a point in Cartesian coordinates and translate it to spherical coordinates. Fig. 1 is a Cartesian coordinate system. Points are described in three dimensional space as absolute positions along the X, Y and Z axes.

Spherical coordinates are given as a pan angle in the XY plane, a tilt angle in the YZ plane, and a distance R from the origin. This is exactly what is required because the pan/tilt head movement matches a spherical coordinate system.

The warehouse floor is represented as the XY plane and the origin of the spherical coordinate system is centered on the pan/tilt assembly. Given any point (B, C) (as shown in Fig. 2) we can create two right triangles by drawing lines from that point, parallel to the X and Y axes, and a line connecting (B, C) and the camera location. The pan angle is as shown.

Visualizing the tilt angle is the similar, except we project the pan/tilt unit onto a wall instead of the floor, as shown in Fig. 3. Here the wall is a plane formed by the YZ axes.

Computing the angles needed is a simple application of trigonometry. It involves the application of the inverse circular functions such as arcsine.

Arcsine is the inverse sine, or sin⁻¹ on your scientific calculators. “asin” is usually the name of the inverse sine function in C/C++. When using “asin” you will be
dealing with an input parameter of -1 to +1 and a return value in radians, instead of degrees. This requires us to apply a conversion factor so that we can present degree values to the pan/tilt interface. Note that the warehouse coordinate system has an origin in the corner so warehouse measurements will always be positive. Note also that pan angles are both positive and negative in the camera hardware.
We are careful to distinguish between radians (as required by the trigonometric functions) and degrees (as required by the camera hardware). If you send a value in radians to the camera hardware, it will probably not appear to move.

8. Warehouse Dimensions

Figs. 4a & b show the major dimensions of the warehouse in the plan view.
The origin of the camera coordinate system is 7’ 10” above the ground and 8.25” from the walls.