Live-Action RC Mario Kart™

Bradley Gross | Jonathan Matthews | Nathaniel Rodman

6.111 Final Project Presentation
Mario Kart 64

- Go-kart racing video game
- Four players, multiple character options
- Power-ups on race track
Bringing the Game to Live-Action

● Project track on platform

● Players race RC cars

● Camera detects car position

● FPGA handles logic, control of cars, items...
Modules

- Image Processing
- Control Mapping
- Physics Logic
- Video Logic
- Audio Logic
- Audio DAC
- Image ROMs
- Sound ROMs
- RC Transmitter
- Driver Module
- Memory Blocks
Physical Setup

- Sony VX-AW15 Projector
- Lab-supplied NTSC camera
  *IR filter*
- N64 controller
  *Gut old N64 for ports*
- “Coke Can” Mini RC Cars
  *IR LEDs*
- Computer speakers
Physical Setup

Diagram showing various components and their connections:
- Control Mapping
- Image ROMs
- Sound ROMs
- Audio DAC
- Memory Blocks
- Image Processing
- Game Logic
- Physics Logic
- Video Logic
- Audio Logic
- VGA Module
- RC Transmitter
- Driver Module
- Information Image ROMs
Driving Cars

- N64 controller interface module

- Physics logic decides commands
  Is car on “grass”? slow : normal
  Is car boosting? fast : normal

- Driver module translates to PWM
  *Mini RC cars are binary speed & turning*

- RC controller + transistor array
  sends signals to car
Tracking Cars

- Cars outfitted with IR LEDs
- Camera feed next to projector
  - *IR bandpass filter*
- Image processing locates \([x,y]\) of cars
  - *Center-of-mass algorithm*
  - *Neighborhood of prev. car location*
- Calibrated offset
  - *Camera coordinates → world coordinates*
Displaying Video

Video logic prepares pixels
Which objects are present?
Depth ordering

Pixels loaded from memory

* Nexys 4 has 4.8 Mbits BRAM
* 1 scene images > 3Mbits
* 2+ scenes, need > 25MHz random access
* SD card loader into BRAM for each scene

VGA 640x480 output
Displaying Video
Playing Audio

Audio logic prepares samples

*Background music*
*Item noises*

Samples loaded from memory

*Same issue as video*
*BRAM not enough for video + audio*
*Audio ~6KHz sampling, random access*
*SD loader to cellular RAM, power-on*

Reconstruction filter → speakers
Game Management

Game phase FSM

Start screen, character select, racing

Race state FSM

Car \([x,y]\), power-ups, race time, etc.

Updated by other modules

Menu nav. & race actions from controllers
Car positions from camera
Item updates from physics
Start Your Engines