Pool Game Designed and Implemented Using Major-Minor FSM Setup

Group 10
Gary M. Matthias
Timothy Mwangi
Anthony Quivers
Input Module

• Physical Objects
  – Pool table
  – Pool stick

• Devices
  – Video camera
  – Accelerometer
  – Analog-to-digital converter
The Pool Stick

- Equipped with an accelerometer for speed calculations
- Colored at the tip for recognition by the camera
Camera View of Pool Table
Input Module: Block Diagram
Game Logic Unit

Design Specification

• Independent Set of modules used to control the all balls on the table.
• Responsible for controlling all events and enforcing all rules of 2 player British pool.
• Unit is abstracted away from user input interface and output graphics system.
• The positions of all balls are refreshed for the graphics module once per frame

Control Unit Implementation

• All balls on the table are controlled concurrently
• There are a total of 16 instances of Ball FSM Modules (i.e.: 1 for cue and 15 balls 1 through 15).
• Internal 2D VGA graphics interface for testing and debugging (This doubles as a backup graphics interface)
Ball FSM

All State Buses
42 bits

Separator

Address

Ball Motion FSM

Ball State Lookup Table

Collision Dynamics

Init X Init Y Prog
Init VX Init VY Prog Vel

Enable

X
Y
Vel_X
Vel_Y
Xb
Yb
Vel_Xb
Vel_Yb

Vel_X_f
Vel_Y_f

Ball State Bus
3-D Pool Display

• Perspective ray tracing
• The math
• Block diagram
• Memory requirements
Perspective Ray Tracing

Eye

Virtual screen

To light source
The Math

Intersection of a line and plane

• Parametric equation of a line
  
  \[ x = x_1 + u \, (x_2 - x_1); \quad y = y_1 + u \, (y_2 - y_1); \quad z = z_1 + u \, (z_2 - z_1) \]

• Equation of a plane: \( A \, x + B \, y + C \, z + D = 0 \)

Substitute equation of line into equation of plane

\[ A \, (x_1 + u \, (x_2 - x_1)) + B \, (y_1 + u \, (y_2 - y_1)) + C \, (z_1 + u \, (z_2 - z_1)) + D = 0 \]

Solve for \( u \)

\[ U = \frac{Ax_1 + By_1 + Cz_1 + D}{Ax_1 - x_2 + By_1 - y_2 + Cz_1 - z_2} \]

If \( Ax_1 - x_2 + By_1 - y_2 + Cz_1 - z_2 = 0 \)

• Line is perpendicular to plane, no intersection/infinite intersections
Intersection of a line and sphere

• Equation of a sphere: \((x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = r^2\)

Define:

• \(a = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2\)

• \(b = 2\left[(x_2 - x_1)(x_1 - x_3) + (y_2 - y_1)(y_1 - y_3) + (z_2 - z_1)(z_1 - z_3)\right]\)

• \(c = x_3^2 + y_3^2 + z_3^2 + x_1^2 + y_1^2 + z_1^2 - 2[x_3 x_1 + y_3 y_1 + z_3 z_1] - r^2\)

Quadratic equation: \(au^2 + bu + c = 0\); Solution: \(-b \pm \sqrt{b^2 - 4ac}/2a\)

The exact behavior is determined by the expression within the square root:

\(b^2 - 4ac\)

• If this is less than 0 then the line does not intersect the sphere.

• If it equals 0 then the line is a tangent to the sphere intersecting it at one point, namely at \(u = -b/2a\).

• If it is greater than 0 the line intersects the sphere at two points.
Signal description

Bx_i, By_i = x and y position of ball i
Bp_i = Boolean, Is ball i still on the table?
X_t, Y_t = x and y position of cue tip
X_p, Y_p = x and y position of pivot
P = Current player
p_c = pixel count
L_c = line count
E_P_L = Vector from eye to pixel
Nv_i = Normal vector
O_i = Object
P_L_S = Vector from point to light source.
NI = Number of intersections
PI_i = Point of i^th intersection
N_L_D = dot product of normal at point and point to light vector.
Lum = Luminosity in YUV absolute colorscale
Memory requirements

• For each pixel $0 \leq \text{Luminosity} \leq 120$; Luminosity can be equal to 240 but in this 3-D renderer the maximum used is 120. – Requires 7 bits/pixel

• For each pixel $0 \leq O_i \leq 120$ is stored requires 7-bits/pixel

• Total memory requirement for double buffer:

  $$= \frac{(7+7)}{8} \times (1024 \times 768) \times 2 \times 10^{-6} = 2.76 \text{MB}.$$