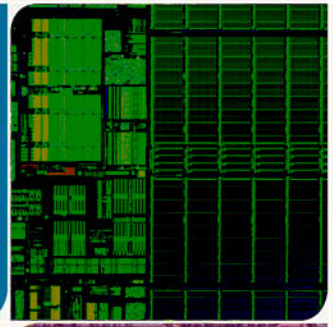
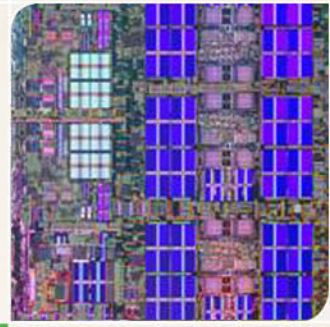
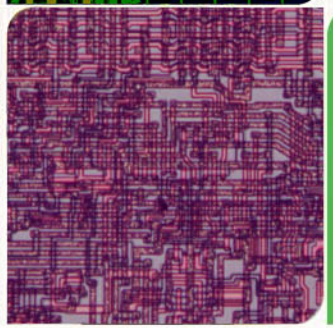
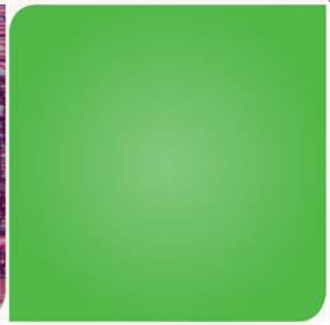
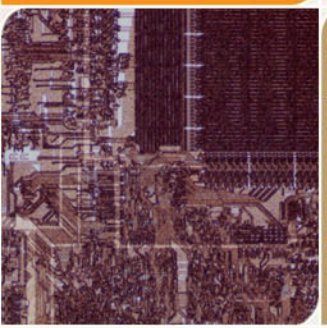



CS4803/CS8803 PGC Design and Programming of Game Console

Spring 2012

Prof. Hyesoon Kim

Georgia Tech College of Computing



Assignment #2

- An image controller
- What to learn
  - Graphics mode
  - Fixed operations
  - Sound
- No need to write assembly code. You can use any APIs

Requirements

- UP/DOWN- move up/down
- LEFT/RIGHT- move left/right
- A- reset to original
- L/R- Rotate
- X/Y- Scale up/ scale down
- Touch- the background should move in the same direction as the stylus. It doesn't need to move as much as the stylus in magnitude- just the same direction.
- Select: save current status (X/Y axis, rotation, scale)
- Start: rollback to the saved status
- **Sound**: At every 10th degree rotation, generate sound

You save the status in the memory.



Graphics Mode

- Dual screen: Main screen, sub screen
- 8 Graphics Modes
- Background Type
  - Frame buffer: manipulate each pixel
  - 3D: OpenGL 3D feature
  - Text: aka tile background
  - Rotation: similar to text but rotation, scale
  - Extended rotation:
  - Large bitmap

Extended Rotation Background

- A bitmap that can be displayed on the screen
- The bitmap could be larger than the physical size of the screen
- Hardware scrolling, rotation, scaling, shearing
- Register sets



Graphics Modes

Main 2D Engine				
Mode	BG0	BG1	BG2	BG3
Mode 0	Text/3D	Text	Text	Text
Mode 1	Text/3D	Text	Text	Rotation
Mode 2	Text/3D	Text	Rotation	Rotation
Mode 3	Text/3D	Text	Text	Extended
Mode 4	Text/3D	Text	Rotation	Extended
Mode 5	Text/3D	Text	Extended	Extended
Mode 6	3D	-	Large Bitmap	-
Frame Buffer	Direct VRAM display as a bitmap			

Graphics Modes

SUB 2D Engine				
Mode	BG0	BG1	BG2	BG3
Mode 0	Text/3D	Text	Text	Text
Mode 1	Text/3D	Text	Text	Rotation
Mode 2	Text/3D	Text	Rotation	Rotation
Mode 3	Text/3D	Text	Text	Extended
Mode 4	Text/3D	Text	Rotation	Extended
Mode 5	Text/3D	Text	Extended	Extended

- Frame buffer can be used by only one screen



Initialization

- initVideo
- initBackground



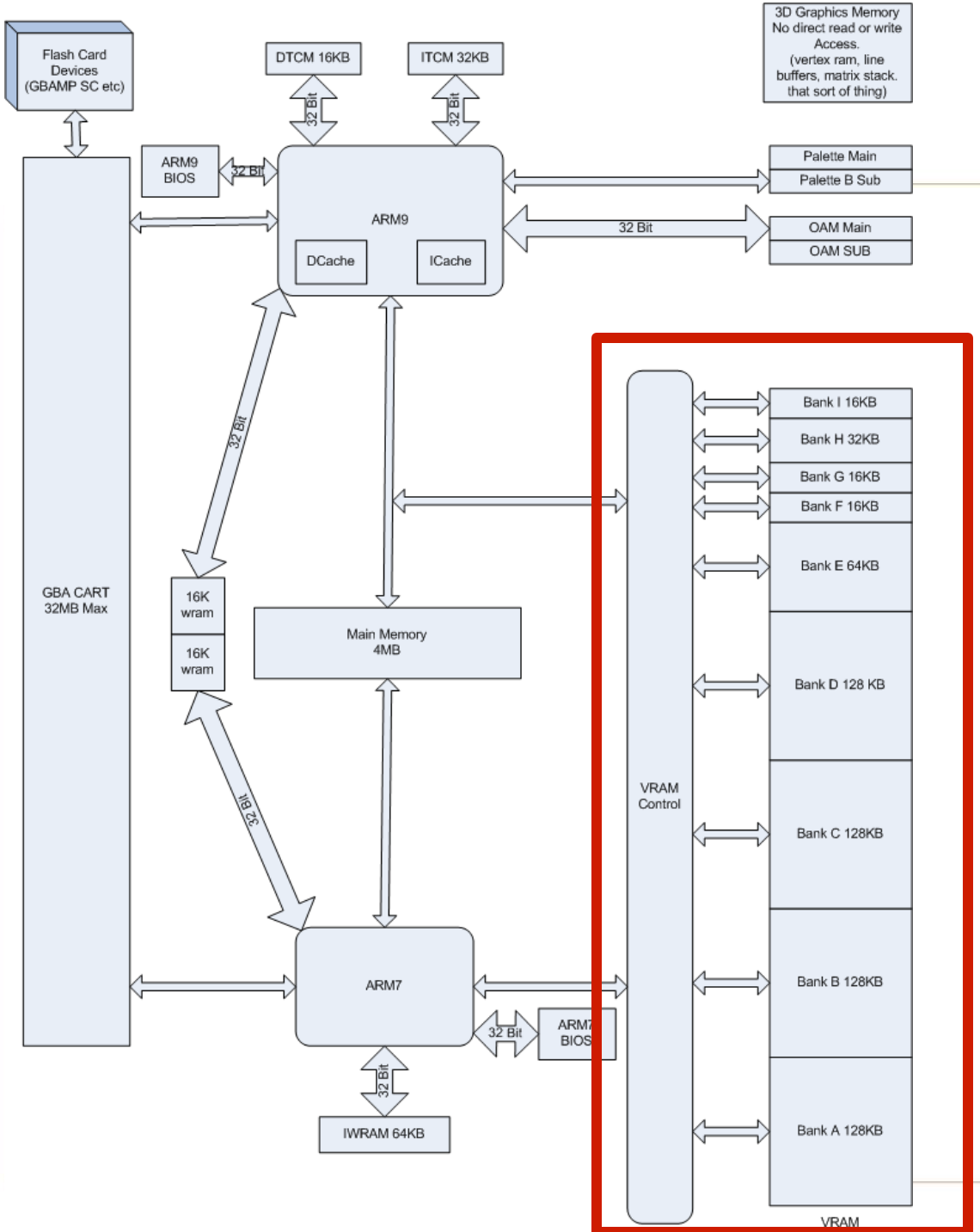
Mode 5 Set

```
/*  Set the video mode on the main screen. */

videoSetMode(MODE_5_2D | // Set the graphics mode to Mode 5
     DISPLAY_BG2_ACTIVE | // Enable BG2 for display
     DISPLAY_BG3_ACTIVE); //Enable BG3 for display

/*  Set the video mode on the sub screen. */
     videoSetModeSub(MODE_5_2D | // Set the graphics mode to Mode 5
                                 DISPLAY_BG3_ACTIVE); // Enable BG3 for display
```

Video RAM (VRAM)



VRAM works as working space for display.
A certain memory space will be drawn depending on Mode
9 banks

Bank	Control Register	Size
VRAM_A	VRAM_A_CR	128KB
VRAM_B	VRAM_B_CR	128KB
VRAM_C	VRAM_C_CR	128KB
VRAM_D	VRAM_D_CR	128KB
VRAM_E	VRAM_E_CR	64KB
VRAM_F	VRAM_F_CR	16KB
VRAM_G	VRAM_G_CR	16KB
VRAM_H	VRAM_H_CR	32KB
VRAM_I	VRAM_I_CR	32KB



VRAM allocation

- We need to allocate right amount of video memory to the correct memory address for a mode
  - Large enough to have the bitmap

```
void initVideo() {
    /*
    *  Map VRAM to display a background on the main and sub screens.
    *  The vramSetMainBanks function takes four arguments, one for each of the
    *  major VRAM banks. We can use it as short hand for assigning values to
    *  each of the VRAM bank's control registers.
    *  We map banks A and B to main screen background memory. This gives us
    *  256KB, which is a healthy amount for 16-bit graphics.
    *  We map bank C to sub screen background memory.
    * We map bank D to LCD. This setting is generally used for when we aren't
    *  using a particular bank.
    */
    vramSetMainBanks(VRAM_A_MAIN_BG_0x06000000,
                     VRAM_B_MAIN_BG_0x06020000,
                     VRAM_C_SUB_BG_0x06200000,
                     VRAM_D_LCD);
// graphics mode setting (previous slide code)
}
```



Other Control Parameters

- BG_{32x32|32x64|64x32|64x64}; used for text backgrounds
- BG_RS_{16x16|32x32|64x64|128x128}; used for rotation backgrounds
- BG_BMP{8|16}_{128x128|256x256|512x256|512x512}: extended rotation background variants, bit per pixel and resolution
- BG_BMP8_1024x512 and BG_BMP8_512x1024: is used only for MODE 6
- BG_WRAP_ON: if you scroll to the end of the image, it will wrap. This way, you can "scroll forever".
- BG_PRIORITY(n) or BG_PRIORITY_n: the priority of the background: 0 is the highest priority, 3 the lowest. A background with a higher priority will be printed on top of backgrounds with lower priorities. If there is a sprite with the same priority, it will be printed on top of the background.
- BG_MOSAIC_ON: You have to set this flag if you want to use the mosaic effect BG_TILE_BASE(n): each tile-block is 16KB. This parameter selects, which block we want to use. For tile-based backgrounds only.
- BG_MAP_BASE(n): each map-block is 2KB. This parameter selects, which block we want to use. For tile-based backgrounds only.
- BG_BMP_BASE(n): each bitmap-block is 16KB. This parameter selects, which block we want to use. For bitmap backgrounds only.



More Registers to Control BackGround

- BGn_X0: this controls where the left origin of the screen maps to the background
- BGn_Y0: this controls where the top of the screen maps to the background
- With n = 0,1,2 or 3. If we use an extended rotation background we have even more registers:
- **BGn_XDX: this controls the x-axis scaling, it's a 0.8.8 fixed point number. If you don't want to scale at all, set it to 1.0 (which is 1 << 8). Increase the value to "zoom out." F0or example, 2.0 (1 << 9) will show the background at half its width.**
- BGn_XDY: this is for rotating and shearing
- BGn_YDX: this is for rotating and shearing
- BGn_YDY: this controls the y-axis scaling and works the same as BGn_XDX.
- BGn_CX: this controls where the left origin of the screen maps to the background (in 0.8.8 fixed point, too).
- BGn_CY: this controls where the top of the screen maps to the background (in 0.8.8 fixed point, too).
- With n = 2 or 3 and the same for the sub screen (SUB_BG2_X0 etc.).
- When using a rotation background or extended rotation background the BGn_CX and BGn_CY registers replace BGn_X0 and BGn_Y0.



initBackgrounds()

```
/* Set up affine background 3 on main as a 16-bit color background. */
    REG_BG3CNT = BG_BMP16_256x256 | BG_BMP_BASE(0) | // The starting
    place in memory
BG_PRIORITY(3); // A low priority /* Set the affine transformation matrix for the
    main screen background 3 * to be the identity matrix. */
REG_BG3PA = 1 << 8;
REG_BG3PB = 0;
REG_BG3PC = 0;
REG_BG3PD = 1 << 8; /* Place main screen background 3 at the origin (upper
    left of the * screen). */
REG_BG3X = 0; REG_BG3Y = 0;
```

Affine Transformation Matrix

- Each background is also transformed by its affine transformation matrix
- X-> Ax+B
- 4 registers [ABCD]
- A: REG_BG3PA, B:REG_BG3PB, C:REG_BG3PC, D: REG_BG3PD

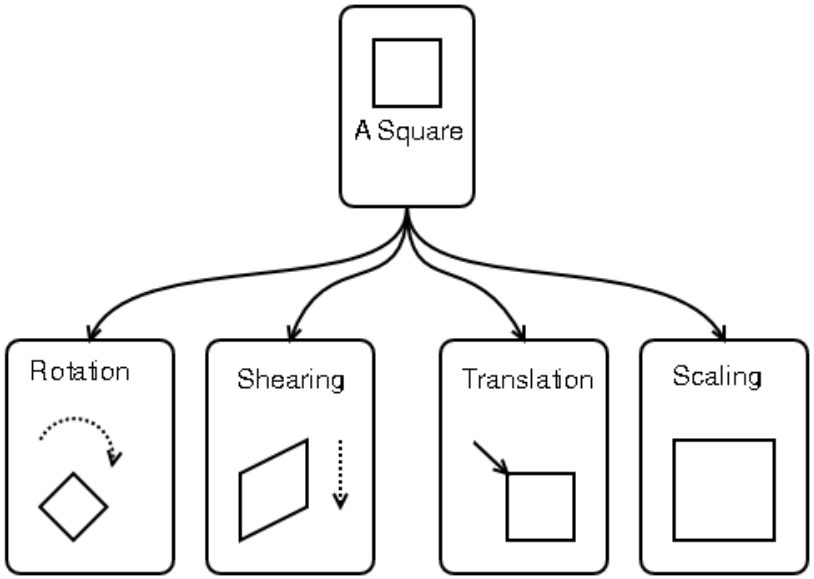


Figure 2. Affine transform Matrixes in Homogenous space

Rotation matrix

cos o	-sin o	0
sin o	cos o	0
0	0	1

Shear matrix

1	a	0
0	1	0
0	0	1

Scaling matrix

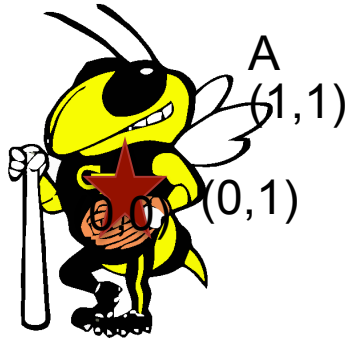
Sx	0	0
0	Sy	0
0	0	1

Translation matrix

1	0	dx
0	1	dy
0	0	1

Exercise

Write an affine transformation matrix and write down the new coordinates of A, B. The yellowjackets becomes 2 times bigger, 90 rotation (counter clock wise) and move + 1 x-axis



Fixed Point Operation

- Use integer operators to calculate floating point operations
- Fixed point integer

```
   12.23          1223
  +20. 41       +  2041
--------------  ----------
   32.64          3264
```

Fixed fraction

```
   12.23          1223
 x  20.41       x  2041
--------------  ----------
   249.61         2496143
```

- a 1.15.16 fixed point number: one bit sign, 15-bit integer, 16-bit fraction. 1+15+16 = 32

Fixed Point Operation Types in DS

- v16 = 1.3.12 fixed point number (used for 3D)
- t16 = 1.11.4 fixed point number
- f32 = 1.19.12 fixed point number (used for matrices)
- v10 = 1.0.9 fixed point number (whoops! 10 bits don't fit into a normal integer? but 3 v10 numbers fit into a 32 bit integer, so this format is used for normals in 3D (it's also ok for normals to be between -1 and 1, so this is why these fixed point numbers have a long fraction but no integer part!)
- **0.8.8 fixed point number**: this format is used for the scaling of the extended rotation backgrounds and doesn't have a typedef in the NDSlib.



0.8.8 fixed Point Conversion

Fixed point number: 0x300

Value ?

3

Fixed point number: 0x104

Value?

0x104=b1 0000 0100 = 1 + 0*1/2+0*1/4+0*1/8+0*1/16+0*1/32+1*1/64+0*1/128+0*/256

Value 4.75 → Fixed point?

4.75 = 4 + 0.5+ 0.25 → (4 << 8 | 1 << 7 | 1 << 6)

= 0x4C0



Background Images

- Use hex array values: provided in the homepage
- Use DMA to read background images

```
/* Select a low priority DMA channel to perform our background *
    copying. */
 static const int DMA_CHANNEL = 3;
  void displayStarField() {
dmaCopyHalfWords(DMA_CHANNEL, starFieldBitmap, /* This variable
    is generated for us by * grit. */
 (uint16 *)BG_BMP_RAM(0), /* Our address for main * background 3 */
    starFieldBitmapLen); /* This length (in bytes) is generated * from grit.
    */
}
```



Sound

Only ARM 7 can play sound , but the library will handle this.

```
int main(int argc, char ** argv)
 {
    [...]
    // Turn on Sound
        soundEnable ();
…
   soundChannelID=soundPlayNoise (10000, 100, 64); //frq can be
    changed here. Current frq is 10k Hz.
    swiDelay(10000000);
    soundKill (soundChannelID);
    }
    }
```

ReadingKeys using APIs

```
If (keysHeld()&KEY_LEFT)
{
   // features for left key
}
If (keysHeld()&KEY_X)
{
   // features for left key
}
```



Things to Know

- You might see two images
- Please start early. (This is longer than the previous assignment !!)
- Download the startup code
- See more info for background setup
  - http://patater.com/files/projects/manual/manual.html#id2612791