Venita Boodhoo
Janie Liu
Jing Wang

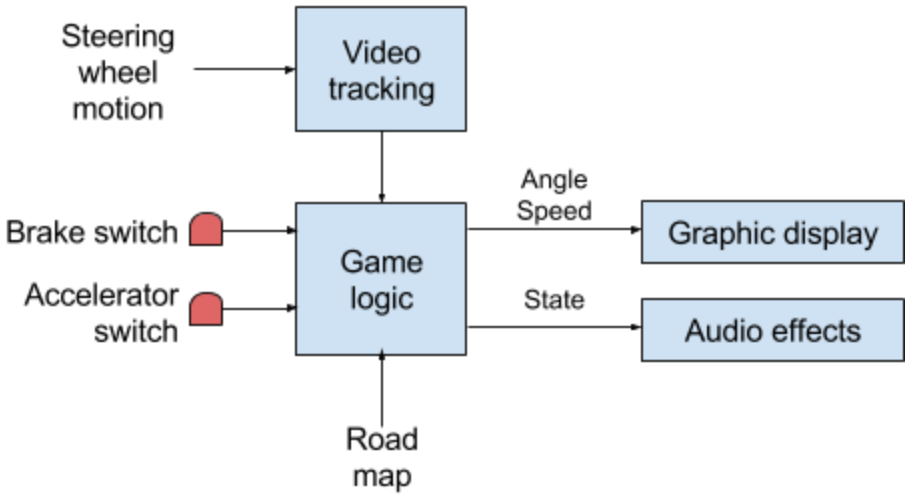
Motion Controlled Driving Game

1. Overview

"You are a tired computer engineer coming home from work. Due to extreme sleep deprivation, the world around you looks strange…"

We would like to design a game that uses motion tracking, graphics, and audio effects. The game is a augmented reality driving game that will be displayed using graphics generated by the FPGA onto the computer monitor. The user is inside the car and can control how the car moves by holding out their two hands and moving them around as if they are holding the steering wheel. There will also be brakes and acceleration pedals for the feet. We can use a VGA camera or an accelerometer to track the movements of the hands and feet. The goal of the game is to reach the final destination without hitting any objects on the road. Sound effects will be played when you crash. Game logic may include assessing how well the driver drove i.e. not crashing into many obstacles.

2. Block Diagram

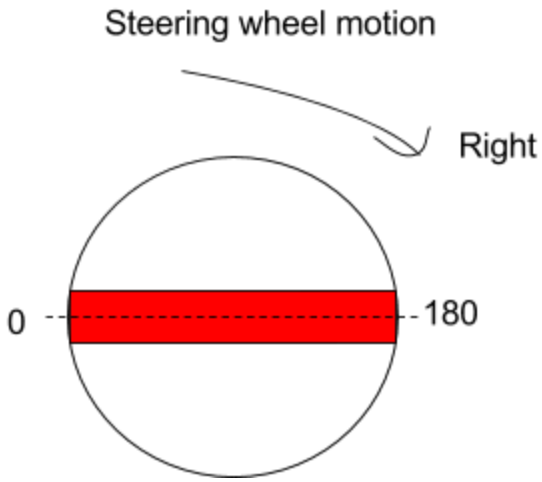


3. Steering wheel (Jing)

In this game, the driver can control how the car in the display moves along the road by turning a physical steering wheel. The steering wheel will be built by cutting out a plastic or wooden circle and adding a rectangular bar across the middle, which will be of a different color

from the circle. As the driver turns the wheel, the colored rectangle will change its angle, which will provide the angle of rotation. The video camera will be aimed at the wheel as the driver plays to provide motion information in real time.

If the colored bar is horizontal, then the angle of the perpendicular to the bar would be 90° and the resulting direction of the car should be straight. The angle from the horizontal would determine how much the car would deviate from driving straight on the one-lane road. We are restricting the angles from 0° to 180° from the steering wheel to range across the direction on the road. Thus, it is not possible for the car to reverse or completely turn around.



4. Image processing (Jing)

The images are processed in MATLAB to detect the colored rectangle of of the steering wheel and calculate what angle the wheel has been turned. This can be done by simply selecting the shape formed by pixels of the selected color and sending the value of the angle of the sides in degrees (rounded to the nearest integer) to the Verilog game logic.

5. Brakes and acceleration (Jing)

For the brake and acceleration, there will be two switches that the driver can step on to accelerate or decelerate the car. The change in speed is constant so long as the switches are pressed - if the driver steps on the brake, the car will decelerate at a constant rate (and vice versa for the accelerator) but continue moving at a constant speed if the switch is released. The output values from these switches (0 or 1 for on or off) will be sent into the Verilog game logic.

6. Game logic (Venita)

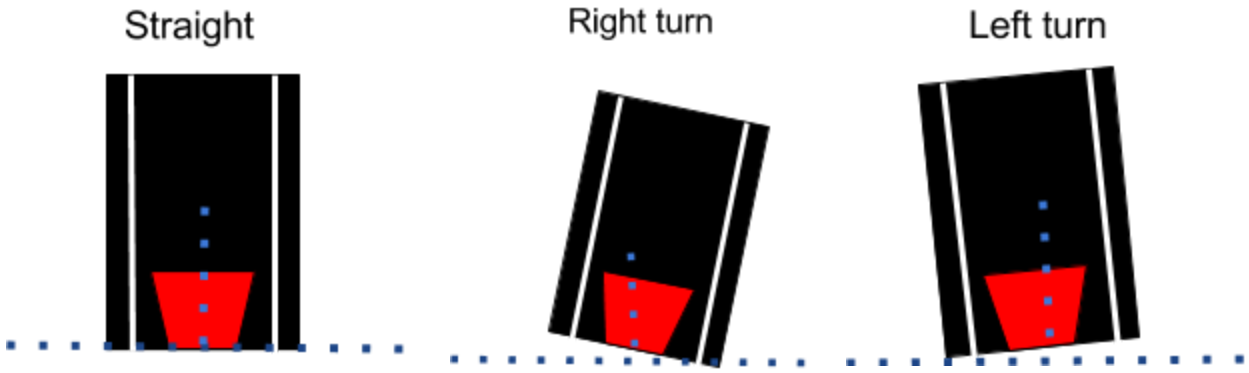
There are 4 inputs that will determine what it is displayed on the screen. These are the brake switch, accelerator switch, road map and steering wheel motion. The switches' values can be stored in registers. If the brake switch is on, then the car should gradually slow down on the

screen. If the accelerator switch is on, the car should speed up. Just as with a real car, one cannot press both the accelerator and brake at the same time so this is an invalid state for our system but we will have a default case of the car moving at a constant slow speed.

These states can be represented as summarized as:

Accelerator Switch	Brake Switch	Motion of car on screen
OFF	OFF	Constant slow speed
OFF	ON	Decelerating
ON	OFF	Accelerating

The road map would be defined by the user in a text file where the user can type 'L', 'R' or 'S' to indicate a road curved to the left, a road curved to the right or a straight road (see images below). Then, the steering wheel input would determine the car's direction: left, right or straight. A car is considered to have driven off the road if its direction does not correspond to a bend in the road. For example, if we consider the first image below, the road is straight and if the steering wheel turns right or left, then the car will crash off of the road.



The game logic module will then send to the graphics module 3 pieces of information: if the car has crashed or not, a map of the road and the speed of the car.

7. Graphics (Janie)

Inputs:

Crash (If through the game logic we've determined that we crashed)
Speed (to render trees coming from background to foreground).
What road angle should be displayed (S, L, or R)

Straight: / \

Left: | \

Right: / |

Outputs: Output values through the VGA to the computer monitor.

Beforehand, process images in MATLAB and store them in flash memory:

- Scenery
- Car view (dashboard)
- Trees

Also the road lines. Start off with three possible street configurations:

- Road is straight
- Road is angled left
- Road is angles right

8. Audio (Janie)

Input: Values denoting what sounds to play.
Output: Will output music to the speaker, after choosing from the recorded, stored sound bytes.

Using the microphone from lab 5, record sounds for:

- driving
- crashing
- beeping
- accelerating
- brakes

And store them in flash memory, cached.

9. Complexity

Most of the project will not require complex arithmetic with the exception of drawing the road, which will require multiplication to calculate the correct slope. Therefore, pipelining might be required for the module that draws the road. Within the game logic, reading the text file that encodes the road might also consume a large amount of time when it is first being loaded in. As described in earlier sections, flash memory will be used to store the graphics and sound effects for the game.

10. Testing

Initially the individual modules will be unit tested to ensure that they work independently:

- To test the brakes and accelerator, we can measuring the output of the switches with a multimeter.

- To test the steering wheel, we can stream the value of the angle that the wheel is turned on MATLAB or Python to see that the program correctly recognizes the angle that the wheel is turned.
- To test the game logic, we can simulate data from the brakes and acceleration by using switches on the FPGA, and the angle of the steering wheel by the up and down buttons on the labkit. This can also be done on Modelsim.
- To test the graphic outputs we can manually input data about the speed and angle of the car.
- To test the audio output, we can manually input data about the state of the game.

Once we ensure that the individual modules function, we will integrate them together one at a time and test that they work together as a whole.

11. Timeline

11/7-11/9: Project presentation

11/12: Implement modules with base level performance

11/19: Complete unit testing for individual modules

11/26: Initial integration and continued work on individual modules

12/03: Continued integration of modules

12/06: Debugging

12/11 Project checkoff

12. Stretch Goals

- Allowing the user to choose between a horse, bike or car
- Allowing the user to change a road while driving
- Fuel light indicator on the dashboard
- Speed limit warning indicator
- Reversing the vehicle
- Obstacles in the road