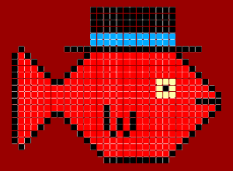

DTU

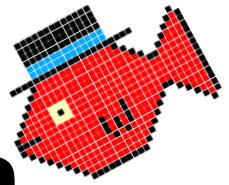


02113

Digital Systems Design Project

Luca Pezzarossa
Technical University of Denmark
DTU Compute

General information



Course information

- The course runs every day during the 3-week period from 9:00 to 17:00 in Building 324 - Room 020.
- The room is reserved from 8:00, so you are free to come earlier to work on your project if you want.
- You will use your own Basys3 FPGA board. We will provide monitors (to show your game) and any needed additional material.
- If you do not have a board, you can borrow one from us or you can buy one at the DTU Bookshop (in Building 101).

Course information

- We will have and use a Discord server (https://discord.gg/pkjUqGjRGh)

- The Discord server is asking questions, sharing ideas, and collaborating with your peers
- The Discord server will be also be actively monitored by the teachers to provide assistance
- The full material for the course is available in the Content section in DTU-Learn

About DTU-Learn

- We rely on **messages/announcements** to communicate with the classroom
- We recommend to **activate notifications** (mail-based or mobile based) in DTU-Learn.
- You can do it by clicking on your Profile Name -> Notifications -> Enable instant notifications.

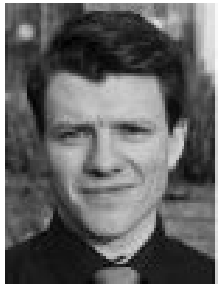
Course Information

- Teacher and course responsible
  - Luca Pezzarossa - lpez@dtu.dk

- Other teachers (helping now and then)

Oliver Keszöcze

Tjark Petersen



Emad Jacob Maroun

A Bit About Myself

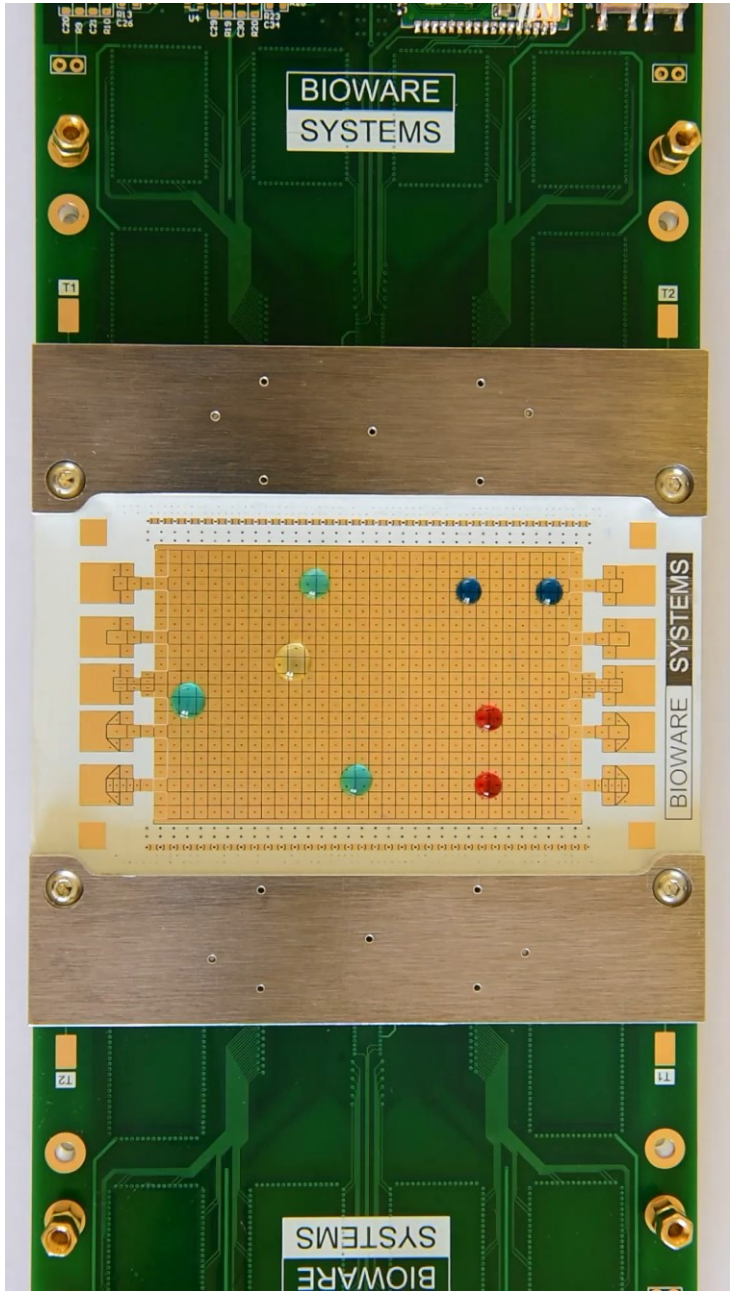
- Teacher and course responsible
  - Luca Pezzarossa - lpez@dtu.dk

- Italian, 35 year old
- BSc and MSc in Electronic Engineering from the *Polytechnic University of Marche*, Ancona, Italy
- PhD in Reconfigurable real-time systems from DTU
- PostDoc at DTU working with digital microfluidics biochips
- Associate professor in Computer Architecture

A Bit About Myself

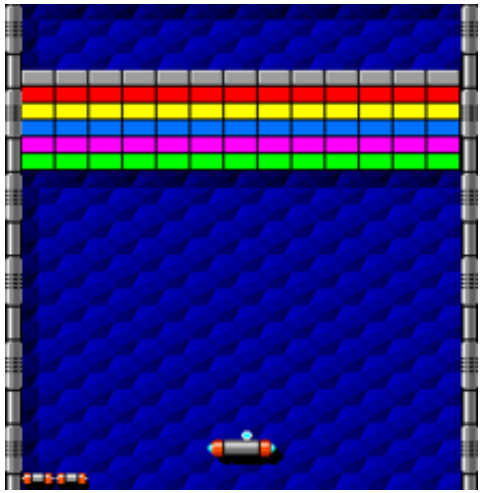
- Teacher and course responsible
  - Luca Pezzarossa - lpez@dtu.dk

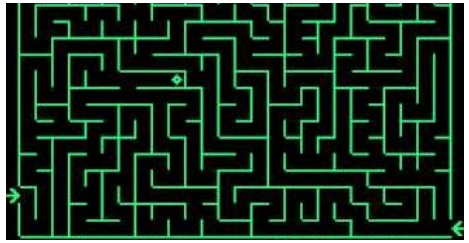
- Italian, 35 year old
- BSc and MSc in Electronic Engineering from the *Polytechnic University of Marche*, Ancona, Italy
- PhD in Reconfigurable real-time systems from DTU
- PostDoc at DTU working with digital microfluidics biochips
- Associate professor in Computer Architecture

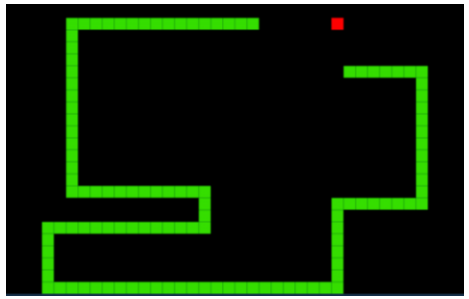


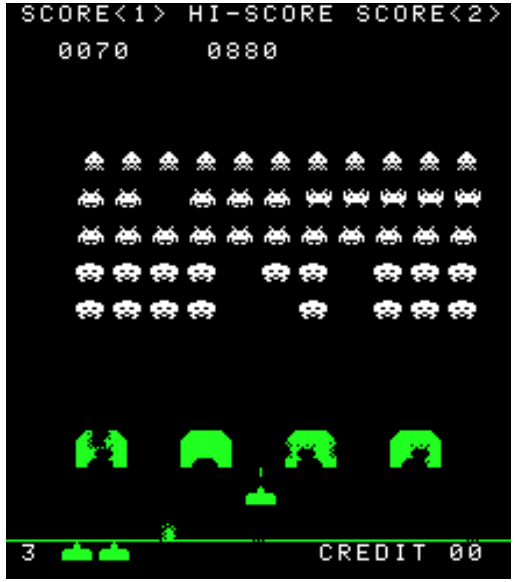
Course Objective

- You will create a 2D arcade-style game entirely **in hardware** using the Chisel hardware description language.









Course Objective

- Design and implement game logic for a 2D arcade game
- Use provided graphical engine and VGA display logic
- Implement on Basys3 FPGA board with VGA output
- Main inputs:
  - 5 buttons on the Basys3 board
  - switches and LEDs for debugging

Course Structure

- 3-week hands-on experience
- Focus on practical hardware design
- Encourages creativity and resource management
- Read the course manual thoroughly before starting
  - **Course manual**

Group Forming

- Groups of 3 people (groups of 2 are possible but less preferred)
- Select your own group members
- Register groups on DTU-Learn
- Align expectations among group members

Presence, Schedule, and Review Meetings

- Attend course hours in the classroom
- Equipment cannot be moved from the classroom
- Three mandatory review meetings:
  - Initial, Midway, Final
  - Prepare to present progress and get feedback

Presence, Schedule, and Review Meetings

	Thu. 6	Fri. 7	Mon. 10	Tue. 11	Wed. 12	Thu. 13	Fri. 14	Mon. 17	Tue. 18	Wed. 19	Thu. 20	Fri. 21	Mon. 24	Tue. 25	Wed. 26
08.00					Initial review meeting				Midway review						
09.00	Introductory lecture			Main project work					meeting						Final review meeting
10.00															(exam)
11.00															
12.00	Lunch break														
13.00															
14.00	Installing tools and														
15.00	working on the learning														
16.00	tasks														
17.00															

- Lectures
- Installing tools and working on the learning tasks
- Main project work
- Review meetings
- Lunch break

Access to Help

- Teachers available in the classroom every day
  - Most of the time? If possible ;-)
- Daily question slots
- Online discussion forum (Discord) monitored by teachers
- Seek help whenever needed
- Note: I will be away for the F-SiC conference on the 19 – 21 June

Course Deliverables

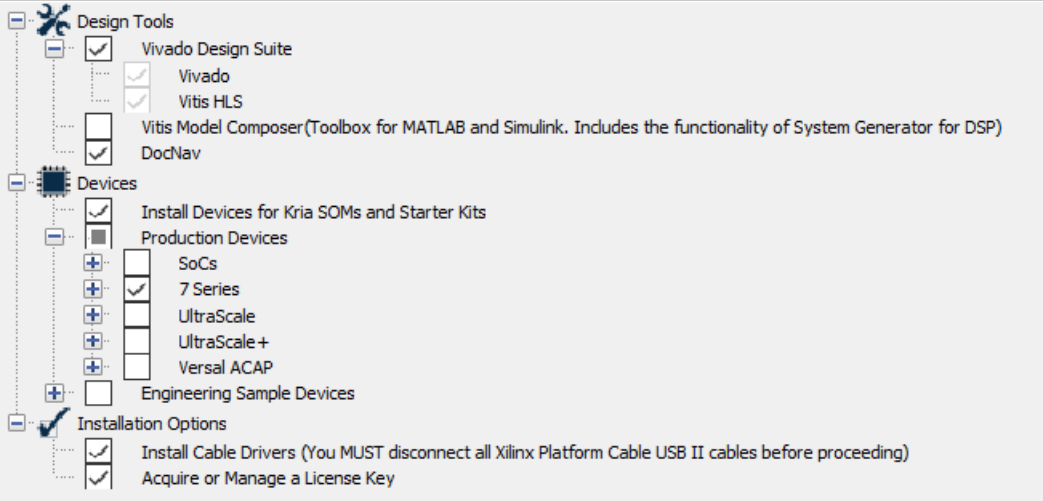
- **Short Report:** Approach, design decisions, challenges, solutions
- **Source Files:** Chisel code, well-documented and organized
- **Demonstration Video:** Showcasing the game on Basys3 board
- **README File:** Instructions for setup and running the game

Tool Installation

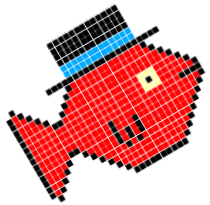
- Required tools: Chisel, GTK waves, Vivado 20231
  - Windows, Linux, and macOS (no Vivado in macOS)

- Installation guides provided for each tool

- Known issues and
- solutions included



Technical content



Course Objectives

- Hands-on approach to hardware design
- Develop a 2D arcade-style game
- Use graphical engine and VGA display logic
- Focus on FSMs and datapaths

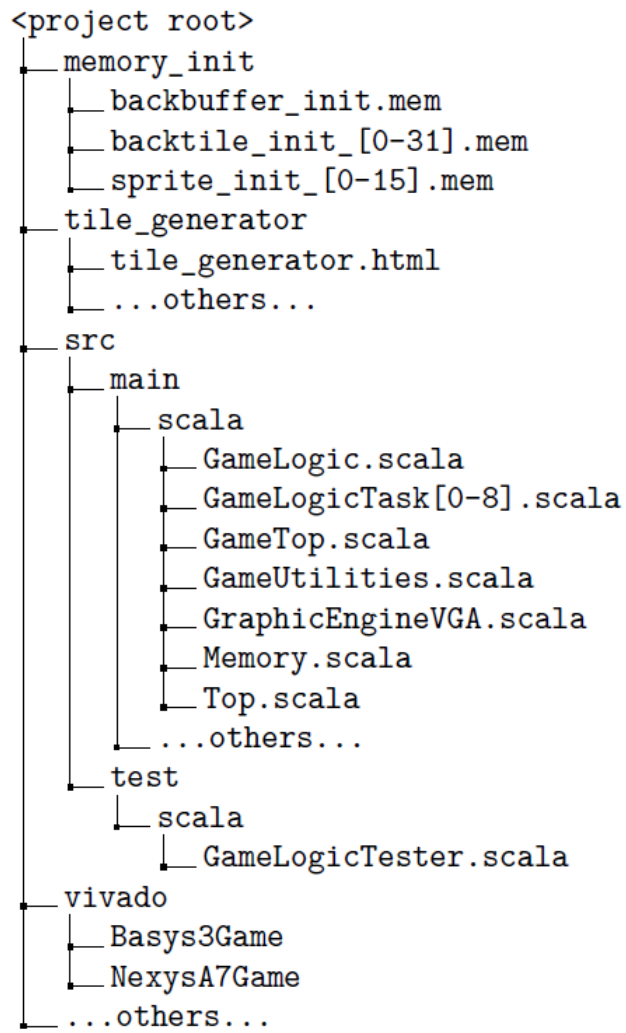
Getting the Code

- Provided as a ZIP archive on DTU-Learn
- Location:

  `DTU-Learn/Course content/Content/Code`

- Unzip and explore the folder structure

Code Structure Overview



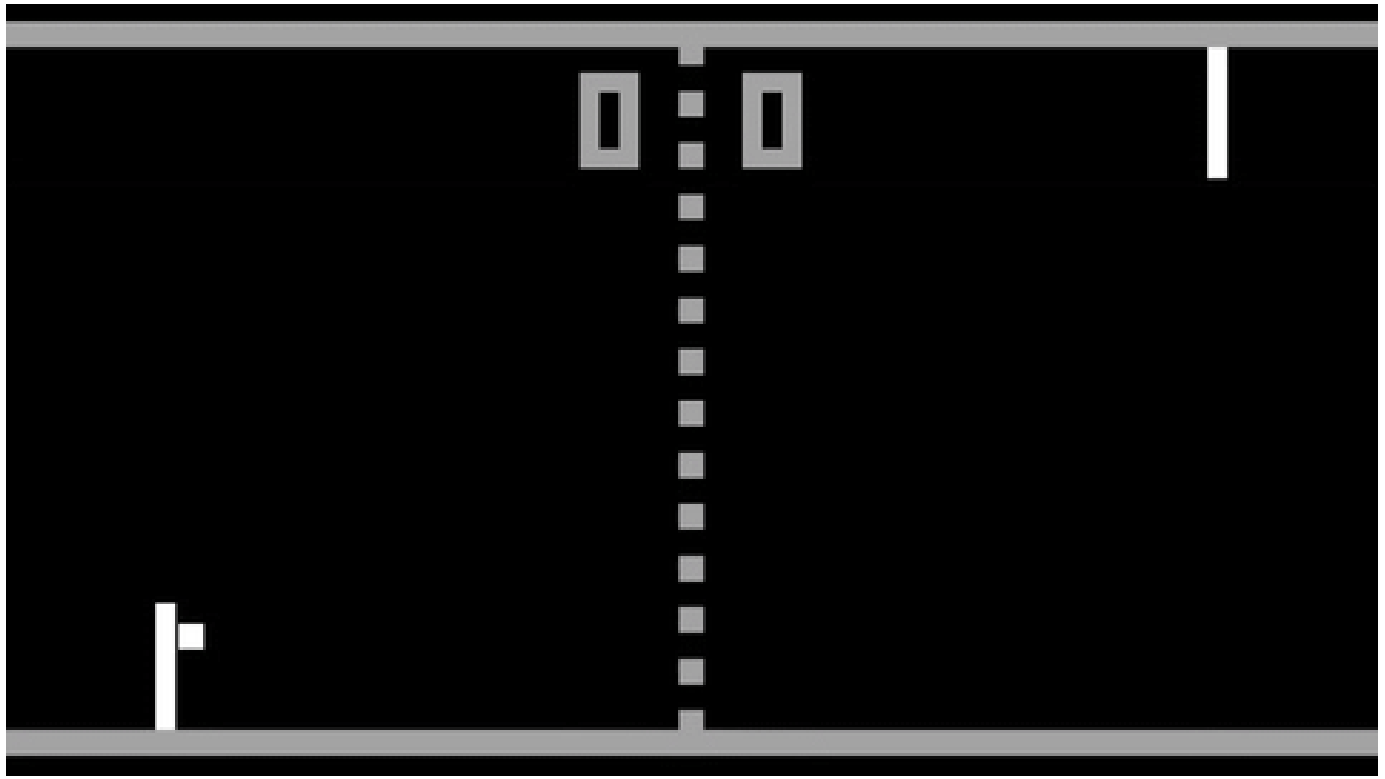
The Graphic Engine

- Manages sprites and backgrounds
- Controls sprite positions, visibility, and flipping
- Larger-than-screen background for scrolling
- Use ROM initialization files for sprites and background tiles

The Graphic Engine

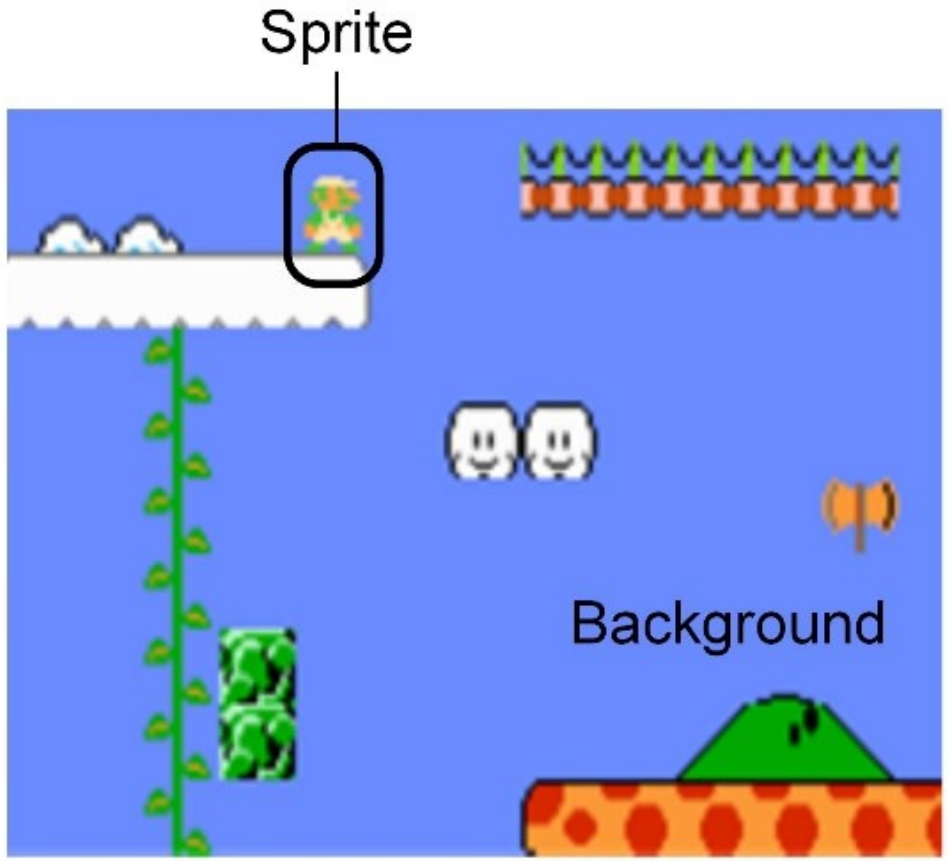
- Manages sprites and backgrounds
- Controls sprite positions, visibility, and flipping
- Larger-than-screen background for scrolling
- Use ROM initialization files for sprites and background tiles

Sprites and Background

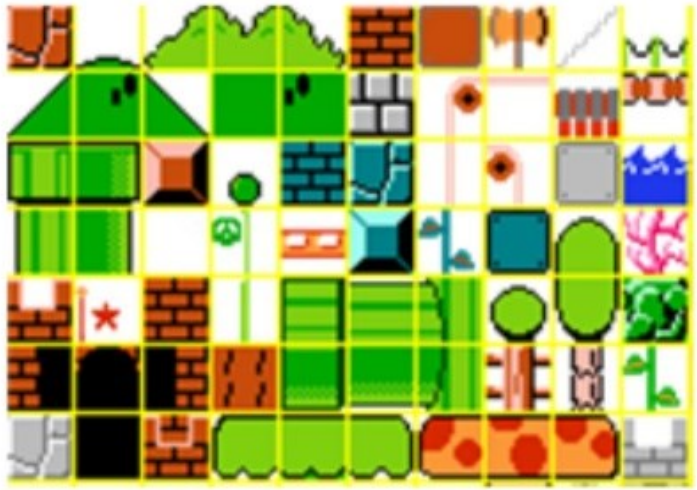


Sprites and Background

Sprites and Background



Background tiles



Sprite tiles

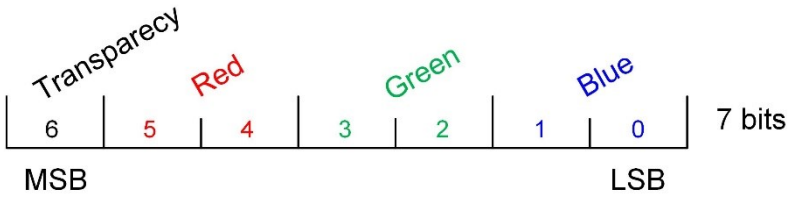
The Graphic Engine

- Manages sprites and backgrounds
- Controls sprite positions, visibility, and flipping
- Larger-than-screen background for scrolling
- Use ROM initialization files for sprites and background tiles

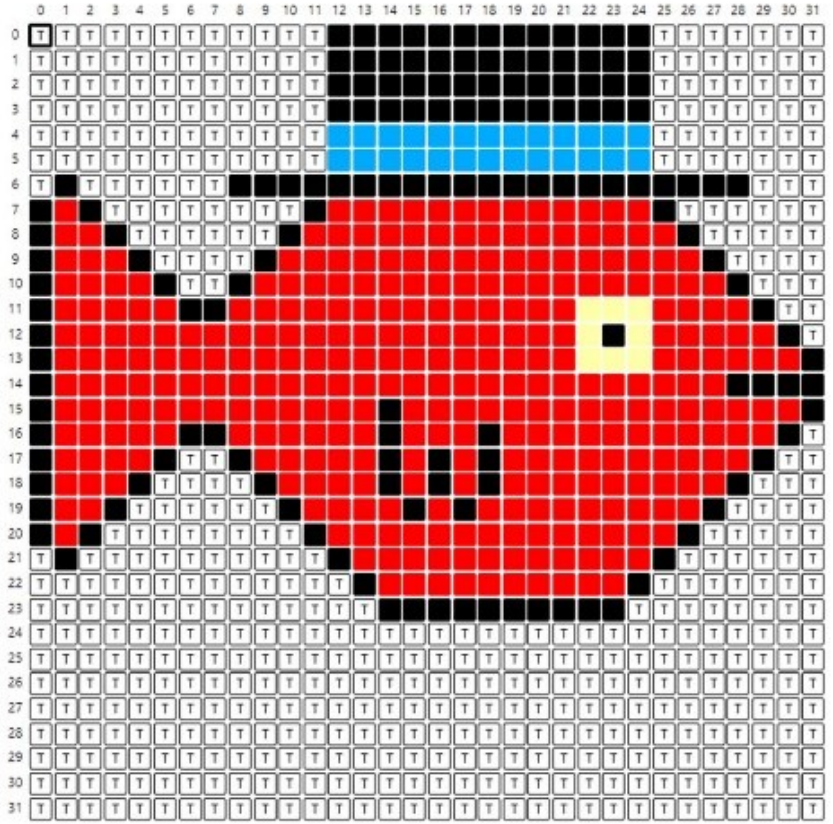
The Graphic Engine

- Manages sprites and backgrounds
- Controls sprite positions, visibility, and flipping
- Larger-than-screen background for scrolling
- Use ROM initialization files for sprites and background tiles

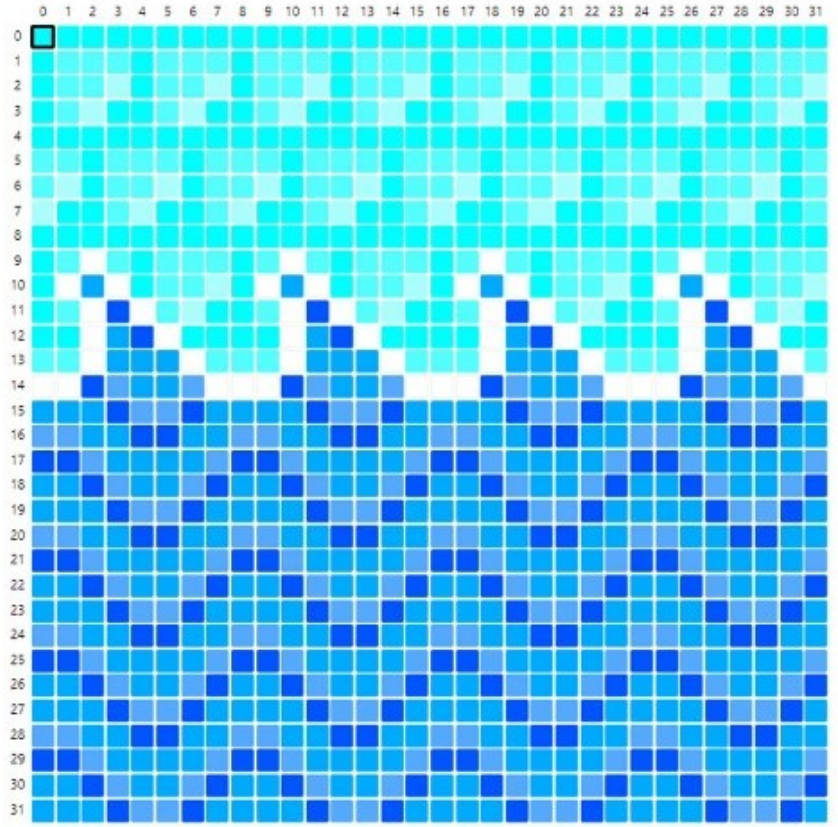
Tiles and Pixel Format



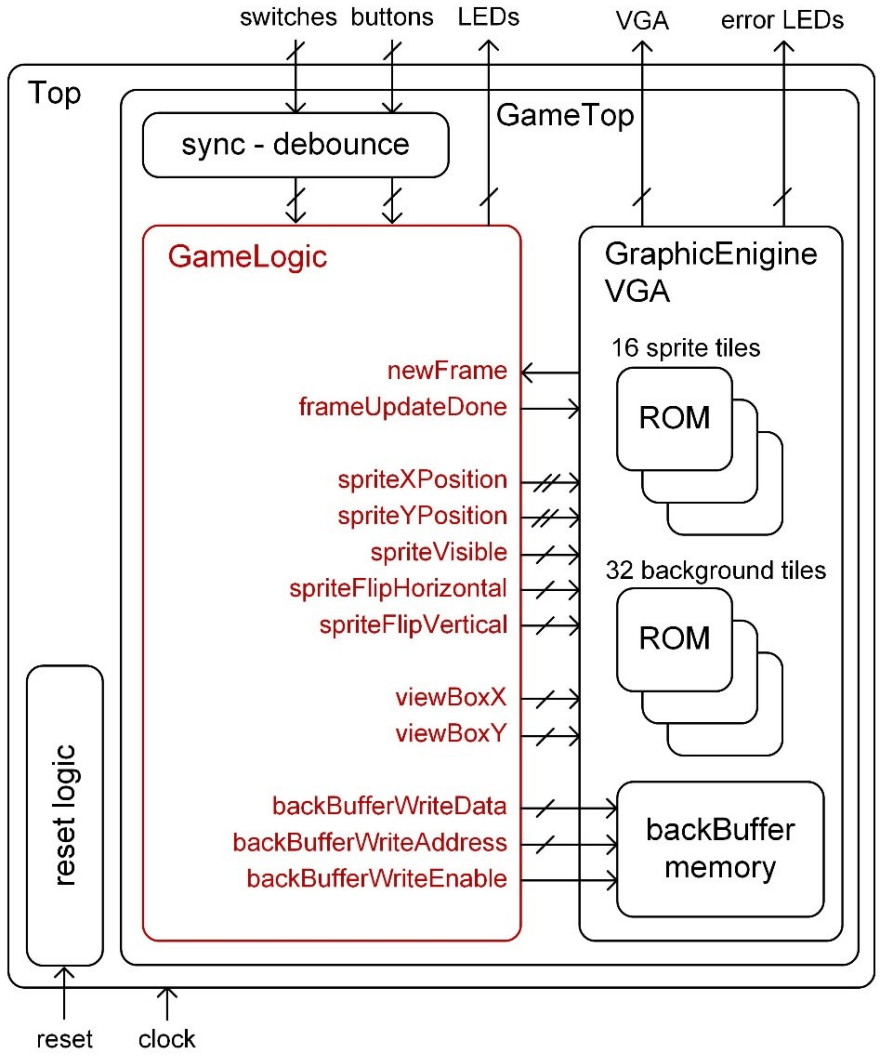
Sprite tile



Background tile

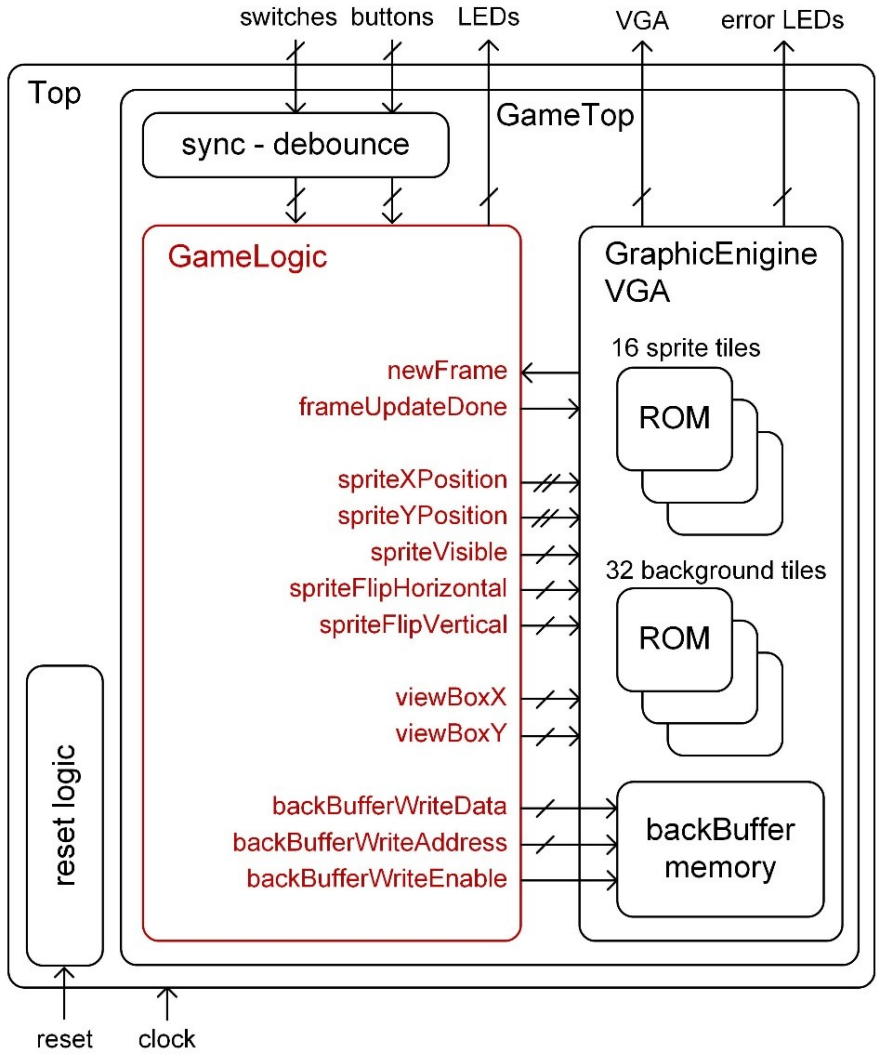


System Architecture



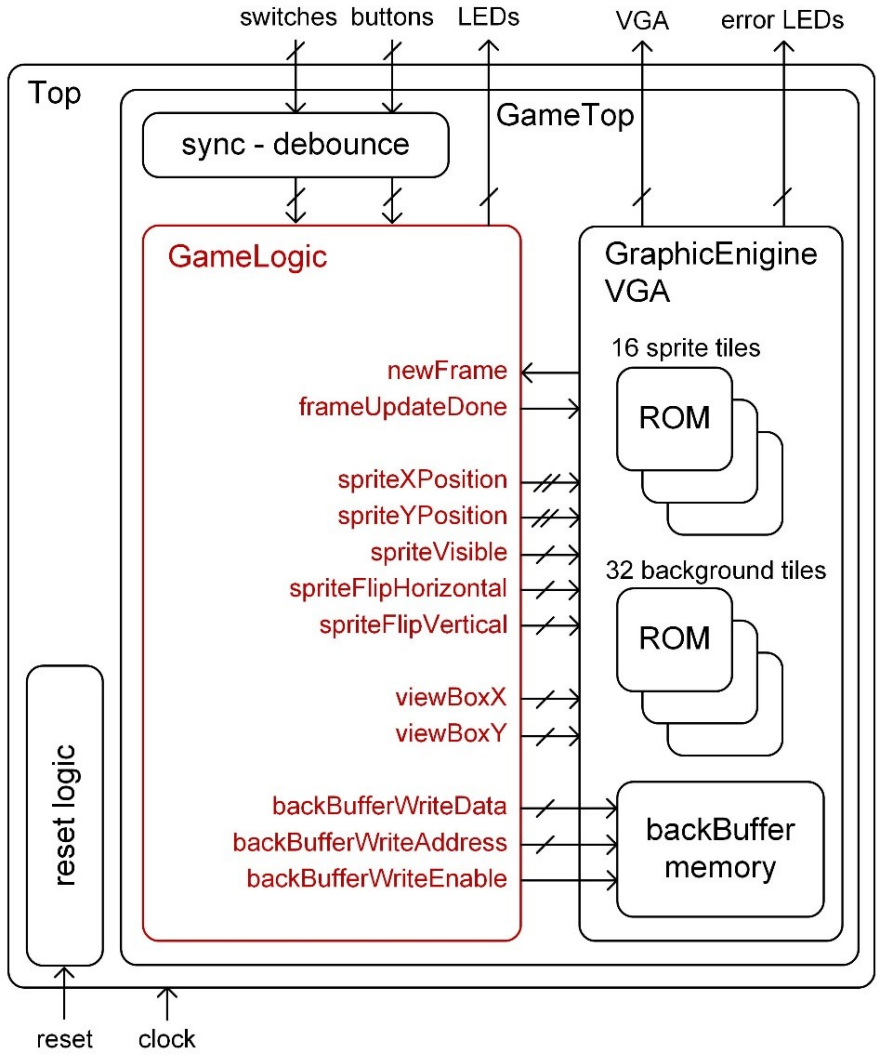
- **Top:** Reset logic
- **GameTop:** Handle inputs/outputs debouncing and synchronization
- **GraphicEngineVGA:** Manages sprite and background rendering
- **GameLogic:** Implement your custom game logic

Interface of the GameLogic Module

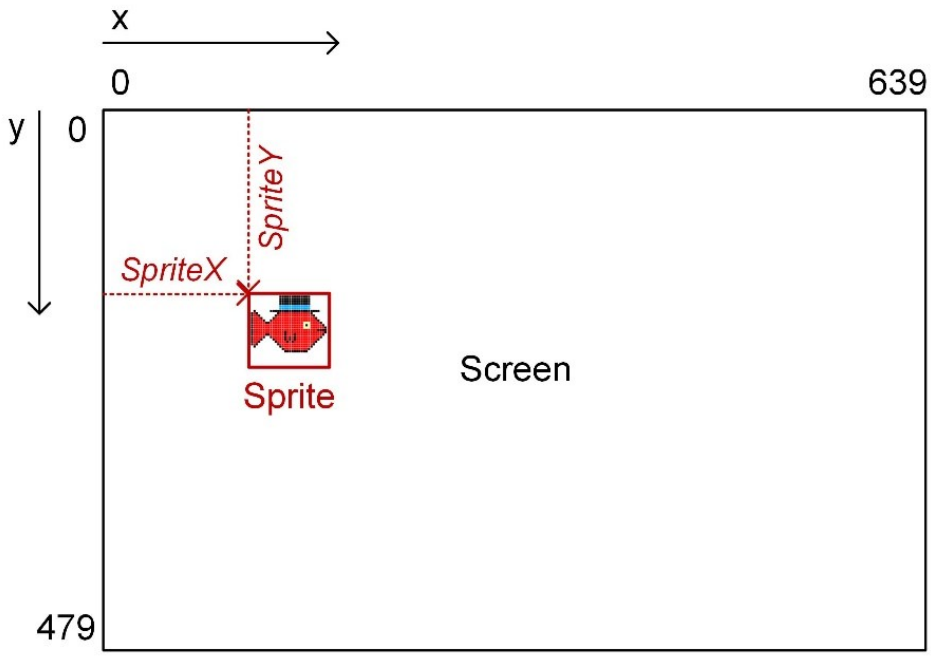


- **Buttons:** 5 Bool inputs (btnC, btnU, btnL, btnR, btnD)
- **Switches:** 8-element Vec of Bool inputs
- **LEDs:** 8-element Vec of Bool outputs

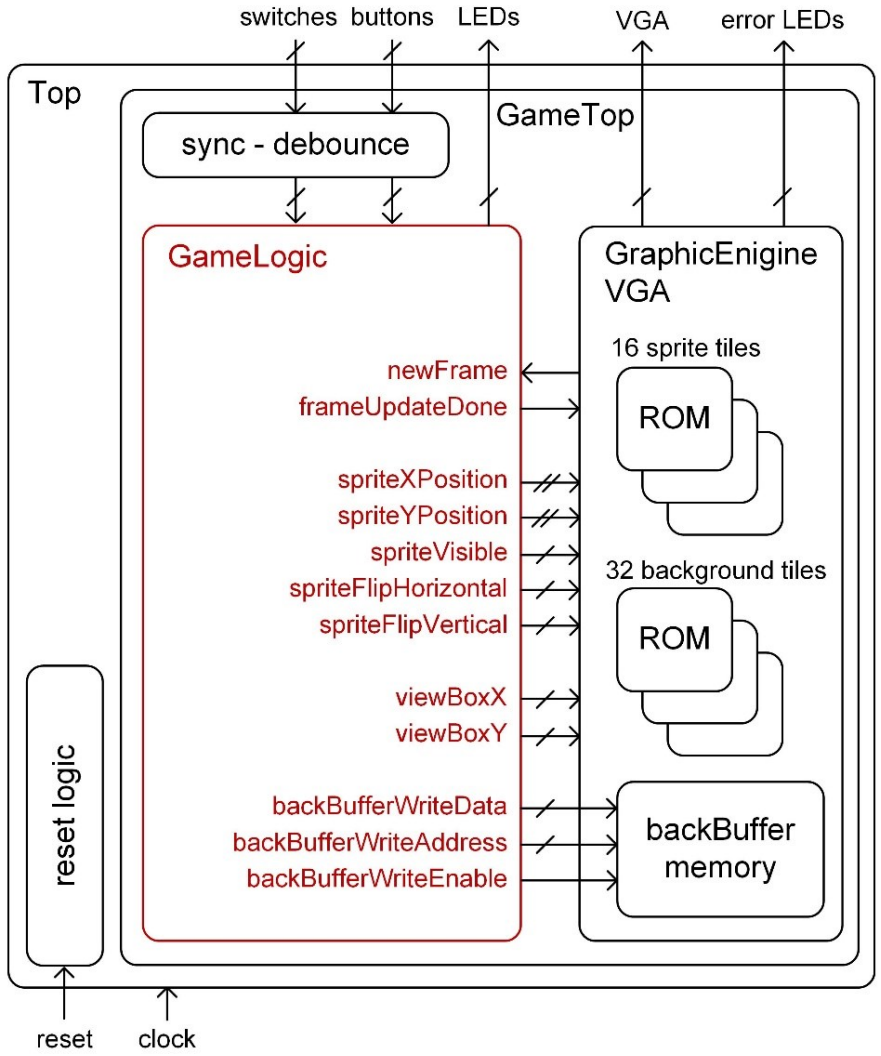
Interface of the GameLogic Module



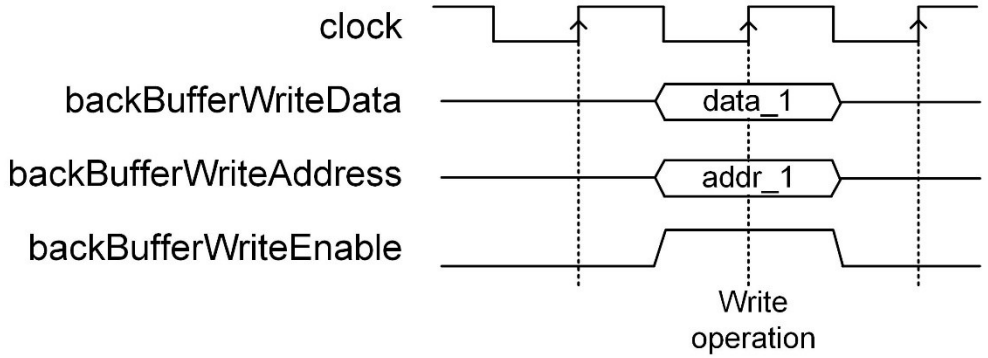
- **Sprite Control:** 16 of
  - X / Y positions (SInt)
  - Visibility (Bool)
  - H / V Flipping (Bool)



Interface of the GameLogic Module



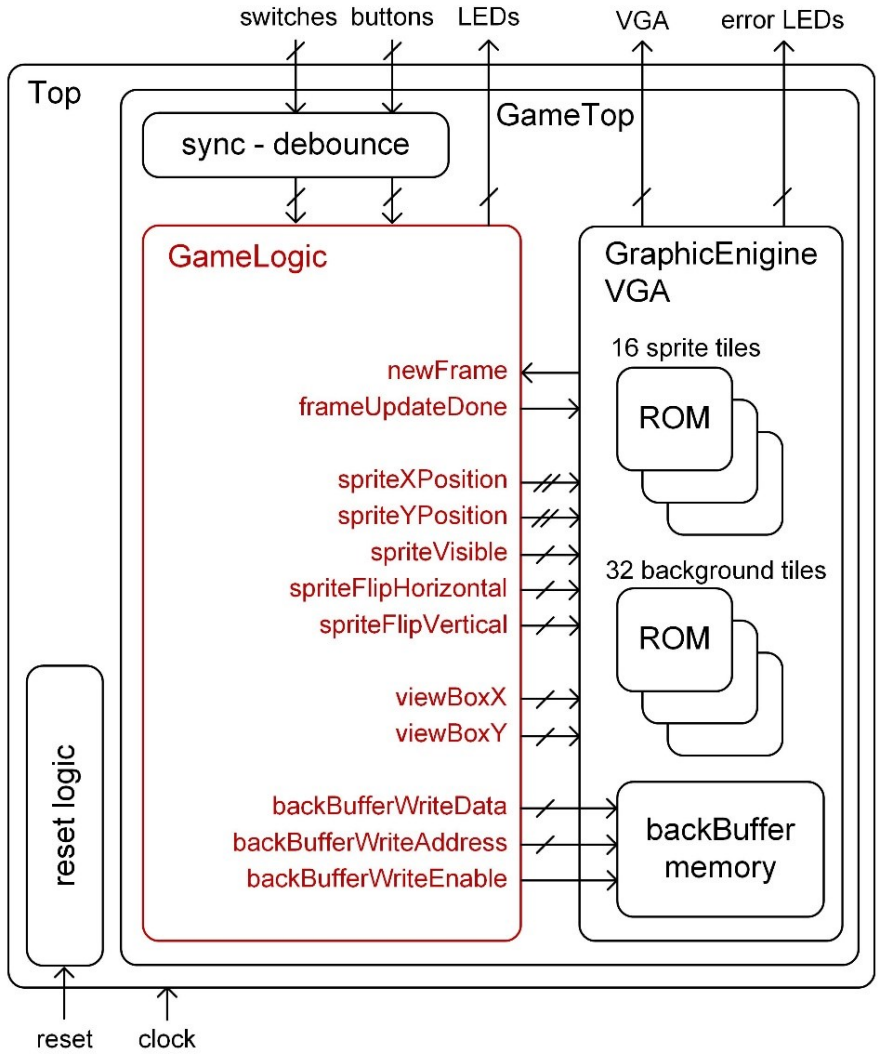
- **Viewbox Control:**
  - X/Y coordinates of the viewbox (UInt)
- **Background Buffer:**
  - write data
  - address
  - enable



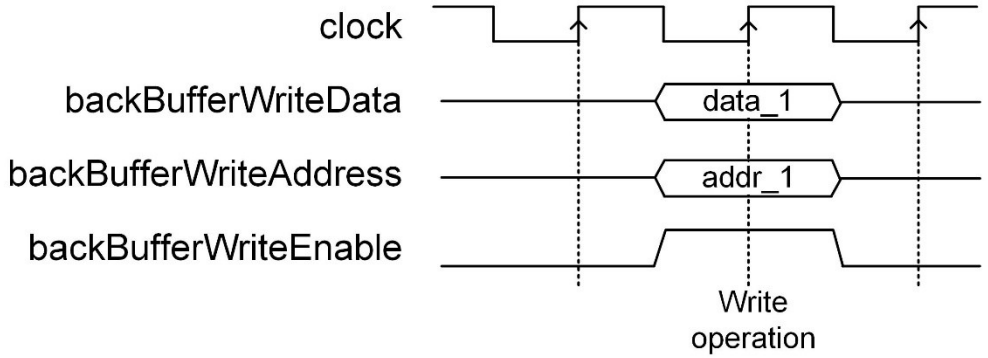
Background and Viewbox

- Background is larger than the screen
- Viewbox is 640x480 pixels
- Background consists of 40x30 tiles (each specified by a memory address)
- Viewbox can scroll for dynamic backgrounds

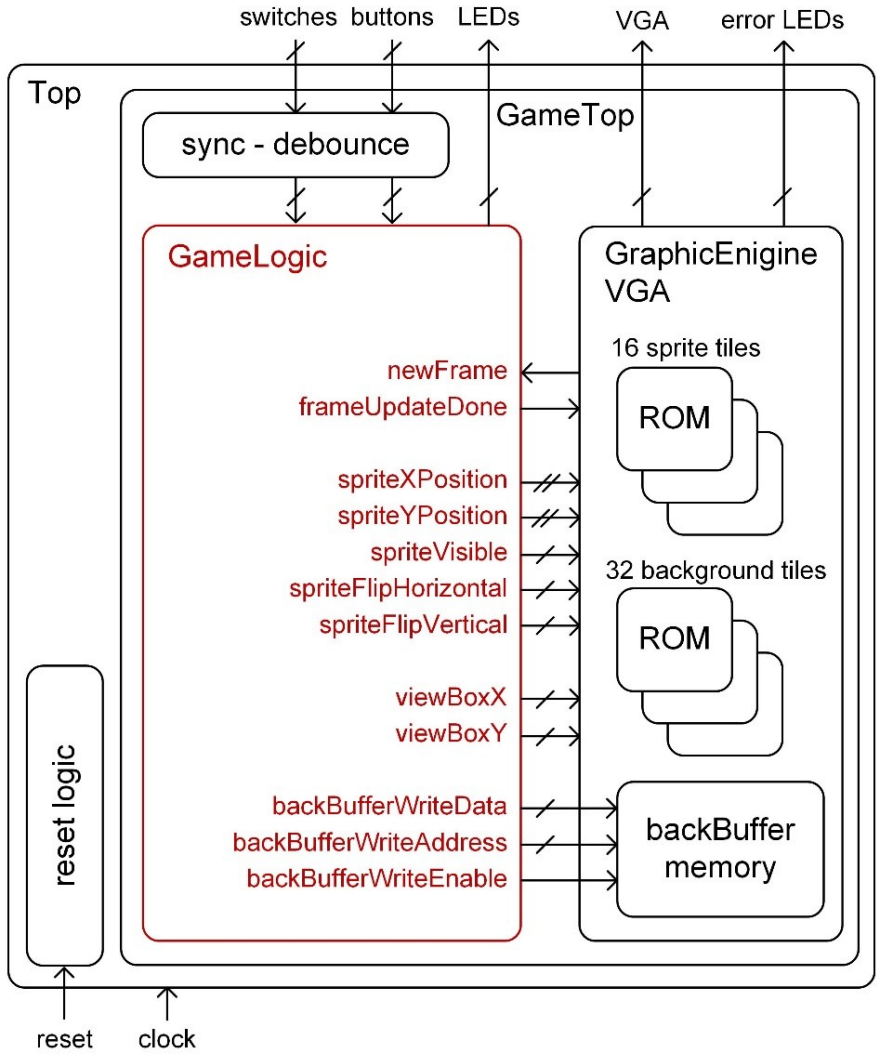
Interface of the GameLogic Module



- **Viewbox Control:**
  - X/Y coordinates of the viewbox (UInt)
- **Background Buffer:**
  - write data
  - address
  - enable

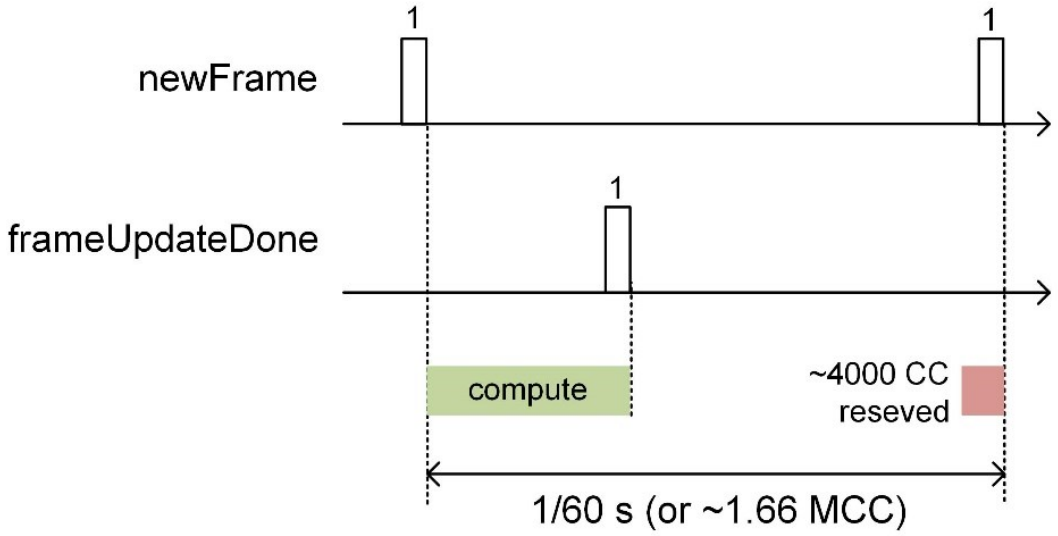


Interface of the GameLogic Module



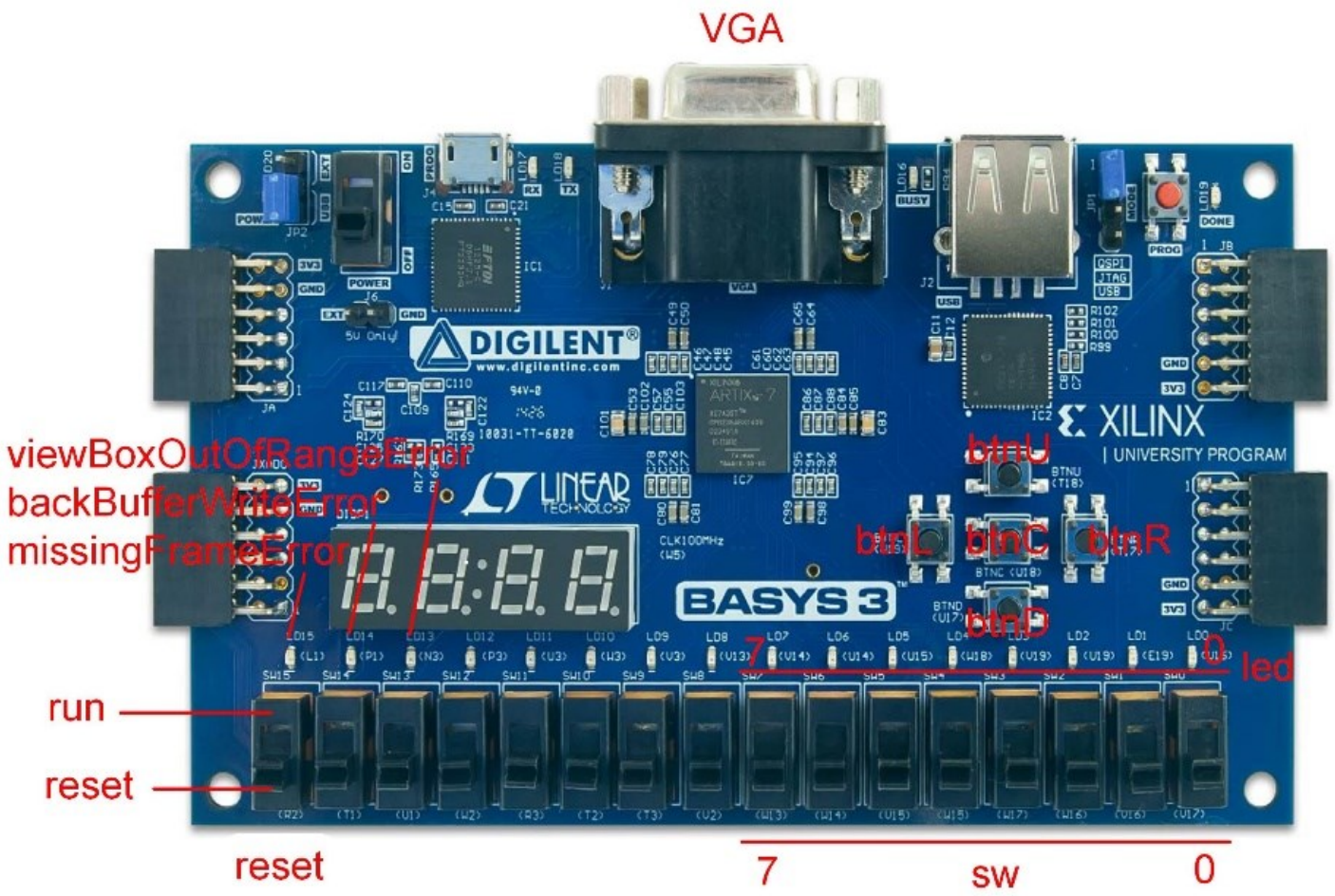
- **Frame Status:**
  - newFrame input
  - frameUpdateDone output

Frame Timing

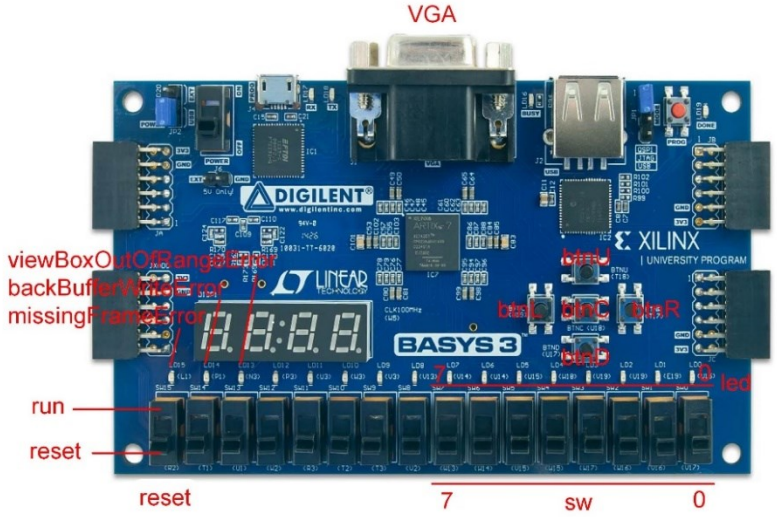


- 60 frames per second
- Each frame has 1.66 million clock cycles
- newFrame signal indicates start of a new frame
- frameUpdateDone signal indicates completion of a frame update
- Reserved period at the end of each frame (no backBuffer memory write here)

Mapping to the Basys3 Board

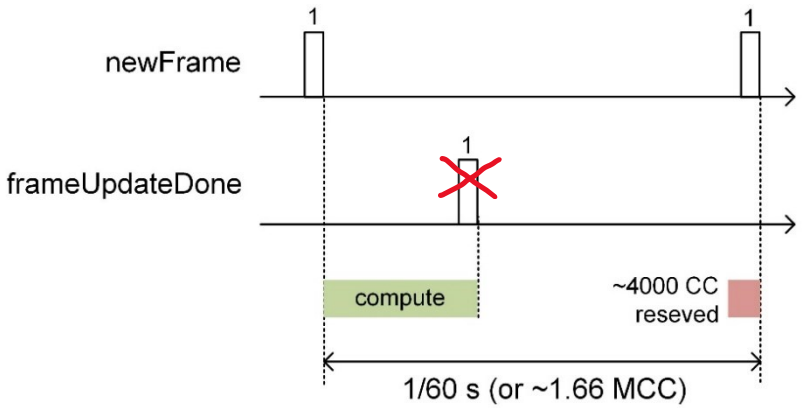


Error LEDs

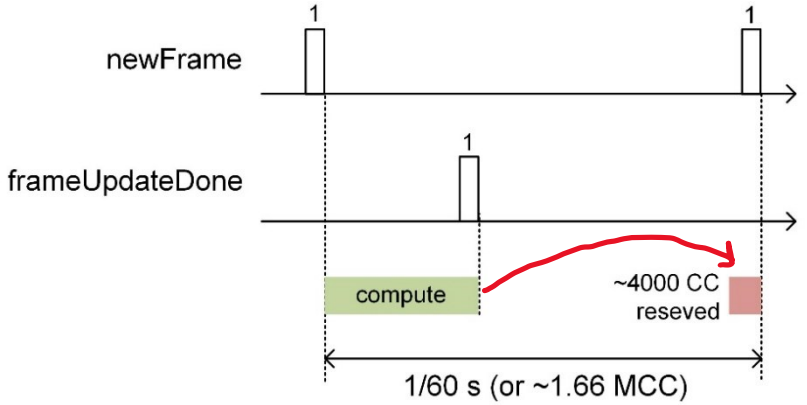


viewBoxOutOfRangeError

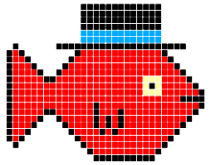
missingFrameError



backBufferWriteError



Learning tasks



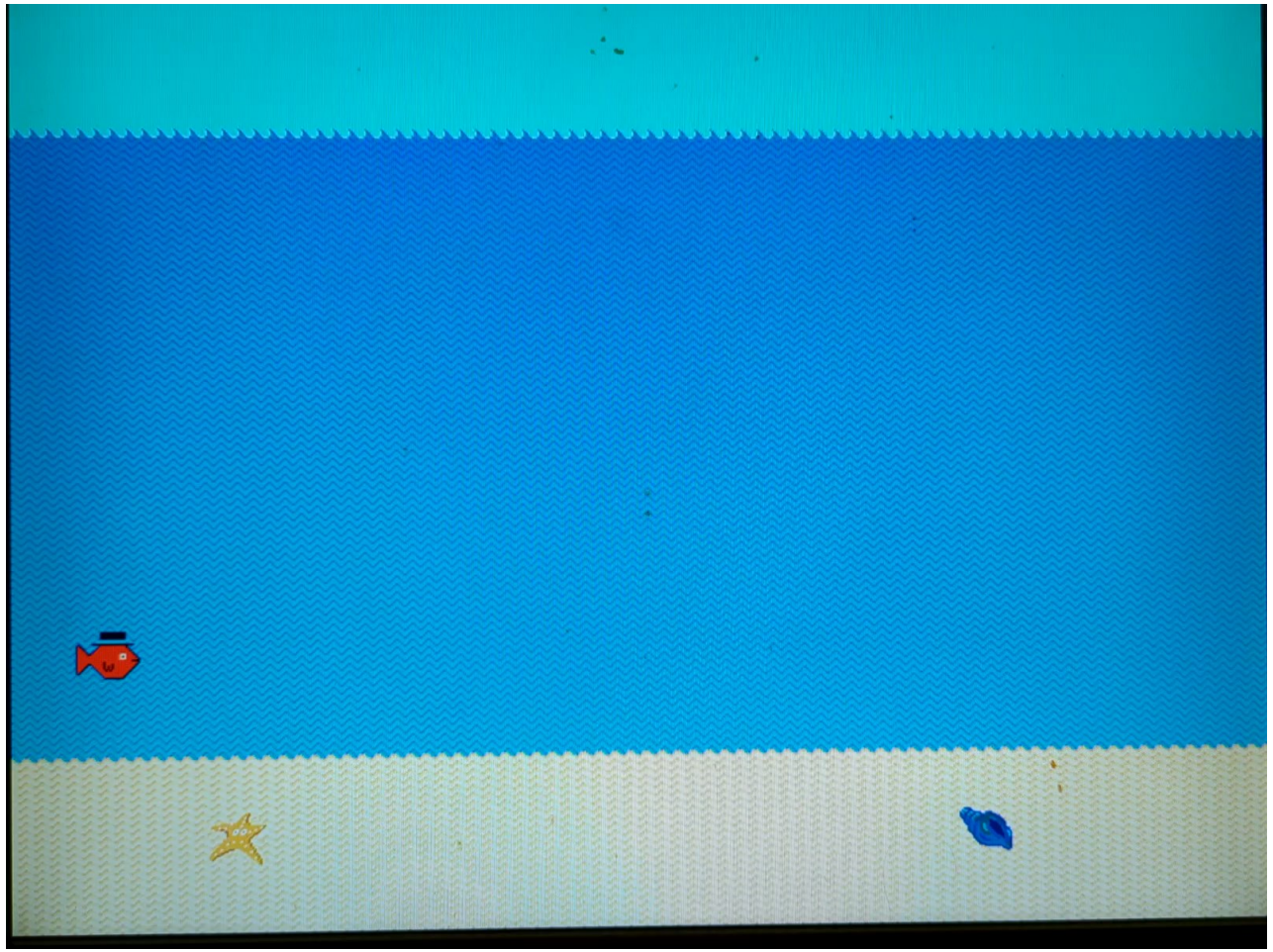
Learning Tasks Overview

- Incremental tasks to understand the system
- Based on a simple demo game: Sea Explorer

Task 0: Run the Demo Game

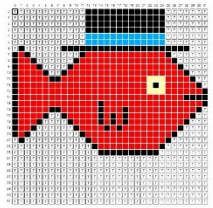
- Build, synthesize, and run the demo game
- Observe the demo game running on the VGA screen
- Analyze the provided implementation for understanding

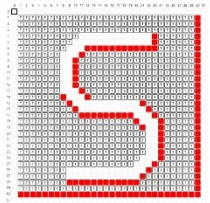
Task 0: Run the Demo Game

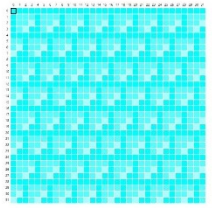


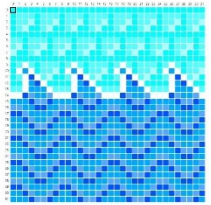
YouTube link:
https://www.youtube.com/watch?v=OLD0g6AezHQ

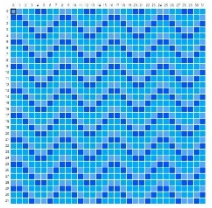
Task 0: Run the Demo Game

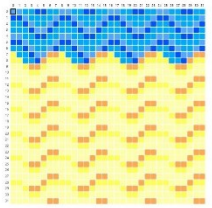

sprite_init_0.mem

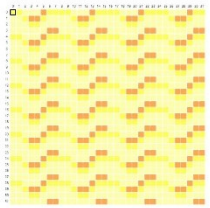

sprite_init_[1-15].mem

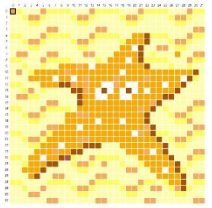

backtile_init_0.mem

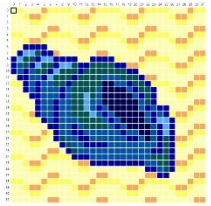

backtile_init_1.mem

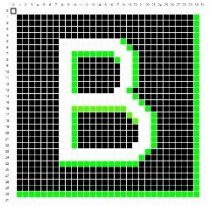

backtile_init_2.mem


backtile_init_3.mem


backtile_init_4.mem


backtile_init_5.mem


backtile_init_6.mem

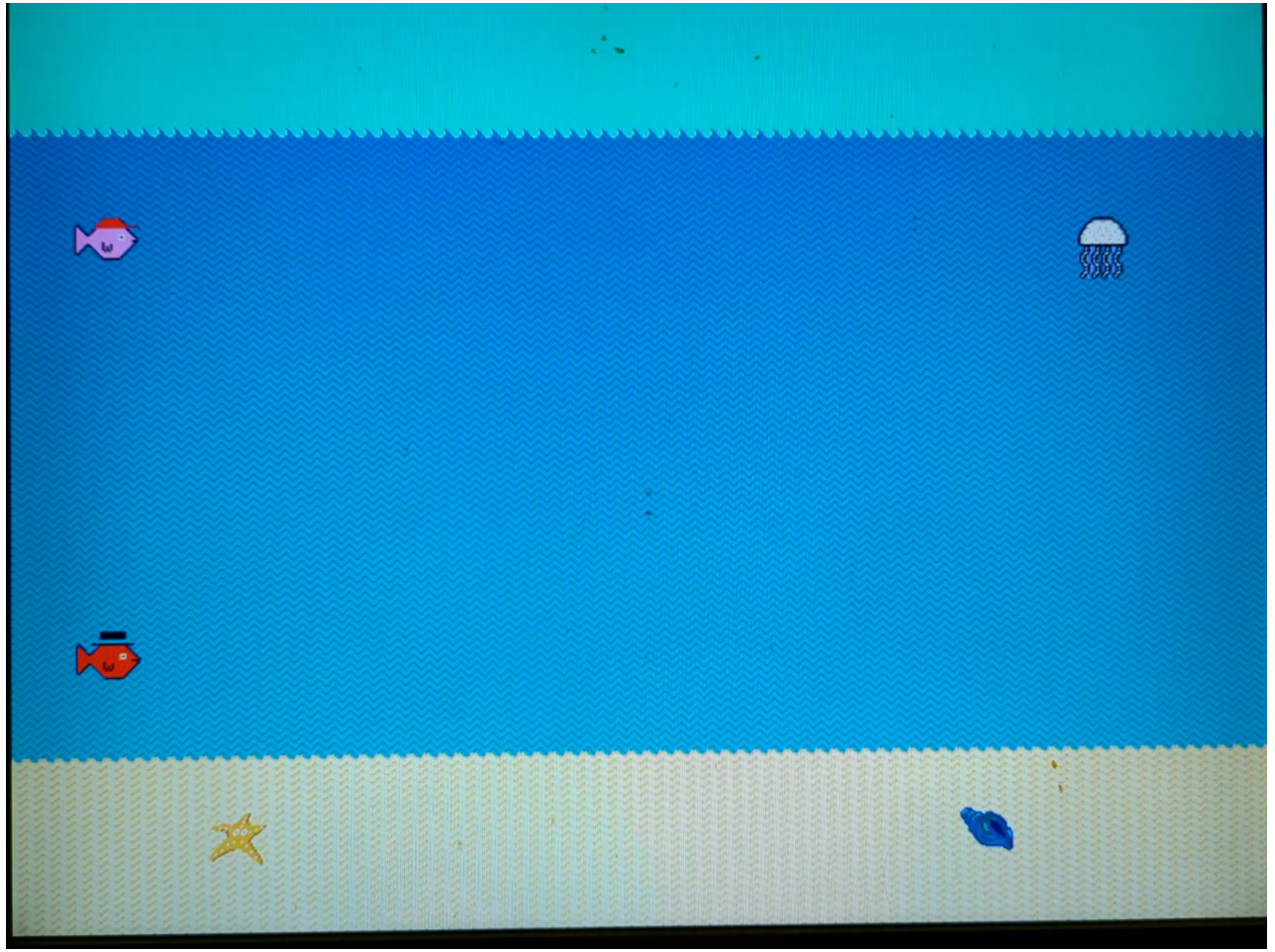

backtile_init_[7-31].mem

```
<project root>
├── memory_init
│   ├── backbuffer_init.mem
│   ├── backtile_init_[0-31].mem
│   └── sprite_init_[0-15].mem
├── tile_generator
│   ├── tile_generator.html
│   └── ...others...
├── src
│   ├── main
│   │   ├── scala
│   │   │   ├── GameLogic.scala
│   │   │   ├── GameLogicTask[0-8].scala
│   │   │   ├── GameTop.scala
│   │   │   ├── GameUtilities.scala
│   │   │   ├── GraphicEngineVGA.scala
│   │   │   ├── Memory.scala
│   │   │   └── Top.scala
│   │   └── ...others...
│   └── test
│       └── scala
│           └── GameLogicTester.scala
├── vivado
│   ├── Basys3Game
│   └── NexysA7Game
└── ...others...
```

Task 1: Draw and Show Your Own Sprites

- Use tile generator tool to design new sprites
- Save generated sprite initialization files
- Make new sprites visible on the screen
- Place new fish and jellyfish at specified positions

Task 1: Draw and Show Your Own Sprites

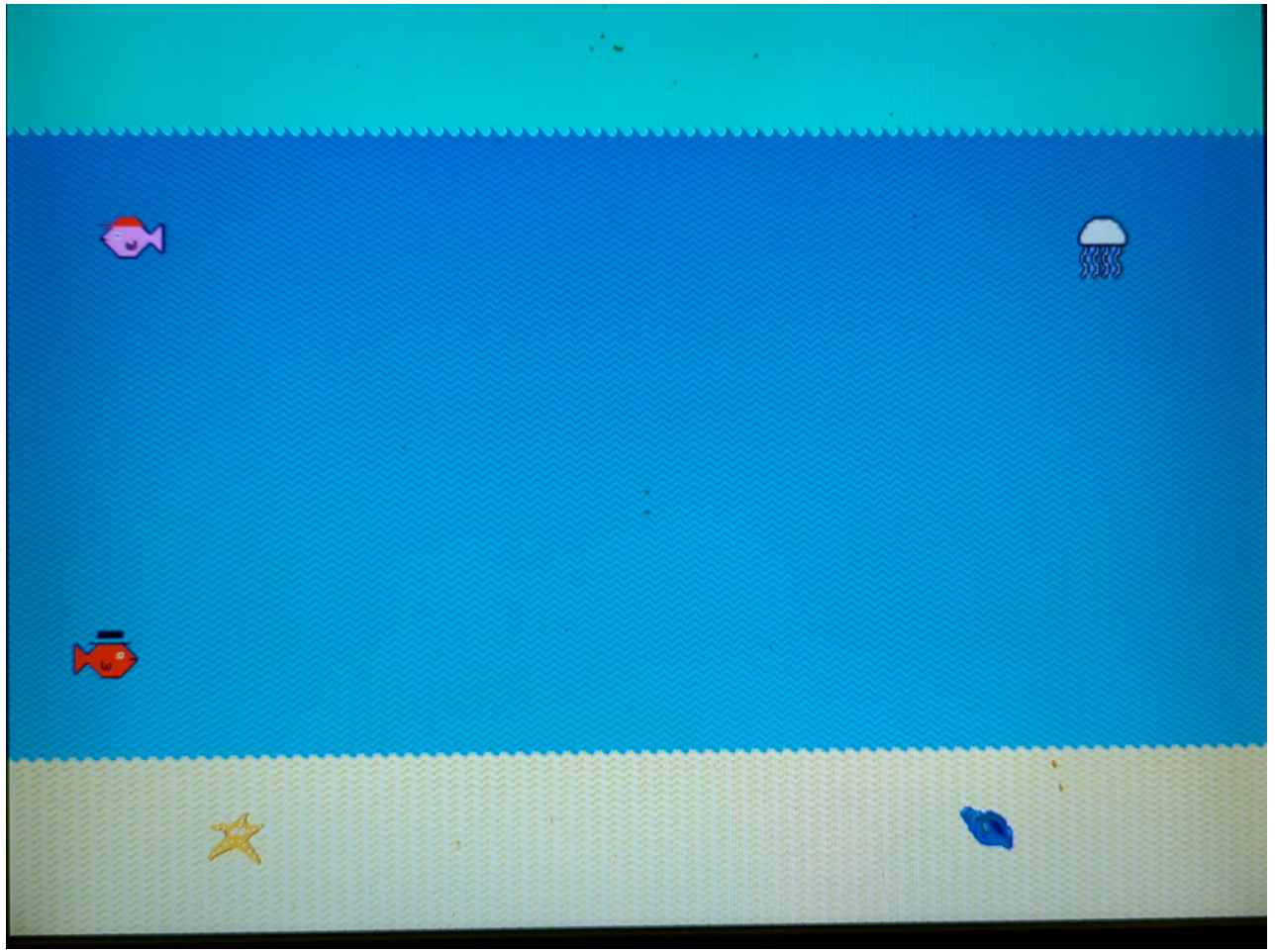


YouTube link:
https://www.youtube.com/watch?v=WCVUoM9Ggwo

Task 2: Move a Sprite

- Implement logic to move sprites autonomously
- Use counters for horizontal and vertical movement
- Update sprite positions in each frame using FSM
- Ensure spriteFlipHorizontal changes with direction

Task 2: Move a Sprite

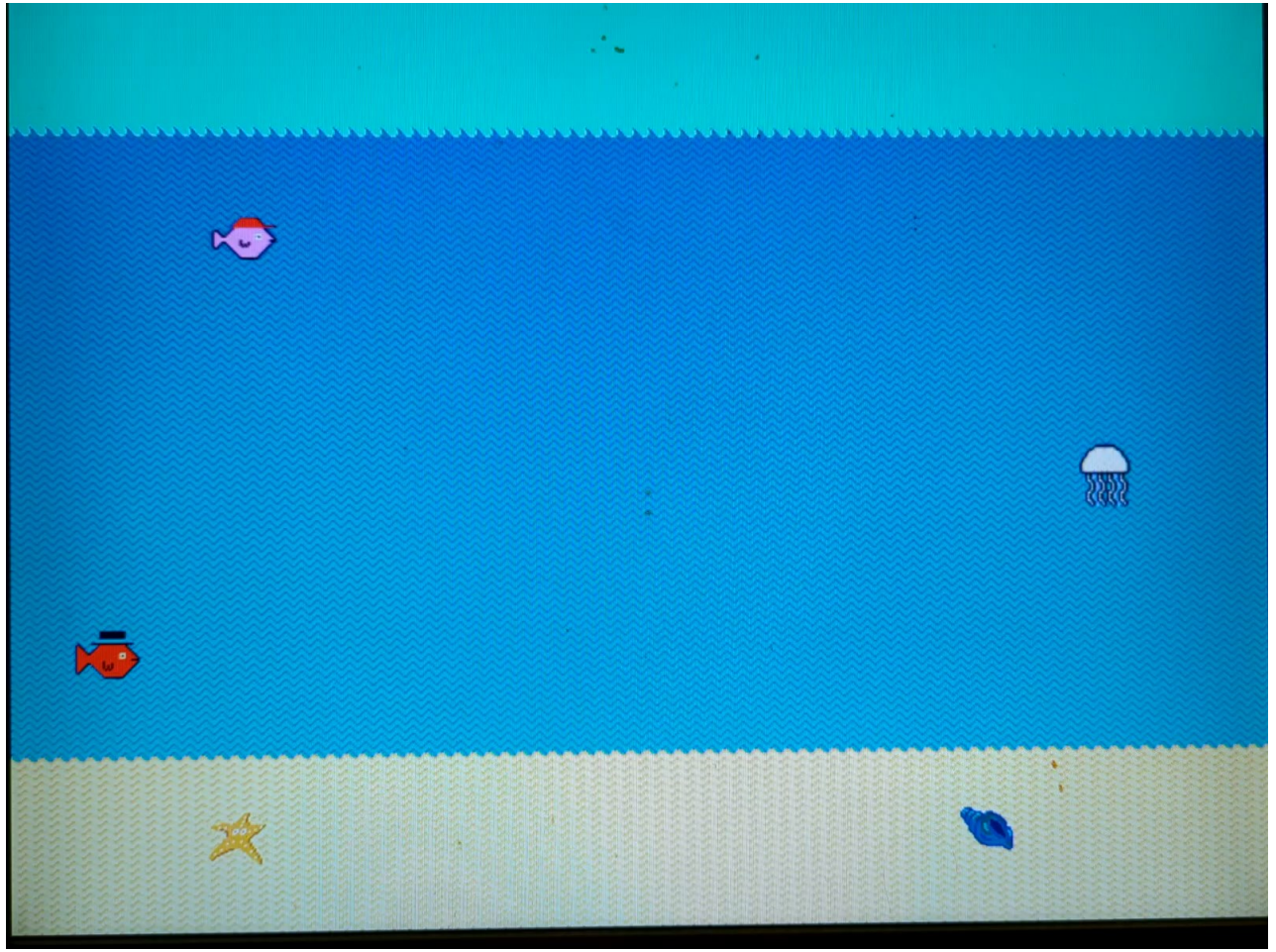


YouTube link:
https://www.youtube.com/watch?v=geQ9UdTJclA

Task 3: Animate a Sprite

- Create an animation effect for a sprite
- Design a second sprite with slight changes
- Toggle visibility between sprites for animation
- Use FSM to control animation timing

Task 3: Animate a Sprite

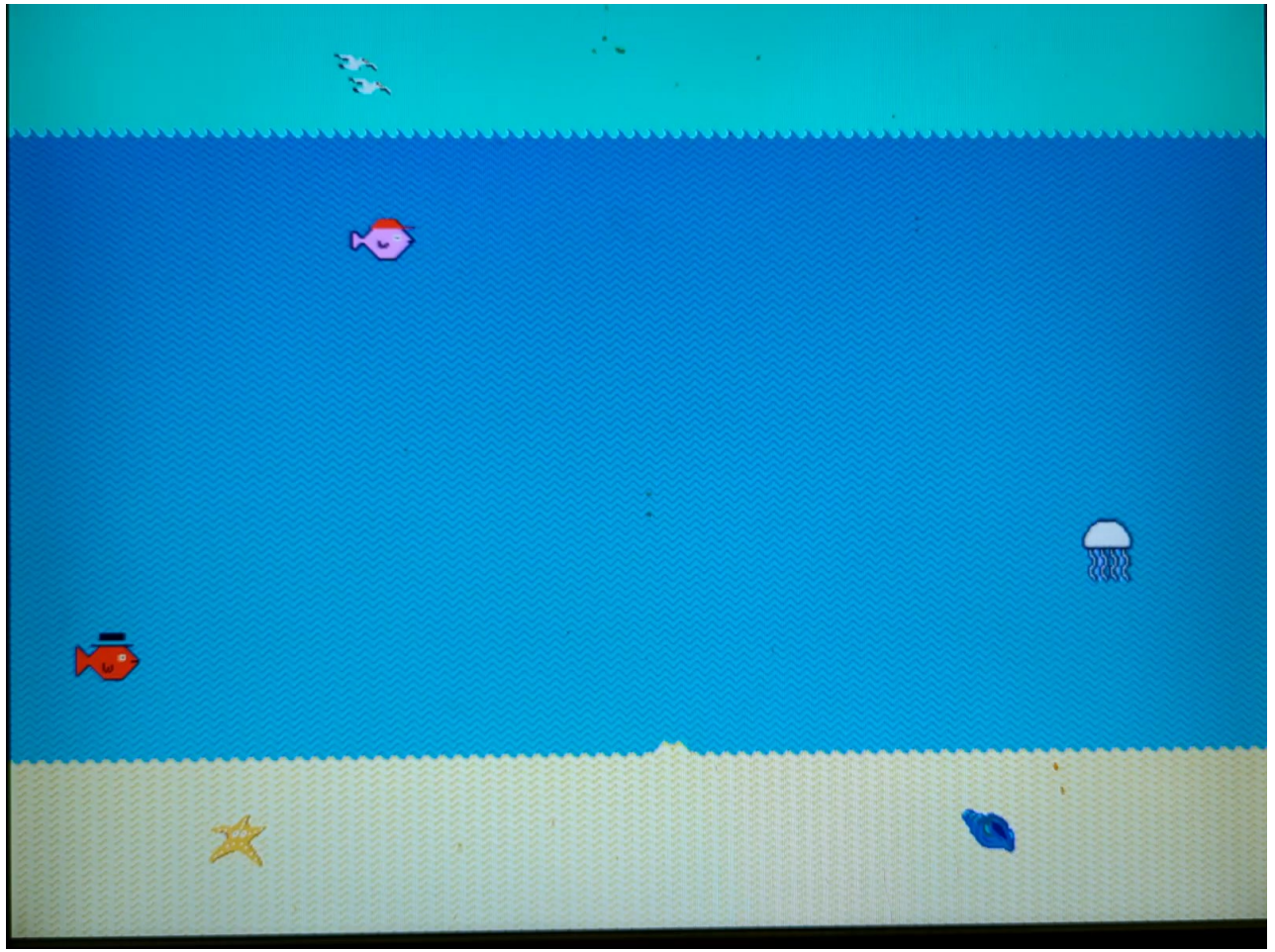


YouTube link:
https://www.youtube.com/watch?v=PiXEN5hguHY

Task 4: Change the Background

- Modify background tiles using tile generator
- Add seagulls and a volcano to the background
- Update backBufferInitmem file with new tile IDs
- Ensure new tiles appear at correct positions

Task 4: Change the Background

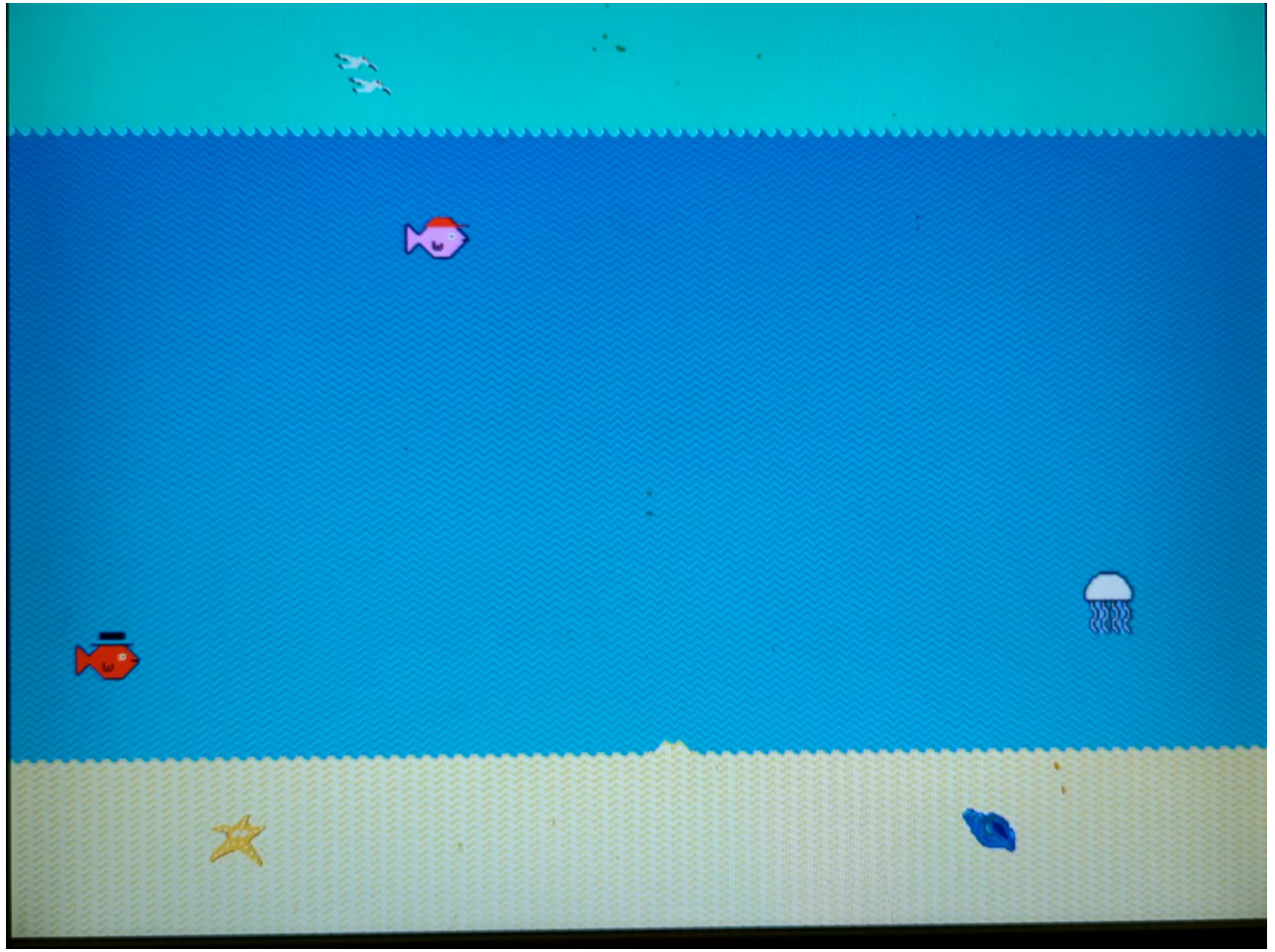


YouTube link:
https://www.youtube.com/watch?v=pEatO8SDPks

Task 5: Animate the Background (Optional)

- Design animated background tiles
- Periodically update background memory with new tiles
- Use FSM to control background animation
- Enhance visual appeal with dynamic background effects

Task 5: Animate the Background (Optional)

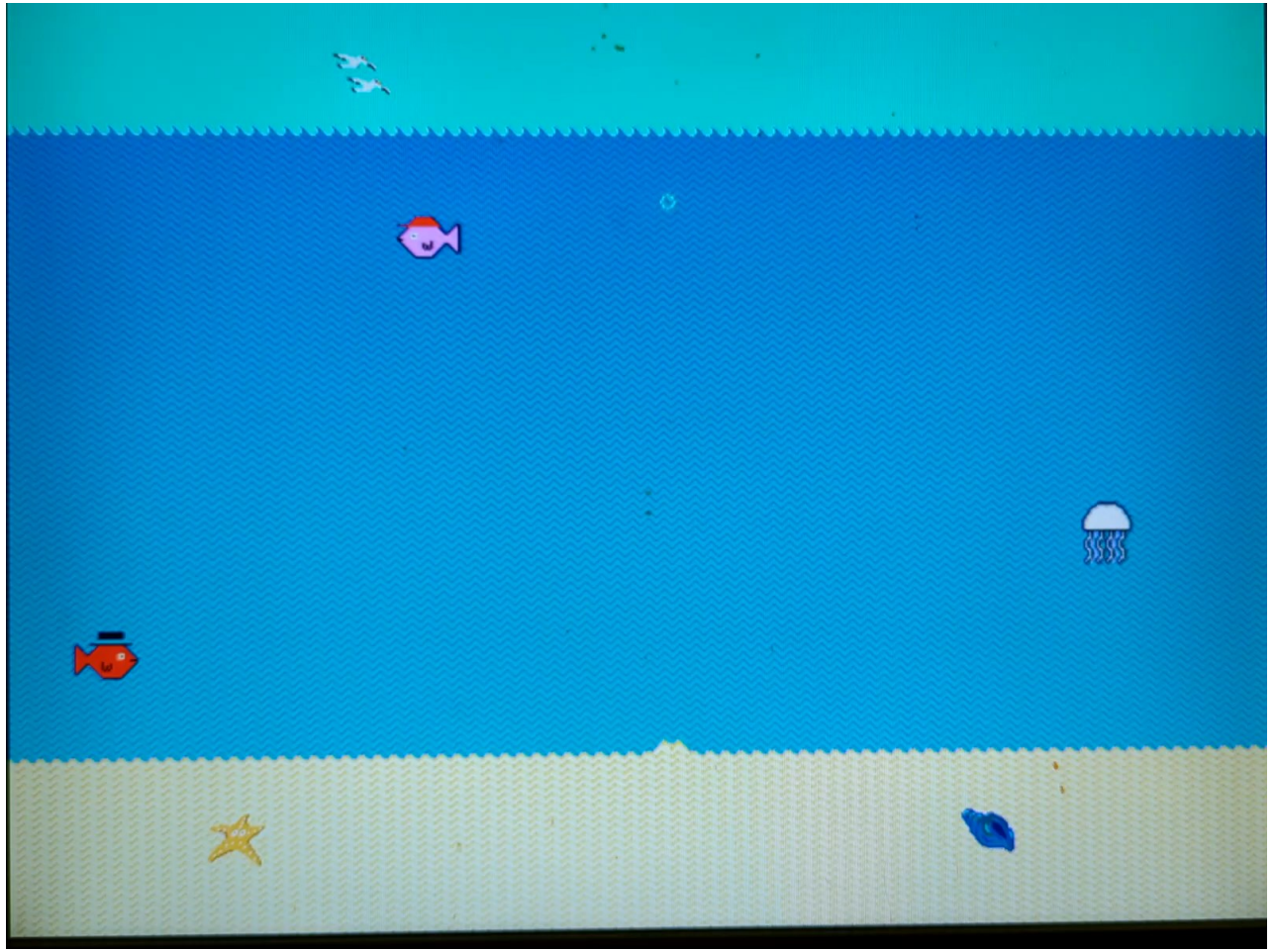


YouTube link:
https://www.youtube.com/watch?v=Ad5RRLGAC4Q

Task 6: Add One More Moving Sprite

- Create a new bubble sprite
- Animate the bubble rising to the water's surface
- Make the bubble disappear at the surface and repeat the cycle
- Use FSM to control the bubble's movement

Task 6: Add One More Moving Sprite

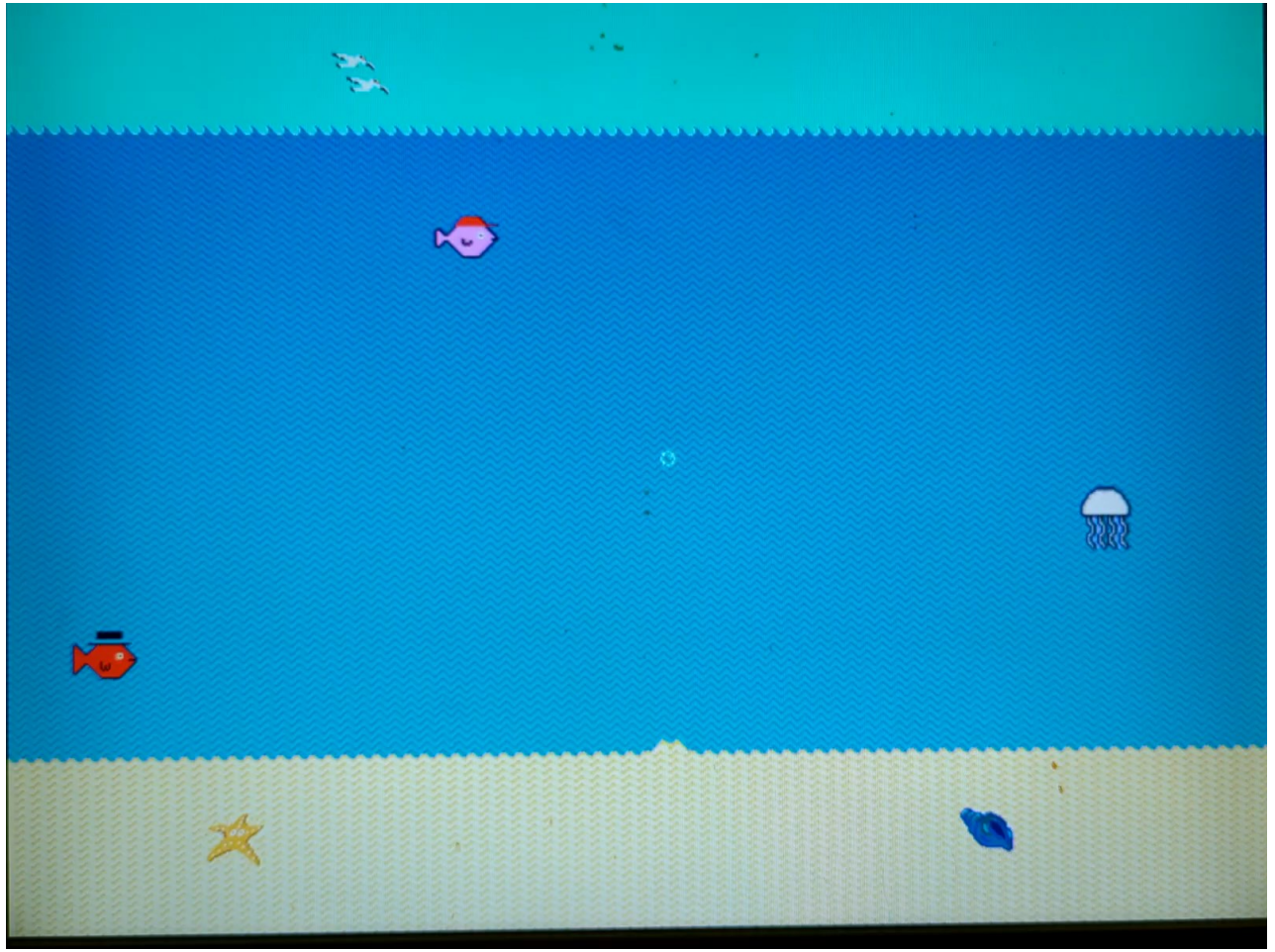


YouTube link:
https://www.youtube.com/watch?v=6nPcnI3DSas

Task 7: Sprite Interaction (Optional)

- Implement collision detection between sprites
- Make bubble disappear when touched by the red fish
- Use bounding box overlap or distance calculation methods
- Complex task requiring precise logic

Task 7: Sprite Interaction (Optional)

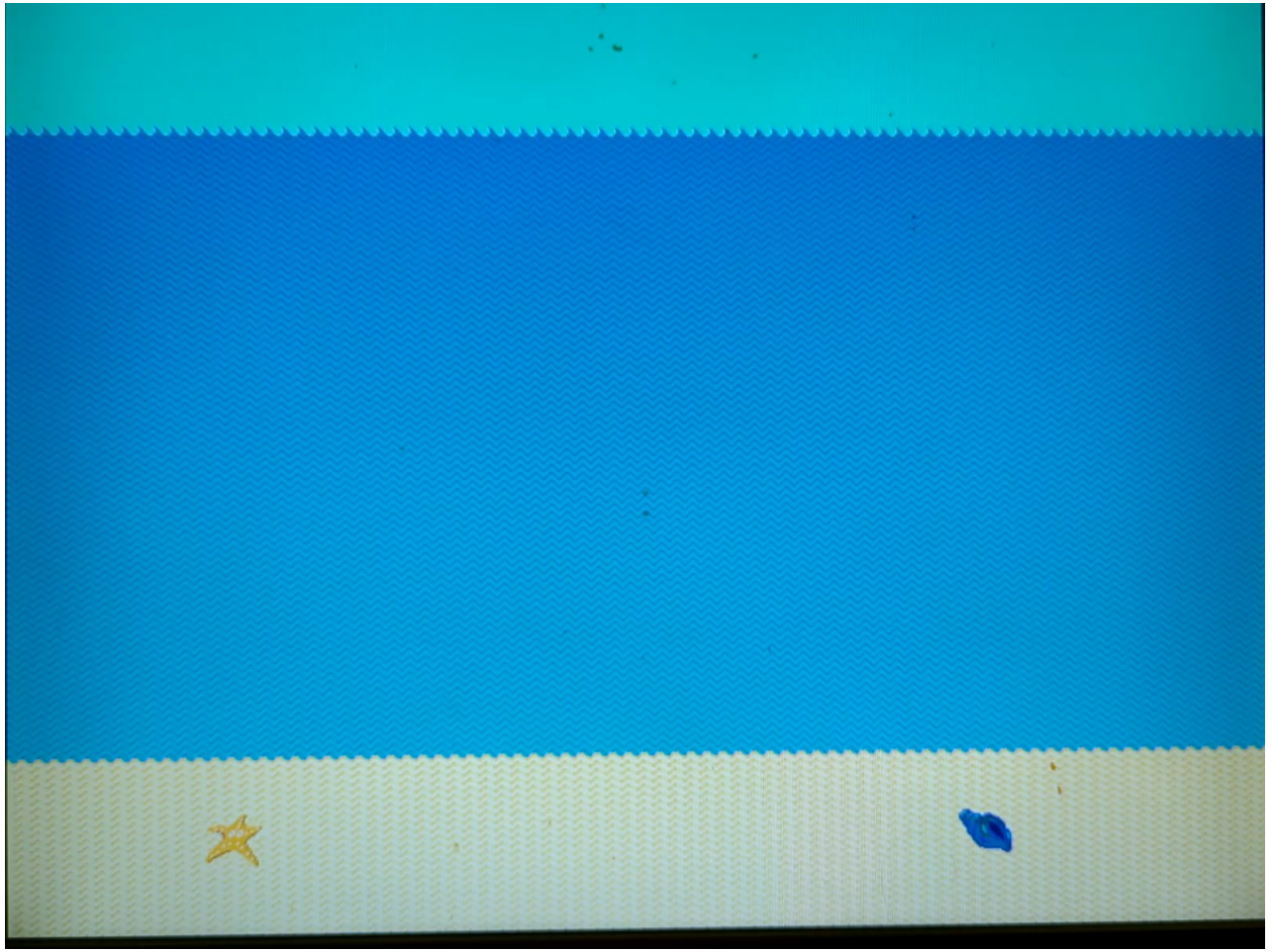


YouTube link:
https://www.youtube.com/watch?v=YYqk9ocKKJU

Task 8: Moving the Viewbox (Given)

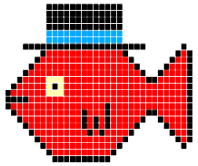
- Explore advanced feature of background scrolling
- Move the viewBox on a large background using buttons
- Pre-implemented task to understand viewBox control
- Examine the given code and try moving the background

Task 8: Moving the Viewbox



YouTube link:
https://www.youtube.com/watch?v=GmdVN8mCsZc

Main project

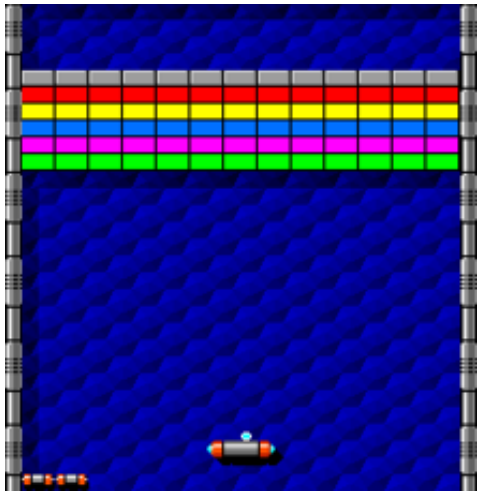


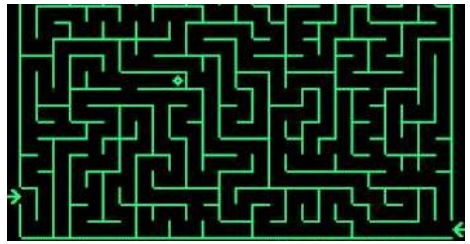
Main Project: Develop Your Game

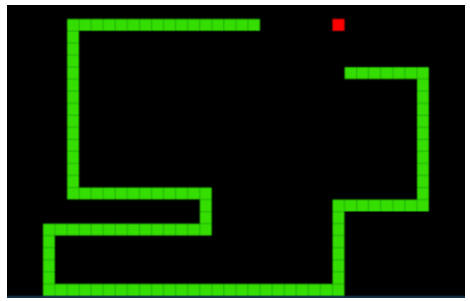
- Implement your own arcade-style game
- Use the provided graphic engine and Chisel
- Develop game logic in the GameLogic block

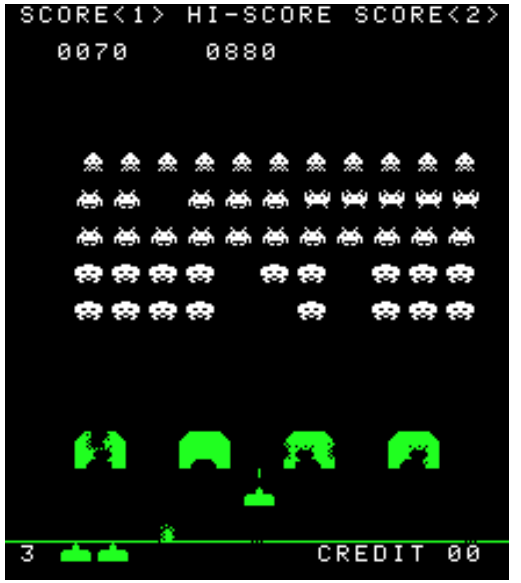
Game Concept

- Start with a simple yet captivating design (start easy!)
- Invent your own unique game concept (get inspired by existing ones)
- Sketch preliminary drafts to visualize the game









Design Phase

- Break down game concept into smaller parts
- Background design: static or scrolling
- Sprite design: size, appearance, animations
- Interaction logic: movement patterns, collision detection

- Define logic for sprites, background, and interactions
- Specify how collisions will be detected and handled
- User input handling using Basys3 buttons

- Develop FSMs for managing game states and interactions
  – States, transitions, and conditions for each FSM

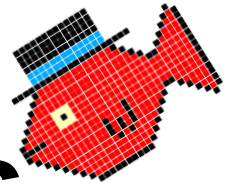
Implementation Phase

- Translate design into Chisel code
- Start with basic functionality and test incrementally
- Ensure each component works before adding new features
- Verify functionality on the Basys3 board (and Chisel tests)
- Conduct thorough testing to identify and fix issues

Enhancements

- Explore advanced features for enhanced gameplay
- Dynamic background effects and sprite animations
- Interactive elements and complex physics
- Experiment within hardware constraints

Final notes



Time Management

Days	What to work on
1	Form groups and register on DTU-Learn
1	Setup development environment and carry out Task 0
1 - 3	Work on the learning tasks
4	Brainstorm and finalize game concepts
4 - 5	Begin initial design drafts
5	Start implementing basic functionality in Chisel
5	Test and verify initial implementations on the board
6 - 10	Incrementally add and test new features
11 - 12	Begin integrating enhancements
13	Finalize all core features and enhancements
11 - 13	Conduct thorough testing (and debugging)
11 - 14	Write and finalize the project report
14	Prepare the final demonstration video
14	Submit the final report and source files
15	Present your game and demonstrate its functionality

Report Requirements

- Maximum **4 pages** using mandatory template

- Introduction
- Design overview
- Implementation details
- Enhancements
- **Contributions (individual contributions of team members)**

Evaluation Criteria

- Grading is based upon the following:
  - **Learning tasks:** completion and understanding
  - **Design:** clarity, completeness, feasibility
  - **Implementation:** accuracy and efficiency
  - **Enhancements:** creativity and effectiveness
  - **Report:** quality and detail of explanations

02113 - Best Game Award

- Celebrate the conclusion of the course
- Event on the last day of the course
- Peer testing and voting for the Award
  - There will be a symbolic prize!
  - Grade is not affected.

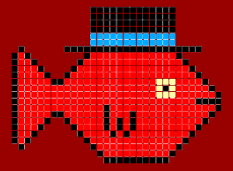
What To Do Now?

- Read the course manual
- Form and register groups
- Install the tools
- Start with the learning task

A Final Note

The course aims to encourage
creativity and smart hardware design.

Let’s aim to have a relaxed, fun,
and productive 3-week course.



02113

Digital Systems Design Project

Luca Pezzarossa
Technical University of Denmark
DTU Compute


DTU