Overview

• Game play
  – One player
  – Three stages (more can be easily added)
  – Two modes: “easy” (E) and “crazy” (C)

• Keyboard control
  – Move: “A,S,W,D”
  – Fire: “Space”
  – Start game: “Enter”
Design Architecture

The diagram shows a design architecture with the following components:

- SRAM
- Audio_Driver
- CPU
- SRAM_Controller
- Audio_Controller
- Keyboard_Controller
- VGA_Controller
- Keyboard
- VGA_Driver
- Preloaded Images
- Avalon Bus
- NIOS II
Software Design

Start

Scenario initialization

Any key pushed?

Yes:

Which Key?

move:

Update the players’ tanks’ positions

fire:

Fire a bullet

No:

Any objects hit?

Yes:

Update the bullets’ positions

No:

Update sprites and display animation
Software Design

- Sprites are stored in integers:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Types</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>23-15</td>
<td>14-6</td>
<td>5-2</td>
<td>1-0</td>
</tr>
</tbody>
</table>

- Backgrounds are split into 13 by 13 blocks, and each of them is stored in an integer:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Blocks</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13 12</td>
<td>11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>


Main Tasks
- Load the scenario setup and locate images on the screen.
- Adjust the tanks and bullets’ positions
- Handle the overlapping problem
- Display animations
• VGA Architecture
• Image Processing:

The user screen is divided into 169 squares, each column or row contains 13 squares as indicated below:
• **Image Processing:**

The game scenario is constructed by 169 different images that loaded from RAM. VGA_Controller will determine each image and its position.
• Image type:
  - Static Scenario

Static Scenarios are predefined before the game starts. Software controls the scenario mapping and send massages to VGA_Driver. Then location information will be stored in RAM and we only focus on the changes after scenario initialization.
• Image type:
  - Sprites

Former method for static scenario is impractical. To locate the sprites, we use the left top point and image length and width.
• **Image Processing:**

The sub-image is formed by 36X36 pixels. The image below is the tank image.
• Animation Effect:

To achieve explosion effect, two explosion images of different sizes are used as frames.

Display the two images in different clock periods and the dynamic effect will be shown.
• Overlapping issue:
  - Set the overlapping area of upper layer to black.
  - Judge the color of image, if it is black, write the data of lower layer to the non-overlapping area.
• **Color conversion:**

  Pixels of tank are represented by 24-bits. The color information stored of tanks occupied a large space of memory.

  Since the appearances of player tank and enemy are the same, color conversion is implemented to save memory space.
• Data Compression

1- bit representation:

2- bit representation:

Reduced size:
• Color conversion:

Pixels of tank are represented by 24-bits. The color information stored of tanks occupied a large space of memory.

Since the appearances of player tank and enemy are the same, color conversion is implemented to save memory space.
• Audio architecture
• 2 kinds of music

➤ Welcome music
  • composed of sin wave with different frequency.
  • Using sin wave data in Lab 3

➤ Sound effect
  • fire and explosion.
  • convert wave to mif, save and play.
  • using adder at output to play two kinds of sound effect simultaneously if needed.
Audio Design

- State machine

```
S0
Waiting for CPU command
CPU_cmd = x"00000100"
```

- **S1**
  - Ready for playing welcome music
  - CPU_cmd = x"00000100"

- **S2**
  - Playing welcome music
  - CPU_cmd = x"0000f00"

- **S3**
  - System reset

```
else
else
else
else
else
else
```

```Python
if CPU_cmd == x"00000100"
    if state == S0:
        state = S1
    else:
        state = S3
else
    if state == S0:
        state = S1
    else:
        state = S3
```

```Python
if state == S1:
    if CPU_cmd == x"0000f00"
        state = S3
```

```Python
if state == S2:
    if CPU_cmd == x"0000f00"
        state = S3
```

```Python
if state == S3:
    if CPU_cmd == x"0000f00"
        state = S3
```

```Python
else
    if state == S0:
        state = S1
    else:
        state = S3
```

```Python
else
    if state == S0:
        state = S1
    else:
        state = S3
```

```Python
else
    if state == S0:
        state = S1
    else:
        state = S3
```
• Images are stored in RAM as a pixel matrix.
• Read the bmp file in Hex starting from the 0x36th byte
• Store RGB information into .mif file.
Wave to mif

<table>
<thead>
<tr>
<th>Acid</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
</tr>
<tr>
<td>001</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
</tr>
<tr>
<td>002</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
</tr>
<tr>
<td>003</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
<td>00D</td>
<td>00D</td>
<td>00B</td>
</tr>
</tbody>
</table>

Table: fire.mif

<table>
<thead>
<tr>
<th>Acid</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>G2DF</td>
<td>D2CD</td>
<td>D2AD</td>
<td>1728</td>
<td>27D3</td>
<td>F601</td>
<td>D003</td>
<td>E2DB</td>
</tr>
<tr>
<td>001</td>
<td>25D4</td>
<td>2740</td>
<td>E94D</td>
<td>D281</td>
<td>FE72</td>
<td>3298</td>
<td>1A48</td>
<td>D95F</td>
</tr>
<tr>
<td>002</td>
<td>D56E</td>
<td>1125</td>
<td>3305</td>
<td>0297</td>
<td>C2EE</td>
<td>E583</td>
<td>2617</td>
<td>3211</td>
</tr>
<tr>
<td>003</td>
<td>D010</td>
<td>D069</td>
<td>CAA8</td>
<td>878F</td>
<td>3696</td>
<td>1F09</td>
<td>D913</td>
<td>D122</td>
</tr>
<tr>
<td>004</td>
<td>D020</td>
<td>D049</td>
<td>185F</td>
<td>CE2B</td>
<td>DBE4</td>
<td>1D2E</td>
<td>3681</td>
<td>F2AF</td>
</tr>
<tr>
<td>005</td>
<td>E02A</td>
<td>EB53</td>
<td>3069</td>
<td>E6B7</td>
<td>CB02</td>
<td>BD58</td>
<td>320E</td>
<td>21CD</td>
</tr>
<tr>
<td>006</td>
<td>E05B</td>
<td>D345</td>
<td>9579</td>
<td>333B</td>
<td>1653</td>
<td>D634</td>
<td>0003</td>
<td>118B</td>
</tr>
<tr>
<td>007</td>
<td>103A</td>
<td>E05D</td>
<td>04F2</td>
<td>E2CC</td>
<td>1752</td>
<td>31C9</td>
<td>37F3</td>
<td>D1A9</td>
</tr>
<tr>
<td>008</td>
<td>E669</td>
<td>2242</td>
<td>3001</td>
<td>F42C</td>
<td>DB05</td>
<td>E6F8</td>
<td>264B</td>
<td>2C64</td>
</tr>
<tr>
<td>009</td>
<td>E8FD</td>
<td>D31E</td>
<td>F510</td>
<td>2A8F</td>
<td>2861</td>
<td>E98C</td>
<td>D19B</td>
<td>F53A</td>
</tr>
<tr>
<td>010</td>
<td>28F7</td>
<td>2120</td>
<td>E650</td>
<td>2C8B</td>
<td>FA4B</td>
<td>2097</td>
<td>2399</td>
<td>E377</td>
</tr>
<tr>
<td>011</td>
<td>G94F</td>
<td>FEF6</td>
<td>292F</td>
<td>1B4A</td>
<td>E555</td>
<td>1E98</td>
<td>FAEC</td>
<td>2366</td>
</tr>
<tr>
<td>012</td>
<td>16B5</td>
<td>E5A5</td>
<td>DA3C</td>
<td>FDFC</td>
<td>2088</td>
<td>1B5F</td>
<td>E4F5</td>
<td>D64B</td>
</tr>
<tr>
<td>013</td>
<td>86FC</td>
<td>471C</td>
<td>0C6C</td>
<td>F4CD</td>
<td>196D</td>
<td>166D</td>
<td>FC65</td>
<td>29E7</td>
</tr>
<tr>
<td>014</td>
<td>82B5</td>
<td>3157</td>
<td>07F6</td>
<td>FE45</td>
<td>1B98</td>
<td>E529</td>
<td>D637</td>
<td>2C8F</td>
</tr>
<tr>
<td>015</td>
<td>72F2</td>
<td>20DF</td>
<td>E717</td>
<td>C201</td>
<td>F2BT</td>
<td>2986</td>
<td>2290</td>
<td>E27D</td>
</tr>
<tr>
<td>016</td>
<td>023C</td>
<td>F66A</td>
<td>1B9A</td>
<td>296C</td>
<td>F08F</td>
<td>D105</td>
<td>E213</td>
<td>23A8</td>
</tr>
<tr>
<td>017</td>
<td>0245</td>
<td>F5FF</td>
<td>D0F5</td>
<td>E6F7</td>
<td>2804</td>
<td>3C07</td>
<td>F6F5</td>
<td>D116</td>
</tr>
<tr>
<td>018</td>
<td>0F61</td>
<td>1A03</td>
<td>3705</td>
<td>0402</td>
<td>E1F5</td>
<td>D9F8</td>
<td>1F30</td>
<td>2904</td>
</tr>
<tr>
<td>019</td>
<td>0CC9</td>
<td>D314</td>
<td>D6D1</td>
<td>365C</td>
<td>12A1</td>
<td>D72C</td>
<td>D569</td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>0651</td>
<td>33B9</td>
<td>10FA</td>
<td>E401</td>
<td>0408</td>
<td>FA95</td>
<td>281D</td>
<td>2285</td>
</tr>
<tr>
<td>021</td>
<td>F369</td>
<td>D741</td>
<td>EA3D</td>
<td>1890</td>
<td>2687</td>
<td>E987</td>
<td>2E7B</td>
<td>F629</td>
</tr>
<tr>
<td>022</td>
<td>00FC</td>
<td>235A</td>
<td>1896</td>
<td>F9B7</td>
<td>E038</td>
<td>E1AF</td>
<td>30EB</td>
<td>2129</td>
</tr>
<tr>
<td>023</td>
<td>85FB</td>
<td>F408</td>
<td>E2F7</td>
<td>F05F</td>
<td>0AC1</td>
<td>183E</td>
<td>1311</td>
<td>FAF5</td>
</tr>
<tr>
<td>024</td>
<td>0D00</td>
<td>0EAD</td>
<td>03D4</td>
<td>2154</td>
<td>1F8B</td>
<td>F719</td>
<td>0068</td>
<td>057F</td>
</tr>
<tr>
<td>025</td>
<td>0G17</td>
<td>SF66</td>
<td>1F09</td>
<td>E72B</td>
<td>C0GF</td>
<td>E639</td>
<td>2162</td>
<td>371F</td>
</tr>
<tr>
<td>026</td>
<td>07D6</td>
<td>D010</td>
<td>D013</td>
<td>F925</td>
<td>24EF</td>
<td>3246</td>
<td>70BD</td>
<td>CE0D</td>
</tr>
<tr>
<td>027</td>
<td>059E</td>
<td>0B1C</td>
<td>3B40</td>
<td>252B</td>
<td>E900</td>
<td>0C47</td>
<td>E256</td>
<td>22E2</td>
</tr>
</tbody>
</table>
Challenges

• **VGA**
  – A lot of sprites
  – Frequent display switch between sprites and static backgrounds
  – Memory size is limited

• **Audio**
  – 2 kinds of music. Need to play both sin wave and wave file
  – Need to play 2 kinds of sound effect simultaneously sometimes

• **Software**
  – A lot of sprites, like bullets, tanks, and explosions
  – Complex game logics, such as bullets collision with obstacles, tanks and even other bullets.
Lessons Learned

• Appropriate design partition is a key for working as a team
• Good data structure is important for implementing complex functions
• Backup source files regularly
• Dropbox helps sharing project files