Module 13B: Using The ISprite APIs

BREW^{TM} Developer Training

Module Objectives

◆Describe the animated image capabilities provided by the ISprite Interface

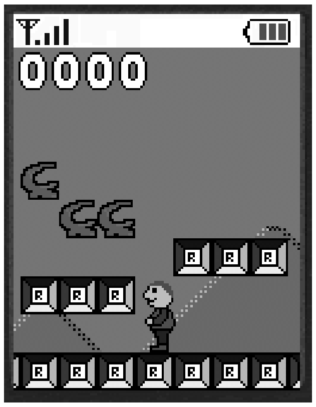
Sprites Overview

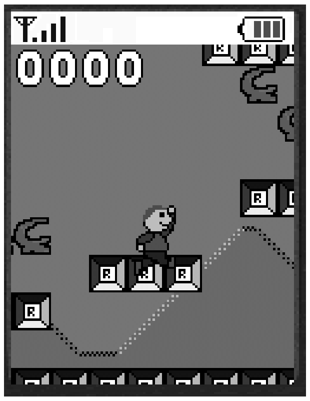
- Definitions of **Sprites** on the Web:
  - Disembodied spirits, elves, fairies or daemons; often the term used for the Air elemental known as "sylphs," or as the name of the elementals of Spirit. www.spiritualitea.com/articles/paganglossary.shtml
  - A small bitmap image, often used in animated games. www.siprep.org/clubs/tech/main/glossary/
- ISprite introduced in BREW V2.0:

3

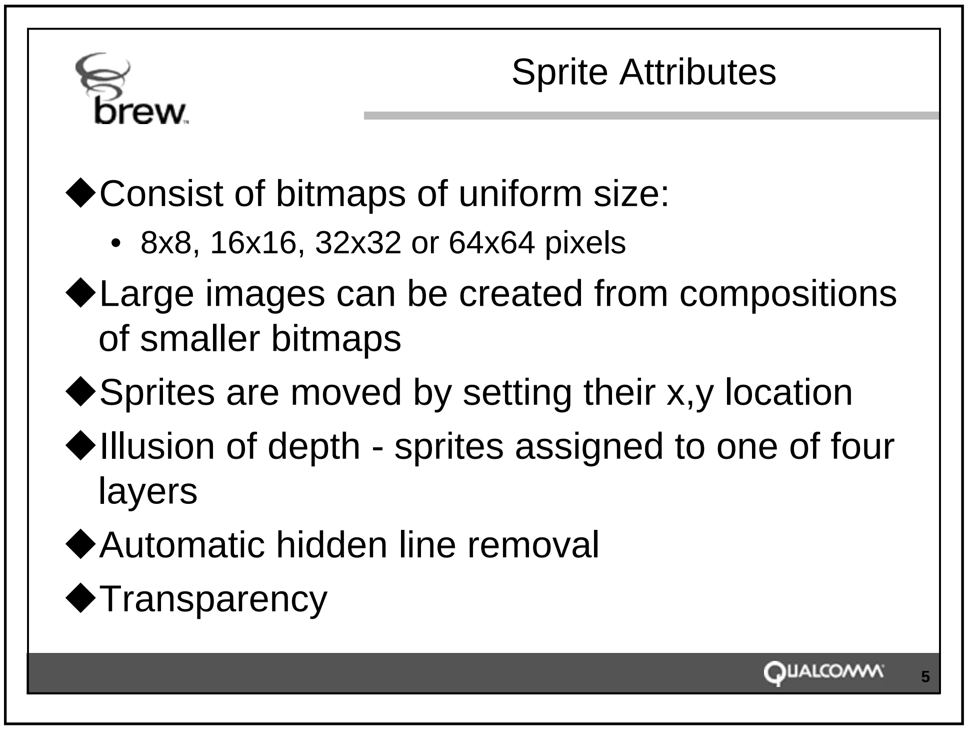
Sprite Game example

- Downloaded from BREW Developer site





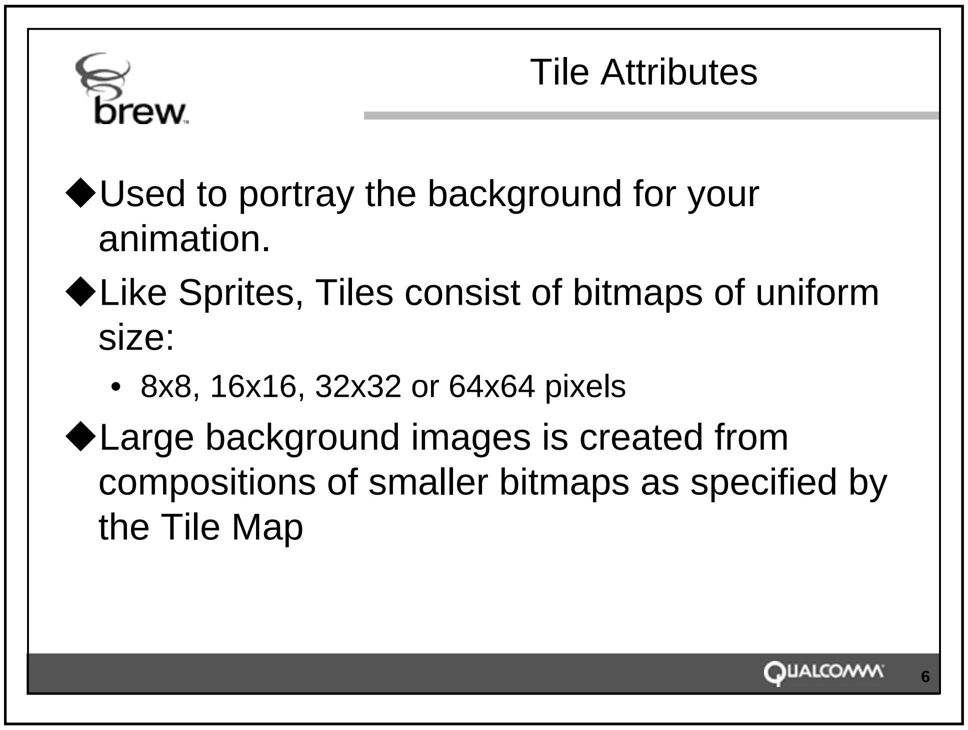
4



Sprite Attributes

- ◆Consist of bitmaps of uniform size:
  - 8x8, 16x16, 32x32 or 64x64 pixels
- ◆Large images can be created from compositions of smaller bitmaps
- ◆Sprites are moved by setting their x,y location
- ◆Illusion of depth - sprites assigned to one of four layers
- ◆Automatic hidden line removal
- ◆Transparency

5



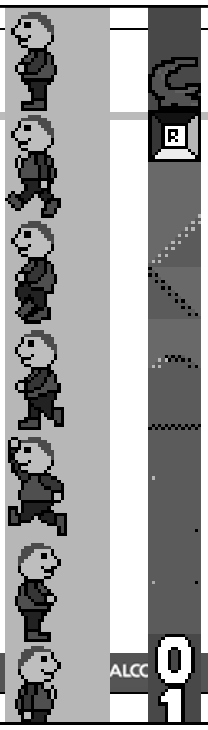
Tile Attributes

- ◆Used to portray the background for your animation.
- ◆Like Sprites, Tiles consist of bitmaps of uniform size:
  - 8x8, 16x16, 32x32 or 64x64 pixels
- ◆Large background images is created from compositions of smaller bitmaps as specified by the Tile Map

6

Sprite / Tile Creation

- Tiles and Sprites are stored in independent bitmaps
- Down in one long column, one image after the next.
- First NxN image is numbered 0; next is 1; etc.
- Notice use of pure green as transparent color.



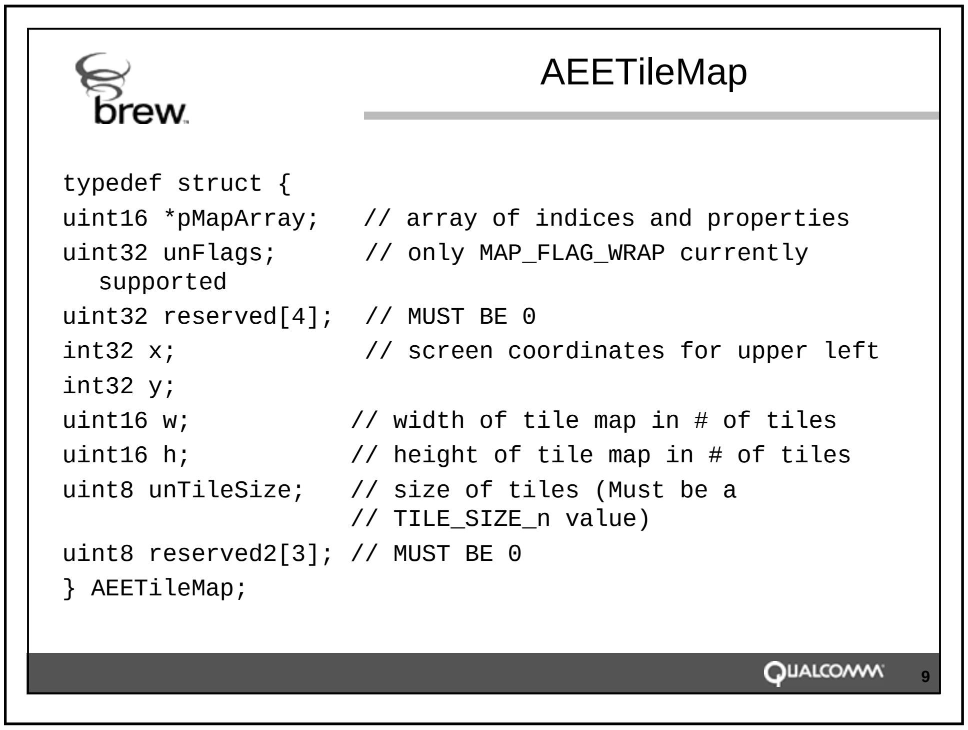
Common ISprite Functions

- ISPRITE_SetDestination()
  Binds a previously created IBitmap to the ISprite Interface
  Target buffer of Draw commands

- ISPRITE_SetTileBuffer()
- ISPRITE_SetSpriteBuffer()
  Passes address of previously opened IBitmap object and number of items contained within. Source buffer for Draw commands

- ISPRITE_DrawTiles()
  Draws all tiles as defined in AEETileMap array

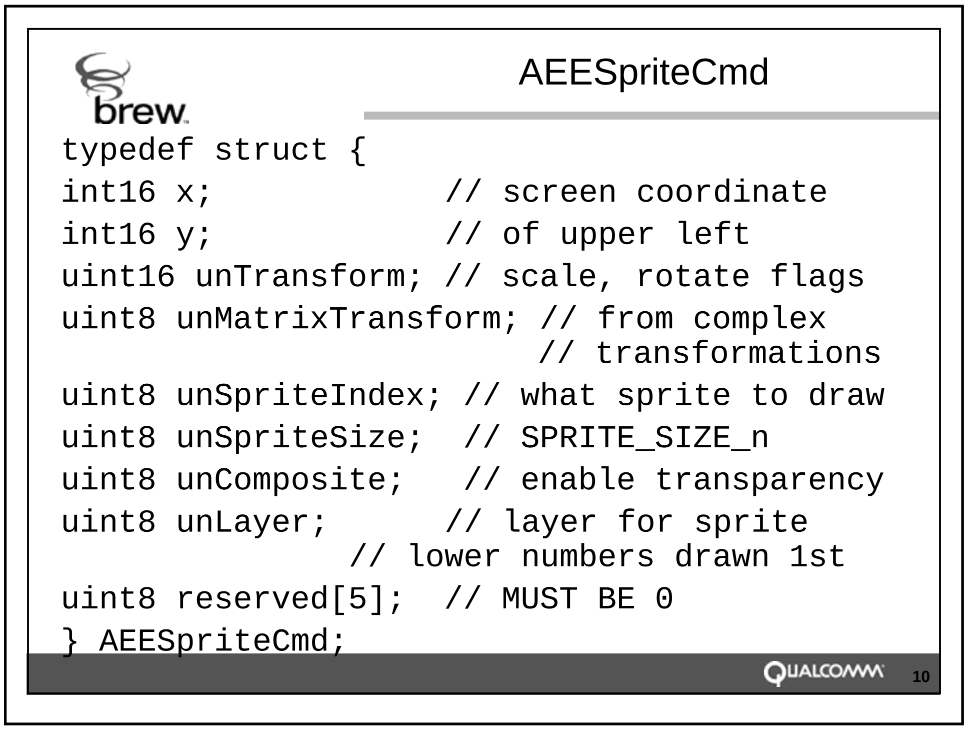
- ISPRITE_DrawSprites()
  Draw all sprites as defined in AEESpriteCmd array



AEETileMap

```
typedef struct {
uint16 *pMapArray;   // array of indices and properties
uint32 unFlags;      // only MAP_FLAG_WRAP currently
  supported
uint32 reserved[4];  // MUST BE 0
int32 x;             // screen coordinates for upper left
int32 y;
uint16 w;           // width of tile map in # of tiles
uint16 h;           // height of tile map in # of tiles
uint8 unTileSize;   // size of tiles (Must be a
                    // TILE_SIZE_n value)
uint8 reserved2[3]; // MUST BE 0
} AEETileMap;
```

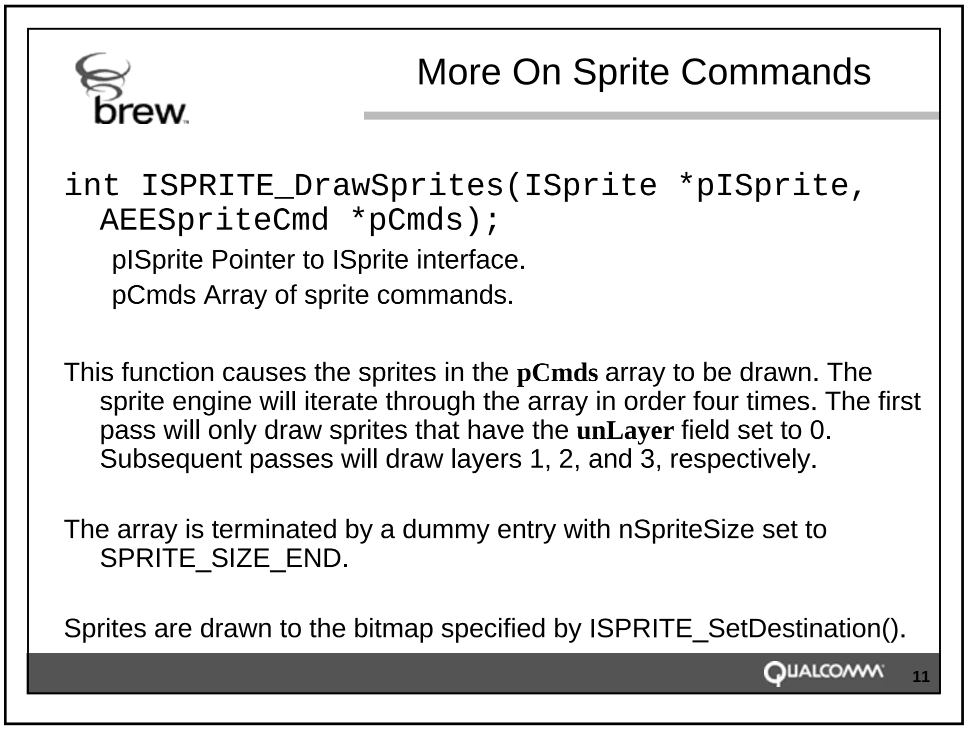
9



AEESpriteCmd

```
typedef struct {
int16 x;            // screen coordinate
int16 y;            // of upper left
uint16 unTransform; // scale, rotate flags
uint8 unMatrixTransform; // from complex
                         // transformations
uint8 unSpriteIndex; // what sprite to draw
uint8 unSpriteSize;  // SPRITE_SIZE_n
uint8 unComposite;   // enable transparency
uint8 unLayer;      // layer for sprite
               // lower numbers drawn 1st
uint8 reserved[5];  // MUST BE 0
} AEESpriteCmd;
```

10



More On Sprite Commands

```
int ISPRITE_DrawSprites(ISprite *pISprite,
  AEESpriteCmd *pCmds);
```

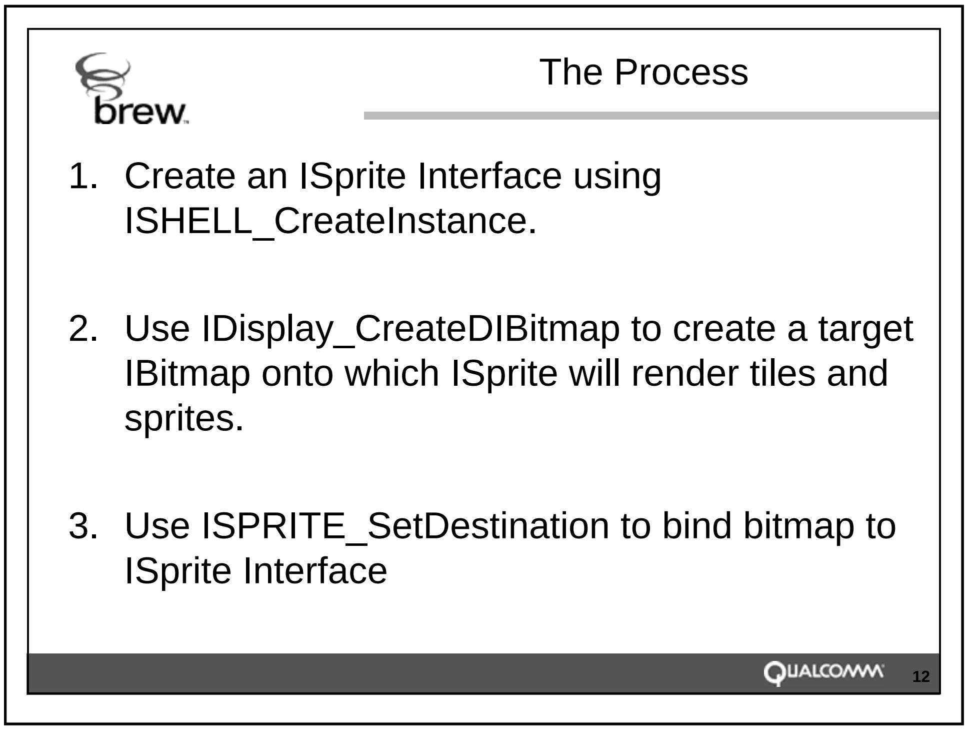
pISprite Pointer to ISprite interface.
pCmds Array of sprite commands.

This function causes the sprites in the **pCmds** array to be drawn. The sprite engine will iterate through the array in order four times. The first pass will only draw sprites that have the **unLayer** field set to 0. Subsequent passes will draw layers 1, 2, and 3, respectively.

The array is terminated by a dummy entry with nSpriteSize set to SPRITE_SIZE_END.

Sprites are drawn to the bitmap specified by ISPRITE_SetDestination().

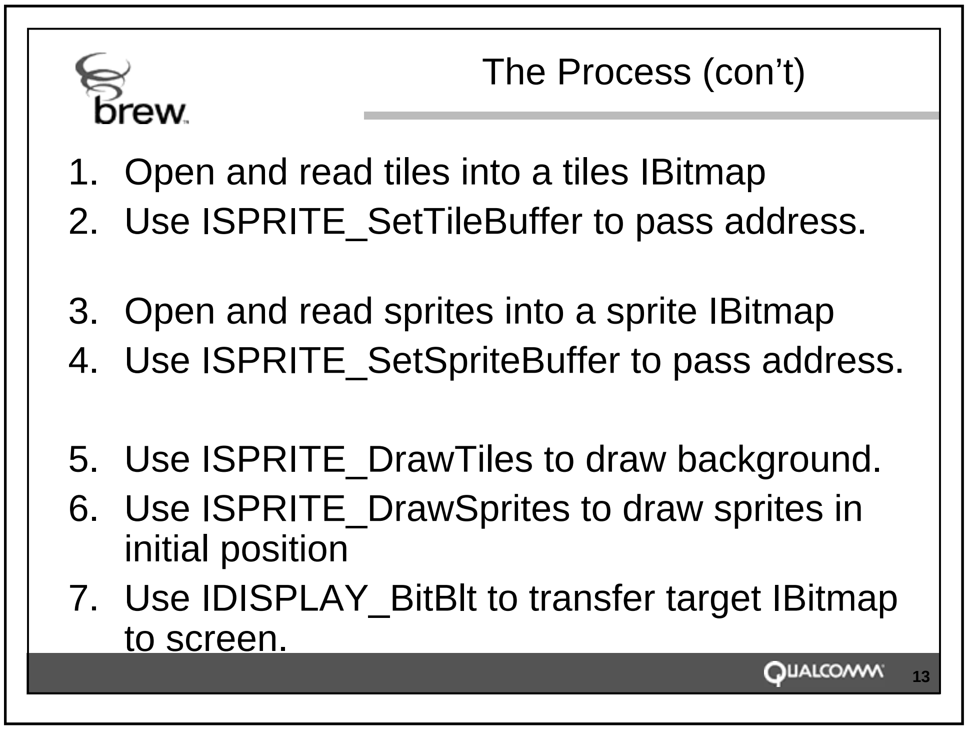
11



The Process

1. Create an ISprite Interface using ISHELL_CreateInstance.
2. Use IDisplay_CreateDIBitmap to create a target IBitmap onto which ISprite will render tiles and sprites.
3. Use ISPRITE_SetDestination to bind bitmap to ISprite Interface

12



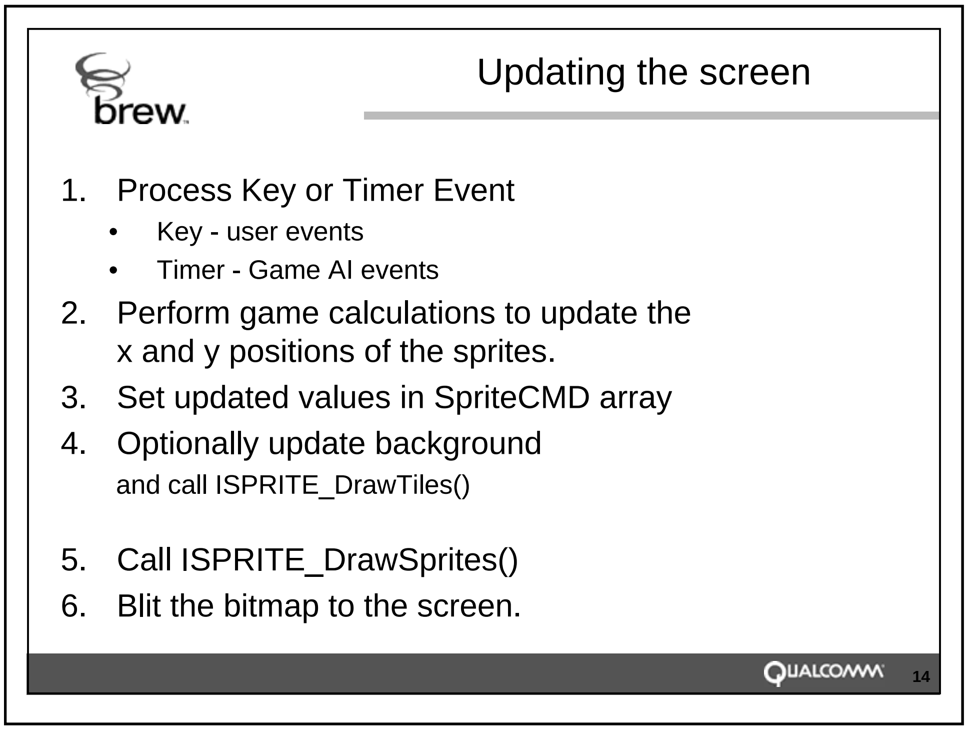
The Process (con't)

1. Open and read tiles into a tiles IBitmap
2. Use ISPRITE_SetTileBuffer to pass address.

3. Open and read sprites into a sprite IBitmap
4. Use ISPRITE_SetSpriteBuffer to pass address.

5. Use ISPRITE_DrawTiles to draw background.
6. Use ISPRITE_DrawSprites to draw sprites in initial position
7. Use IDISPLAY_BitBlt to transfer target IBitmap to screen.

13

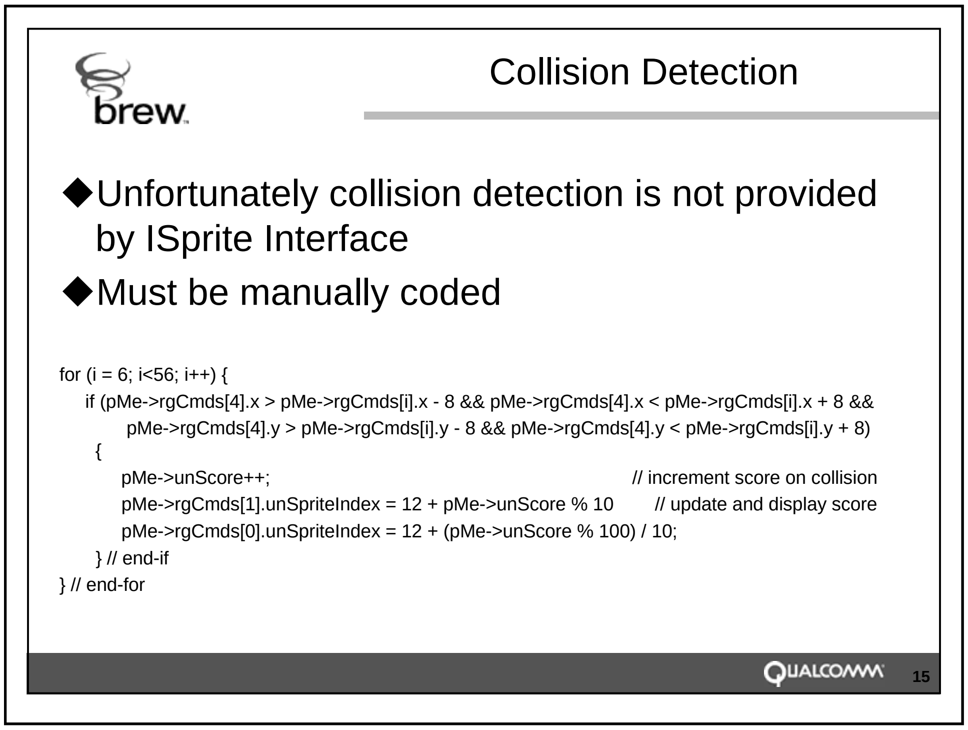


Updating the screen

1. Process Key or Timer Event
   - Key - user events
   - Timer - Game AI events
2. Perform game calculations to update the x and y positions of the sprites.
3. Set updated values in SpriteCMD array
4. Optionally update background
   and call ISPRITE_DrawTiles()

5. Call ISPRITE_DrawSprites()
6. Blit the bitmap to the screen.

14



Collision Detection

- ◆Unfortunately collision detection is not provided by ISprite Interface
- ◆Must be manually coded

```
for (i = 6; i<56; i++) {
    if (pMe->rgCmds[4].x > pMe->rgCmds[i].x - 8 && pMe->rgCmds[4].x < pMe->rgCmds[i].x + 8 &&
          pMe->rgCmds[4].y > pMe->rgCmds[i].y - 8 && pMe->rgCmds[4].y < pMe->rgCmds[i].y + 8)
     {
          pMe->unScore++;                                             // increment score on collision
          pMe->rgCmds[1].unSpriteIndex = 12 + pMe->unScore % 10      // update and display score
          pMe->rgCmds[0].unSpriteIndex = 12 + (pMe->unScore % 100) / 10;
     } // end-if
} // end-for
```

15