# Car Racing Game

<table>
<thead>
<tr>
<th>Name</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jing Shi</td>
<td>Sprite Display Modules and Driver</td>
</tr>
<tr>
<td>Mingxin Huo</td>
<td>Audio Modules and Driver</td>
</tr>
<tr>
<td>Yifan Li</td>
<td>Software for game logic</td>
</tr>
<tr>
<td>Siwei Su</td>
<td>Software for game logic</td>
</tr>
</tbody>
</table>
Sprite

7 Layers Merging Priority

Background Cloud Tree Golden Coin Car Number Speed

1 mph
**Submodule:** ROM

**Parameter:** HEIGHT, WIDTH

**Input:** hc, vc, clk, (X, Y)

**Output:** R, G, B, Valid

**Signals:** ROM address,

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEIGHT, WIDTH</td>
<td>Define the height and width of this Sprite Picture</td>
</tr>
<tr>
<td>hc, vc</td>
<td>Horizontal Count and Vertical Count, define current pixel location</td>
</tr>
<tr>
<td>(X, Y)</td>
<td>The coordinate of the Sprite, determines where to put the Sprite</td>
</tr>
<tr>
<td>R, G, B</td>
<td>The RGB value of the Sprite</td>
</tr>
<tr>
<td>Valid</td>
<td>Whether the VGA scan location (hv, cv) is within the Sprite region</td>
</tr>
<tr>
<td>ROM Address</td>
<td>Calculate the address of current pixel RGB value in ROM</td>
</tr>
</tbody>
</table>
Figure 6 The Background Sprite
1. How to debug in hardware
2. Need to handle the positive or negative signed of calculation
3. Top-to-Down Design Methodology
4. The importance of team work especially when design the interface of different modules (like the input/output ports, and timing), otherwise it would become a nightmare when try to assemble all the modules
Audio Files Preparation

Original Data ➔ Breaking and Reassembling ➔ Data Type Conversion ➔ MIF

Original acceleration sample data plot

Reassembled acceleration sample data plot

Modified acceleration unsigned integer type sample data (partial) plot
Audio Controller

Control Signal ➔ Fetch Audio Data ➔ SSM2603 Audio Codec ➔ Sound

.mif files ➔ Audio ROM Blocks ➔ Audio Effect Block ➔ Clock

Audio ROM Blocks ➔ address ➔ Audio Effect Block

Audio Effect Block ➔ data ➔ Audio Codec Interface

Configuration Parameters ➔ Audio Codec Configuration Interface

Audio Codec Configuration Interface ➔ parameters ➔ Audio Codec Interface

SSM2603 Audio Codec
Overview

* Motivated by car racing video games
* Using keyboard to control your car and trying your best to get first place in the game
* Play issues:
  1. Be careful not to hit other cars and road margin, otherwise you’ll get penalty
  2. Try your best to eat more coins as much as possible to get a speedup reward.
Movement issues

• By calculating vector AB and AC, we could easily control the location and movement of trees and clouds.

\[
ab(x, y) = \left( \frac{A(x) - B(x)}{\sqrt{(A(x) - B(x))^2 + (A(y) - B(y))^2}}, \frac{A(y) - B(y)}{\sqrt{(A(x) - B(x))^2 + (A(y) - B(y))^2}} \right)
\]

\[
ac(x, y) = \left( \frac{A(x) - C(x)}{\sqrt{(A(x) - C(x))^2 + (A(y) - C(y))^2}}, \frac{A(y) - C(y)}{\sqrt{(A(x) - C(x))^2 + (A(y) - C(y))^2}} \right)
\]
Pass Event

* Three variables to track how far each car travelled: car1_dist (player), car2_dist, car3_dist
* car1_dist =+ car1_speed
car2_dist =+ car2_speed
car3_dist =+ car3_speed
* Compare |car1_dist-car2_dist| and |car1_dist-car3_dist| with a set passing range
* If within passing range, there could be a pass event
* If the player’s car is falling behind and speeding up, it will pass the opponent car; if it is ahead and not speeding up, it will be passed by the opponent car.
Problems Encountered

* Unable to move the trees with road
* Missing libusb
* Keyboard function not working in main()
* Unable to make the car move continuously
* Unable to define reasonable pass range due to high frame rate
Solutions

* Set two vectors that are controlled by one point’s coordinates
* Move everything from lab3 to lab2
* Create a thread and put the whole keyboard function in it
* Change timeout variable in libusb_interrupt_transfer
* Divide the distance variables by an integer
Lessons Learned

* Software and hardware design
* Time management
* Teamwork
* Keyboard control