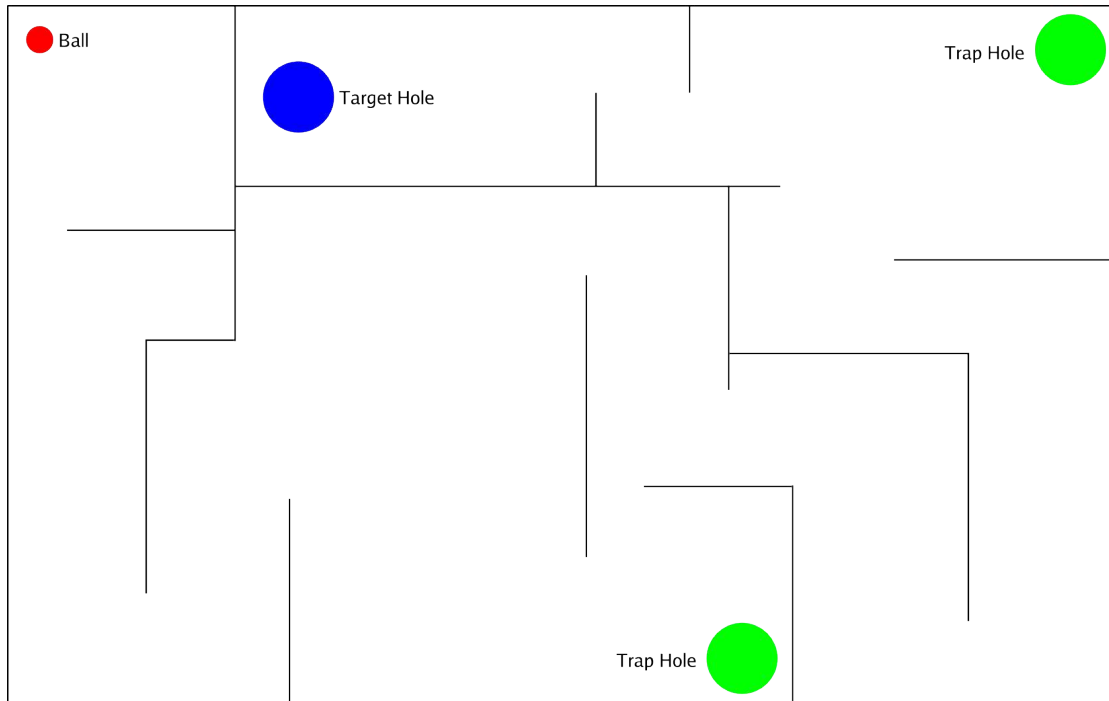


Tilting Maze Game

Matt Fishburn
Hongyi Hu

TA: Jae Lee

Game Overview

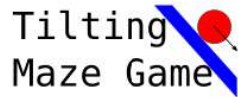


Sample Screenshot

- Tilting the Board Moves the Ball
- Objects on Board:
 - Walls
 - Ball
 - Traps
 - Destination

Source Control / Documentation

http://hawkeye.mit.edu/cgi-bin/trac.cgi/file/trunk/src/game_state.v



Search

logged in as fishbum Logout Settings Help/Guide About Trac

Wki Timeline Roadmap Browse Source View Tickets New Ticket Search

Revision Log

root / trunk / src / game_state.v

View revision: 14

Revision 14 (by hongyihu, 11/08/05 19:11:34) Adding first revisions of code

```
module game_state(reset, clk_27, hz_60, x_acc, y_acc, mask_computed, x, y, vx, vy, inHole, targetReached, new_x, new_y, new_vx, new_vy, level);
// global input signals
input reset; //
input clk_27; // clock signal running at 27 MHz
input hz_60; // 60 Hz signal to indicate next frame

// physics input signals
input signed [31:0] x_acc, y_acc; // x and y acceleration data from physics module

// mask input signals
input mask_computed; // has the mask for the current level been computed?

// collision input detection signals
input [11:0] x, y; // x and y position of ball
input signed [31:0] vx, vy; // x and y velocity of ball
input inHole; // Did the ball fall into a hole in the last time step?
input targetReached; // Did the ball reach the target in the last time step?

// outputs to collision detection unit
output [11:0] new_x, new_y; // new x and y position of ball
output signed [31:0] new_vx, new_vy; // new x and y velocity of ball
output [2:0] level; // current level number

// stored data
reg [2:0] level; // current level number

always @(posedge clk_27) begin
// check if the mask has been computed
end
endmodule
```

Note: See [TracBrowser](#) for help on using the browser.

Download in other formats:

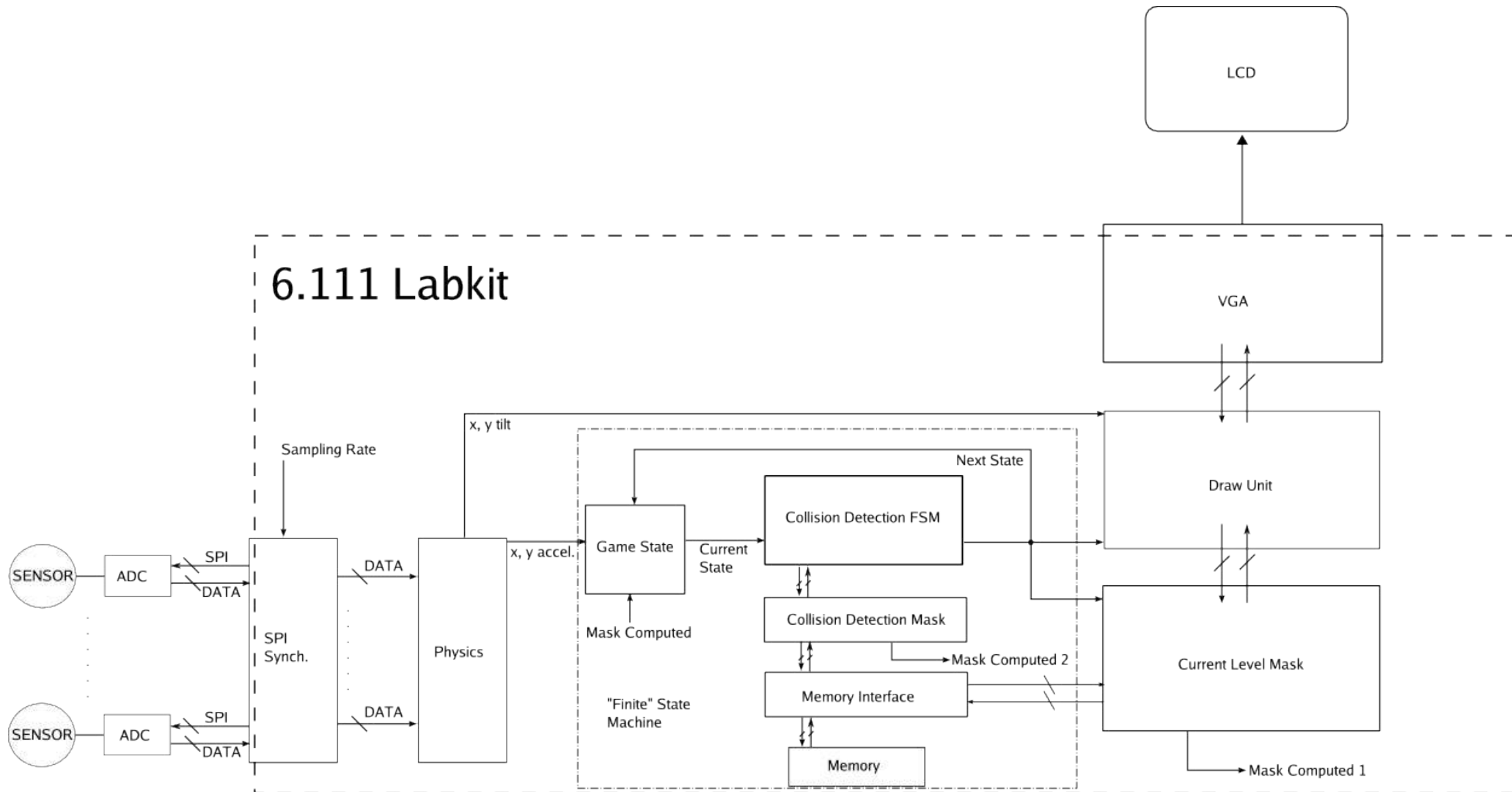
[Original Format](#) | [Plain Text](#)



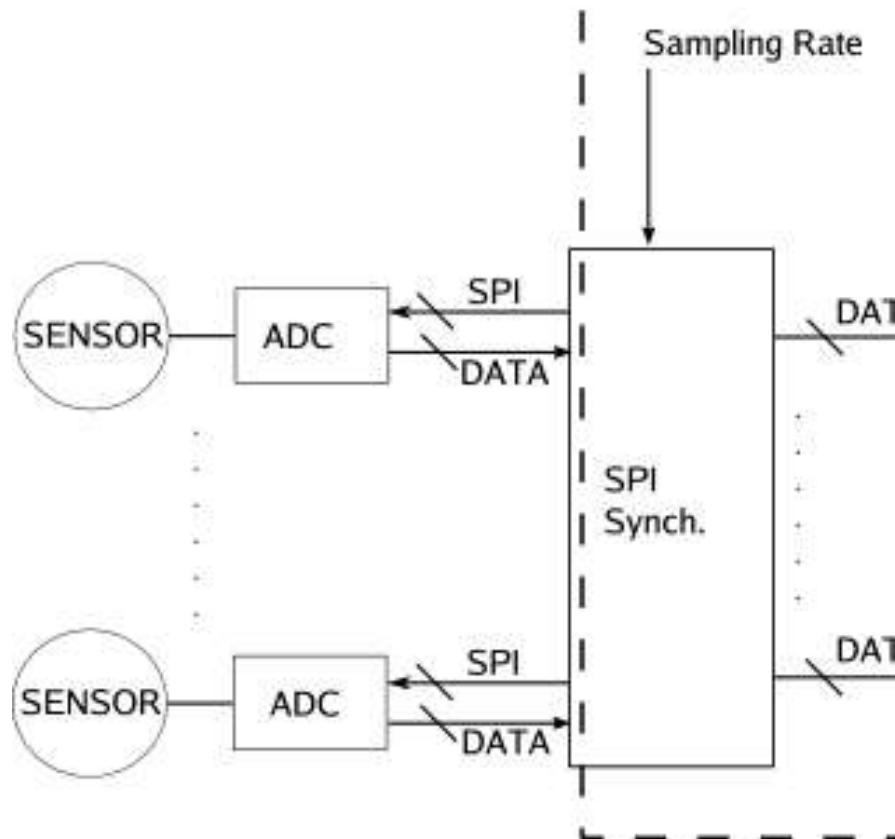
Powered by Trac 0.8.4
By Edgewall Software

Visit the Trac open source project at
<http://trac.edgewall>

Block Diagram Overview

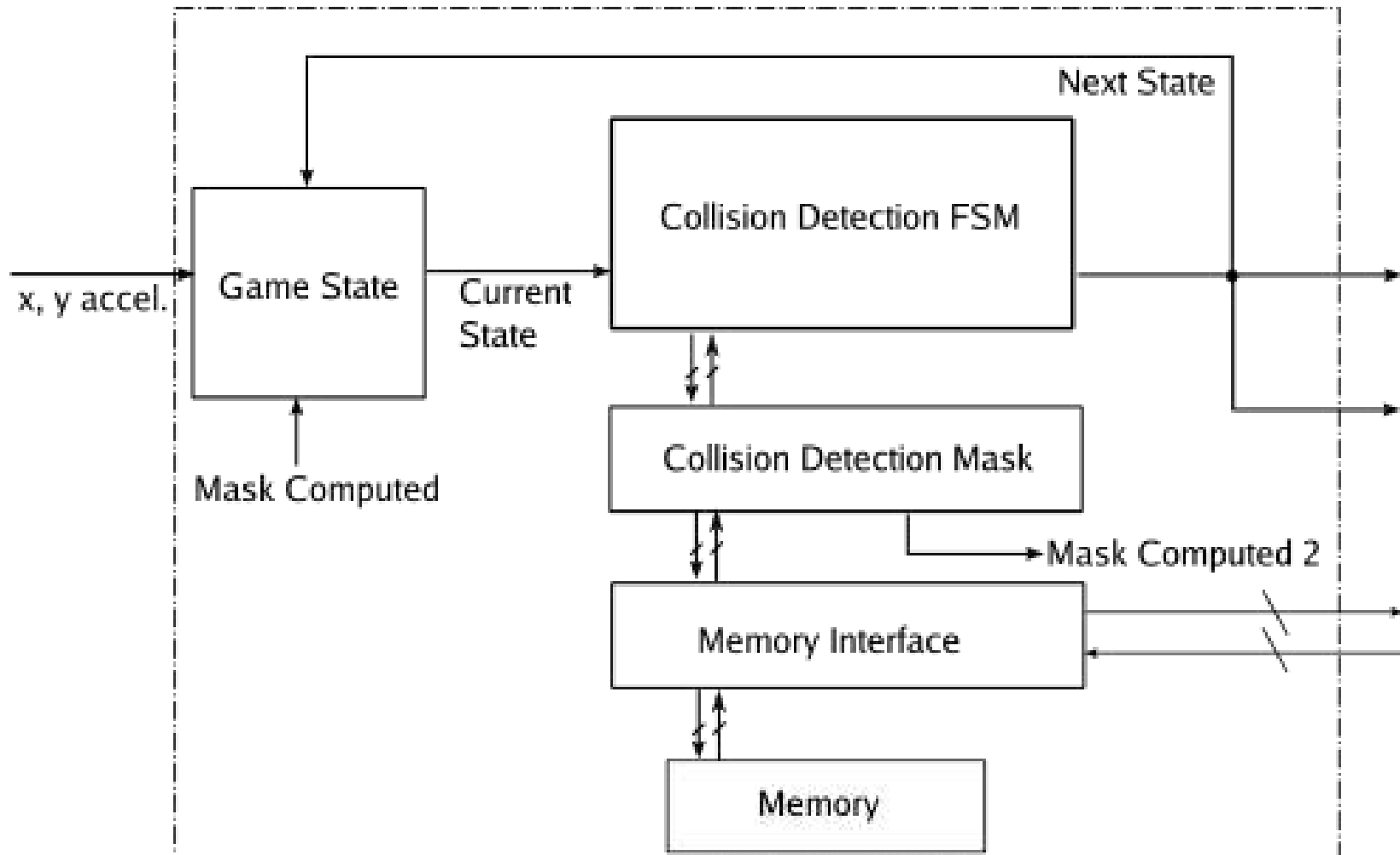


External Sensor Interfaces



- Two Gyros & Three Accelerometers
 - 0 to 5 V output
 - 10-200 Hz sampling
- Two 3-Channel ADCs
- SPI Synchronizer Module

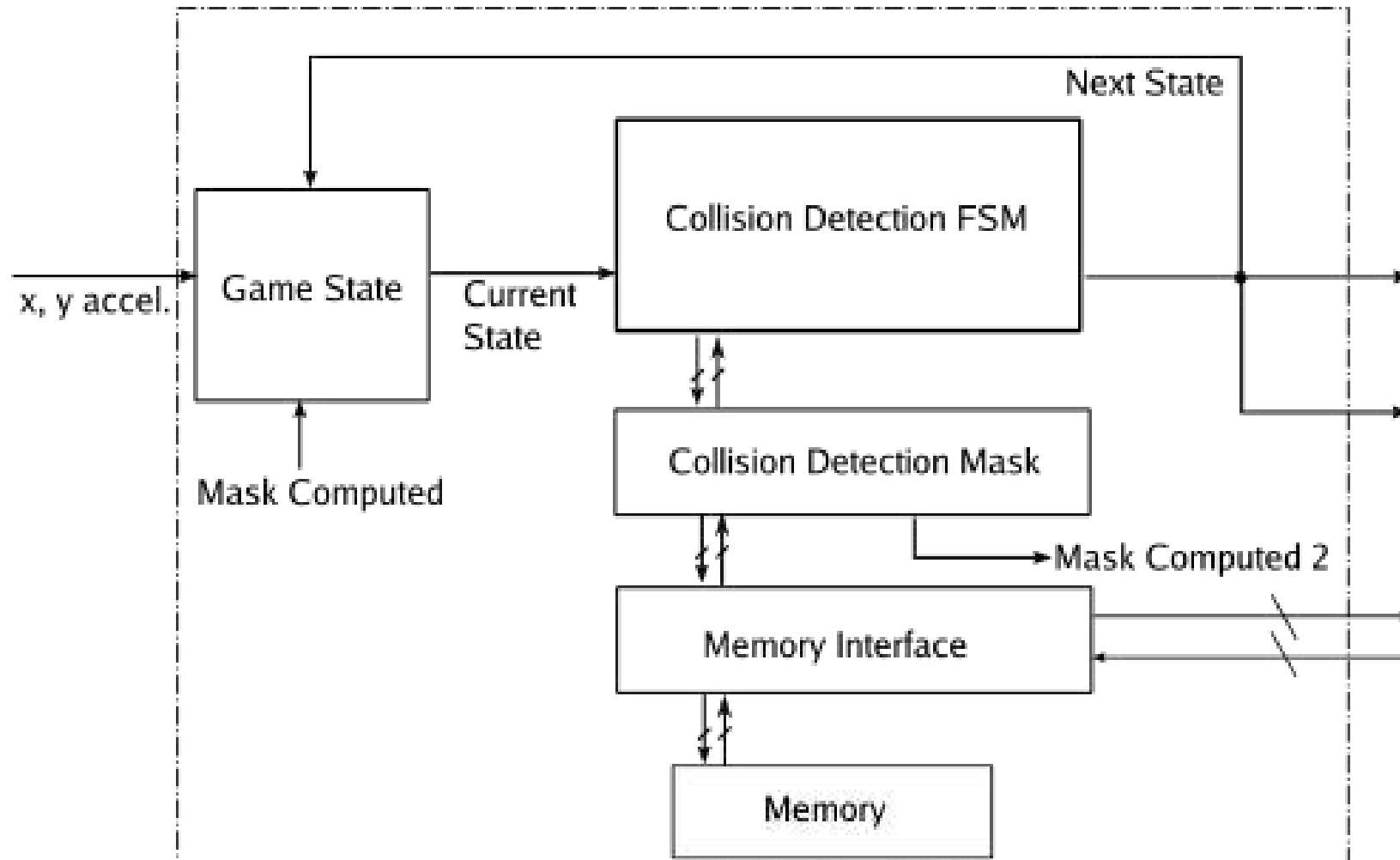
FSM



Game State

- Stores and updates the state of the game
- Inputs:
 - Acceleration in x and y axis (signed fractionals)
 - New ball position and velocity, game state data
- Outputs:
 - Current ball position and updated velocity

Collision Detection



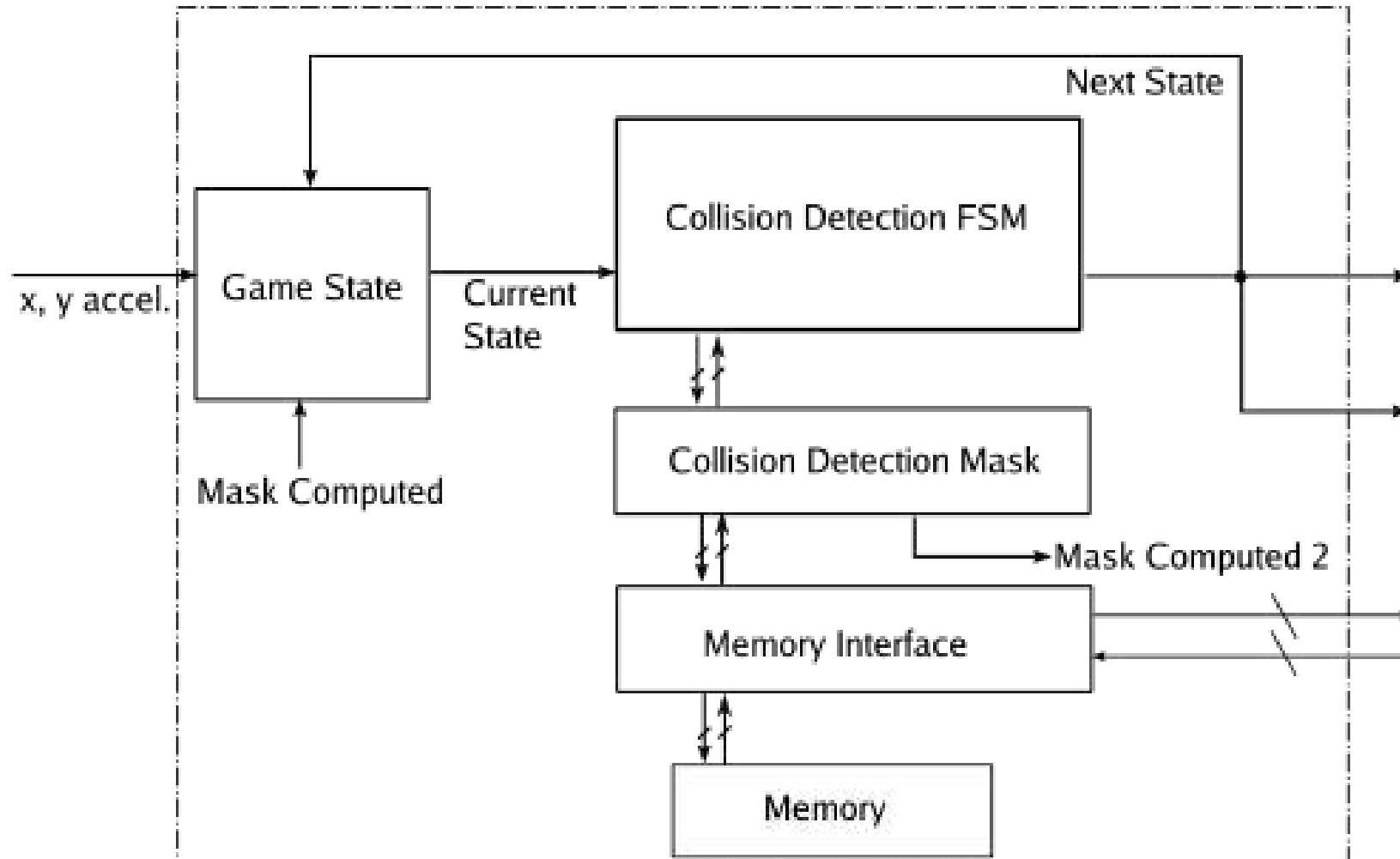
Collision Detection

- Moves the ball and checks for collisions
- Inputs:
 - Current position and velocity of ball
 - Collision data from CD mask
- Outputs:
 - New position and velocity of ball
 - Collision query to CD mask

Collision Detection Process

- Check four sides and center of ball
- Set ball's velocity along an axis to 0 if the ball will collide with a wall
- Reset level if ball's center is over a trap hole
- Move on to next level if ball's center is over a target hole
- Otherwise move the ball to its new position and repeat

Collision Detection Mask



Collision Detection Mask

- Stores location information for every obstacle in current level
- Inputs
 - Collision query from Collision Detection module
 - Level data from Memory Interface
- Outputs
 - Collision data
 - Level query to Memory Interface
- Level Mask similar

Memory Interface

- Needs to communicate with both collision mask and level mask
- What is stored?
 - Type of each 16 x 16 pixel block stored in three bits
 - Location in memory signifies position on map
 - Each level takes up 4Kb of memory

Draw and XVGA Units

- Receives ball information from the FSM
- For every pixel:
 - Sees if the ball should be drawn
 - Checks with level mask if a wall/hole should be drawn
- Sends appropriate color for every pixel to the screen