數電實驗期末專題
P2P賽車遊戲

第四組
B10901163張顥譽
B10901176蔡弘祥
B10901179鄭承瑞

Outline

1. Introduction
2. System Architecture
3. Hardware Design
4. Algorithm
5. Workflow
6. Problem Solved/ Lesson learned

INTRODUCTION

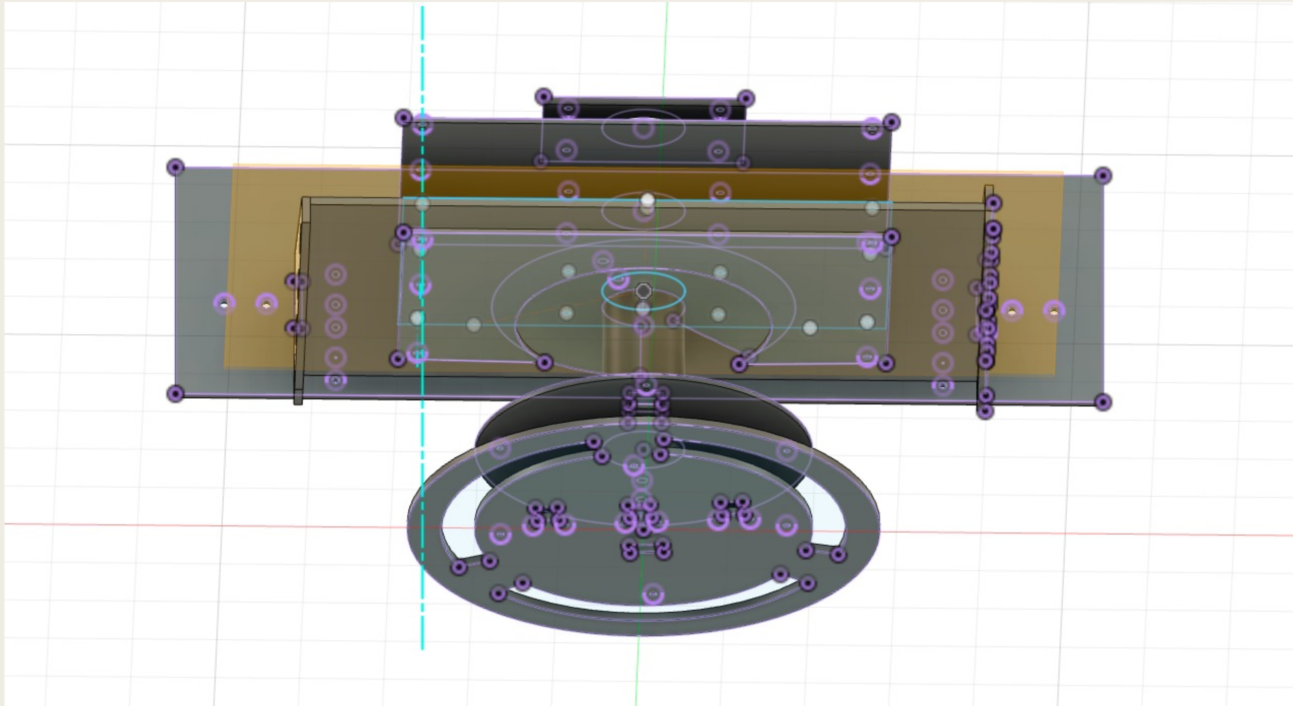
Intro

- We made a realistic P2P race game with FPGA
- Incorporate the wheel, paddle to make a great experience
- Shows the game on screen by VGA
- Motor shake the wheel when driving
- Implement the collision algorithm

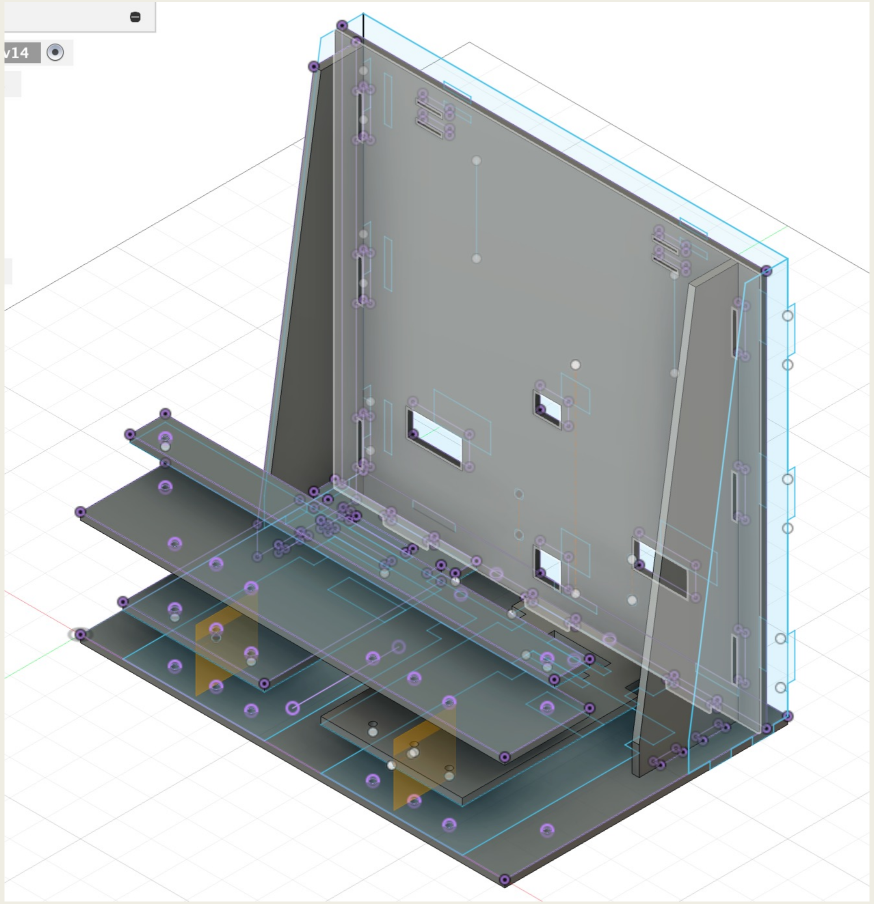
HARDWARE DESIGN

Wheel

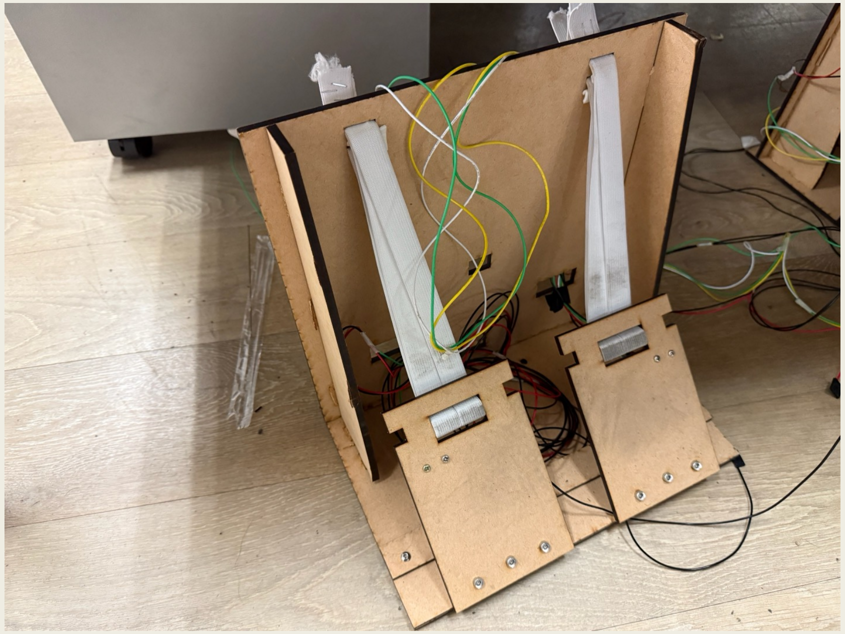
Using three layer to stabilize wheel
Using spring to have damping touching
A lot of work in assembly and adjust

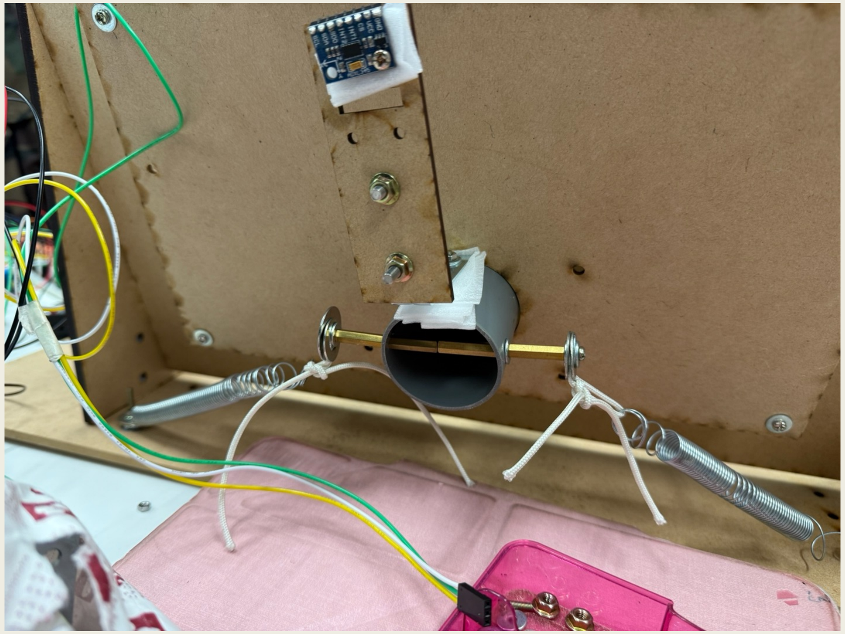


Paddle



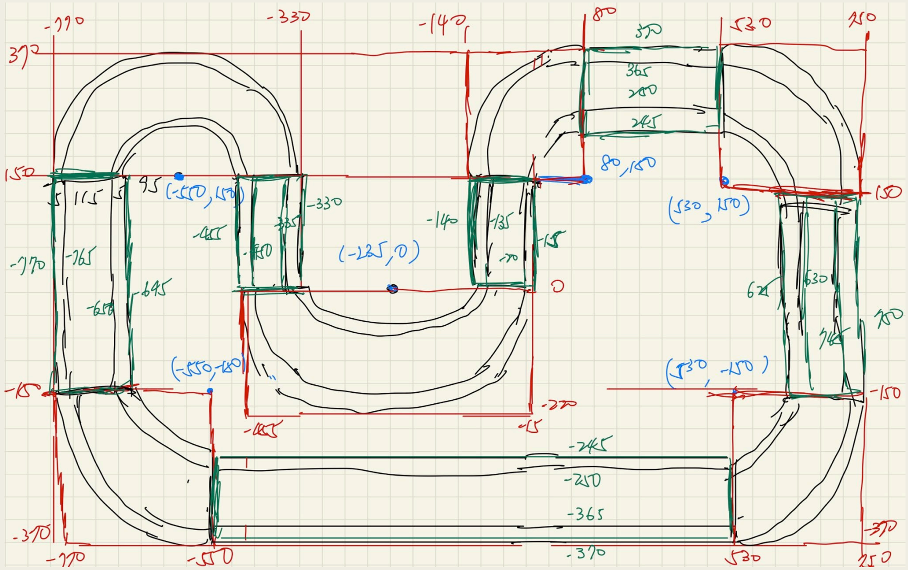
Force feedback by spring





Track Design

- Use specific function and bounding box to define the map
  - *Horizontal Line*
  - *Vertical Line*
  - *Circle*
- Different types of track
  - *Ordinary*
  - *Sand*
  - *Rock*



ARCHITECTURE

Image Preprocess
Color Compression

- Use python to convert 24-bit to 4-bit
  - *With alpha channel: convert to 15 colors, one is used to save transparent or not*
    - Car
  - *Without alpha channel: convert to 16 colors*
    - Map
    - Bar
    - Bar Digit
    - Start Caption
    - Win Caption
    - Lose Caption
    - Idle Background
    - QBlock

Image Preprocess
Data Storage

- Use python to convert ROM (LUT) or binary files (.bin)
  - *ROM (LUT): Verilog module*
    - Car
  - *Binary files (.bin): DE2-115 Control Panel pre-write*
    - Map
    - Bar
    - Bar Digit
    - Start Caption
    - Win Caption
    - Lose Caption
    - Idle Background
    - QBlock

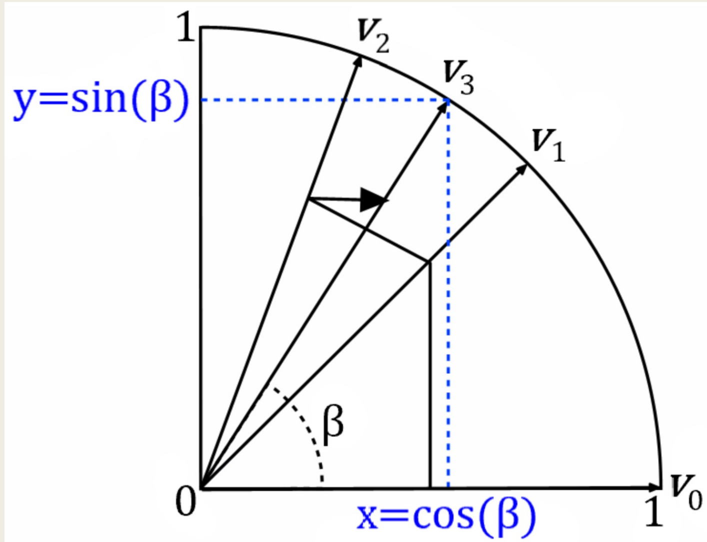
Sram

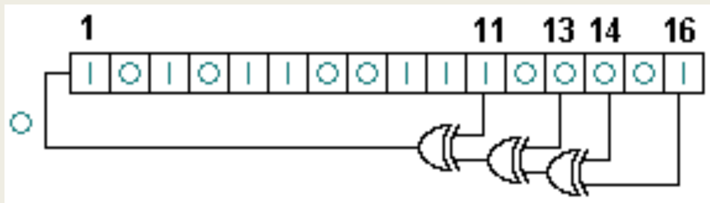
- Total 2^20 addresses, each address is composed of 16-bit data
- Each pixel is encoded into 4-bit, each address saves 4 pixels
- Decode them back to 24-bit when rendering
- Total used 1.6MB/2MB

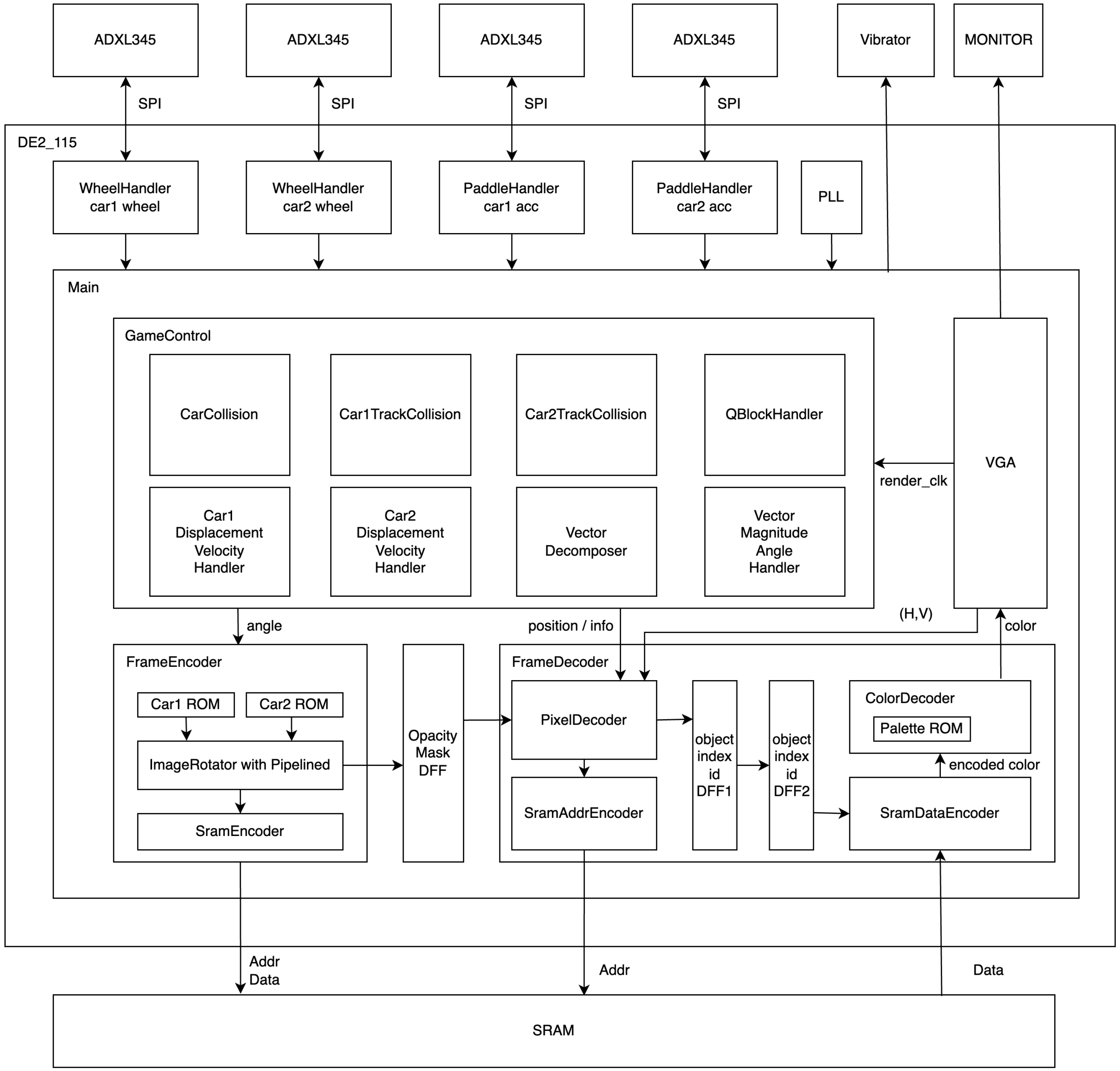
Object	# of pixels	# of address used
Map	1600*800	320000
Bar	1600*100	40000
Bar Digit	26*38*10	2470
Start Caption	664*56	9296
Win Caption	200*60	3000
Lose Caption	200*60	3000
Idle Background	1600*900	360000
QBlock	40*40*4	1600
Car	40*40*2	800

Math

- Vector Rotation / Angle / Magnitude
  - *Cordic Algorithm*
  - *Data scale up to increase accuracy*
  - *16 stages w/wo pipelined*
- Square root
  - *Support Fixed Point Number*
- Random
  - *LFSR from LAB1*

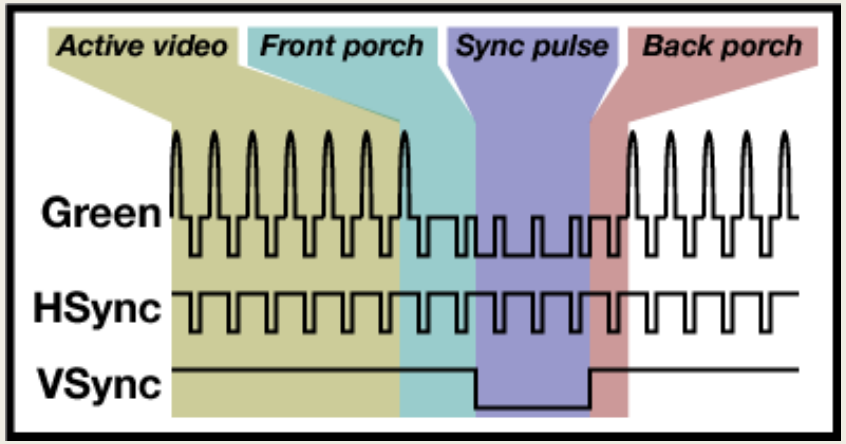





ADXL345
ADXL345
ADXL345
ADXL345
Vibrator
MONITOR
SPI
SPI
SPI
SPI
DE2_115
WheelHandler car1 wheel
WheelHandler car2 wheel
PaddleHandler car1 acc
PaddleHandler car2 acc
PLL
Main
GameControl
CarCollision
Car1TrackCollision
Car2TrackCollision
QBlockHandler
Car1 Displacement Velocity Handler
Car2 Displacement Velocity Handler
Vector Decomposer
Vector Magnitude Angle Handler
render_clk
VGA
(H,V)
color
angle
position / info
FrameEncoder
Car1 ROM
Car2 ROM
ImageRotator with Pipelined
SramEncoder
Opacity Mask DFF
FrameDecoder
PixelDecoder
SramAddrEncoder
object index id DFF1
object index id DFF2
ColorDecoder
Palette ROM
encoded color
SramDataEncoder
Addr
Data
Addr
Data
SRAM

VGA

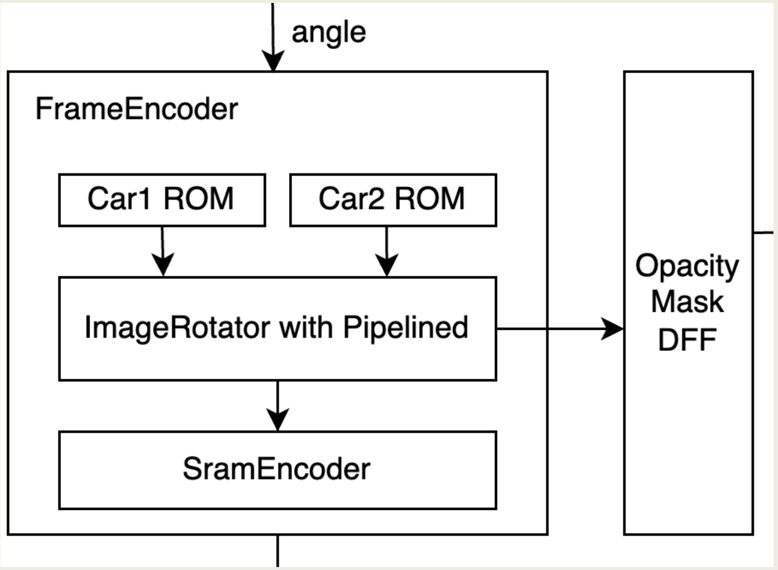
- Two modules to help rendering
  - *FrameEncoder*
    - Front porch state
    - Sync pulse state
    - Back porch state
  - *FrameDecoder*
    - Active state



Special Thanks: 石博允、UCB強者、渠立宇

FrameEncoder

- Handle car rotation and opacity mask
- For faster rendering
  - *Save the original car images in ROM*
  - *Use pipelined cordic to finish rotation before VGA active*
  - *Directly generate the opacity masks so that we only need to access Sram once when rendering*

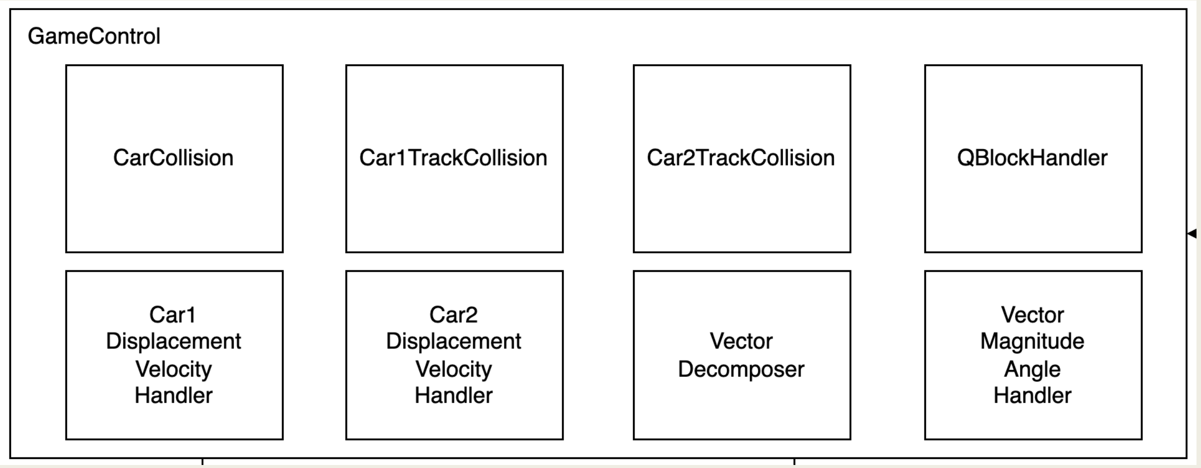


FrameDecoder

- PixelDecoder
  - *Determine which object to render and the pixel index*
- SramAddrEncoder
  - *Use object ID and pixel index to set Sram address*
- SramDataDecoder
  - *Use pixel index to find which 4-bit to read*
- ColorDecoder
  - *Use Palette ROM with object ID as input to decode the color back to 24 bits*

GameControl

- Car Status
  - *Position*
  - *Velocity Magnitude*
  - *Car Angle*
  - *Lap*
  - *Mass Level*
- Two Cars Collision
- Car/Track Collision
- Qblock



GameControl
Two Cars Collision

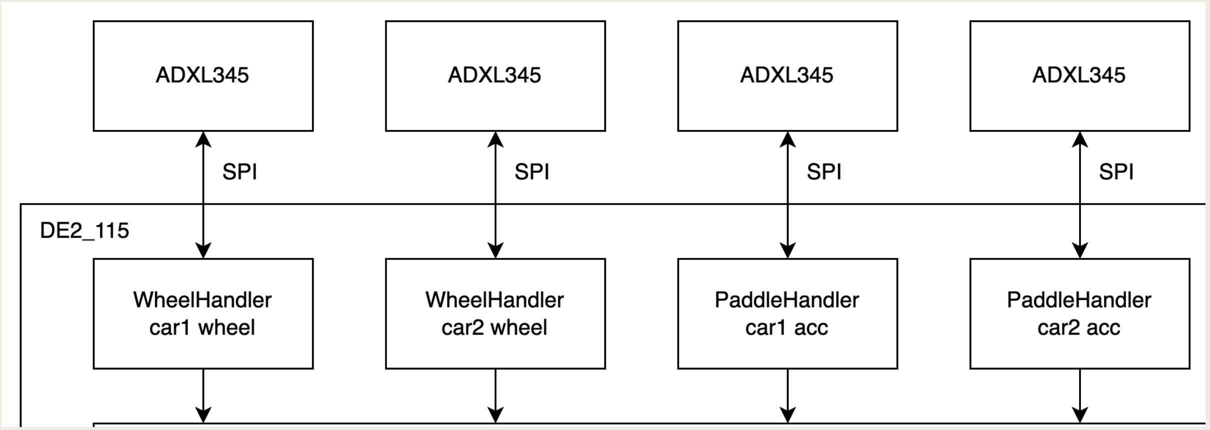
- Elastic Collision
- Decompose the velocity into x, y components
- Calculate the velocity after collision
- Calculate the final angle (arctan) and magnitude (square root)

$$\mathbf{v}'_1 = \mathbf{v}_1 - \frac{2m_2}{m_1 + m_2} \frac{\langle \mathbf{v}_1 - \mathbf{v}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2} (\mathbf{x}_1 - \mathbf{x}_2)$$

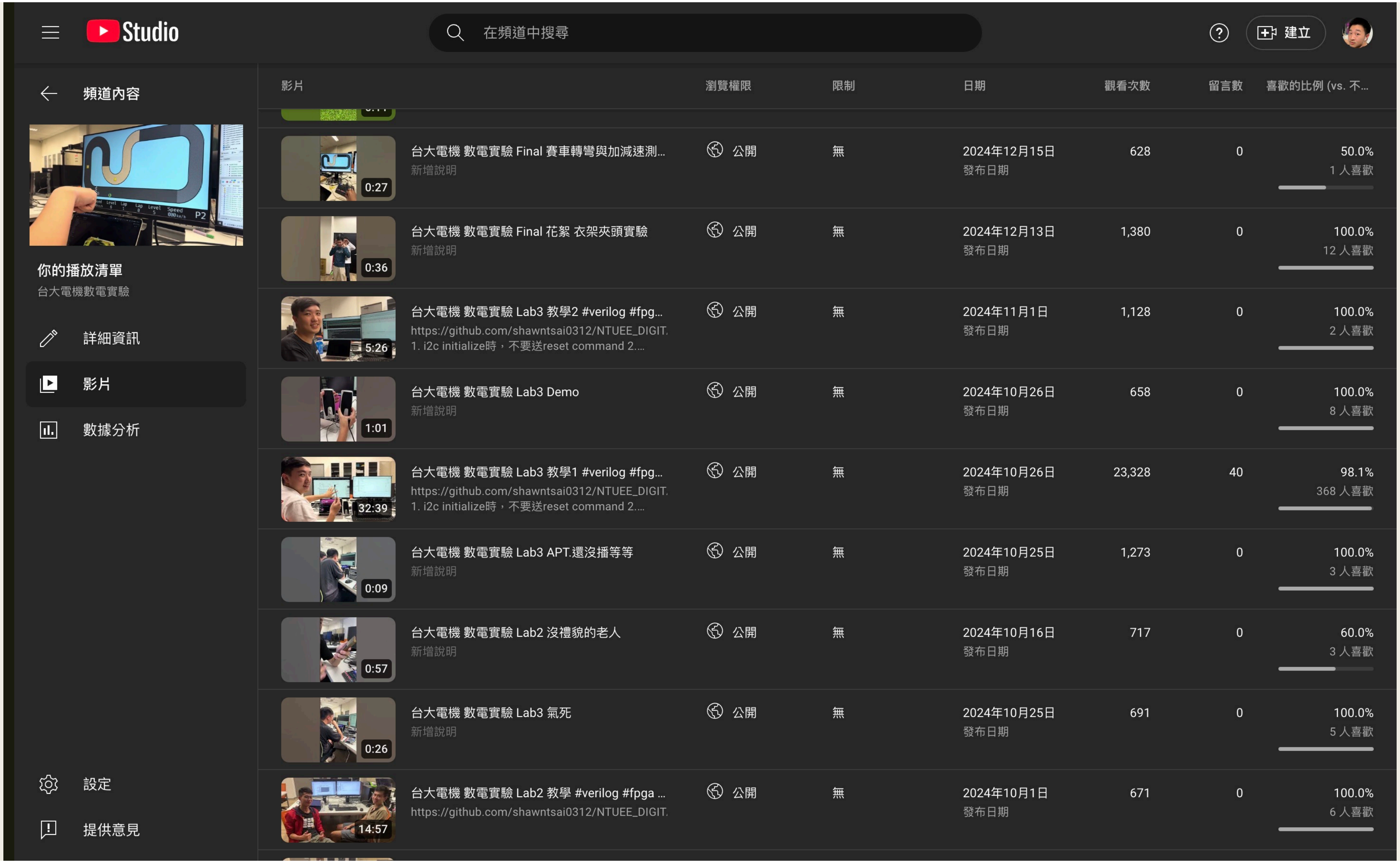
$$\mathbf{v}'_2 = \mathbf{v}_2 - \frac{2m_1}{m_1 + m_2} \frac{\langle \mathbf{v}_2 - \mathbf{v}_1, \mathbf{x}_2 - \mathbf{x}_1 \rangle}{\|\mathbf{x}_2 - \mathbf{x}_1\|^2} (\mathbf{x}_2 - \mathbf{x}_1)$$

ADXL345

- SPI 4 wire
- 100kHz clock
- 10-bit precision
- Read gx, gy and convert to the corresponding rotation/acceleration



Special Thanks: 石博允、UCB強者、渠立宇



Studio

在頻道中搜尋

建立

頻道內容

你的播放清單

台大電機數電實驗

詳細資訊

影片

數據分析

設定

提供意見

影片	瀏覽權限	限制	日期	觀看次數	留言數	喜歡的比例 (vs. 不...
0:27 台大電機 數電實驗 Final 賽車轉彎與加減速測... 新增說明	公開	無	2024年12月15日 發布日期	628	0	50.0% 1 人喜歡
0:36 台大電機 數電實驗 Final 花絮 衣架夾頭實驗 新增說明	公開	無	2024年12月13日 發布日期	1,380	0	100.0% 12 人喜歡
5:26 台大電機 數電實驗 Lab3 教學2 #verilog #fpg... https://github.com/shawntsai0312/NTUEE_DIGIT. 1. i2c initialize時，不要送reset command 2....	公開	無	2024年11月1日 發布日期	1,128	0	100.0% 2 人喜歡
1:01 台大電機 數電實驗 Lab3 Demo 新增說明	公開	無	2024年10月26日 發布日期	658	0	100.0% 8 人喜歡
32:39 台大電機 數電實驗 Lab3 教學1 #verilog #fpg... https://github.com/shawntsai0312/NTUEE_DIGIT. 1. i2c initialize時，不要送reset command 2....	公開	無	2024年10月26日 發布日期	23,328	40	98.1% 368 人喜歡
0:09 台大電機 數電實驗 Lab3 APT.還沒播等等 新增說明	公開	無	2024年10月25日 發布日期	1,273	0	100.0% 3 人喜歡
0:57 台大電機 數電實驗 Lab2 沒禮貌的老人 新增說明	公開	無	2024年10月16日 發布日期	717	0	60.0% 3 人喜歡
0:26 台大電機 數電實驗 Lab3 氣死 新增說明	公開	無	2024年10月25日 發布日期	691	0	100.0% 5 人喜歡
14:57 台大電機 數電實驗 Lab2 教學 #verilog #fpga ... https://github.com/shawntsai0312/NTUEE_DIGIT.	公開	無	2024年10月1日 發布日期	671	0	100.0% 6 人喜歡


1482低能兒
@1482dinenger · 806位訂閱者 · 596 部影片
按讚訂閱分享開啟小鈴鐺 ...顯示更多
自訂頻道
管理影片